PATROL CAR ALLOCATION MODEL: PROGRAM DESCRIPTION

PREPARED FOR THE OFFICE OF POLICY DEVELOPMENT AND RESEARCH, DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT,

AND FOR THE NATIONAL INSTITUTE OF LAW ENFORCEMENT AND CRIMINAL JUSTICE, L.E.A.A., DEPARTMENT OF JUSTICE

JAN M. CHAIKEN
PETER DORMONT

R-1786/3-HUD/DOJ
SEPTEMBER 1975

THE NEW YORK CITY RAND INSTITUTE

Rand
SANTA MONICA, CA. 90406

PATROL CAR ALLOCATION MODEL: PROGRAM DESCRIPTION

PREPARED FOR THE OFFICE OF POLICY DEVELOPMENT AND RESEARCH, DEPARTMENT OF HOUSING AND URBAN DEVELOPMENT,

AND FOR THE NATIONAL INSTITUTE OF LAW ENFORCEMENT AND CRIMINAL JUSTICE, L.E.A.A., DEPARTMENT OF JUSTICE

JAN M. CHAIKEN
PETER DORMONT

R-1786/3-HUD/DOJ
SEPTEMBER 1975

THE
NEW YORK CITY
RAND
INSTITUTE

Rand
SANTA MONICA, CA. 90406

PREFACE

This report provides installation instructions and an annotated program listing for a computer program written in the FORTRAN language, designed to assist police departments in determining the number of patrol cars to have on duty in each geographical command at different times of the day and week. The program--called the Patrol Car Allocation Model--is described in two companion reports:

- R-1786/1, *Patrol Car Allocation Model: Executive Summary*
- R-1786/2, *Patrol Car Allocation Model: User's Manual.*

The first of these, written for police department administrators and planning officials who wish to understand how the Patrol Car Allocation Model can be used in policy analysis, serves as the Summary of both the present volume and the User's Manual. The second provides all the information needed to use the program once it has been installed on a computer system.

The current report (R-1786/3) is written for data processing personnel. Most users will want to read only the first two chapters, which describe installation procedures, file formats, core requirements, and minor program modifications. For the benefit of users who may wish to make substantial modifications, the remainder of the report contains complete documentation, including a program listing with a detailed description of each subprogram.* A separate program used to calculate some of the data input for the Patrol Car Allocation Model is also described and listed in Appendix B, coauthored by David Jaquette.

Preparation of this report was supported by the Office of Policy Development and Research of the U.S. Department of Housing and Urban Development (HUD) under contract H-2164 with The New York City-Rand Institute. Among the objectives of this HUD contract are the development, field testing, and documentation of methods to improve resource

---

*The listings are printed 95 percent of actual size, in order to meet printing space constraints.

allocation procedures in municipal emergency service agencies in the United States.

Design of the Patrol Car Allocation Model was partially funded by HUD and partially by the National Institute of Law Enforcement and Criminal Justice under grant 75NI-99-0012 to The Rand Corporation.

This report is one of a series that documents several different deployment models for police, fire, and ambulance agencies. Further information can be obtained by writing to the addresses in Appendix D.

CONTENTS

INDEX OF PROGRAM SEGMENTS

GLOSSARY

ALGORITHM
A procedure for performing a calculation.

ALLOCATE
1. Assign a certain number of cars to each shift.
2. Divide a fixed total number of car-hours among shifts.

AMPERSAND (&)
At the end of a line of PCAM instructions, signifies that the command continues on the following line.

ASTERISK (*)
1. At the start of a line of output from the DISP command, indicates that the tour is overlaid by another tour.
2. In output tables, indicates a limiting constraint.
3. In input commands, represents the current number of car-hours allocated.

AVAILABLE
1. Ready to be dispatched to a call for service.
2. Not engaged in cfs work or non-cfs work.
3. On preventive patrol.

BATCH
A mode of operating a computer program in which all instructions are prepared on cards or other input device prior to program execution, and output is received later from a high-speed printer. Contrasted with INTERACTIVE.

BLOCK, TIME
A period of time (whole number of hours) over which the number of patrol cars on duty does not change. One or two time blocks constitute a tour.

CALL RATE
Average number of calls for service received per hour.

CALL RATE PARAMETER
A parameter for each day in each precinct. When multiplied by the hourly call-rate factor, gives the expected number of calls for service in the hour.

CAR (see PATROL CAR)

CAR-HOUR (ACTUAL)
One patrol car on duty for one hour.

CAR-HOUR (EFFECTIVE)
One hour spent by one patrol car on any activities other than non-cfs work.

CFS
Call for service.

CFS WORK
1. All activities of a patrol car from the time it is dispatched to a call for service until the time it is available again for dispatch.
2. Number of car-hours spent on such activities.

CFS WORKLOAD
1. Loosely speaking, the extent to which cfs work is a burden on a patrol car.
2. Technically, the number of car-hours of cfs work in a given period of time.

COMMAND
1. An instruction to the PCAM program.
2. An administrative unit in a police department that is supervised by a superior officer. (Used in the expression *geographical command*.)

CONSTRAINING MEASURE
Same as LIMITING CONSTRAINT.

CONSTRAINT
A number specified as the largest or smallest value permitted for a performance measure.

CURRENT-DATA
Some or all of the data in DATABASE, which have been read into the computer memory by a READ command and are used and/or modified by PCAM commands.

DATABASE
The data prepared by the user for input into PCAM.

DAY
A 24-hour period used for organizing PCAM data. Not necessarily a calendar day.

DELAY, TOTAL
Sum of queuing delay and travel time. (Same as TOTAL RESPONSE TIME.)

DELIMITER
Any character other than a letter, digit, parenthesis, asterisk, hyphen, period, or ampersand. Examples of delimiters are blanks, commas, colons, and equal signs.

DESCRIPTIVE MODE
Capability to calculate and display performance measures by time of day and geographical command when the numbers of patrol cars on duty in each shift have been specified.

DIVISION
A combination of precincts. Some police departments use the word "division" for a precinct. This is permitted in PCAM by changing the keyword PRECINCT.

EFFECTIVE CAR
The equivalent of a patrol car that does not engage in any non-cfs work.

EXPONENTIALLY DISTRIBUTED
A random variable T is exponentially distributed if there is a parameter μ such that

$$\text{Prob}(T > t) = e^{-\mu t}.$$

The mean of T is $1/\mu$. The assumption that service times for calls to the police are exponentially distributed is not verified by data, but the assumption is technically necessary in PCAM. (This is a source of PCAM's simplicity.)

FIELDED
In the field. A patrol car is fielded if it is on duty.

FILLER WORD
One of the following words, which may be entered in a PCAM command if desired, but will be ignored by the program: FOR, CAR, HOUR, HOURS, TO, ON, BY, DATA.

HOURLY CALL RATE FACTOR
A parameter for a single hour in a single precinct. When multiplied by the call rate parameter for the day, gives the expected number of calls in the hour.

HOURLY SERVICE TIME FACTOR
A parameter for a single hour in a single precinct. When multiplied by the service time parameter for the day, gives the expected service time (in minutes) for calls received during the hour.

INTERACTIVE
A mode of operating a computer program whereby the user enters instructions at a terminal and receives output immediately at the same terminal. Contrasted with BATCH.

KEYWORD
A character string that has a special meaning to the PCAM program. These are either filler words or one of the following: DAY, P, C, T, F, ADD, ALOC, DISP, END, LIST, MEET, READ, SET, WRITE, TOUR (or a substitute provided by the user), DIVISION (or a substitute), PRECINCT (or a substitute).

LIMITING CONSTRAINTS
When meeting constraints, the particular performance measures whose constrained values lead to a need for the largest number of patrol cars. (If these constraints were eliminated, a smaller number of patrol cars would meet all the constraints.)

LIST
Command that causes PCAM to print out the values of the data items associated with all precincts, days, and tours within its scope.

MINIMUM ALLOCATION
The smallest whole number of actual patrol cars that can be assigned to a shift to keep the average utilization of an *effective car* under 1 in every hour.

NEW-DATA
A permanent file which is created by the WRITE command from all or part of CURRENT-DATA.

NON-CFS WORK
1. Any activity of a patrol car that makes the car unavailable for dispatch but was not generated by a previous dispatch to a call for service.
2. Number of car-hours spent on such activities.

OBJECTIVE FUNCTION
The performance measure to be minimized by an allocation.

OPTIMAL
Yielding the smallest possible value of the objective function.

OUTPUT ORDER
A choice of displaying output tables either by tour within day within precinct, or by precinct within tour within day.

OVERLAY TOUR
A tour that begins during one tour and ends during the following tour.

PARAMETER
A number that characterizes a particular hour, block, shift, day, or precinct. See also SERVICE TIME PARAMETER and CALL RATE PARAMETER.

PATROL CAR
A mobile vehicle that can respond to calls for service from the public. Includes vehicles other than automobiles that serve the same function, e.g., scooters.

PATROL FREQUENCY
Number of times per hour that a random point will be passed by a car on preventive patrol.

PCAM
Patrol Car Allocation Model.

PLUS (+)

At the start of a line of output from the DISP command, indicates that the tour is an overlay.

POISSON PROCESS

In the PCAM context, the occurrence of calls for service in a given precinct during a given hour constitutes a Poisson process if there is a parameter λ such that the time between calls has the distribution

$$\text{Prob(time between calls} > t) = e^{-\lambda t}.$$

This assumption is well verified by data.

PRECINCT

A geographical area that is treated as independent from other areas by the patrol car dispatcher. Each patrol car is assigned to an entire tour in one precinct, although it may work in only part of the precinct.

PRESCRIPTIVE MODE

Capability to suggest the number of patrol cars that should be on duty during each shift, so as to meet standards of performance specified by the user.

PREVENTIVE PATROL

The practice of driving a patrol car through an area, with no particular destination in mind, looking for criminal incidents or opportunities, suspicious occurrences, etc.

PRIORITY

Importance of a call for service. PCAM permits three priority levels. Priority 1 calls are so important that the dispatcher will violate ordinary dispatching practices to get a patrol car to respond immediately. Priority 2 calls are important enough that a rapid response is preferred over a slow response. Priority 3 calls can wait in queue without deleterious effect.

QUALIFIER

Phrase(s) associated with a computer command, defining the scope of the command. May be any subset of these phrases, separated by delimiters: 'TOUR=⟨NAMELIST⟩', 'DAY=⟨NAMELIST⟩', 'DIVISION=⟨NAMELIST⟩', 'PRECINCT=⟨NAMELIST⟩'.

QUEUE

In the PCAM context, a collection of calls for service that are waiting to be assigned to a patrol car because no patrol car is available at the moment.

QUEUING DELAY

The length of time a call for service waits in queue.

REGRESSION ANALYSIS
A procedure for fitting a straight line to data so as to minimize the sum of the squares of the deviations of the data from the straight-line estimate.

RESPONSE TIME, TOTAL
Sum of queuing delay and travel time. (Same as TOTAL DELAY.)

SCOPE
The collection of precincts, tours, and days to which the action of a PCAM command applies.

SERVICE TIME
Number of minutes a patrol car will be unavailable from the time it is dispatched to a call until it is available to respond to another call.

SERVICE TIME PARAMETER
A parameter for each day in each precinct. When multiplied by the hourly service time factors, gives the expected service time in each hour.

SHIFT
A particular tour in a particular precinct on a particular day.

SQUARE-ROOT LAW
An equation for the average travel distance D in a region of area A when N patrol units are available:

$$D = \sqrt{\frac{A}{N}} .$$

STEADY STATE
In the PCAM context, a situation where the probability of finding n cars available does not change over the course of an hour.

SUPPRESSIBLE CRIMES
Any crimes whose occurrence might conceivably be affected by the amount of preventive patrol. (It is not known whether any crimes are actually "suppressed" by preventive patrol.) The PCAM user is free to define this category of crimes however he chooses.

TIME BLOCK
See BLOCK, TIME.

TOTAL DELAY
Same as TOTAL RESPONSE TIME; the sum of queuing delay plus travel time.

TOUR
A period of time (whole number of hours) beginning when a patrol officer starts work for the day and ending when the officer finishes work. In PCAM, tours are assumed to start at the same time in every precinct on every day (but overlay tours need not be present on every day in every precinct).

TRAVEL TIME

The length of time from the moment a patrol car is dispatched to an incident until the moment it arrives at the scene.

UNAVAILABILITY PARAMETERS

A pair of constants B1 and B2 for each precinct that give the best regression fit to the linear equation

$$\begin{pmatrix}\text{fraction of time}\\ \text{on non-cfs work}\end{pmatrix} = \text{B1} \times \begin{pmatrix}\text{fraction of time}\\ \text{on cfs work}\end{pmatrix} + \text{B2} .$$

UTILIZATION

The fraction of time a patrol car is busy on cfs work.

I. PROGRAM INSTALLATION

INTRODUCTION

The Patrol Car Allocation Model (PCAM) is a computer program designed to help police departments determine the number of patrol cars to have on duty in each of their geographical commands. Typically, the number of patrol cars needed will vary according to the season of the year, day of the week, and hour of the day.

A companion user's manual describes applications of the program, explains the meaning of the various items of data to be included in the data base, and gives complete instructions for operating the program once it is installed:

- Jan M. Chaiken and Peter Dormont, *Patrol Car Allocation Model: User's Manual*, R-1786/2.

The PCAM program is written in the FORTRAN language and is compatible with most FORTRAN compilers. The particular compiler features required by the program will be described below.

Successful use of the PCAM program requires little or no expertise in the use of computers. The user controls the program with a sequence of simple commands. These can be punched on cards for operation in batch mode (where the program's output is produced on a line printer), or they can be entered at a teletype or other slow-speed terminal for operation in interactive mode (in which case the program's output is displayed immediately at the terminal). Some of the facilities provided by the commands are:

- Data selection
- Allocation of patrol cars to meet constraints on performance measures
- Allocation of patrol cars to best achieve specified objectives
- Display of measures describing expected patrol car performance under particular allocations.

The data required for processing these commands must be supplied to the program in an external file that we call DATABASE. The format for this file is described in Chapter II.

Installing PCAM on a computer system is a simple and straightforward operation. However, various computer systems differ with respect to their conventions for accessing files in the FORTRAN language, and this may have to be taken into account in program installation. In addition, users may wish to optimize the amount of run-time storage reserved, with respect to the size of their data base and intended use of the program.

The program as listed in Chapter V and distributed by Rand is set up to run in batch mode, with changes for interactive mode indicated (see "Minor Program Modifications," below). However, on request we will supply the program in a form suitable for interactive operation. If the program is to be used primarily in batch mode, many users will wish to make program changes to enhance the appearance of the output.

This chapter provides the information needed to install the program and make the indicated types of changes. The user wishing to make more substantial changes will have to familiarize himself with Chapters IV and V. Refer to the Glossary and the User's Manual for definitions of unfamiliar terms.

PCAM SOURCE LANGUAGE AND COMPILATION

The PCAM program is written in the FORTRAN language. The program conforms closely to ANSI* standards, but two extensions were used to simplify coding. These are:

- o Generalized subscripts (subscripts can be coded as arbitrary integer-valued expressions), e.g.,
  IP=ICDAT(LPARM+(IPARM-1)*2+LBLKTB(2))
- o Use of quoted literals in FORMAT statements, e.g.,
  1   FORMAT(' THIS FORMAT STATEMENT IS ALLOWED').

When all desired modifications to the source code have been made,

*American National Standards Institute, *FORTRAN*, X-3, 9-1966.

the PCAM program should be compiled, and the object program saved in an execution-ready form. On IBM 360 or 370 systems running under OS or similar operating systems, the following JCL might be used to accomplish this:

```
//     jobcard
//STEP1 EXEC FORTGCL
//FORT.SYSIN DD *
             :
            PCAM source program
             :
/*
//LKED.SYSLMOD DD   definition of load module library
//LKED.SYSIN DD *
    NAME PCAM
/*
```

FILE STRUCTURE AND CONVENTIONS

The basic inputs to the PCAM program are (1) a sequence of commands, supplied by the user on cards or through an interactive terminal, which control the functions performed by the program, and (2) the DATABASE file on a direct access or magnetic tape device, which describes the characteristics of a city that are relevant to PCAM's modeling of its police patrol operations. PCAM's basic operations can only be performed on that part of DATABASE which resides in the computer's main memory. The user directs part or all of the data to be read from DATABASE by means of a READ command, as described in the User's Manual. The term CURRENT-DATA refers to the data that have been read from DATABASE and are available for processing.

The user can modify the contents of CURRENT-DATA through various commands. PCAM also has the capability of writing out a file containing part or all of the information in CURRENT-DATA. The file created by this operation is called NEW-DATA and is written in response to a WRITE command (see the User's Manual). It is in the same format as DATABASE and can be used in its place in subsequent runs of PCAM.

PCAM references all files through INTEGER variables that contain FORTRAN unit numbers. This facilitates changing the unit numbers used by PCAM to conform to the conventions of a particular operating system.

Table 1 describes PCAM's files in terms of these file reference variables and gives the values of the variables in the distributed program. All file reference variables (except the variable for NEW-DATA) are in COMMON/SYSTEM/. To change the values of these file reference variables, the DATA statement numbered 3870 in Chapter V, located in the BLOCK DATA subprogram, should be modified.

Table 1

PCAM FILES

Variable Name	Device	Description	Unit Number
SYSIN	Teletype, terminal, card reader, disk, tape, etc.	User's command input	4
SYSOUT	Teletype, terminal printer, disk, tape, etc.	Printed output from program	5
IFILE	Tape, direct access	Input data (DATABASE)	19
NUNIT	Tape, direct access	Output data base (NEW-DATA)	Value determined from user input in the WRITE command
LIT	Tape, direct access	Scratch file for literals (space needed for three 80-character records)	20

Whether the user changes the unit numbers or not, most operating systems require the user to prepare data definition statements that identify the device or file corresponding to each unit number. For example, the following JCL is used to run the program on the Rand Computation Center's IBM 370 computer:

```
//     jobcard
//S1   EXEC PGM=PCAM,REGION=160K
//STEPLIB DD DSN= name of load module library,DISP=SHR
//GO.FT04F001 DD DSN=name of command file,UNIT=USER,DISP=SHR
//GO.FT05F001 DD SYSOUT=A,DCB=(RECFM=FA,LRECL=81,BLKSIZE=81)
//GO.FT06F001 DD SYSOUT=A
//GO.FT18F001 DD DSN=name of NEW-DATA file,UNIT=USER,
//    VOL=SER=volume,SPACE=(TRK,(4,1),RLSE),DISP=(NEW,CATLG),
//    DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//GO.FT19F001 DD DSN=name of DATABASE file,UNIT=USER,DISP=SHR
//GO.FT20F001 DD DSN=&&PCAM,UNIT=TEMP,VOL=SER=TEMP10,
//    SPACE=(TRK,(10,2),RLSE),DISP=(NEW,DELETE)
```

The FT18 DD statement allows the user to write a NEW-DATA file on FORTRAN unit 18. In other words, the user is permitted to enter the command WRITE [DATA] [ON] 18 [FOR] <QUALIFIER>.

STORAGE ALLOCATION

Most run-time storage that PCAM uses is allocated dynamically from a large one-dimensional array (see Chapter IV). The array is named CDAT (an abbreviation for CURRENT-DATA) and is contained in COMMON/STORE/. Wherever COMMON/STORE/ occurs in the program, an array ICDAT of the same size as CDAT is defined and is equivalenced to CDAT. The minimum amount of storage that must be reserved for CDAT depends on the size of the user's data base and on how much of the data base will be accessed in a single READ command.

Four different classes of information are stored in CDAT. The storage requirements for each class are given below. The sum of these requirements is the minimum size for array CDAT.

1. Permanent Tables

Tables that are allocated at the start of each PCAM run require the following number of words of storage:

$$(9 \cdot \text{NDAYDT}) + (13 \cdot \text{NTRDT}) + (3 \cdot \text{NBLDT}) + (8 \cdot \text{NDIVDT}),$$

where NDAYDT = number of days of data in DATABASE, NTRDT = number of tours in each day in DATABASE, NBLDT = number of blocks for each day in DATABASE, and NDIVDT = number of divisions in DATABASE.

2. Variable Size Tables

Tables whose size depends on the number of divisions, days, and tours selected in a READ command qualifier require the following number of words of storage:

NDAYRD + NTRRD + NDIVRD,

where NDAYRD = number of days read into CURRENT-DATA, NTRRD = number of tours read, and NDIVRD = number of divisions read.

3. Data Storage

Data read from DATABASE into CURRENT-DATA by a READ command require the following number of words of storage:

NPCTRD · NWDPCT,

where NPCTRD = number of precincts included in CURRENT-DATA and the calculation of NWDPCT will be explained below in Chapter IV (Table 7).

4. Temporary Storage

Temporary storage is used for names and numbers* during command interpretation and execution. The exact amount of this storage that is needed for any command is given by:

8 · (number of names in command)
+ 2 · (number of numbers in command).

Temporary storage is always released when the execution of a command is complete. An allocation of about 150 words for this type of storage will be sufficient for most applications.

The program as distributed allows 11,000 words for the sum of these four requirements. Most users will find this amount of space adequate. If the user's DATABASE is too large, an error message will be printed when the program is run, and then it will be necessary to calculate the actual requirements as listed above.

---

*Names appear in user commands to label entities such as precincts, days, and tours. Numbers are used to identify output tables, objective functions, constraints, files, and numerical quantities. For more detailed information, see Chapter II and also the User's Manual.

If the user's DATABASE requires less than 11,000 words for CDAT, the program will operate properly, and a message will be printed after the END command indicating the total number of words actually used for the first three requirements listed above. The user can then reduce the amount of space allocated to CDAT, if he desires to do so. The only advantage in making this modification will be a reduction in the cost of running the program.

In order to change the space allocation, either an increase when necessary or a decrease when desired, the dimensions of CDAT and ICDAT must be changed on the following pairs of lines of the program (see Chapter V):

13, 15	2210, 2212
81, 83	2263, 2265
215, 217	2311, 2313
260, 262	2365, 2367
565, 567	2419, 2421
618, 620	2573, 2575
996, 998	2749, 2751
1153, 1155	2875, 2877
1203, 1205	2953, 2955
1232, 1234	2976, 2978
1365, 1367	3242, 3244
1450, 1452	3285, 3287
1493, 1495	3322, 3324
1532, 1534	3402, 3404
1586, 1588	3444, 3446
1685, 1687	3486, 3488
1775, 1777	3623, 3625
1947, 1949	3862, 3864

In addition, in BLOCK DATA, NWORDS must be set equal to the dimension of CDAT. This occurs on line 3866.

CORE REQUIREMENTS

A moderate amount of core is required to run the PCAM program. Although the exact amount will vary from one computer system to another, the figures given below will serve as a good guideline.

The core requirements for the PCAM program depend directly on the size of the array CDAT; call this NWORDS. The amount of core required to run PCAM on the Rand Computation Center IBM System 370 computer is given by: (115 + 4 · NWORDS/1024) K bytes. The program as distributed has NWORDS=11,000, and therefore requires 158K bytes of core, but we suggest requesting 160K bytes. For installations with other than IBM 360/370 computers, the equivalent requirement can be obtained by using the fact that there are four bytes in a word on IBM 360/370 computers and that 1K byte = 1024 bytes.

For the assistance of potential users who are severely restricted in the amount of core, we point out that the core requirements for PCAM can be reduced by means of a technique called chaining.* (We do not recommend chaining the PCAM program unless absolutely necessary.) This technique allows only those parts of the program that are required to perform a particular function to be resident in core. The remainder of the program can remain in external storage until required. For example, an examination of the program listing in Chapter V and the cross-referenced listing of program segments in Appendix C reveals that while a LIST command is being executed (by subroutine LIST) there is no need for subroutine WRITE, which implements the WRITE command, to be core-resident.

The best way to break up the PCAM program for chaining will vary from installation to installation and depends upon the amount of core available to run the program and the way in which it is used. In general, those subprograms required for the execution of all, or most, commands should be core-resident throughout the program's execution. The subprograms that are needed to execute particular commands should be grouped so that only the group required to execute one command resides in core with the continuously resident subprograms.

MINOR PROGRAM MODIFICATIONS

To convert the PCAM batch program into an interactive program, five lines must be removed from the program (43, 44, 51, 416, and 417), and two lines must be inserted (25 and 26). These are described in Chapter V under the headings "MAIN Program" and "Subroutine GETTKN."

---

*This is frequently called "overlaying," but we do not use this terminology because the word "overlay" has been assigned a specific (and different) meaning in the User's Manual and the Glossary.

If the program is operated in batch mode, the user may wish to change the appearance of output, since there is more room on a page of output from a high-speed printer than there is on output from a teletype or similar interactive terminal. Changes in spacing of columns of output, number of decimal places displayed, and column labels may be made as follows:

- o For the table displayed by the LIST command, change format statements in Subroutine LIST.
- o For the tables displayed by the DISP command, change format statements in Subroutine TITLE and Subroutine PRTBL (print table).

In addition, output data of no interest to the user can be completely suppressed by modifying the same subroutines.

To change pagination, or to provide a different heading at the top of the first page (for example, the name of the police department or the date of the run), modify the MAIN program or the INIT subroutine.

If all tours* have the same length, the department may prefer to have PCAM allocate *cars* rather than *car-hours*. The program as distributed will accept commands referring to cars, such as ALOC 24 CARS BY F(2), but it will interpret the expression "24 cars" to mean 24 car-hours. To change the program so that the input number is interpreted as cars, line 3103 of the program must be changed following the instructions on lines 3098-3102 (see Chapter V).

COSTS

The cost of running the PCAM program will vary from installation to installation. However, we can give a rough idea of the range of costs based on our experience with two computer systems. Compiling the program costs approximately $10, and this is more expensive than

---

*In the PCAM documentation, the word "tour" is used to designate the period of time during which a patrol car is on duty. The program itself employs whatever term the user specifies--watch, shift, platoon, or whatever.

most runs of the program after compilation. It is therefore desirable to save the object code from the compiled program.

The demonstration of the program illustrated in the figures in Chapter III of the User's Manual was run from object code at a cost under $2. Typical applications should involve costs under $10 for machine time unless numerous ADD or ALOC commands are entered, or these commands are performed on a large number of shifts simultaneously. (In PCAM terminology, a *shift* is a tour in all precincts at once. The number of shifts is the product of the number of precincts, days, and tours included explicitly or implicitly in a command qualifier.) In general, PCAM is an inexpensive program to operate and compares favorably with any other program that could answer similar policy questions.

II. PCAM DATA FILE FORMAT

This chapter describes the format of the DATABASE and NEW-DATA files mentioned in Chapter I. Figure 1 and the demonstration DATABASE in Appendix A may assist the user in interpreting the instructions in this chapter. The format items shown are those used to read the DATABASE file. Those used to write NEW-DATA files are different in some respects as noted, but always produce a file that can later be read according to the formats shown for DATABASE. The reader is referred to the Glossary and the User's Manual for the definitions of unfamiliar terms. Appendix B documents a computer program (not part of PCAM) that may assist some departments in calculating values for the unavilability parameters B1 and B2 that appear in the precinct header record, described below.

The DATABASE file must be prepared in standard 80-column records on a disk or other rewindable storage device. The PCAM program uses the variable name IFILE for DATABASE and assumes it is located on unit 19. The user may change the unit number in COMMON/SYSTEM/, which is initialized on line 3870 in BLOCK DATA (see Chapter V).

1. Control record. This is the first record in the data base.

Columns	Format*	Comments	Description
1-8	A8	Left justify	The word DIVISION, or whatever word the department uses for aggregations of precincts.
11-18	A8	Left justify	The word PRECINCT, or whatever word the department uses for precincts.
21-28	A8	Left justify	The word TOUR, or whatever word the department uses for tour.
30-31	I2	Right justify	Number of divisions in the data base.
33-35	I3	Right justify	Number of precincts in the data base.
37-39	I3	Right justify	Number of days of data which are supplied for each precinct.
41-42	I2	Right justify	Number of time blocks in each day.
44-45	I2	Right justify	Number of tours in each day.
47	I1		Indicator for overlay tour. Enter 0 or 1 as described below.

*All A8 formats are read as 8A1.

PCAM permits the following possibilities for overlay tours: (a) there are no overlay tours, (b) every day in every precinct has an overlay tour, or (c) some days and/or some precincts have an overlay tour. Enter 0 for the overlay tour indicator in case (a); enter 1 in case (b) or (c). In cases (b) and (c), the last tour in the data for every day in every precinct must be the overlay tour. However, in case (c) the overlay tour data will be blank for some days and/or precincts.

2. Day name record(s). If there are ten days or fewer in the data base, this is the second record. Otherwise, continuation records will be needed; supply as many as required.

Columns	Format	Comments	Description
Begin in 1	10A8	Left justify	Name for each day in the data base. For each precinct, day data will have to be in the same order as the names on this record.

3. Blocks descriptor record. This follows the day name record(s).

Columns	Format	Comments	Description
Begin in 1	24(I2,1X)	Right justify	Last hour of each time block. Supply as many hours as there are time blocks in a day, up to 24. The hours must be in increasing order.

4. Tour descriptor records. If there are ten or fewer days in the data base, and each day has N tours, the tour descriptor records will be the 4th, 5th, ..., 3 + Nth records. There is one such record for each tour, with the overlay tour (if any) last.

Columns	Format	Comments	Description
1-8	A8	Left justify	Name of tour.
10-11	I2	Right justify	Ordinal number of the first time block (or only time block) in the tour.
13-14	I2	Right justify	Ordinal number of the second time block in the tour. Zero or blank if the tour has only one time block.

5. Precinct header record. The first such record follows the last tour descriptor record. The next such record follows all the data for the first precinct. In total, the number of precinct header records will equal the number of precincts in the data.

Columns	Format	Comments	Description
1-8	A8	Left justify	Precinct name.
10-17	A8	Left justify	Division name for this precinct.
20-24	F5.0*		Area of precinct (in square miles).
26-30	F5.0**		Total length of streets in precinct (in miles).
32-36	F5.0***		Unavailability parameter B1.
38-42	F5.0***		Unavailability parameter B2.

6. Day detail records. There are three of these records for each day in each precinct. The first three follow the first precinct header, the next three appear after all data for shifts and blocks in the first day in the first precinct, etc.

Record	Columns	Format	Description
1	1-5	F5.0	Call rate parameter.
	7-11	F5.0	Service time parameter.
	13	I1	An indicator for the presence or absence of an overlay tour for this day for this precinct. Enter 0 if there is no overlay tour, 1 if there is an overlay tour.
2	1-72	24(F3.2)+	Call rate factors for each hour of the day. The product of one of these factors and the call rate parameter in record 1 should be the number of calls occurring in the precinct in the corresponding hour of the day.
3	1-72	24(F3.2)+	Service time factors for each hour of the day. The product of one of these factors and the service time parameter in record 1 should be the average service time for calls occurring in the precinct in the corresponding hour of the day.

*F5.2 in NEW-DATA.

**F5.1 in NEW-DATA.

***F5.3 in NEW-DATA.

+No decimal points in NEW-DATA.

7. Shift detail records. There is one such record for each tour for each day for each precinct. After the third day detail record for each day in each precinct, there will be N of these records, describing the N tours in that day.

Columns	Format	Description
1-5	F5.0*	Average number of cars on duty during the shift.
7-11	F5.0*	Average speed of cars when responding to calls (in miles/hour).
13-17	F5.0*	Average speed of cars when on preventive patrol (in miles/hour).
19-23	F5.0**	Fraction of calls which are of priority 1.
25-29	F5.0**	Fraction of calls which are of priority 2.

8. Block detail records. There is one such record for each day for each precinct. It must follow the shift detail records for that day and precinct. The number and order of the entries in this record must be the same as in the blocks descriptor record.

Columns	Format	Description
Begin in 1	24F3.1	Average total number of suppressible crimes (or outside robberies, etc.) occurring in each time block. These may not be zero.

---

*F5.1 in NEW-DATA.

**F5.4 in NEW-DATA.

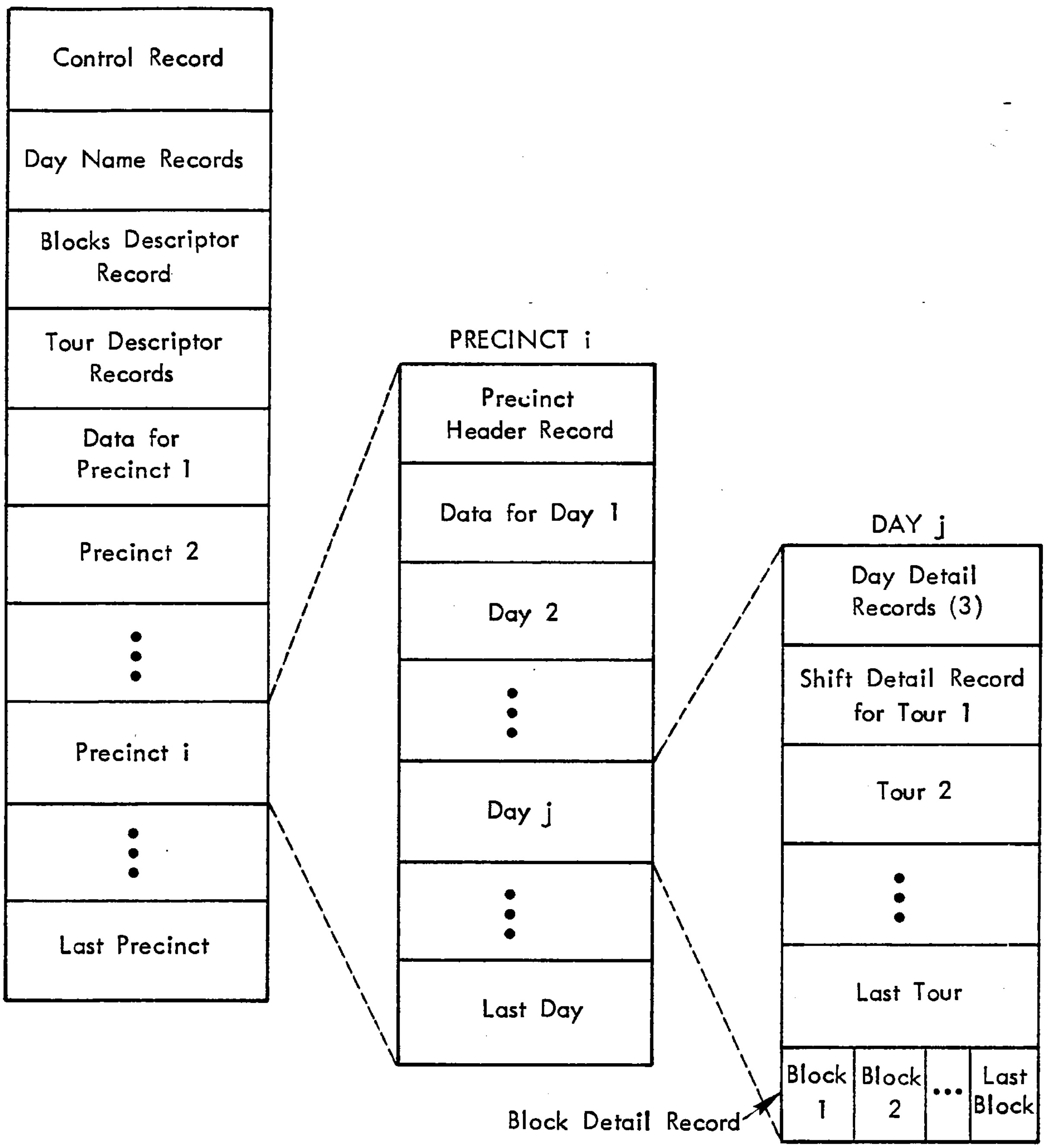

Fig. 1 — Order of data in DATABASE

III. ALGORITHMS FOR CONVERSION OF ALLOCATIONS BETWEEN TOURS AND BLOCKS

This chapter documents the algorithms that PCAM uses to convert allocations of cars to tours into block allocations, and vice versa. Recall that a tour is a period of time over which a patrol car can be on duty, and a block is a part of a tour during which the number of patrol cars on duty is constant.

CONVERSION OF TOUR ALLOCATIONS TO BLOCK ALLOCATIONS

Given an allocation of cars to the tours of a day in a precinct, PCAM determines the resulting allocation of cars to blocks as follows:

1. Set the number of cars assigned to each block of the day to zero.
2. For each tour of the day, including the overlay tour if any, add the number of cars assigned the tour to the number of cars assigned to each of its blocks. For example, consider four blocks named 1, 2, 3, and 4, and three tours named A, B, and C. Tour A works blocks 1 and 2, tour B works blocks 2 and 3, and tour C works blocks 3 and 4. Let N_i be the number of cars on duty during block i, and let M_a be the number of cars assigned to tour a. Then $N_1 = M_A$, $N_2 = M_A + M_B$, $N_3 = M_B + M_C$, $N_4 = M_C$.

CONVERSION OF BLOCK ALLOCATIONS TO TOUR ALLOCATIONS

PCAM uses two different algorithms to convert block allocations to tour allocations. One algorithm is used to determine the allocation of cars to tours after constraints have been met for blocks. The other algorithm is used to determine the required increase in car allocation when the number of cars allocated to tours is not enough cars to handle the call-for-service workload in all hours of a day.

1. The algorithm to determine the tour allocation after constraints are met for all blocks of a day in a precinct is the following. For each tour of the day:
   a. If the tour is not involved in an overlay, assign the maximum of the number of cars in its blocks.
   b. If an overlay tour starts during the tour, assign the number of cars assigned to its first block. Save this as N_1. Save the number of cars assigned to its second block as N_2.
   c. If an overlay tour ends during the tour, assign the number of cars assigned to its second block. Save this as N_4. Save the number of cars assigned to its first block as N_3.
   d. If the tour is an overlay tour, assign $N = \max(N_2 - N_1, N_3 - N_4, 0)$ cars.
   e. If N equals 0, STOP.
   f. Let $\delta = \max(N_2 - N_1, 0) - \max(N_3 - N_4, 0)$.
   g. If $\delta > 0$ and the overlay tour is longer than the first overlaid tour, add δ cars to the first overlaid tour and remove δ cars from the overlay tour. If $\delta < 0$ and the overlay tour is longer than the second overlay tour, add δ cars to the second overlay tour and remove δ cars from the overlay tour. If $\delta = 0$ or the overlay tour is not longer than the overlaid tour, make no adjustments.
2. Under conditions where the allocation of cars to the blocks of a day is insufficient to handle the call-for-service workload in each hour (this can result from the execution of a READ or SET command), the following algorithm is used to determine where to increase the assignment of cars.
   a. Determine the minimum number of cars required for each deficient block and assign that number of cars. Mark such blocks as deficient.
   b. For each tour of the day (the overlay tour last):
      i. If the tour is not involved in an overlay and has a deficient block, assign the maximum number of cars assigned to its blocks.

ii. If an overlay tour starts during the tour, save the number of cars assigned to its first block as N_1. If its first block is deficient, assign N_1 cars to the tour.

iii. If an overlay tour ends during the tour, save the number of cars assigned to its second block as N_4. If its second block is deficient, assign N_4 cars to the tour.

iv. If the tour is an overlay and one or both of its blocks is deficient, assign cars as follows. Let N_2 be the number of cars assigned to its first block and N_3 be the number of cars assigned to its second block (these include the effect of increasing a block assignment to meet the workload restriction in step a above. However, note that at this stage in the algorithm, any assignments made to tours in steps b and c have not been converted to block assignments). The number of cars assigned to the overlay tour is then max(N_2 - N_1, N_3 - N_4, number cars currently assigned).

c. The algorithm described above (Conversion of Tour Allocations to Block Allocations) is then used to redetermine the block assignments.

Chapter IV
INTERNAL DATA STRUCTURES

This chapter describes PCAM's internal data structures and its run-time storage management system. An understanding of these aspects of the program is necessary only if the user wishes to interpret the program listings in Chapter V or modify the program. This section assumes a familiarity with the data base format given in Chapter II. Refer to the Glossary and the User's Manual for definitions of unfamiliar terms.

STORAGE MANAGEMENT

The amount of core storage required by PCAM will vary according to the size of the data base and the portion of the data base selected in each READ command. To allow for this variation, while enabling the program to run with the minimum amount of core storage required for a particular data base, PCAM dynamically allocates much of the storage that it uses.

Dynamic storage allocation is accomplished by reserving a large, one-dimensional array, the size of which can be set when the program is compiled. Then, when a variable amount of storage is required for some purpose, it can be allocated from the array, at which time the subscript of the first word allocated is saved for future reference. The array is referenced by the variable name CDAT when REAL data are accessed and by ICDAT when INTEGER data are accessed.

Two subroutines are used to allocate storage from CDAT; these are GETBOT and GETTOP. GETBOT allocates storage from the "bottom" of the array. This storage is used for three types of data: (a) tables whose dimensions depend only on certain parameters describing the data base and do not vary during program execution; (b) tables whose dimensions can change as a result of the number of days and tours selected in a READ command, and (c) data read from the data base by a READ command. Subroutine GETTOP allocates storage from the "top" of CDAT. This is basically "scratch pad" storage, used during interpretation of all commands and sometimes during command execution.

The storage management system is actually rather simple. Storage is allocated on a last-in-first-out basis. Each routine that requests storage has the responsibility of releasing it or not, depending upon intended future use. Storage is released by setting a pointer to a subscript which represents the highest or lowest free word of CDAT, depending upon whether storage is being freed from the top or bottom. Thus, care has been exercised so that storage is not prematurely freed and unrecoverable "holes" are not left in allocated storage.

TABLE POINTERS

This section describes the dynamically allocated tables used by PCAM. Pointers to these tables and table dimensions are saved in COMMON/PNTRS/. This common block also contains certain variables relating to overlay tours. For completeness, these variables will also be described in this section. Table 2 below lists each variable in COMMON/PNTRS/, its contents, and the routine where its value is set. If storage is allocated for a table in a routine other than the one in which its entries are made, the name of the routine making the table entries appears in parentheses. Variables beginning with 'N' are dimensions or counters, while those beginning with 'L' are pointers.

Table 2

VARIABLES IN COMMON/PNTRS/

Name	Contents	Where set (entered)
NPCTDT	Number of precincts in the data base.	INIT
NPCTRD	Number of precincts read by the last READ command.	READ
LPCTDT	Pointer to (subscript in CDAT of) data read by last READ command.	READ
LNMLST(1)	Pointer to list of day names in current command qualifier (stored one character to a word, eight characters to a name).	GTDSPC

Table 2--continued

Name	Contents	Where set (entered)
LNMLST(2)	Pointer to list of tour names in current command qualifier.	GTDSPC
LNMLST(3)	Pointer to list of division names in current command qualifier.	GTDSPC
LNMLST(4)	Pointer to list of precinct names in current command qualifier.	GTDSPC
NNAMES(1)	Number of day names in current command qualifier.	GTDSPC
NNAMES(2)	Number of tour names in current command qualifier.	GTDSPC
NNAMES(3)	Number of division names in current command qualifier.	GTDSPC
NNAMES(4)	Number of precinct names in current command qualifier.	GTDSPC
NDAYDT	Number of days of data in the data base for each precinct.	INIT
LDAYNM	Pointer to table of names of all days in the data base (8*NDAYDT words). These are in the same order as the day data for each precinct in the data base.	INIT
LDYRFL	Pointer to table of day "read" flags (NDAYDT words). Each entry corresponds to one day in the data base. An entry value of zero indicates that no data are to be read for that day. A nonzero value indicates that data are to be read. If the value is nonzero, then it is the ordinal position of that day among days read. If there are three days' data for each precinct in the data base, and the user selects the first and third in a READ command, then the entries in this table will be 1, 0, 2.	INIT(READ)
NDAYRD	Number of days of data selected in the last READ command qualifier.	READ

Table 2--continued

Name	Contents	Where set (entered)
LDYWFL	Pointer to table of day "work" flags (NDAYRD words). Each entry corresponds to one day for which data have been read. An entry value of zero indicates that the current command will not operate on data for that day. A nonzero value indicates that the day is to be included in the command scope. If the entry is nonzero, then it is the ordinal position of the selected day among all days in the data base. Continuing the above example, if the user selects the second of the days read in a command, then the entries in this table would be 0, 3.	READ(SETWFL)
NTRDT	Number of tours in the data base for each day.	INIT
LTRTB(1)	Pointer to table of blocks (NTRDT words). Each entry corresponds to a tour, the order being the same as in the data base. Entry values are the ordinal position among blocks of the first block in a tour.	INIT
LTRTB(2)	The same as LTRTB(1), except gives the position of the second block for each tour. A zero-valued entry indicates that there is no second block for the tour.	INIT
LTRST	Pointer to starting hours of tours (NTRDT words). Each entry corresponds to a tour. The value of each entry is the starting hour (1-24) of that tour.	INIT
LTREND	The same as LTRST, but ending hours.	INIT
LTRRFL	Pointer to table of tour "read" flags. This is the same as LDYRFL, but for tours.	INIT(READ)
LTRNM	Pointer to table of tour names (8*NTRDT words). These are in the same order as the tour data for each day in the data base.	INIT

Table 2--continued

Name	Contents	Where set (entered)
NTRRD	Number of tours selected in the last READ command qualifier.	READ
LTRWFL	Pointer to table of tour "work" flags (NTRRD words). This is the same as LDYWFL, but for tours.	READ(SETWFL)
NBLDT	Number of blocks for each day in the data base.	INIT
LBLKTB(1)	Pointer to table of starting hours for each block (NBLDT words).	INIT
LBLKTB(2)	Pointer to table of ending hours for each block (NBLDT words).	INIT
LBLRFL	Pointer to table of block "read" flags (NBLDT words). This is the same as LTRRFL, but for blocks.	INIT(READ)
NBLRD	Number of blocks read by a READ command for each day (function of the number of tours selected).	READ
LBLWFL	Not used.	
NDIVDT	Number of divisions into which precincts are aggregated.	INIT
NDIVRD	Number of divisions selected by a READ command.	READ
LDIVNM	Pointer to list of names of divisions selected by a READ command (8*NDIVDT words). Includes those selected by a request for all precincts.	INIT(READ)
LDIVFL	Pointer to list of flags that select divisions for current command (not READ) (NDIVRD words). Each entry corresponds to a division name in LDIVNM. A nonzero entry value indicates that the division was selected; a zero entry indicates that it was not.	READ(SETWFL)

Table 2--continued

Name	Contents	Where set (entered)
IOVRLY	A flag that indicates whether or not there are overlay tours in the data base. A value of 1 indicates that the last tour of each day in the data base is an overlay tour; a value of 0 indicates that there are no overlay tours.	INIT
IOVTR(1)	The position of the first overlaid tour among the tours specified in a READ command. A value of n indicates that the nth tour of the tours read for each day is the tour during which the overlay tour starts.	READ
IOVTR(2)	The position of the second overlaid tour among the tours specified in a READ command. A value of m indicates that the mth tour of the tours read for each day is the tour during which the overlay tour ends (of course, IOVTR(2)=IOVTR(1)+1).	READ

DATA STORAGE

PCAM stores the data read by a READ command in array CDAT in a structure parallel to the way the data are stored in DATABASE (see Chapter II and in particular Fig. 1). For each precinct, a constant-size area of storage contains certain data that describe the precinct as a whole; then a variable-size area contains data for each day read for the precinct. Within the area allocated for each day, a constant-size area contains data for the day as a whole and two variable-size areas contain data for each tour and each block (the size of each area depends on the number of tours read for a day).

Each element of precinct, day, tour, and block data is referenced by a pointer which is the subscript within CDAT of the data for the precinct, day, tour, or block, plus an *offset* that corresponds to the type of data being referenced. For example, the word containing the area of a precinct is referenced in the program by CDAT(LPCT+ARPOFF), where LPCT is the previously determined pointer to the data for the precinct and ARPOFF is the relative position within precinct data (for all precincts) of the word containing the precinct's area. Tables 3, 4, 5, and 6 give the layout of the constant data for precincts, days, tours, and blocks. LPCT, LDAY, LTOUR, and LBLK are pointers to particular precincts, days, tours, and blocks, respectively.

Table 3

DESCRIPTION OF PRECINCT DATA

Data Item	Mode	Reference	Offset Value
Name (8 words)	Character	(LPCT+NMPOFF)	0
Division number (relative position in LDIVNM of division name: 0 for none)	Integer	(LPCT+DVPOFF)	8
Area (square miles)	Real	(LPCT+ARPOFF)	9
Total street length (miles)	Real	(LPCT+SMPOFF)	10
B1 (unavailability parameter)	Real	(LPCT+B1POFF)	11
B2 (unavailability parameter)	Real	(LPCT+B2POFF)	12
Data for days		(LPCT+DYPOFF)	13

Table 4

DESCRIPTION OF DAY DATA

Data Item	Mode	Reference	Offset Value
Call rate parameter	Real	(LDAY+CPDOFF)	0
Service time parameter	Real	(LDAY+SPDOFF)	1
Overlay indicator for day	Integer	(LDAY+OVDOFF)	2
Hourly call rates (24 words)	Real	(LDAY+CRDOFF)	3
Hourly service times (24 words)	Real	(LDAY+STDOFF)	27
Data for tours	(a)	(LDAY+TRDOFF)	51
Data for blocks	(a)	(LDAY+BLDOFF)	Depends on number of tours

[a]See Table 5.

Table 5

DESCRIPTION OF TOUR DATA

Data Item	Mode	Reference	Offset Value
Difference in objective function value per car-hour if one car is added to tour	Real	(LTOUR+QDTOFF)	0
Difference in objective function value per car-hour if a car is removed from an overlay tour and one car is added to each of the tours that it overlays (overlay tours only)	Real	(LTOUR+QXTOFF)	1
Number of calls during tour	Real	(LTOUR+CRTOFF)	2
Objective function value with current allocation	Real	(LTOUR+QOTOFF)	3
Objective function value with one more car	Real	(LTOUR+QNTOFF)	4
Number of most limiting constraint	Integer	(LTOUR+CTTOFF)	5
Tour type (1=ignore, 2=standard, 3=first in overlay, 4=second in overlay, 5=overlay tour)	Integer	(LTOUR+TYTOFF)	6
Actual cars assigned to start tour	Real	(LTOUR+ACTOFF)	7
Response speed (mph)	Real	(LTOUR+RVTOFF)	8
Patrol speed (mph)	Real	(LTOUR+PVTOFF)	9
Fraction of priority 1 calls	Real	(LTOUR+HFTOFF)	10
Fraction of priority 2 calls	Real	(LTOUR+MFTOFF)	11
Fraction of priority 3 calls	Real	(LTOUR+LFTOFF)	12

Table 6

DESCRIPTION OF BLOCK DATA

Data Item	Mode	Reference	Offset Value
Effective cars (including overlay effects)	Real	(LBLK+EFBOFF)	0
Actual cars on duty (including overlays)	Real	(LBLK+ACBOFF)	1
Average workload during hours of block (hours of servicing calls per hour)	Real	(LBLK+AWBOFF)	2
Total calls during block	Real	(LBLK+CRBOFF)	3
Maximum workload over all hours of block	Real	(LBLK+RMBOFF)	4
Number of suppressible crimes during block	Real	(LBLK+OCBOFF)	5
Number of most limiting constraint	Real	(LBLK+CTBOFF)	6
Objective function value with current allocation	Real	(LBLK+QOBOFF)	7
Objective function value with one additional car	Real	(LBLK+QNBOFF)	8

The offsets for all data items described above are contained in COMMON/OFFSET/. Other variables in COMMON/OFFSET/ give the storage requirements for precincts, days, tours, and blocks. These are described in Table 7.

Table 7

OTHER CONTENTS OF COMMON/OFFSET/

Variable	Contents	Value
NWDBL	Number of words required for a block	9
NWDTR	Number of words required for a tour	13
NWDDY	Number of words required for a day	51+NWDTR*NTRRD +NWDBL*NBLRD
NPRIO	Number of priority classes	3
NWDPCT	Number of words required for a precinct	13+NWDDY*NDAYRD

Chapter V

LISTING* AND DESCRIPTION OF THE PCAM FORTRAN PROGRAM

The discussions in this chapter assume the reader's familiarity with the contents of Chapter IV (Internal Data Structures) and the User's Manual. Refer to Appendix C for a cross-reference listing of program segments and common blocks. An alphabetized list of their names was given immediately following the table of contents.

MAIN PROGRAM

The MAIN program primarily controls the execution of the subroutines that implement the various PCAM commands. It operates in a continuous loop, determining which subroutine to call by examining successive command identifiers, until an END command is encountered.

Execution begins with a call to subroutine INIT to initialize permanent tables, etc. Then, if operating in interactive mode, a message prompting for the user's next command is written. Subroutine SCAN is called to obtain the command identifier. If the identifier is valid, the appropriate subroutine is called to complete command interpretation and execution. When command execution is completed, the MAIN program proceeds to the next command.

The listing provided here is for a batch program. To convert to an interactive program, three clearly indicated changes must be made:

1. Remove the comment 'C' on cards 25 and 26. This will cause the program to prompt for the next command. The user may also have to change the dollar sign on line 26, which serves the purpose of leaving the interactive terminal's print head in place.
2. Remove lines 43 and 44. These cause the printer to eject to a new page after displaying tables of output.
3. Remove line 51. This causes page ejection after listing data.

---

*In order to meet space constraints, the listing has been photographically reduced by 5 percent.

```
      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      INTEGER TYPE,VAL
      DIMENSION VAL(2)
      BOT=1
      TOP=NWORDS+1
      CALL INIT
      LGETT=TOP
 ****  NEXT 2 LINES NEEDED FOR INTERACTIVE MODE  ****
       WRITE(SYSOUT,1)
       FORMAT(/' COMMAND? '$)
      TYPE=SEND
      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. CMD) GO TO 20
      WRITE(SYSOUT,2)
      FORMAT(/' ***INVALID COMMAND - REENTER.')
      TOP=LGETT
      GO TO 10
      ICMD=VAL(1)-TYPOFF(CMD)
      GO TO (100,200,300,400,500,600,700,800,900),ICMD

      CALL ADDALC(2)
      GO TO 10
      CALL ADDALC(0)
      GO TO 10
      CALL DISP
 **** REMOVE NEXT TWO LINES FOR INTERACTIVE MODE ****
      WRITE(SYSOUT,3)
      FORMAT(1H1)
      GO TO 10
      WRITE(SYSOUT,4) MAXBOT
      FORMAT(/' MAXIMUM SIZE OF CURRENT-DATA WAS ',I5,' WORDS')
      STOP
      CALL LIST
 **** REMOVE NEXT LINE FOR INTERACTIVE MODE ****
      WRITE(SYSOUT,3)
      GO TO 10
```

```
00      CALL MEET
        GO TO 10
00       CALL READ
        GO TO 10
00       CALL SET
         GO TO 10
00      CALL WRITE
        GO TO 10
        END
```

Subroutine INIT

This subroutine performs initialization tasks for PCAM. It is called only once (from MAIN).

The initialization tasks consist primarily of reading control information from the data base and allocating storage for tables whose dimensions will not change during program execution. In addition, starting hours for blocks and starting and ending hours for tours are computed.

The array KEYWD, which contains all command language keywords, is initialized by writing literals on file LIT. (This is a variable name containing a FORTRAN unit number. See Chapter I.) File LIT is read back under A format. This procedure eliminates the neea for a DATA statement, which some compilers restrict to initializing only one array element per entry, and simplifies modification of keywords. In addition, the program determines the relative position among keywords (in KEYWD) of the first keyword of each "syntactic type."* The relative positions of the first keywords of each type are saved in array TYPOFF.

---

*See description of Subroutine SCAN for syntactic types.

```
      SUBROUTINE  INIT

   SUBROUTINE TO INITIALIZE PERMANENT TABLES, ETC.

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      WRITE(SYSOUT,6)


        WRITE KEYWORD SCRATCH FILE

      WRITE(LIT,11)
      WRITE(LIT,12)
      WRITE(LIT,13)
      REWIND LIT
      READ(LIT,10) ((KEYWD(I,J),I=1,8),J=1,NKYWD)


        READ CONTROL CARD FROM DATA BASE

      REWIND IFILE
      READ(IFILE,1) DCLSNM,PCLSNM,TOURNM,NDIVDT,NPCTDT,NDAYDT,NBLDT,
     1NTRDT,IOVRLY


        ALLOCATE STORAGE SPACE

      N=8*NDAYDT
      CALL GETBOT(N,LDAYNM)
      CALL GETBOT(NDAYDT,LDYRFL)
      NL=LDAYNM+N-1
      READ(IFILE,2) (CDAT(I),I=LDAYNM,NL)

      CALL GETBOT(NBLDT,LBLKTB(1))
      CALL GETBOT(NBLDT,LBLKTB(2))
      CALL GETBOT(NBLDT,LBLRFL)
      K=LBLKTB(2)-1
      READ(IFILE,3) (ICDAT(K+I),I=1,NBLDT)
      ICDAT(LBLKTB(1))=1
```

```
      DO 100 I=2,NBLDT
0     ICDAT(LBLKTB(1)+I-1)=ICDAT(LBLKTB(2)+I-2)+1

      N=8*NTRDT
      CALL GETBOT(N,LTRNM)
      CALL GETBOT(NTRDT,LTRTB(1))
      CALL GETBOT(NTRDT,LTRTB(2))
      CALL GETBOT(NTRDT,LTRRFL)
      L2=LTRNM-1
      DO 120 I=1,NTRDT
      L1=L2+1
      L2=L1+7
      J=LTRTB(1)+I-1
      K=LTRTB(2)+I-1
0     READ(IFILE,4) (CDAT(L3),L3=L1,L2),ICDAT(J),ICDAT(K)
      CALL GETBOT(NTRDT,LTRST)
      CALL GETBOT(NTRDT,LTREND)


        CALCULATE STARTING AND ENDING HOURS

      DO 130 ITOUR=1,NTRDT
      IBLK=ICDAT(LTRTB(1)+ITOUR-1)
      ISTART=ICDAT(LBLKTB(1)+IBLK-1)
      ICDAT(LTRST+ITOUR-1)=ISTART
      IEND=ICDAT(LBLKTB(2)+IBLK-1)
      ICDAT(LTREND+ITOUR-1)=IEND
      IBLK=ICDAT(LTRTB(2)+ITOUR-1)
      IF(IBLK .EQ. 0) GO TO 130
      IEND=ICDAT(LBLKTB(2)+IBLK-1)
      ICDAT(LTREND+ITOUR-1)=IEND
)     CONTINUE

      CALL GETBOT(8*NDIVDT,LDIVNM)
      RDBOT=BOT

           ASSIGN TYPES TO KEYWORDS

      I=0
      DO 150 ITYPE=1,NTYPES
      N=TYPOFF(ITYPE)
      TYPOFF(ITYPE)=I
      DO 140 J=1,N
      I=I+1
      IF(I .LE. NKYWD) GO TO 140
      WRITE(SYSOUT,5) ITYPE
      STOP
)     WDTYPE(I)=ITYPE
)     CONTINUE
      REWIND IFILE
      RETURN

      FORMAT(8A1,2X,8A1,2X,8A1,1X,I2,1X,I3,1X,I3,1X,I2,1X,I2,1X,I1)
      FORMAT(80A1)
      FORMAT(24(I2,1X))
      FORMAT(8A1,1X,I2,1X,I2)
      FORMAT(///26X,'NEW YORK CITY - RAND INSTITUTE'/27X,
     C 'PATROL CAR ALLOCATION MODEL'///)
```

```
      FORMAT(
     C'DAY     ',
     C'TOUR    ',
     C'DIVISION',
     C'PRECINCT',
     C'P       ',
     C'C       ',
     C'T       ',
     C'F       ',
     C'ADD     ',
     C'ALOC    ')
      FORMAT(
     C'DISP    ',
     C'END     ',
     C'LIST    ',
     C'MEET    ',
     C'READ    ',
     C'SET     ',
     C'WRITE   ',
     C'FOR     ',
     C'CAR     ',
     C'CARS    ')
      FORMAT(
     C'TO      ',
     C'BY      ',
     C'DATA    ',
     C'HOUR    ',
     C'HOURS   ',
     C'ON      ')
    FORMAT(80A1)
      FORMAT(/' ***INTERNAL ERROR: TOO MANY KEYWORDS AT TYPE ',
     C I2,' - EXECUTION TERMINATED')
      END
```

Subroutine GETBOT

Subroutine GETBOT (get bottom) allocates storage from the "bottom" of array CDAT.* The input parameter N specifies the number of words of storage that are needed. The variables TOP and BOT in COMMON/STORE/ contain the subscripts of the highest free word plus one and the lowest free word in CDAT, respectively. If N words of storage are available, the output parameter L is set to the subscript of the first word allocated, BOT is updated, and the storage obtained is set to zeros. If N words of storage are not available, execution is terminated.

```
      SUBROUTINE GETBOT(N,L)
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
C
C        ALLOCATE STORAGE
C
      L=BOT
      BOT=L+N
C
C        ERROR CONDITION  .  INSUFFICIENT SPACE
      IF(BOT .LT. TOP) GO TO 10
      WRITE(SYSOUT,1)
1     FORMAT(/' *** INSUFFICIENT STORAGE FOR TABLES OR DATA - ',
     1'EXECUTION TERMINATED')
      STOP
C
C         SET DATA TO ZERO
C
10    K=BOT-1
      DO 20 I=L,K
20    ICDAT(I)=0
      IF(BOT .GT. MAXBOT) MAXBOT=BOT
      RETURN
      END
```

*See Chapter IV for a description of the storage management system.

Subroutine SCAN

This subroutine scans the user's command input for the next syntactic element (e.g., command identifier, name list, number list, etc.) Its two parameters STYPE and SVAL are set to the type and value, respectively, of the element obtained. SVAL is a two-word array; the meaning of each word depends on the type of the syntactic element, as shown in Table 8.

Table 8

SYNTACTIC TYPES RETURNED FROM SCAN

Type Identifier (STYPE)	Description	Form of SVAL
SEND	End of command encountered	--
CMD	Command identifier	(Position of identifier in KEYWD table, --)
NUMLST	Number list	(Number of elements in list, pointer to list)
NAMLST	Name list	(Number of elements in list, pointer to list)
FSPEC	Function identifier (objective function, constraint, data type, table)	(Position of identifier in KEYWD table, --)
DSPEC	Data type specification (DAY, TOUR, PRECINCT, DIVISION)	(Position of identifier in KEYWD table, --)
ERR	Invalid element	--

SCAN calls GETTKN to get the next lexical element (number, word, paren, etc.) from the command text. If STYPE indicates that the last element type was "end of command," then SCAN instructs GETTKN to start reading a new command by setting TYPE to indicate "end of command." (See description of Subroutine GETTKN; the parameter is called LTYPE in SCAN.)

The elements of name lists are stored in the "top" of array CDAT in storage allocated by calls to GETTOP. Names are stored eight characters to a name, one character to a word. Elements of name lists occupy contiguous words of storage in the order opposite to that in which they were entered.

Numbers are stored in word pairs. The first word of a pair contains the integer representation and the second word the floating point representation of the number. Word pairs in a number list occupy contiguous words of storage in the same order as that in which they were entered.

```
      SUBROUTINE  SCAN(STYPE,SVAL)

 SCANS USER COMMAND INPUT FOR NEXT LEXICAL ELEMENT

      COMMON/SYSTEM/SYSIN,SYS3UT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPCFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/LCODES/LEND,WORD,NJM,LP,RP
      INTEGER WORD,RP

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLSF,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      INTEGER STYPE,SVAL
      DIMENSION SVAL(2),LVAL(8)

      LGETT=TOP
      IF(STYPE .EQ. SEND) LTYPE = LEND

           GET NEXT LEXICAL ELEMENT FROM COMMAND

      CALL GETTKN(LTYPE,LVAL)
      GO TO (100,200,300,400,405),LTYPE

           END OF COMMAND REACHED

0     STYPE=SEND
      RETURN

           BEGINNING OF A WORD ENCOUNTERED

0     I=LKP8(LVAL,KEYWD,NKYWD)
      IF(I .EQ. 0) GO TO 220
      STYPE=WDTYPE(I)
      IF(STYPE .EQ. DUM) GO TO 10
      SVAL(1)=I
      RETURN

0     STYPE=NAMLST
      SVAL(1)=1
      CALL GETTOP(8,I)
      CALL MOVE(LVAL,CDAT(I),8)
      SVAL(2)=I
      RETURN

           NEXT LEXICAL ELEMENT IS A NUMBER
```

```
C
300     STYPE=NUMLST
        SVAL(1)=1
        CALL GETTOP(2,I)
        ICDAT(I)=LVAL(1)
        ICDAT(I+1)=LVAL(2)
        SVAL(2)=I
        RETURN
C
C            NEXT LEXICAL ELEMENT IS A LEFT PARENTHESIS
C
400     CALL GETTKN(LTYPE,LVAL)
        IF(LTYPE .EQ. NUM)GO TO 450
        IF(LTYPE .EQ. WORD .OR. LTYPE .EQ. RP) GO TO 410
C
C            ERROR ENCOUNTERED IN COMMAND FORMAT
C
405     WRITE(SYSOUT,1)
1       FORMAT(/' *** INVALID LIST FORMAT - REENTER.')
        STYPE=ERR
        RETURN
C
C            NAMELIST ENCOUNTERED
C
410     N=0
        STYPE=NAMLST
415     IF(LTYPE .EQ. RP) GO TO 430
        IF(LTYPE .EQ. WORD) GO TO 420
        WRITE(SYSOUT,2)
2       FORMAT(/' *** INVALID NAME LIST ELEMENT - REENTER.')
        TOP=LGETT
        STYPE=ERR
        RETURN
420     N=N+1
        CALL GETTOP(8,LOC)
        CALL MOVE(LVAL,CDAT(LOC),8)
        CALL GETTKN(LTYPE,LVAL)
        GO TO 415
430     SVAL(1)=N
        SVAL(2)=LOC
        RETURN
C
C            NUMBERLIST ENCOUNTERED
C
450     N=0
        STYPE=NUMLST
460     IF(LTYPE .EQ. RP) GO TO 480
        IF(LTYPE .EQ. NUM) GO TO 470
        WRITE(SYSOUT,3)
3       FORMAT(/' *** INVALID NUMBER LIST FORMAT - REENTER.')
        STYPE=ERR
        TOP=LGETT
        RETURN
C
C            STORE NUMBERS
C
470     N=N+1
        CALL GETTOP(2,LOC)
```

```
      ICDAT(LOC)=LVAL(1)
      ICDAT(LOC+1)=LVAL(2)
      CALL GETTKN(LTYPE,LVAL)
      GO TO 460
0     SVAL(1)=N
      SVAL(2)=LOC
      NSW=N/2
      IF(NSW .LT. 1) RETURN
      J=LOC
      K=LOC+(N-1)*2
      DO 490 I=1,NSW
      IT1=ICDAT(J)
      IT2=ICDAT(J+1)
      ICDAT(J)=ICDAT(K)
      ICDAT(J+1)=ICDAT(K+1)
      ICDAT(K)=IT1
      ICDAT(K+1)=IT2
      J=J+2
      K=K-2
0     CONTINUE
      RETURN
      END
```

Subroutine GETTKN

Subroutine GETTKN (get token) obtains the next lexical element in the user's command input. Its two parameters TYPE and VAL are set to the class and value, respectively, of the element obtained. VAL is an eight-word array; its use depends on the type of element scanned, as shown in Table 9.

Table 9

LEXICAL TYPES RETURNED FROM GETTKN

Type Identifier (TYPE)	Description	Form of VAL
LEND	End of command	--
WORD	Character string of up to eight characters, starting with a letter	Each computer word contains one character as if it were read in A1 format. WORDs are left adjusted in VAL and padded with blanks
NUM	Number	The first word of VAL contains the integer representation of the number, and the second word contains its floating point representation.
LP	Left parenthesis	--
RP	Right parenthesis	--

At entry, if TYPE indicates that an "end of command" was the last element scanned, a new record is read from the input file and scanned from its start. A new record is also read if an ampersand is encountered while scanning for the next element.

Function LKP1 is invoked to determine the type and value of each character scanned.

Two lines of this subroutine, 416 and 417, must be removed to obtain an interactive program. These print out the command that has just been read, which is unnecessary and annoying if the user has typed the command into his terminal.

```
      SUBROUTINE GETTKN(TYPE,VALUE)

 GETS NEXT LEXICAL ELEMENT IN USER'S COMMAND INPUT

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/LCODES/LEND,WORD,NUM,LP,RP
      INTEGER WORD,RP

      INTEGER TYPE,VALUE,CHAR,CARD,COL,ALPHNM,DIGIT,CHARBL,CHARLP,
     1CHARRP,CHARAM,CHARST

      DIMENSION VALUE(8),CARD(81),ALPHNM(38),LETTER(26),DIGIT(12)

      EQUIVALENCE (LETTER,ALPHNM),(DIGIT,ALPHNM(27)),
     1(IXVAL,XVAL)

      DATA ALPHNM(1)/1HA/,ALPHNM(2)/1HB/,ALPHNM(3)/1HC/,
     1ALPHNM(4)/1HD/,ALPHNM(5)/1HE/,ALPHNM(6)/1HF/,ALPHNM(7)/1HG/,
     2ALPHNM(8)/1HH/,ALPHNM(9)/1HI/,ALPHNM(10)/1HJ/,ALPHNM(11)/1HK/,
     3ALPHNM(12)/1HL/,ALPHNM(13)/1HM/,ALPHNM(14)/1HN/,ALPHNM(15)/1HO/,
     4ALPHNM(16)/1HP/,ALPHNM(17)/1HQ/,ALPHNM(18)/1HR/,ALPHNM(19)/1HS/,
     5ALPHNM(20)/1HT/,ALPHNM(21)/1HU/,ALPHNM(22)/1HV/,ALPHNM(23)/1HW/,
     6ALPHNM(24)/1HX/,ALPHNM(25)/1HY/,ALPHNM(26)/1HZ/

      DATA DIGIT(1)/1H0/,DIGIT(2)/1H1/,DIGIT(3)/1H2/,DIGIT(4)/1H3/,
     1DIGIT(5)/1H4/,DIGIT(6)/1H5/,DIGIT(7)/1H6/,DIGIT(8)/1H7/,
     2DIGIT(9)/1H8/,DIGIT(10)/1H9/,DIGIT(11)/1H./,CHARLP/1H(/,
     3CHARRP/1H)/,CHARAM/1H&/,CHARBL/1H /,DIGIT(12)/1H-/,CHARST/1H*/

      DATA CARD(81)/1H /


      IF(TYPE .NE. LEND .AND. COL .LT. 81) GO TO 120
)     READ(SYSIN,2) (CARD(I),I=1,80)
      FORMAT(80A1)
    **** NEXT TWO LINES NOT NEEDED FOR INTERACTIVE MODE ****
      WRITE (SYSOUT,3) (CARD(I),I=1,80)
      FORMAT(1H ,80A1)
      COL=0
5     COL=COL+1
      IF(COL .LT. 81) GO TO 120
      TYPE=LEND
      RETURN

            FIND NEXT NON-DELIMITER

)     CHAR=CARD(COL)
      IF(CHAR .EQ. CHARBL) GO TO 115
      I=LKP1(CHAR,ALPHNM,38)
      IF(I .NE. 0) GO TO 200
      IF(CHAR .EQ. CHARAM) GO TO 100
      IF(CHAR .EQ. CHARRP) GO TO 150
      IF(CHAR .EQ. CHARLP) GO TO 160
      IF(CHAR .NE. CHARST) GO TO 115
```

```
            FOUND '*'

        TYPE=WORD
        DO 125 J=2,8
5       VALUE(J)=CHARBL
        VALUE(1)=CHAR
        COL=COL+1
        RETURN

            FOUND RIGHT PAREN

0       TYPE=RP
        COL=COL+1
        RETURN

            FOUND LEFT PAREN

0       TYPE=LP
        COL=COL+1
        RETURN

            FOUND WORD

0       IF(I .GT. 26) GO TO 300
        TYPE=WORD
        DO 210 J=2,8
0       VALUE(J)=CHARBL
        J=0
)       J=J+1
        IF (J .GT. 8) GO TO 230
        VALUE(J)=CHAR
)       COL=COL+1
        CHAR=CARD(COL)
        IF(LKP1(CHAR,ALPHNM,38) .NE. 0) GO TO 220
        RETURN

            FOUND NUM OR '-*'

)       TYPE=NUM
        I=I-26
        IVAL=0
        ISIGN=1
        IF(I .NE. 12) GO TO 310
        ISIGN=-1
        IF(CARD(COL+1) .NE. CHARST) GO TO 320
        COL=COL+2
        TYPE=WORD
        DO 305 J=3,8
5       VALUE(J)=CHARBL
        VALUE(1)=CHAR
        VALUE(2)=CHARST
        RETURN

            GET INTEGER VALUE

)       IF(I .EQ. 11) GO TO 350
        IVAL=IVAL*10+I-1
)       COL=COL+1
```

```
        CHAR=CARD(COL)
        I=LKP1(CHAR,DIGIT,11)
        IF (I .NE. 0) GO TO 310
        IVAL=IVAL*ISIGN
        VALUE(1)=IVAL
        XVAL=IVAL
        VALUE(2)=IXVAL
        RETURN

             GET REAL VALUE (WITH FRACTION, IF PRESENT)

0       XVAL=IVAL
        POWER=1.
)       COL=COL+1
        CHAR=CARD(COL)
        I=LKP1(CHAR,DIGIT,10)
        IF(I .EQ. 0) GO TO 370
        POWER=POWER*10.
        XVAL=XVAL+(I-1)/POWER
        GO TO 360
)       XVAL=XVAL*ISIGN
        VALUE(2)=IXVAL
        VALUE(1)=IVAL*ISIGN
        RETURN
        END
```

Function LKP1

Function LKP1 determines the relative position of a one-word argument in a list. Parameter LIST is the list to be searched for IARG. N is the number of entries in LIST. The function value returned is the relative position of IARG in LIST or zero if IARG is not found in LIST.

```
      FUNCTION LKP1(IARG,LIST,N)
C
C  DETERMINES WHETHER IARG IS IN LIST, AND, IF SO, WHERE
C
      DIMENSION LIST(N)
      LKP1=0
      IF(N .EQ. 0) RETURN
      DO 10 I=1,N
      IF(IARG .EQ. LIST(I)) GO TO 20
10    CONTINUE
      RETURN
20    LKP1=I
      RETURN
      END
```

Function LKP8

Function LKP8 is called to determine the relative position of an eight-character name in a list of names. ARG is the name to be found, LIST is the list to be searched, and N is the number of entries in LIST.

The value returned is the position of ARG in LIST, or zero if ARG is not in LIST. Note that this function is frequently invoked with both ARG and LIST as parts of one-dimensional arrays.

```
      FUNCTION LKP8(ARG,LIST,N)
C
C  DETERMINES WHETHER ARG IS IN LIST, AND, IF SO, WHERE
C
      REAL LIST
      DIMENSION ARG(8),LIST(8,N)
      L=0
      LKP8=0
      NT=N
10    IF(NT .EQ. 0) RETURN
      L=L+1
      DO 20 I=1,8
      IF(ARG(I) .NE. LIST(I,L)) GO TO 30
20    CONTINUE
      LKP8=L
      RETURN
30    NT=NT-1
      GO TO 10
      END
```

Subroutine MOVE

Subroutine MOVE is called to move N words from array S to array T. S and T frequently represent parts of larger arrays.

```
      SUBROUTINE MOVE(S,T,N)
C
C  MOVES N WORDS FROM ARRAY S TO ARRAY T
C
      DIMENSION S(N),T(N)
      IF(N .LE. 0) RETURN
      DO 10 I=1,N
10    T(I)=S(I)
      RETURN
      END
```

Subroutine GETTOP

This subroutine operates like GETBOT, except that allocated storage is obtained from the top of array DATA and is not initialized when allocated.

```
      SUBROUTINE GETTOP(N,L)
C
C  ALLOCATES STORAGE AT TOP OF DATA ARRAY
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
C
      L=TOP-N
      TOP=L
      IF(TOP .GT. BOT) RETURN
      WRITE(SYSOUT,1)
1     FORMAT(/' *** INSUFFICIENT TEMPORARY STORAGE - EXECUTION',
     1' TERMINATED')
      STOP
      END
```

Subroutine READ

Subroutine READ implements the READ command. Its function is to read selected data from the data base and to make these data available to subsequent commands. See Chapter IV for a description of the organization of the data after they have been read.

READ calls GTDSPC to scan the command qualifier and MRGORD to set the default output order for subsequent DISP commands. Storage is then obtained for the various "work" flags (see Table 2, Chapter IV) and all "read" and "work" flags are initialized. If an overlay tour has been specified in the command qualifier, a check is performed to insure that the overlaid tours have also been specified.

When reading data, precincts are selected on the basis of whether or not their precinct or division name appears in the qualifier. If no precinct or division names are specified, then all precincts are selected. Instead of storing a division name for each precinct read, a division number that refers to an entry in table LDIVNM (see Chapter IV) is used.

Within selected precincts, days are selected on the basis of the values of entries in table LDYRFL (see Chapter IV). For each day, hourly call rates and service times are computed from the corresponding parameters and hourly factors; service times are converted from minutes to hours.

Within days, tours are selected on the basis of the values of entries in table LTRRFL. A "tour type" is determined for each shift on the basis of its relationship to overlays (see Table 2). To facilitate indexing through the data, we have required that the same number of tours be stored for each day read, regardless of whether the day has an overlay tour or not. Therefore, the type of a tour is set to "ignore" when it occupies a position that would be held by an overlay tour, but the data base indicates that there is no overlay tour for the day. The meanings of other type codes should be apparent from Table 2.

Blocks are selected by entries in table LBLRFL. The constraint indicator for each block is set to -1 when it is read to indicate that it has not been in the scope of a prescriptive command (MEET, ADD, or ALOC).

When the data for all tours and blocks of a day have been read, subroutine DERIVE is called to determine the numbers of actual and effective cars on duty in each block and to insure that enough cars are available in each hour of the day to handle the call-for-service workload.

```
      SUBROUTINE READ

UBROUTINE TO READ SELECTED DATA FROM DATA BASE

      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))





      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      INTEGER TYPE,VAL,ORDER,OVRLY
      DIMENSION VAL(2),TPCTNM(8),TDIVNM(8),TPCTDT(20),TDATA(24),
     1ORDER(4)
      EQUIVALENCE (TPCTNM,TPCTDT),(TDIVNM,TPCTDT(9)),
     1(TPCTDT,TDATA),(OVRLY,IOVRLY)

            INITIALIZE

      NPCTRQ=0
      BOT=RDBOT
      LGETB=BOT
      LGETT=TOP
```

```
NDIVRQ=0
NDIVRD=0
NPCTRD=0
NDAYRD=0
NTRRD=0
NBLRD=0

      INTERPRET QUALIFIERS

TYPE=CMD
CALL SCAN(TYPE,VAL)
CALL GTDSPC(TYPE,VAL,ORDER)
IF(TYPE .NE. ERR) GO TO 10
TOP=LGETT
RETURN

      SET DEFAULT OUTPUT OJRDER FOR DISP

CALL MRGORD(ORDER,PORDER,RORDER)

      GET STORAGE FOR WORK FLAGS.
      INITIALIZE FLAGS
             DATA FOR DAYS

IT=1
L=LNMLST(IT)
N=NNAMES(IT)
IF(N .NE. 0) GO TO 30
CALL GETBOT(NDAYDT,LDYWFL)
DO 20 I=1,NDAYDT
ICDAT(LDYWFL+I-1)=I
ICDAT(LDYRFL+I-1)=I
NDAYRD=NDAYDT
GO TO 100
CALL GETBOT(N,LDYWFL)
DO 40 I=1,NDAYDT
ICDAT(LDYRFL+I-1)=0
DO 50 I=1,N
J=LKP8(CDAT(L),CDAT(LDAYNM),NDAYDT)
IF(J .EQ. 0) GO TO 900
ICDAT(LDYRFL+J-1)=1
L=L+8
NDAYRD=N
IDAY=0
DO 60 I=1,NDAYDT
IF(ICDAT(LDYRFL+I-1) .EQ. 0) GO TO 60
IDAY=IDAY+1
ICDAT(LDYRFL+I-1)=IDAY
ICDAT(LDYWFL+IDAY-1)=I
CONTINUE

             DATA FOR TOURS

IT=2
N=NNAMES(IT)
IF(N .NE. 0) GO TO 120
CALL GETBOT(NTRDT,LTRWFL)
DO 110 I=1,NTRDT
```

```
ICDAT(LTRWFL+I-1)=I
ICDAT(LTRRFL+I-1)=I
NTRRD=NTRDT
GO TO 200
L=LNMLST(IT)
DO 130 I=1,NTRDT
ICDAT(LTRRFL+I-1)=0
CALL GETBOT(N,LTRWFL)
DO 140 I=1,N
J=LKP8(CDAT(L),CDAT(LTRNM),NTRDT)
IF(J .EQ. 0) GO TO 900
ICDAT(LTRRFL+J-1)=1
L=L+8
NTRRD=N
ITOUR=0
DO 150 ITYPE=1,NTRDT
IF(ICDAT(LTRRFL+ITYPE-1) .EQ. 0) GO TO 150
ITOUR=ITOUR+1
ICDAT(LTRRFL+ITYPE-1)=ITOUR
ICDAT(LTRWFL+ITOUR-1)=ITYPE
CONTINUE

           DATA FOR DIVISIONS

IT=3
N=NNAMES(IT)
NDIVRQ=N
IF(N .EQ. 0) GO TO 300
IF( N .GT. NDIVDT) GO TO 910
L=LNMLST(IT)
CALL MOVE(CDAT(L),CDAT(LDIVNM),8*N)

IT=4
NPCTRQ=NNAMES(IT)
IF(NPCTRQ .GT. NPCTDT) GO TO 910
LPCTNM=LNMLST(IT)

DO 305 I=1,NBLDT
ICDAT(LBLRFL+I-1)=0
N=NTRDT
IF(OVRLY .NE. 0) N=N-1
DO 315 I=1,N
IF(ICDAT(LTRRFL+I-1) .EQ. 0) GO TO 315
DO 310 J=1,2
K=ICDAT(LTRTB(J)+I-1)
IF(K .EQ. 0) GO TO 315
NBLRD=NBLRD+1
ICDAT(LBLRFL+K-1)=NBLRD
CONTINUE
CONTINUE

     CHECK OVERLAY TOURS

IF(OVRLY .EQ. 0 .OR. ICDAT(LTRRFL+NTRDT-1) .EQ. 0) GO TO 340
IBLK1=ICDAT(LTRTB(1)+NTRDT-1)
IBLK2=ICDAT(LTRTB(2)+NTRDT-1)
I=LKP1(IBLK1,ICDAT(LTRTB(2)),NTRDT)
IF(I .NE. 0) GO TO 325
```

```
      N1 = 1
)     WRITE(SYSOUT,9) N1,TOURNM
      STOP
5     IOVTR(1)=ICDAT(LTRRFL+I-1)
      IF(IOVTR(1) .EQ. 0) GO TO 320
      I=LKP1(IBLK2,ICDAT(LTRTB(1)),NTRDT)
      IF(I .NE. 0) GO TO 335
      N2 = 2
)     WRITE(SYSOUT,9) N2,TOURNM
      STOP
6     IOVTR(2)=ICDAT(LTRRFL+I-1)
      IF(IOVTR(2) .EQ. 0) GO TO 330

)     NWDDY=TRDOFF+NTRRD*NWDTR+NBLRD*NWDBL
      NWDPCT=DYPOFF+NDAYRD*NWDDY
      NDIVRD=NDIVRQ
      BLDOFF=TRDOFF+NTRRD*NWDTR
      REWIND IFILE
      CALL SKIP(IFILE,(1+(NDAYDT-1)/10+1+1+NTRDT))

            READ PRECINCT HEADER RECORD

      DO 450 IPCT=1,NPCTDT
      READ(IFILE,1) TPCTDT
      IF(NPCTRQ+NDIVRQ .NE. 0) GO TO 350
      IDIV=LKP8(TDIVNM,CDAT(LDIVNM),NDIVRD)
      IF(IDIV .NE. 0) GO TO 370
      NDIVRD=NDIVRD+1
      IF(NDIVRD .LE. NDIVDT) GO TO 347
      WRITE(SYSOUT,8) DCLSNM
      STOP
      CALL MOVE(TDIVNM,CDAT(LDIVNM+(NDIVRD-1)*8),8)
      IDIV=NDIVRD
      GO TO 370

      IF(NDIVRQ .EQ. 0) GO TO 355
      IDIV=LKP8(TDIVNM,CDAT(LDIVNM),NDIVRQ)
      IF(IDIV .NE. 0) GO TO 370
      IF(NPCTRQ .EQ. 0) GO TO 360
      J=LKP8(TPCTNM,CDAT(LPCTNM),NPCTRQ)
      IF(J .NE. 0) GO TO 345
      CALL SKIP(IFILE,NDAYDT*(4+NTRDT))
      GO TO 450
      NPCTRD=NPCTRD+1
      CALL GETBOT(DYPOFF,LPCT)
      IF(NPCTRD .EQ. 1) LPCTDT=LPCT
      CALL MOVE(TPCTNM,CDAT(LPCT+NMPOFF),8)
      ICDAT(LPCT+DVPOFF)=IDIV
      CALL MOVE(TPCTDT(17),CDAT(LPCT+ARPOFF),4)

            READ DAY DETAIL RCORDS FOR THIS PRECINCT

      DO 440 IDAY=1,NDAYDT
      IF(ICDAT(LDYRFL+IDAY-1) .NE. 0) GO TO 375
      CALL SKIP(IFILE,(4+NTRDT))
              GO TO 440
      CALL GETBOT(TRDOFF,LDAY)
      LCR=LDAY+CRDOFF-1
```

```
      READ(IFILE,2) CDAT(LDAY+CPDOFF),CDAT(LDAY+SPDOFF),
     C ICDAT(LDAY+OVDOFF),(CDAT(LCR+I),I=1,48)
      CPARM=CDAT(LDAY+CPDOFF)
      SPARM=CDAT(LDAY+SPDOFF)

           CALCULATE CALL RATES AND SERVICE TIMES

      DO 380 I=1,24
      I1=I-1
      CDAT(LDAY+CRDOFF+I1)=CDAT(LDAY+CRDOFF+I1)*CPARM
      CDAT(LDAY+STDOFF+I1)=CDAT(LDAY+STDOFF+I1)*SPARM/60.

           READ SHIFT DETAIL RECORDS
           FOR THIS DAY AND PRECINCT

      DO 400 ITOUR=1,NTRDT
      IF(ICDAT(LTRRFL+ITOUR-1) .NE. 0) GO TO 390
      CALL SKIP(IFILE,1)
      GO TO 400
      CALL GETBOT(NWDTR,LTOUR)
      READ(IFILE,3) (CDAT(LTOUR+ACTOFF+I-1),I=1,5)
      CDAT(LTOUR+QXTOFF)=-1.
      ICDAT(LTOUR+TYTOFF)=2
      IF(OVRLY .EQ. 0 .OR. ITOUR .LT. NTRDT) GO TO 400
      IF(ICDAT(LDAY+OVDOFF) .NE. 0) GO TO 395
      ICDAT(LTOUR+TYTOFF)=1
      GO TO 410
      ICDAT(LTOUR+TYTOFF)=5
      ICDAT(LDAY+TRDOFF+(IOVTR(1)-1)*NWDTR+TYTOFF)=3
      ICDAT(LDAY+TRDOFF+(IOVTR(2)-1)*NWDTR+TYTOFF)=4
      GO TO 410
      CONTINUE

           READ BLOCK DETAIL RECORD
           FOR THIS DAY AND PRECINCT
      READ(IFILE,4) TDATA
      DO 420 I=1,NBLDT
      IF(ICDAT(LBLRFL+I-1) .EQ. 0) GO TO 420
      CALL GETBOT(NWDBL,LBLOCK)
      CDAT(LBLOCK+OCBOFF)=TDATA(I)
      ICDAT(LBLOCK+CTBOFF)=-1
      CONTINUE

           CHECK THAT MINIMUM ALLOCATION IS PRESENT
           AND CALCULATE AVERAGES

      CALL DERIVE(LPCT,LDAY)
      CONTINUE
      CONTINUE
      IF(NPCTRD .GT. 0) GO TO 460
      WRITE(SYSOUT,7) PCLSNM
      TOP=LGETT
      BOT=LGETB
      RETURN
      CALL GETBOT(NDIVRD,LDIVFL)
      TOP=LGETT
      RETURN
      WRITE(SYSOUT,5) (KEYWD(I,IT),I=1,8),(CDAT(L+I-1),I=1,8)
```

```
      GO TO 920
      WRITE(SYSOUT,6) (KEYWD(I,IT),I=1,8)
      TOP=LGETT
      BOT=LGETB
      RETURN
            FORMAT FOR PRECINCT HEADER RECORD
      FORMAT(8A1,1X,8A1,1X,4(1X,F5.0))
            FORMAT FOR DAY DETAIL RECORDS
      FORMAT(2(F5.0,1X),I1/24F3.2/24F3.2)
            FORMAT FOR SHIFT DETAIL RECORD
      FORMAT(13(F5.0,1X))
            FORMAT FOR BLOCK DETAIL RECORD
      FORMAT(24F3.1)
      FORMAT(/' ***',2(1X,8A1),' NOT IN DATA - REENTER')
      FORMAT(/' *** TOO MANY ',8A1,'S SPECIFIED - REENTER')
      FORMAT(/' *** NO ',8A1,' DATA SELECTED - REENTER.')
      FORMAT(/' *** BLOCK ',I1,' FOR OVERLAY ',8A1,' NOT FOUND',
     1' - EXECUTION TERMINATED')
      FORMAT(/' *** DATA BASE ERROR: MORE UNIQUE ',8A1,' NAMES THAN',
     C 'DECLARED - EXECUTION TERMINATED')
      END
```

Subroutine GTDSPC

Subroutine GTDSPC (get data specification) is called to scan command qualifiers. It obtains up to four lists of names which are the user's specifications for days, tours, divisions, and precincts. The lists are stored in array CDAT. List pointers are stored in array LNMLST and list lengths are stored in array NNAMES (see Chapter IV).

GTDSPC parameters TYPE and VAL are passed to subroutine SCAN, which is called to obtain syntactic elements from the input stream. Thus, GTDSPC's calling program can determine what syntactic element followed the qualifier in the input stream. At entry, GTDSPC assumes that TYPE and VAL have been set by a previous call to SCAN so that they describe the first element of the qualifier, or the next input element if the qualifier is null.

GTDSPC also returns a three-element array (ORDER) that specifies the order of the phrase types in the qualifier (which in turn determines the DISP command default output order). The elements of ORDER correspond to phrases in the qualifier, e.g., ORDER(1) refers to the first phrase, ORDER(2) to the second phrase, ORDER(3) to the third phrase. The values of the elements of ORDER indicate the type of phrase in each position as follows: 1 = DAY phrase; 2 = TOUR phrase; 3 = DIVISION or PRECINCT phrase (the numbers are derived from the order of the keywords in table KEYWD; DIVISION and PRECINCT are considered equivalent in this context, and the second one entered is ignored).

```
      SUBROUTINE GTDSPC(TYPE,VAL,ORDER)

;ETS DATA SPECIFICATION BY SCANNING QUALIFIERS

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      INTEGER TYPE,VAL,ORDER
      DIMENSION VAL(2),ORDER(3)
      LGETT=TOP
      IORDER=0
      DO 5 I=1,3
      ORDER(I)=0
      NNAMES(I)=0
      NNAMES(4)=0
      GO TO 12

            GET PHRASE TYPE

      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. DSPEC) GO TO 15
      IF(TYPE .EQ. ERR .OR. TYPE .EQ. FSPEC .OR. TYPE .EQ. SEND)
     1   RETURN
      WRITE(SYSOUT,2)
      FORMAT(/' *** INVALID QUALIFIER - REENTER')
      TYPE=ERR
      RETURN
      KEYVAL=VAL(1)
      IT=KEYVAL-TYPOFF(DSPEC)

            GET NAME LIST

      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. NAMLST) GO TO 20
      WRITE(SYSOUT,1)(KEYWD(I,KEYVAL),I=1,8)
      FORMAT(/' *** INVALID ',8A1,' SPECIFICATION - REENTER.')
      TYPE=ERR
      TOP=LGETT
      RETURN

 TERMINE OOUTPUT ORDER SPECIFIED BY THIS COMMAND
```

```
LNMLST(IT)=VAL(2)
NNAMES(IT)=VAL(1)
IORDER=IORDER+1
IF(IORDER .GT. 3) GO TO 10
IF(IT .EQ. 4) IT=3
IF(LKP1(IT,ORDER,3) .NE. 0) GO TO 10
ORDER(IORDER)=IT
GO TO 10
END
```

Subroutine MRGORD

Subroutine MRGORD (merge order) is called to set a new default output order from the qualifier of a READ or DISP command. Its arguments are arrays which fit the description of the ORDER parameter of subroutine GTDSPC. NEWORD represents the ordering of qualifier phrases in the last qualifier scanned. OLDORD represents the previously existing ordering of output phrases. OUTORD represents an ordering of output phrases resulting from merging the "old" ordering with the "new" ordering.

Subroutine MOVE is called to move the contents of OLDORD to a temporary storage (TMPORD) where elements can be "erased" without affecting the original values in OLDORD. Elements of NEWORD are moved to OUTORD in their current order and the phrase types moved are erased from TMPORD. Any elements of OUTORD left unfilled by this process are filled by moving elements from TMPORD in the order in which they occur. Thus, a new DISP command output order is established.

```
      SUBROUTINE MRGORD(NEWORD,OLDORD,OUTORD)
C
C  SETS OUTPUT ORDER FOR DISP COMMAND.   MERGES NEW INFORMATION
C  INTO OLD TO ESTABLISH OUTPUT ORDER
C
      INTEGER OLDORD,OUTORD,TMPORD
      DIMENSION NEWORD(3),OLDORD(3),OUTORD(3),TMPORD(3)
C
      CALL MOVE(OLDORD,TMPORD,3)
      DO 30 IORD=1,3
      IF(NEWORD(IORD) .EQ. 0) GO TO 10
      OUTORD(IORD)=NEWORD(IORD)
      I=LKP1(NEWORD(IORD),TMPORD,3)
      IF(I .NE. 0) TMPORD(I)=0
      GO TO 30
10    DO 20 I=1,3
      IF(TMPORD(I) .EQ. 0) GO TO 20
      OUTORD(IORD)=TMPORD(I)
      TMPORD(I)=0
      GO TO 30
20    CONTINUE
      RETURN
30    CONTINUE
      RETURN
      END
```

Subroutine DERIVE

Subroutine DERIVE is called by READ and SET after reading or modifying the data for a day for a precinct. Its two parameters LPCT and LDAY are pointers to the data for the precinct and the day, respectively. DERIVE's primary function is to determine, for each block of the day: number of actual cars on duty; average cfs workload; maximum cfs workload in any hour; total calls for service; and number of effective cars on duty. For each tour of the day, DERIVE determines the fraction of calls in the lowest priority class, and the total number of calls during the tour.

Subroutine SBLACT (set block actual cars) is called to determine the number of actual cars on duty in each block of a day from the number of cars assigned to each tour of the day. The number of effective cars on duty in a block is computed using the formula given in Chapter III.

DERIVE also checks to determine whether or not each block has enough effective cars to handle the call-for-service workload in its busiest hour. If a block lacks sufficient effective cars, then the algorithm described in Chapter III is used to determine where to increase the assignment of cars to tours of a day so that each block will have enough effective cars. After this algorithm has been applied, subroutine SBLEF (set block effective cars) is called to redetermine the number of effective cars in each block of the day.

```
      SUBROUTINE DERIVE(LPCT,LDAY)

CALCULATES, FOR EACH BLOCK IN DAY AND PRECINCT,
AVERAGES OF INPUT DATA

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
   1(TOURNM,KEYWD(1,2))

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
   1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
   2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
   3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
   4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
   1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
   2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
   3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/PNTRS/IOVRLY,IOVTR(2),
   1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
   2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
   3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
   4LDIVNM,LDIVFL

      REAL LFR
      DIMENSION IBERR(24),ACB(2),ACTR(2)

      IDERR=0
      B1=CDAT(LPCT+B1POFF)
      B2=CDAT(LPCT+B2POFF)

           FIND NUMBER OF CARS ON DUTY IN EACH BLOCK

      CALL SBLACT(LPCT,LDAY)
      DO 20 IBLDT=1,NBLDT
      IBLK=ICDAT(LBLRFL+IBLDT-1)
      IF(IBLK .EQ. 0) GO TO 20
      LBLK=LDAY+BLDOFF+(IBLK-1)*NWDBL
      IBERR(IBLK)=0
      ISTART=ICDAT(LBLKTB(1)+IBLDT-1)
      IEND=ICDAT(LBLKTB(2)+IBLDT-1)
      BLKLN=IEND-ISTART+1

           CALCULATE AVERAGE AND MAXIMUM CALL RATE IN BLOCK

```

```
RMAX=0.
CRATE=0.
AWL=0.
LB=LDAY-1
DO 10 I=ISTART,IEND
IB=I+LB
CR=CDAT(IB+CRDOFF)
CRATE=CRATE+CR
R=CR*CDAT(IB+STDOFF)
IF(R .GT. RMAX) RMAX=R
AWL=AWL+R

AWL=AWL/BLKLN
CDAT(LBLK+CRBOFF)=CRATE
CDAT(LBLK+AWBOFF)=AWL
ACT=CDAT(LBLK+ACBOFF)

      CALCULATE EFFECTIVE CARS AND CHECK WHETHER
      MINIMUM ALLOCATION IS ACHIEVED

EF=ACT*(1.-((B1*AWL/ACT)+B2))
NEF=EF
IF(NEF.GT. RMAX) GO TO 15
IBERR(IBLK)=1
ACT=CEIL((EF+B1*AWL)/(1.-B2))
EF=ACT*(1.-((B1*AWL/ACT)+B2))
NEF=EF
IF(NEF.GT.RMAX) GO TO 13
ACT=ACT+1.
GO TO 12
CDAT(LBLK+ACBOFF)=ACT
CDAT(LBLK+EFBOFF)=EF
CDAT(LBLK+RMBOFF)=RMAX
CONTINUE

DO 100 ITYPE=1,NTRDT
ITOUR=ICDAT(LTRRFL+ITYPE-1)
IF(ITOUR .LT. 1) GO TO 100
ITERR=0
LTOUR=LDAY+TRDOFF+(ITOUR-1)*NWDTR
IF(ICDAT(LTOUR+TYTOFF) .EQ. 1) GO TO 100

      CALL RATE IN TOURS

IF(NPRIO .LT. 2) GO TO 40
LFR=1.
N=NPRIO-1
DO 30 I=1,N
LFR=LFR-CDAT(LTOUR+HFTOFF+I-1)
CDAT(LTOUR+HFTOFF+N)=LFR
CRATE=0.
DO 50 IBLK=1,2
ACB(IBLK)=0.
IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
IF(IBLD .LT. 1) GO TO 50
IBLR=ICDAT(LBLRFL+IBLD-1)
LBLK=LDAY+BLDOFF+(IBLR-1)*NWDBL
ACB(IBLK)=CDAT(LBLK+ACBOFF)
```

```
      CRATE=CRATE+CDAT(LBLK+CRBOFF)
      IF(IBERR(IBLR) .EQ. 0) GO TO 50
      ITERR=ITERR+IBLK
      CONTINUE

      CDAT(LTOUR+CRTOFF)=CRATE

            CALCULATE NEW NUMBER OF ACTUAL CARS IN TOURS,
            IF THERE HAS BEEN A CHANGE IN THE BLOCKS

      ID=ICDAT(LTOUR+TYTOFF)-1
      GO TO (60,65,70,75),ID
      IF(ITERR .EQ. 0) GO TO 90
      ACT=AMAX1(ACB(1),ACB(2))
      GO TO 85
      ACTR(1)=ACB(1)
      IF(ITERR .EQ. 2 .OR. ITERR .EQ. 0) GO TO 90
      ACT=ACB(1)
      GO TO 85
      ACTR(2)=ACB(2)
      IF(ITERR .EQ. 1 .OR. ITERR .EQ. 0) GO TO 90
      ACT=ACB(2)
      GO TO 85
      IF(ITERR .EQ. 0) GO TO 90
      ACT=CDAT(LTOUR+ACTOFF)
      TACT=ACT
      DO 80 I=1,2
      ACT=AMAX1(ACT,ACB(I)-ACTR(I))
      IF(TACT .GE. ACT) GO TO 90
      LPNM=LPCT+NMPOFF-1
      LTNM=LTRNM+(ITYPE-1)*8-1
      IDAY=(LDAY-DYPOFF-LPCT)/NWDDY
      IDAY=ICDAT(LDYWFL+IDAY)
      LDNM=LDAYNM+(IDAY-1)*8-1
      WRITE(SYSOUT,2) ACT,PCLSNM,(ICDAT(LPNM+I),I=1,8),
     1TOURNM,(ICDAT(LTNM+I),I=1,8),(ICDAT(LDNM+I),I=1,8)
      FORMAT(/' ***',F4.0,' CARS NEEDED IN',2(1X,8A1),' FOR',
     12(1X,8A1),' ON DAY ',8A1)
      CDAT(LTOUR+ACTOFF)=ACT
      IDERR=1
      CONTINUE
      CONTINUE

      IF(IDERR .EQ. 0) RETURN

            REDETERMINE EFFECTIVE CARS IF CHANGE IN ACTUAL CARS

      CALL SBLACT(LPCT,LDAY)
      CALL SBLEF(LPCT,LDAY)
      RETURN
      END
```

Subroutine SBLACT

Subroutine SBLACT (set block actual cars) is called to determine the number of actual cars on duty in each block of a day in a precinct, based on the number of cars assigned to each tour of the day. Parameters LPCT and LDAY are pointers to the data for the precinct and day for which the block allocations are to be determined. The algorithm used to determine the number of cars on duty in each block of a day from the number of cars on duty in each tour of the day is given in Chapter III.

```
      SUBROUTINE SBLACT(LPCT,LDAY)

 CALCULATES  NUMBER OF ACTUAL CARS IN BLOCKS,
 BASED ON NUMBER OF CARS IN TOURS

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      DO 10 IBLK=1,NBLRD
      LBLK=LDAY+BLDOFF+(IBLK-1)*NWDBL
0     CDAT(LBLK+ACBOFF)=0.
      DO 30 ITYPE=1,NTRDT
      ITOUR=ICDAT(LTRRFL+ITYPE-1)
      IF(ITOUR .LT. 1) GO TO 30
      LTOUR=LDAY+TRDOFF+(ITOUR-1)*NWDTR
      IF(ICDAT(LTOUR+TYTOFF) .EQ. 1) GO TO 30
      DO 20 IB=1,2
      IBLKDT=ICDAT(LTRTB(IB)+ITYPE-1)
      IF(IBLKDT .EQ. 0) GO TO 20
      IBLKRD=ICDAT(LBLRFL+IBLKDT-1)
      LBLK=LDAY+BLDOFF+(IBLKRD-1)*NWDBL
      CDAT(LBLK+ACBOFF)=CDAT(LBLK+ACBOFF)+CDAT(LTOUR+ACTOFF)
0     CONTINUE
0     CONTINUE
      RETURN
      END
```

Subroutine SBLEF

Subroutine SBLEF (set block effective cars) determines the number of effective cars on duty in each block of a day. Parameters LPCT and LDAY are pointers to the data for the precinct and day for which the calculations are to be performed. Chapter II and Appendix B of the User's Manual give the formula used to compute effective cars from actual cars and average workload.

```
      SUBROUTINE SBLEF(LPCT,LDAY)

 CONVERTS ACTUAL CARS TO EFFECTIVE CARS IN EACH BLOCK

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      B1=CDAT(LPCT+B1POFF)
      B2=CDAT(LPCT+B2POFF)
      DO 10 IBLK=1,NBLRD
      LBLK=LDAY+BLDOFF+(IBLK-1)*NWDBL
      AWL=CDAT(LBLK+AWBOFF)
      ACT=CDAT(LBLK+ACBOFF)
      CDAT(LBLK+EFBOFF)=ACT*(1.-((B1*AWL/ACT)+B2))
      RETURN
      END
```

Subroutine LIST

Subroutine LIST implements the LIST command. It prints input data (and some derived values) for selected precincts, days, and tours.

Subroutine GTDSPC is called to scan the qualifier and SETWFL is called to define the subset of precincts, days, and tours for which data will be listed. Function NXPCT is called to set a pointer (LPCT) to the data for the next precinct selected. A pointer value of zero at entry to NXPCT requests the first precinct selected; a pointer value of zero returned from NXPCT means no more precincts have been selected. The name, area, street miles, and unavailability parameters are printed for each precinct selected.

After a precinct pointer has been obtained and the data for the precinct printed, function NXDAY is called to find the days for which data are to be listed. As with NXPCT, the value of the day data pointer (LDAY) at entry to NXDAY indicates whether the first day for a precinct is to be located, and the value of the day pointer returned from NXDAY is zero if there are no more days selected. For each selected day its name, call rate parameter, and service time parameter are printed; in addition, column headings are printed for the tour data which follow.

Function NXTOUR is used in the same manner as NXPCT and NXDAY to index through the tours of each day. For each tour selected, LIST computes average call rate and service time over all its hours and the average number of effective cars in its blocks. These are printed along with the tour name, actual cars assigned, response speed, patrol speed, and the fraction of calls in each priority class.

```
      SUBROUTINE LIST

IMPLEMENTS THE LIST COMMAND

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      DIMENSION VAL(2),ORDER(3)
      INTEGER TYPE,VAL

      LGETT=TOP
      TYPE=CMD

            INTERPRETS QUALIFIER OF LIST COMMAND

      CALL SCAN(TYPE,VAL)
      CALL GTDSPC(TYPE,VAL,ORDER)
      IF(TYPE .NE. ERR) GO TO 10
      TOP=LGETT
      RETURN
      CALL SETWFL(IERR)
      IF(IERR .EQ. 0) GO TO 15
      TOP=LGETT
      RETURN

            FIND NEXT PRECINCT, WRITE HEADER INFORMATION
```

```
      LPCT=0
      LPCT=NXPCT(LPCT)
      IF(LPCT .NE. 0) GO TO 30
      TOP=LGETT
      RETURN
      WRITE(SYSOUT,1) PCLSNM, (ICDAT(LPCT+NMPOFF+I-1),I=1,8),
    1 CDAT(LPCT+ARPOFF),CDAT(LPCT+SMPOFF),CDAT(LPCT+B2POFF),
    2 CDAT(LPCT+B1POFF)

            FIND NEXT DAY.  LIST HEADER INFORMATION.

                LDAY=0
      LDAY=NXDAY(LPCT,LDAY)
      IF(LDAY .EQ. 0) GO TO 20
      IDAY=(LDAY-LPCT-DYPOFF)/NWDDY
      IDAY=ICDAT(LDYWFL+IDAY)
      LDNM=LDAYNM+(IDAY-1)*8-1
      WRITE(SYSOUT,2) (ICDAT(LDNM+I),I=1,8),CDAT(LDAY+CPDOFF),
    1 CDAT(LDAY+SPDOFF),TOURNM

            FIND NEXT TOUR.  CALCULATE AVERAGE CALL RATE,
            SERVICE TIME, EFFECTIVE CARS

      LTOUR=0
      LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) GO TO 40
      LTNM=LTRNM+(ITYPE-1)*8-1
      IF(ICDAT(LTOUR+TYTOFF) .NE. 5) GO TO 55
      WRITE(SYSOUT,3) (ICDAT(LTNM+I),I=1,8),CDAT(LTOUR+ACTOFF)
      GO TO 40
      ISTART=ICDAT(LTRST+ITYPE-1)
      IEND=ICDAT(LTREND+ITYPE-1)
      ST=0.
      DO 60 I=ISTART,IEND
      ST=ST+CDAT(LDAY+STDOFF+I-1)
      TOURLN=IEND-ISTART+1
      ST=ST*60./TOURLN
      CR=CDAT(LTOUR+CRTOFF)/TOURLN
      EF=0.
      DO 70 IBLK=1,2
      IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBDT .LT. 1) GO TO 70
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      BLKLN=ICDAT(LBLKTB(2)+IBDT-1)-ICDAT(LBLKTB(1)+IBDT-1)+1
      EF=EF+BLKLN*CDAT(LBLK+EFBOFF)
      CONTINUE
      EF=EF/TOURLN
      WRITE(SYSOUT,3) (ICDAT(LTNM+I),I=1,8),CDAT(LTOUR+ACTOFF),
    1 EF,CDAT(LTOUR+RVTOFF),
    1CDAT(LTOUR+PVTOFF), ST,CR,(CDAT(LTOUR+HFTOFF+I-1),I=1,3)
      GO TO 50
            FORMAT OF PRECINCT HEADER
      FORMAT(/,1H ,8A1,2H: ,8A1,'; AREA=',F5.1,'; STREET MILES=',
    1 F5.1,'; B2=',F5.3,'; B1=',F5.3)
            FORMAT FOR DAY HEADER AND COLUMN LABELS
      FORMAT(/'   DAY: ',8A1,'; CALL RATE PARM=',F5.2,'; SERVICE TIME'
```

```
1 ,' PARM=',F5.2//21X,
2' AVG.                  AVG.  AVG.  FRAC.  FRAC.  FRAC.'/16X,
3'ACT.  EFF.  RSP.  PTL.  SERV  CALL  OF P1  OF P2  OF P3'/
3 6X,8A1, 2X,
4'CARS  CARS  VEL.  VEL.  TIME  RATE  CALLS  CALLS  CALLS')
        FORMAT FOR ENTRIES IN COLUMNS
   FORMAT(6X,8A1,6(2X,F4.1),3(2X,F5.3))
   END
```

Subroutine SETWFL

Subroutine SETWFL (set work flags) is called to set the "work" flags (LTRWFL, LDYWFL) described in Table 2 after a command has been successfully interpreted. Division flags (LDIVFL) are also set.

These flags define the subsets of days, tours, and divisions (among those that have been read) which will be operated on by the current command. These subsets are determined from the phrases of the command qualifier. The qualifier phrases must have been converted to name lists by subroutine GTDSPC before SETWFL is called. For days and tours, each work flag corresponds to one day or tour that has been read. If a day or tour is selected by a command qualifier, then the value of its work flag will be the relative position of the day or tour among all the days or tours in the data base; otherwise, its value will be zero. If no day or tour names appear in a command qualifier, then all days or tours read are implicitly selected; otherwise, only those named are selected. Names that do not appear in the data base are ignored.

Division flags (in LDIVFL) correspond to names of divisions in a list produced by the READ command (LDIVNM). Flags of divisions named in the command qualifier are set to one (1); others are set to zero. Division flags are referenced by a division number associated with each precinct. The condition of no division names in the command qualifier is detected in subroutine NXPCT.

The parameter IERR is set to 1 if any errors are detected in SETWFL; otherwise, its value on return will be zero.

```
      SUBROUTINE SETWFL(IERR)

 SET WORK FLAGS
 ASED ON QUALIFIER SCANNED BY GTDSPC

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

             SET DAY FLAGS

      IERR=0
      IF(NDAYRD .LT. 1) GO TO 105
      DO 10 I=1,NDAYRD
      ICDAT(LDYWFL+I-1)=0
      LNM=LNMLST(1)
      N=NNAMES(1)
      IF(N.NE. 0) GO TO 30
      IDAY=0
      DO 20 I=1,NDAYDT
      IF(ICDAT(LDYRFL+I-1) .EQ. 0) GO TO 20
      IDAY=IDAY+1
      ICDAT(LDYWFL+IDAY-1)=I
      CONTINUE
      GO TO 100
      IF(N .EQ. 0) GO TO 100
      I=LKP8(ICDAT(LNM),ICDAT(LDAYNM),NDAYDT)
      IF(I .EQ. 0) GO TO 40
      IDAY=ICDAT(LDYRFL+I-1)
      IF(IDAY .LT. 1) GO TO 40
      ICDAT(LDYWFL+IDAY-1)=I
      LNM=LNM+8
      N=N-1
      GO TO 30

             SET TOUR FLAGS

      IF(NTRRD .GT. 0) GO TO 108
      WRITE(SYSOUT,1)
      FORMAT(/' *** NO PRIOR READ COMMAND - REENTER.')
      IERR=1
```

```
RETURN
DO 110 I=1,NTRRD
ICDAT(LTRWFL+I-1)=0
LNM=LNMLST(2)
N=NNAMES(2)
IF(N .NE. 0) GO TO 130
ITOUR=0
DO 120 I=1,NTRDT
IF(ICDAT(LTRRFL+I-1) .EQ. 0) GO TO 120
ITOUR=ITOUR+1
ICDAT(LTRWFL+ITOUR-1)=I
CONTINUE
GO TO 200
IF(N .EQ. 0) GO TO 200
I=LKP8(ICDAT(LNM),ICDAT(LTRNM),NTRDT)
IF(I .EQ. 0) GO TO 140
ITOUR=ICDAT(LTRRFL+I-1)
IF(ITOUR .LT. 1) GO TO 140
ICDAT(LTRWFL+ITOUR-1)=I
LNM=LNM+8
N=N-1
GO TO 130

     SET DIVISION FLAGS

IF(NDIVRD .EQ. 0) RETURN
DO 210 I=1,NDIVRD
ICDAT(LDIVFL+I-1)=0
LNM=LNMLST(3)
N=NNAMES(3)
IF(N .EQ. 0) RETURN
I=LKP8(ICDAT(LNM),ICDAT(LDIVNM),NDIVRD)
IF(I .EQ. 0) GO TO 230
ICDAT(LDIVFL+I-1)=1
LNM=LNM+8
N=N-1
GO TO 220
END
```

Function NXPCT

Function NXPCT (next precinct) is called during command execution to determine the next precinct selected by a command qualifier. Its argument (LPCT) is a pointer to the data for a precinct in array DATA. On entry, LPCT points to the last precinct selected (zero if none) and the value of the function is a pointer to the next precinct selected (zero if none).

A precinct is selected if any of the following criteria are met:

- No precinct or division names appear in the command qualifier
- The entry in table LDIVFL (see Chapter IV) corresponding to the precinct's division is nonzero
- The precinct's name appears in the PRECINCT phrase of the command qualifier.

```
      FUNCTION NXPCT(LPCT)

FINDS THE NEXT PRECINCT SELECTED AFTER LPCT

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      NXPCT=LPCT
      IF(LPCT .NE. 0) GO TO 10
      NXPCT=LPCTDT
      GO TO 20
      NXPCT=NXPCT+NWDPCT
      IF(NXPCT .LE. LPCTDT+(NPCTRD-1)*NWDPCT) GO TO 30
      NXPCT=0
      RETURN
      IF(NNAMES(3)+NNAMES(4) .EQ. 0) RETURN
      IF(NNAMES(3) .EQ.0) GO TO 40
      IDIV=ICDAT(NXPCT+DVPOFF)
      IF(ICDAT(LDIVFL+IDIV-1) .NE. 0) RETURN
      IF(NNAMES(4) .EQ. 0) GO TO 10
      I=LKP8(ICDAT(NXPCT+NMPOFF),ICDAT(LNMLST(4)),NNAMES(4))
      IF(I .EQ. 0) GO TO 10
      RETURN
      END
```

Function NXDAY

Function NXDAY (next day) is used to index through the selected days for a precinct during command execution. Its arguments LPCT and LDAY are pointers to the data for a precinct and to the data for the last day selected for the precinct, respectively (see Chapter IV). The value of the function is a pointer to the next day selected after LDAY (zero if none). On entry, a value of zero for LDAY indicates that the first day selected for the precinct is to be located.

A day is selected if and only if the value of its corresponding work flag is nonzero (see the discussion of Subroutine SETWFL).

```
      FUNCTION NXDAY(LPCT,LDAY)

 FINDS THE NEXT DAY SELECTED IN PRECINCT LPCT AFTER LDAY

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      NXDAY=LDAY
      IF(LDAY .NE. 0) GO TO 10
      NXDAY=LPCT+DYPOFF
      GO TO 20
0     NXDAY=NXDAY+NWDDY
0     IDAY=(NXDAY-LPCT-DYPOFF)/NWDDY+1
      IF(IDAY .LE. NDAYRD) GO TO 30
      NXDAY=0
      RETURN
0     IF(ICDAT(LDYWFL+IDAY-1) .EQ. 0) GO TO 10
      RETURN
      END
```

Function NXTOUR

Function NXTOUR (next tour) is used to index through the selected tours of a day during command execution. Its arguments LDAY and LTOUR are pointers to the data for a day within a precinct and the last tour selected within the day, respectively. The value of the function is a pointer to the next tour selected after LTOUR (zero if none). On entry, a value of zero for LTOUR indicates that the first tour selected for the day is to be located.

A tour is selected if and only if the value of its corresponding work flag is nonzero and the tour type is not equal to 1 (a tour type of 1 indicates that the tour holds a place that would be occupied by an overlay tour, but there is no overlay tour for the specified day).

On return, the parameter ITYPE is set to the value of the tour work flag. This value is the relative position of the tour among all the tours in the data base and is useful for referencing the tables that contain tour starting and ending times and mappings of tours to blocks.

```
      FUNCTION NXTOUR(LDAY,LTOUR,ITYPE)

  FINDS THE NEXT TOUR SELECTED IN LDAY AFTER LTOUR.
  ITYPE IS THE VALUE OF THE TOUR WORK FLAG.

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      NXTOUR=LTOUR
      IF(NXTOUR .NE. 0) GO TO 10
      NXTOUR=LDAY+TRDOFF
      GO TO 20
      NXTOUR=NXTOUR+NWDTR
      ITOUR=(NXTOUR-LDAY-TRDOFF)/NWDTR+1
      IF(ITOUR .LE. NTRRD) GO TO 30
      NXTOUR=0
      RETURN
      ITYPE=ICDAT(LTRWFL+ITOUR-1)
      IF(ITYPE .EQ. 0 .OR. ICDAT(NXTOUR+TYTOFF) .EQ. 1) GO TO 10
      RETURN
      END
```

Subroutine DISP

Subroutine DISP implements the DISP command. Its function is to display selected output tables for selected shifts.

DISP calls SCAN twice to get the user's table specification. Then GTDSPC is called to scan the command qualifier and SETWFL is called to determine the subset of shifts for which output will be displayed. Subroutine MRGORD determines the output order (in DORDER) that results from the previously established output order (RORDER) and the order of qualifier phrases in the current command (ORDER). We consider that the six possible permutations of the values in DORDER define six different ways of printing tables. The permutations are mapped onto unique integers by multiplying the elements of DORDER by successive powers of two. Table 10 gives the output orderings and their corresponding integer labels (after the labels have been transformed into successive integers in the range 1-6).

Table 10

OUTPUT ORDERINGS

Integer Label	Output Order
1	Precinct, Tour, Day
2	Tour, Precinct, Day
3	Precinct, Day, Tour
4	Day, Precinct, Tour
5	Tour, Day, Precinct
6	Day, Tour, Precinct

The integer labels are used as entries into a branch table that controls calls to routines to implement table displays in the various output orderings. In the program documented in this report, only the Day, Tour, Precinct and Precinct, Day, Tour orderings are implemented. The branch table is entered once for each table number specified by the user.

In the program documented here, only Tables 1 and 2 are valid. The subroutine provides flexibility for the user to implement other output orders and/or output tables.

```
      SUBROUTINE DISP

 IMPLEMENTS THE DISP COMMAND

      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      DIMENSION ORDER(3),VAL(2),DORDER(3)
      INTEGER ORDER,TYPE,VAL,DORDER

      LGETT=TOP
      TYPE=CMD

            FINDS WHICH TABLE(S) ARE TO BE DISPLAYED

      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. FSPEC) GO TO 20
      WRITE(SYSOUT,1)
      FORMAT(/' *** INVALID TABLE SPECIFICATION - REENTER.')
      TOP=LGETT
      RETURN
      KEYVAL=VAL(1)
      I=KEYVAL-TYPOFF(FSPEC)
      IF(I .NE. 3) GO TO 10
      CALL SCAN(TYPE,VAL)
      IF(TYPE .NE. NUMLST) GO TO 10
      NPARM=VAL(1)
      LPARM=VAL(2)

            INTERPRET QUALIFIER

      CALL SCAN(TYPE,VAL)
      CALL GTDSPC(TYPE,VAL,ORDER)
      IF(TYPE .NE. ERR) GO TO 30
      TOP=LGETT
      RETURN

            SET WORK FLAGS

      CALL SETWFL(IERR)
```

```
IF(IERR .EQ. 0) GO TO 35
TOP=LGETT
RETURN

     SET OUTPUT ORDER

CALL MRGORD(ORDER,RORDER,DORDER)
IORD=0
DO 40 I=1,3
K=2**(I-1)
IORD=IORD+K*DORDER(I)
IF(IORD .GT. 14) IORD=IORD-1
IORD=IORD-10
DO 700 I=1,NPARM
ITAB=ICDAT(LPARM+(I-1)*2)
IF(ITAB .LT. 1 .OR. ITAB .GT. 2) GO TO 700

     CALL ROUTINE TO DISPLAY TABLE

GO TO (100,200,300,400,500,600),IORD

CALL DSPPDT(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPPDT(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
GO TO 700
CALL DSPDTP(ITAB)
WRITE(SYSOUT,2)
FORMAT(1H /1H )
TOP=LGETT
RETURN
END
```

Subroutine DSPPDT

Subroutine DSPPDT (display by precinct, day, tour) is called by subroutine DISP to print DISP command output tables by precinct, day, and tour. Its parameter ITAB specifies the table number to be printed.

Subroutine ZERO is called to initialize accumulators for averages at the overall, precinct, day, and tour levels. The integer parameter of ZERO specifies the level of the accumulators to be initialized, with one (1) corresponding to the highest level (overall) and four (4) corresponding to the lowest level (tour).

Functions NXPCT, NXDAY, and NXTOUR are used to index through the precincts, days, and tours selected by the user. A labeling line and column headings for tours are printed for each precinct and day displayed.

For each tour selected, a flag (FLAG) is set which contains the character to be printed at the left of each line of table output. Another flag, IADD (which is a parameter for subroutine COMPTB), indicates whether or not the tour is an overlay tour, and therefore whether its measures (level 4) are to be accumulated into the measures for a day (level 3). This may seem redundant here, but the decision to include overlay tour measures in higher levels of aggregation or not must be made at different levels for different output orders. Subroutine COMPTB is called to compute the measures of table ITAB for the tour. Subroutine PRTBL (print table) prints a line of output measures (at level 4, the tour level). Subroutine TOTAL adds measures for a specified level into the accumulators for the next highest level. TOTAL also computes and prints averages for all except level 4 measures. Statistics are not printed for a level of aggregation if there is only one entry for the next lower level.

```
      SUBROUTINE DSPPDT(ITAB)

 DISPLAYS TABLE ITAB IN ORDER OF TOUR WITHIN DAY WITHIN PRECINCT

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      DATA BLANK/1H /,STAR/1H*/,PLUS/1H+/

      CALL ZERO(1)

            FIND PRECINCT

      NPCT=0
      LPCT=0
      LPCT=NXPCT(LPCT)
      IF(LPCT .EQ. 0) GO TO 100
      NPCT=NPCT+1
      NDAY=0
      CALL ZERO(2)
      LDAY=0

            FIND DAY

      LDAY=NXDAY(LPCT,LDAY)
      IF(LDAY .EQ. 0) GO TO 80
      NDAY=NDAY+1
      IDAY=(LDAY-LPCT-DYPOFF)/NWDDY
      IDAY=ICDAT(LDYWFL+IDAY)
```

```
      LDNM=LDAYNM+(IDAY-1)*8-1
      WRITE(SYSOUT,1) PCLSNM,(ICDAT(LPCT+NMPOFF+I-1),I=1,8),
     1 (ICDAT(LDNM+I),I=1,8)
            FORMAT OFR PRECINCT AND DAY HEADER
      FORMAT(/' ',8A1,': ',8A1,'; DAY: ',8A1)
      CALL TITLE(ITAB,TOURNM)
      NTOUR=0
      LTOUR=0
      CALL ZERO(3)

            FIND TOUR

      LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) GO TO 60
      IND=ICDAT(LTOUR+TYTOFF)
      FLAG=BLANK
      IF(IND .LT. 3) GO TO 40
      FLAG=STAR
      IF(IND .EQ. 5) FLAG=PLUS
      NTOUR=NTOUR+1
      CALL ZERO(4)
      IADD=1
      IF(IND .EQ. 5) IADD=0

             COMPUTE OUTPUT MEASURES

      CALL COMPTB(ITAB,LPCT,LDAY,LTOUR,ITYPE,IADD)

            PRINT OUTPUT MEASURES FOR SHIFT

      CALL PRTBL(ITAB,4,FLAG,ICDAT(LTRNM+(ITYPE-1)*8))
      GO TO 30

            ACCUMULATE MEASURES FOR DAYS

      CALL TOTAL(ITAB,3,NTOUR,1)
      GO TO 20
      IF(NDAY .LT. 1) RETURN
      IF(NDAY .GT. 1)WRITE(SYSOUT,2) PCLSNM,
     1(ICDAT(LPCT+NMPOFF+I-1),I=1,8)
            FORMAT FOR PRECINCT HEADER
      FORMAT(/' ',8A1,': ',8A1)

            ACCUMULATE MEASURES FOR PRECINCTS

      CALL TOTAL(ITAB,2,NDAY,1)
      GO TO 10
      IF(NPCT .LT. 2) RETURN
       WRITE(SYSOUT,3)
      FORMAT(/' GRAND')
      CALL TOTAL(ITAB,1,NPCT,0)
      RETURN
      END
```

Subroutine DSPDTP

Subroutine DSPDTP controls DISP command output when the output order is precinct, within tour, within day. Parameter ITAB specifies the output table that is to be displayed.

DSPDTP operates in a manner similar to that of DSPPDT. The primary differences are in the meanings of the different levels of aggregation and in the way the next shift to be displayed is found. The routines NXPCT, NXDAY, and NXTOUR are set up to vary the tour most quickly, followed by the day and precinct. This corresponds exactly to the output order of tour within day within precinct, but not to precinct within tour within day. Therefore, for each day selected, DSPDTP determines the number of times that NXDAY will have to be called after a precinct is located to get to the day. DSPDTP also computes the number of times that NXTOUR must be called to get to a particular tour after a precinct and day have been located.

```
      SUBROUTINE DSPDTP(ITAB)

 DISPLAYS TABLE ITAB IN ORDER OF PRECINCT WITHIN TOUR WITHIN DAY

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      DATA BLANK/1H /,STAR/1H*/,PLUS/1H+/

      NDAY=0
      NXTDAY=0
      CALL ZERO(1)

          FIND POSITION OF NEXT DAY AMONG SELECTED DAYS

      DO 15 I=1,NDAYRD
      IDAY=ICDAT(LDYWFL+I-1)
      IF(IDAY .GT. NXTDAY) GO TO 20
      CONTINUE
      GO TO 100
      NXTDAY=IDAY
      NDAY=NDAY+1
      LDNM=LDAYNM+(IDAY-1)*8-1
      CALL ZERO(2)

          FIND TYPE OF NEXT DAY

      NXTYPE=0
      NTOUR=0
```

```
      DO 40 I=1,NTRRD
      ITYPE=ICDAT(LTRWFL+I-1)
      IF(ITYPE .GT. NXTYPE) GO TO 50
      CONTINUE
      IF(NTOUR .GT. 1) WRITE(SYSOUT,2) (CDAT(LDNM+I),I=1,8)
           FORMAT FOR DAY HEADER
      FORMAT(/' DAY: ',8A1)

           ACCUMULATE MEASURES FOR DAY

      CALL TOTAL(ITAB,2,NTOUR,1)
      GO TO 10
      NXTYPE=ITYPE
      NTOUR=NTOUR+1
      CALL ZERO(3)
      LTNM=LTRNM+(ITYPE-1)*8-1
      WRITE(SYSOUT,1) (ICDAT(LDNM+I),I=1,8),TOURNM,(ICDAT(LTNM+I),
     1 I=1,8)
           FORMAT OFR DAY AND TOUR HEADER
      FORMAT(/' DAY ',8A1,';',2(1X,8A1))
      CALL TITLE(ITAB,PCLSNM)

           FIND NEXT PRECINCT

      LPCT=0
      NPCT=0
      LPCT=NXPCT(LPCT)
      IF(LPCT .NE. 0) GO TO 65
      IADD=1
      IF(IOVRLY .EQ. 1 .AND. NXTYPE .EQ. NTRDT) IADD=0

           ACCUMULATE MEASURES FOR TOUR

      CALL TOTAL(ITAB,3,NPCT,IADD)
      GO TO 30

           GET DAY

      LDAY=0
      DO 70 I=1,NDAY
      LDAY=NXDAY(LPCT,LDAY)

           GET TOUR
      LTOUR=0
      LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) GO TO 60
      IF(ITYPE .NE. NXTYPE) GO TO 80
      NPCT=NPCT+1
      FLAG=BLANK
      IND=ICDAT(LTOUR+TYTOFF)
      IF(IND .LT. 3) GO TO 90
      FLAG=STAR
      IF(IND .EQ. 5) FLAG=PLUS
      CALL ZERO(4)

           COMPUTE AND PRINT MEASURES

      CALL COMPTB(ITAB,LPCT,LDAY,LTOUR,ITYPE,1)
```

```
CALL PRTBL(ITAB,4,FLAG,ICDAT(LPCT+NMPOFF))
GO TO 60
IF(NDAY .LT. 2) RETURN
WRITE(SYSOUT,4)
FORMAT(/' GRAND')
CALL TOTAL(ITAB,1,NDAY,0)
RETURN
END
```

Subroutine ZERO

Subroutine ZERO is called by the subroutines that control table output to clear level N accumulators. ZERO initializes array T, which is used to accumulate weighted sums for output measures, and array S, which is used to accumulate weights.

```
      SUBROUTINE ZERO(N)
C
C INITIALIZE ACCUMULATORS FOR LEVEL 'N' TABLE OUTPUT
C
      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER
C
      DO 10 I=1,8
      S(N,I)=0.
10    T(N,I)=0.
      RETURN
      END
```

Subroutine TITLE

Subroutine TITLE is called by the routines that control DISP command table output to print column headings. Parameter ITAB specifies the table for which headings are to be printed. NAME is an array that contains eight characters in A1 format used as a heading to identify the leftmost column. NAME can identify the column entries as being tour names, precinct names, or day names (although the day name option has not been implemented).

```
      SUBROUTINE TITLE(ITAB,NAME)

 PRINTS COLUMN HEADINGS FOR TABLE ITAB

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      DIMENSION NAME(8)

      GO TO (10,20),ITAB

10    WRITE(SYSOUT,1) NAME
      FORMAT(/,12X,
     1'AVG.  AVG.   AVG.   PATROL    AVG.   AV PTL FREQ'/12X,
     2'UTIL. UTIL.  TRAV.  HRS PER  PATROL  TIMES SUPP  AVG CARS'/
     3 2X,8A1,2X,
     4'(EFF) (ACT)  TIME   SUPP CR   FREQ.  CR PER HR   AVAIL.')
      RETURN

20    WRITE(SYSOUT,2) NAME
      FORMAT(/,12X,
     1'ACT. CAR   CALL  SERV  PROB CALL  AVG P2  AVG P3  AVG TOT'
     1/2X,8A1,2X,
     2'CARS  HRS  RATE  TIME  DELAYED   DELAY   DELAY   DELAY')
      RETURN
      END
```

Subroutine COMPTB

Subroutine COMPTB (compute table) is called from the routines that control DISP command output to compute output measures for one shift. Parameter ITAB specifies the table for which measures are to be computed. LPCT, LDAY, and LTOUR are pointers to the data for the precinct, day, and tour to be used in the computation. ITYPE is the relative position of the tour among all tours in the data base; it provides an index to tour starting and ending times. IADD indicates whether or not the measures computed for the tour are to be included in the next higher level of aggregation (this depends on the DISP command output order and on whether or not the shift is an overlay).

For either output table, weighted sums and weights are computed for all measures and summed over all blocks of the shift. The measures are either computed directly from data items in CDAT or by function references to such routines as AVTT for average travel time or OBJF1 for fraction of calls delayed. Weighted sums and weights are accumulated in the columns of row 4 of arrays T and S, respectively. If requested, the contents of row 4 of arrays T and S are added to the contents of row 3; this represents inclusion of the measures for the shift in the next higher level of aggregation. Finally, averages of the measures over the blocks of the shift are computed by dividing the weighted sums in T by the weights in S. Array CIND is set to print an asterisk next to the value of the limiting constraint (if any).

```
      SUBROUTINE COMPTB(ITAB,LPCT,LDAY,LTOUR,ITYPE,IADD)

COMPUTES ONE OUTPUT LINE OF ONE TABLE

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER

      DATA BLANK/1H /,STAR/1H*/
      DIMENSION ICNSTR(10)
      DATA ICNSTR(1)/1/,ICNSTR(2)/3/,ICNSTR(3)/7/,ICNSTR(4)/4/,
     1 ICNSTR(5)/5/,ICNSTR(6)/1/,ICNSTR(7)/5/,ICNSTR(8)/6/,
     2 ICNSTR(9)/7/,ICNSTR(10)/8/
      LCR=LDAY+CRDOFF
      LST=LDAY+STDOFF
      IEND=ICDAT(LTREND+ITYPE-1)
      ISTART=ICDAT(LTRST+ITYPE-1)
      ILEN=IEND-ISTART+1
      TOURLN=ILEN
      GO TO (10,500),ITAB

          COMPUTE TABLE 1 MEASURES

      DO 20 IBLK=1,2
      IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBLD .LT. 1) GO TO 20
      IBLR=ICDAT(LBLRFL+IBLD-1)
      LBLK=LDAY+BLDOFF+(IBLR-1)*NWDBL
      IBEND=ICDAT(LBLKTB(2)+IBLD-1)
      IBSTRT=ICDAT(LBLKTB(1)+IBLD-1)
      BLKLN=IBEND-IBSTRT+1
      EF=CDAT(LBLK+EFBOFF)

          IF OVERLAY TOUR, GET DATA FROM OVERLAID TOUR

      LTTOUR=LTOUR
```

```
      IF(ICDAT(LTOUR+TYTOFF) .EQ. 5)
     1     LTTOUR=LDAY+TRDOFF+(IOVTR(IBLK)-1)*NWDTR
      RV=CDAT(LTTOUR+RVTOFF)
      AWL=CDAT(LBLK+AWBOFF)

            UTILIZATION (EFFECTIVE)

      X=AWL/EF
      Y=BLKLN*EF
      T(4,1)=T(4,1)+X*Y
      S(4,1)=S(4,1)+Y

            UTILIZATION (ACTUAL)

      ACT=CDAT(LBLK+ACBOFF)
      X=AWL/ACT
      Y=BLKLN*ACT
      T(4,2)=T(4,2)+X*Y
      S(4,2)=S(4,2)+Y

            TRAVEL TIME

      X=AVTT(IBSTRT,IBEND,LPCT,LDAY,RV,EF)
      Y=CDAT(LBLK+CRBOFF)
      T(4,3)=T(4,3)+X*Y
      S(4,3)=S(4,3)+Y

            PATROL HOURS/ SUPP CRIME

      X=BLKLN*(EF-AWL)
      Y=CDAT(LBLK+OCBOFF)
      T(4,4)=T(4,4)+X
      S(4,4)=S(4,4)+Y

            PATROL FREQUENCY

      T(4,5)=T(4,5)+X*CDAT(LTTOUR+PVTOFF)
      S(4,5)=S(4,5)+BLKLN*CDAT(LPCT+SMPOFF)

            PATROL FREQ * SUPP CR/HR

      T(4,6)=T(4,6)+X*CDAT(LTTOUR+PVTOFF)*CDAT(LBLK+OCBOFF)
      S(4,6)=S(4,6)+BLKLN**2*CDAT(LPCT+SMPOFF)

            AVERAGE CARS AVAILABLE

      X=EF-AWL
      T(4,7)=T(4,7)+X*BLKLN
      S(4,7)=S(4,7)+BLKLN
      CONTINUE

            ACCUMULATE MEASURES IF REQUESTED

      IF(IADD .LT. 1) GO TO 40
      DO 30 I=1,7
      IF(S(4,I) .EQ. 0.) T(4,I)=0.
      T(3,I)=T(3,I)+T(4,I)
      S(3,I)=S(3,I)+S(4,I)
```

```
      CONTINUE

C           COMPUTE AVERAGES

      DO 50 I=1,7
      CIND(I)=BLANK
      IF(S(4,I) .EQ. 0.) GO TO 50
      T(4,I)=T(4,I)/S(4,I)
      CONTINUE

C          SET CONSTRAINT INDICATOR

      IC=ICDAT(LTOUR+CTTOFF)
      IF(IC .GT. 5 .OR. IC .LT. 1) RETURN
      CIND(ICNSTR(IC))=STAR
      RETURN

C          COMPUTE TABLE 2 MEASURES

      ACT=CDAT(LTOUR+ACTOFF)
      T(4,1)=ACT
      S(4,1)=1.
      T(4,2)=ACT*TOURLN
      S(4,2)=1.
      DO 520 IBLK=1,2
      IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBLD .LT. 1) GO TO 520
      IBLR=ICDAT(LBLRFL+IBLD-1)
      LBLK=LDAY+BLDOFF+(IBLR-1)*NWDBL
      ISTART=ICDAT(LBLKTB(1)+IBLD-1)
      IEND=ICDAT(LBLKTB(2)+IBLD-1)
      BLKLN=IEND-ISTART+1
      LTTOUR=LTOUR
      IF(ICDAT(LTOUR+TYTOFF) .EQ. 5)
     1     LTTOUR=LDAY+TRDOFF+(IDVTR(IBLK)-1)*NWDTR
      LFR=LTTOUR+HFTOFF

C          CALL RATE

      T(4,3)=T(4,3)+CDAT(LBLK+CRBOFF)
      S(4,3)=S(4,3)+BLKLN

C          SERVICE TIME

      ST=CDAT(LBLK+AWBOFF)*BLKLN*60.
      T(4,4)=T(4,4)+ST
      S(4,4)=S(4,4)+CDAT(LBLK+CRBOFF)

C          PROB. CALL DELAYED

      EF=CDAT(LBLK+EFBOFF)
      X=OBJF1(ISTART,IEND,LCR,LST,EF)
      T(4,5)=T(4,5)+X
      S(4,5)=S(4,5)+CDAT(LBLK+CRBOFF)

C          AVG P2 DELAY

      X=OBJF2(2,ISTART,IEND,LCR,LST,LFR,EF)*60.
```

```
Y=CDAT(LBLK+CRBOFF)*CDAT(LFR+1)
T(4,6)=T(4,6)+X
S(4,6)=S(4,6)+Y

      AVG P3 DELAY

X=OBJF2(3,ISTART,IEND,LCR,LST,LFR,EF)*60.
Y=CDAT(LBLK+CRBOFF)*CDAT(LFR+2)
T(4,7)=T(4,7)+X
S(4,7)=S(4,7)+Y

      AVG TOTAL DELAY

RV=CDAT(LTTOUR+RVTOFF)
X=OBJF3(ISTART,IEND,LPCT,LDAY,RV,EF)*60.
T(4,8)=T(4,8)+X
S(4,8)=S(4,8)+CDAT(LBLK+CRBOFF)
CONTINUE
N=8
IF(IADD .LT.1) N=2
DO 530 I=1,N
IF(S(4,I) .EQ. 0.) T(4,I)=0.
T(3,I)=T(3,I)+T(4,I)
S(3,I)=S(3,I)+S(4,I)
DO 550 I=1,8
CIND(I)=BLANK
IF(S(4,I) .EQ. 0.) GO TO 550
T(4,I)=T(4,I)/S(4,I)
CONTINUE
IC=ICDAT(LTOUR+CTTOFF)
IF(IC .LT. 6 .OR. IC .GT. 10) RETURN
CIND(ICNSTR(IC))=STAR
RETURN
END
```

Subroutine PRTBL

Subroutine PRTBL (print table) prints one line of Table 1 or Table 2 output.* The line can represent any level of aggregation from one shift to an overall average. Parameter ITAB specifies the table number to be printed and that implies the format and number of items to be written. LEV is the level of aggregation of statistics that are to be printed. PRTBL assumes that T(LEV,N) contains the output measure that will be printed in column N+1 of the line of output (this will have been computed by TOTAL or COMPTB, depending on LEV). NAME is an eight-character identifier that will be printed in the first output column. In the current version NAME can be a tour name, a precinct name, or the word "AVERAGE," depending on the output order and the level of aggregation. FLAG is a one-character indicator that is printed at the left of an output line to show the overlay status of a shift.

PRTBL assumes that CIND(N) contains a one-character indicator (asterisk or blank) that will be printed to the left of the (N+1)st column to indicate whether or not the corresponding measure was the limiting constraint in a MEET command.

---

*This refers to the two types of output that can be obtained using the DISP command, not to the tables in the present report.

```
      SUBROUTINE PRTBL(ITAB,LEV,FLAG,NAME)

  PRINTS ONE LINE OF TABLE ITAB

      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      DIMENSION NAME(8)

      GO TO (10,20),ITAB

0     WRITE(SYSOUT,1) FLAG,NAME,(CIND(I),T(LEV,I),I=1,7)
      FORMAT(1H ,A1,8A1,1X,A1,F4.3,1X,A1,F4.3,2X,A1,F4.1,1X,2(2X,A1
    1,F5.2), 3X,A1,F6.3,3X,A1,F6.2)
      RETURN

0     WRITE(SYSOUT,2) FLAG,NAME,(CIND(I),T(LEV,I),I=1,8)
      FORMAT(1H ,A1,8A1,1X,A1,F4.1,A1,F5.1,1X,A1,F4.1,A1,F5.1,3X,A1,
    1 F4.3,3X,3(1X,A1,F6.2))
      RETURN
      END
```

Subroutine TOTAL

Subroutine TOTAL is called from the routines that control DISP command output. Its function is to add weighted sums and weights for a specified level of output measures to the accumulators for the next higher level. Averages are computed for the specified level and printed via a call to PRTBL.

Parameter ITAB specifies the table of output measures being displayed. LEV specifies the level of measures to be printed. N gives the number of observations at level LEV+1 that are reflected in the level LEV sums (if N is less than 2, no level LEV statistics are printed). IADD indicates whether or not the accumulation of level LEV sums into LEV-1 is to take place (this depends on overlay considerations).

```
      SUBROUTINE TOTAL(ITAB,LEV,N,IADD)

ACCUMULATES SUMS FOR WEIGHTED AVERAGES IN TABLES

      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      DIMENSION AV(8)
      DATA AV(1)/1HA/,AV(2)/1HV/,AV(3)/1HE/,AV(4)/1HR/,AV(5)/1HA/,
   1AV(6)/1HG/,AV(7)/1HE/,AV(8)/1H /,FLAG/1H /

      IF(N .LT. 1) RETURN
      IF(LEV .LT. 2) GO TO 15
      IF(ITAB .EQ. 1 .AND. IADD .EQ. 0) GO TO 15
      M=8
      IF(ITAB .EQ. 2 .AND. IADD .EQ. 0) M=2
      LEVM1=LEV-1
      DO 10 I=1,M
      T(LEVM1,I)=T(LEVM1,I)+T(LEV,I)
      S(LEVM1,I)=S(LEVM1,I)+S(LEV,I)
      DO 17 I=1,8
      IF(S(LEV,I) .GT. 0.) GO TO 16
      T(LEV,I)=0.
      GO TO 17
      T(LEV,I)=T(LEV,I)/S(LEV,I)
      CIND(I)=FLAG
      IF(LEV .EQ. 4 .OR. N .LT. 2) RETURN
      WRITE(SYSOUT,1)
      FORMAT(1H )
      CALL PRTBL(ITAB,LEV,FLAG,AV)
      GO TO (20,30),ITAB

      RETURN

      X=T(LEV,1)*S(LEV,1)
      Y=T(LEV,2)*S(LEV,2)
      WRITE(SYSOUT,3) X,Y
      FORMAT('  TOTAL',3X,F6.1,1X,F6.1)
      RETURN
      END
```

Function AVTT

Function AVTT returns the average travel time to incidents over a specified span of hours of a particular day in a precinct. Parameters ISTART and IEND give the first and last hour for which travel time is computed. LPCT and LDAY are pointers to the data for the precinct and day. RV is the response speed of patrol units and EF is the number of effective cars on duty.

The formula used to determine the travel time for each hour is given in Appendix B of the User's Manual. The travel times for each hour are weighted by the number of calls in the hour. The travel time returned is in minutes.

```
      FUNCTION AVTT(ISTART,IEND,LPCT,LDAY,RV,EF)

 CALCULATES AVERAGE TRAVEL TIME

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      TD=0.
      CRT=0.
      LCR=LDAY+CRDOFF-1
      LST=LDAY+STDOFF-1
      A=CDAT(LPCT+ARPOFF)
      STRDNS=CDAT(LPCT+SMPOFF)/A
      G=(STRDNS-1.)/(STRDNS-2.)
      SQRTA=SQRT(A)
      DO 30 I=ISTART,IEND
      CR=CDAT(LCR+I)
      ST=CDAT(LST+I)
      CRT=CRT+CR
      AVAVL=EF-CR*ST

             USE TRAVEL DISTANCE FUNCTION APPROPRIATE FOR AVG
             CARS AVAILABLE IN AN HOUR

      IF(AVAVL .GE. 1.) GO TO 10
      TD=TD+.678*SQRTA*CR
      GO TO 30
      IF(AVAVL .GE.2.) GO TO 20
      TD=TD+SQRTA*(.08 +.598/SQRT(AVAVL))*CR
      GO TO 30
      TD=TD+.711*CR*SQRTA/SQRT(AVAVL)
      CONTINUE

             COMPUTE AVERAGE TRAVEL TIME FROM TRAVEL DISTANCE WITH
             STREET DENSITY CORRECTION

      AVTT=60.*G*TD/(CRT*RV)
      RETURN
      END
```

Function OBJF1

Function OBJF1 (objective function 1) returns the weighted sum of the probability that a call will be delayed over a span of hours of a day in a precinct. It is called from COMPTB for Table 2 output, from KNSTR when constraints are being met, and from OBJFUN when car hours are being allocated. Parameters ISTART and IEND specify the span of hours. LCR and LST are pointers to the hourly call-rate and service-time data for the day, and EF is the number of effective cars on duty.

The probability that a call is delayed in each hour is given by PQUEUE; this is weighted by the number of calls in the hour.

```
      FUNCTION OBJF1(ISTART,IEND,LCR,LST,EF)
:
: CALCULATES WEIGHTED SUM OF PROBABILITY THAT A CALL
: WILL BE DELAYED
:
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWJRDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
:
      CRT=0.
      Q=0.
      DO 10 IHOUR=ISTART,IEND
      CR=CDAT(LCR+IHOUR-1)
      ST=CDAT(LST+IHOUR-1)
      AWL=CR*ST
.0    Q=Q+PQUEUE(AWL,EF)*CR
      OBJF1=Q
      RETURN
      END
```

<u>Function PQUEUE</u>

Function PQUEUE (<u>p</u>robability of <u>queue</u>) computes the probability that a call will be delayed, given the product of the service time and call rate (AWL) and number of effective cars (EF). It is invoked to obtain the probability of a call being queued before dispatch for one hour of a day in a precinct.

The formula used to compute PQUEUE is given in Appendix B of the User's Manual. If EF is not an integer, the value of PQUEUE is computed for the greatest integer less than EF and for the smallest integer greater than EF, and linear interpolation is used to obtain a function value for EF.

```
      FUNCTION PQUEUE(AWL,EF)

C CALCULATES PROBABILITY THAT A CALL WILL BE DELAYED

      N=EF
      XN=N
      C=0.
      SUM=1.
      T=1.
      IF(N .LT. 2) GO TO 110
      NM1=N-1
      DO 100 I=1,NM1
      C=C+1.
      T=T*AWL/C
0     SUM=SUM+T
0     C=C+1.
      T=T*AWL/C
      X=T/(1.-AWL/XN)
      PQUEUE=X/(SUM+X)
      IF(XN .EQ. EF)RETURN
      SUM=SUM+T
      C=C+1.
      T=T*AWL/C
      X=T/(1.-AWL/C)
      PQ=X/(SUM+X)
      PQUEUE=PQUEUE-(PQUEUE-PQ)*(EF-XN)
      RETURN
      END
```

Function OBJF2

Function OBJF2 (objective function 2) returns the weighted sum of the average time that a call of a specified priority can expect to wait before dispatch. The sum is taken over a span of hours of a day in a precinct. The delay is in hours.

Parameter N specifies the priority level of interest. ISTART and IEND specify the span of hours over which the weighted sum is to be taken. LST and LCR are pointers to the hourly service times and call rates for the day. LFR is a pointer to an array that contains the fraction of calls in each priority class for the shift in which the hours occur. EF is the number of effective cars on duty.

The formula used to compute the expected delay in an hour in a specified priority class is given in Appendix B of the User's Manual. The delay for each hour between ISTART and IEND is weighted by the number of calls in the priority class in the hour.

```
      FUNCTION OBJF2(N,ISTART,IEND,LCR,LST,LFR,EF)

COMPUTE WEIGHTED SUM OF PRIORITY N CALL DELAYS OVER
OURS ISTART TO IEND

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      IF(N .LT. 1) GO TO 50
      CMFRN1=0.
      NM1=N-1
      FRN=CDAT(LFR+NM1)
      IF(NM1 .LT. 1) GO TO 20
      DO 10 I=1,NM1
      CMFRN1=CMFRN1+CDAT(LFR+I-1)
      CMFRN=CMFRN1+FRN
      W=0.
      DO 30 I=ISTART,IEND
      CR=CDAT(LCR+I-1)
      ST=CDAT(LST+I-1)
      AWL=CR*ST
      ENMU=EF/ST
      Q=PQUEUE(AWL,EF)
      W=W+CR*FRN*Q/(ENMU*(1.-CR*CMFRN/ENMU)*(1.-CR*CMFRN1/ENMU))
      OBJF2=W
      RETURN

           COMPUTE AVERAGE DISPATCH DELAY FOR ALL CALLS

      W=0.
      DO 60 I=ISTART,IEND
      CR=CDAT(LCR+I-1)
      ST=CDAT(LST+I-1)
      AWL=CR*ST
      Q=PQUEUE(AWL,EF)
      W=W+Q*CR/(EF/ST-CR)
      OBJF2=W
      RETURN
      END
```

Function OBJF3

Function OBJF3 (objective function 3) computes the weighted total delay (queuing + travel time) that a randomly selected call can expect to experience before a patrol car arrives, summed over a span of hours of a day in a precinct.

Parameters ISTART and IEND specify the span of hours over which the weighted sum is to be taken. LPCT and LDAY are pointers to the data for the precinct and day. RV is the response speed of patrol cars and EF is the number of effective cars on duty.

The formulas for computing travel time and expected delay in an hour are given in Appendix B of the User's Manual. The queuing delays and travel times are weighted by the number of calls in each hour.

```
      FUNCTION OBJF3(ISTART,IEND,LPCT,LDAY,RV,EF)

COMPUTES WEIGHTED SUM OF TOTAL RESPONSE TIMES

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      TD=0.
      W=0.
      LCR=LDAY+CRDOFF-1
      LST=LDAY+STDOFF-1
      A=CDAT(LPCT+ARPOFF)
      SQRTA=SQRT(A)
      DO 40 I=ISTART,IEND
      CR=CDAT(LCR+I)
      ST=CDAT(LST+I)
      AWL=CR*ST
      AVAVL=EF-AWL
      IF(AVAVL .GE. 1.) GO TO 10
      TD=.678*SQRTA*CR+TD
      GO TO 40
      IF(AVAVL .GE. 2.) GO TO 20
      TD=TD+SQRTA*(.08+.598/SQRT(AVAVL))*CR
      GO TO 40
      TD=(.711*SQRTA/SQRT(AVAVL))*CR+TD
      W=W+(PQUEUE(AWL,EF)/(EF/ST-CR))*CR
      STRDNS=CDAT(LPCT+SMPOFF)/A
      G=(STRDNS-1.)/(STRDNS-2.)
      TT=G*TD/RV
      OBJF3=W+TT
      RETURN
      END
```

Subroutine SET

Subroutine SET implements the SET command. Its function is to alter the values of specified data items that have been read from the data base.

Successive calls to subroutine SCAN get pointers to lists of numbers that specify the types of data items to be altered and the values they are to assume. If the lists constitute a valid specification, subroutine GTDSPC is called to scan the command qualifier and SETWFL is called to set the "work" flags for the days and tours in the scope of the command.

In the main processing loop of SET, the program indexes through all selected precincts. For each precinct, the progam indexes through all data item-value pairs specified by the user. If a data item applies to precincts as a whole (unavailability parameters are of this type), then the value of the data item for the precinct is changed to the specified value and the next data item-value pair is examined. If a data item applies to days within precincts (e.g., call-rate and service-time parameters) or to tours within days (e.g., actual cars assigned and response speed), then SET indexes through all selected days. If the data item applies to days as a whole, then the change is made for each day in turn. If the data item applies to tours, then the change is made to all selected tours within the day. If the user specifies that the number of suppressible crimes for a tour is to be changed, then proportional changes are made in the number of suppressible crimes in each block of the tour.

When all changes have been applied to all days for a precinct, subroutine DERIVE is called for each day to compute average workloads and effective cars for each block, and to insure that the resulting number of effective cars on duty in each block of each day is sufficient to handle the cfs workload.

```
      SUBROUTINE SET

 IMPLEMENTS THE SET COMMAND

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      DIMENSION ORDER(3),VAL(2)
      INTEGER TYPE,VAL
      LGETT=TOP
      TYPE=CMD
      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. FSPEC) GO TO 20
      WRITE(SYSOUT,1)
      FORMAT(/' *** INVALID PARAMETER SPECIFICATION - REENTER')
      TOP=LGETT
      RETURN
      KEYVAL=VAL(1)
      I=KEYVAL-TYPOFF(FSPEC)
      IF(I .NE. 1) GO TO 10

           GET DATA TYPES, CHECK VALIDITY

      CALL SCAN(TYPE,VAL)
      IF(TYPE .NE. NUMLST) GO TO 10
      NPARM=VAL(1)
```

```
      LPARM=VAL(2)
      DO 25 IPARM=1,NPARM
      I=ICDAT(LPARM+(IPARM-1)*2)
      IF(I .GT. 0 .AND. I .LT. 11) GO TO 25
      WRITE(SYSOUT,4) I
      FORMAT(/' *** PARAMETER ''',I2,''' INVALID - REENTER.')
      TOP=LGETT
      RETURN
      CONTINUE

            GET DATA VALUES

      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. NUMLST) GO TO 30
      WRITE(SYSOUT,2)
      FORMAT(/' *** INVALID PARAMETER VALUE - REENTER.')
      TOP=LGETT
      RETURN
      NVAL=VAL(1)
      LVAL=VAL(2)
      IF(NVAL .EQ. NPARM) GO TO 40
      WRITE(SYSOUT,3)
      FORMAT(/' *** NUMBER OF VALUES DOES NOT MATCH NUMBER OF PARMS'
     1,' - REENTER')
      TOP=LGETT
      RETURN

            SCAN QUALIFIER

      CALL SCAN(TYPE,VAL)
      CALL GTDSPC(TYPE,VAL,ORDER)
      IF(TYPE .NE. ERR) GO TO 50
      TOP=LGETT
      RETURN

            SET WORK FLAGS

      CALL SETWFL(IERR)
      IF(IERR .EQ. 0) GO TO 55
      TOP=LGETT
      RETURN
      LPCT=0

            GET NEXT PRECINCT

      LPCT=NXPCT(LPCT)
      IF(LPCT .EQ. 0) GO TO 140

            LOOK AT DATA TYPES

      DO 130 IPARM=1,NPARM
      NP=ICDAT(LPARM+(IPARM-1)*2)
      IF(NP .GT. 2) GO TO 70
      CDAT(LPCT+SMPOFF+NP)=CDAT(LVAL+(IPARM-1)*2+1)
      GO TO 130

            DAY-SPECIFIC DATA
```

```
70      LDAY=0
75      LDAY=NXDAY(LPCT,LDAY)
        IF(LDAY .EQ. 0) GO TO 130
        IF(NP .GT. 4) GO TO 90
        N=NP-3
        XPARM=CDAT(LVAL+(IPARM-1)*2+1)
        RATIO=XPARM/CDAT(LDAY+N)
        CDAT(LDAY+N)=XPARM
        L1=LDAY+CRDOFF
        IF(N.EQ.1) L1=LDAY+STDOFF
        L2=L1+23
        DO 80 L=L1,L2
0       CDAT(L)=CDAT(L)*RATIO
        GO TO 75

              TOUR-SPECIFIC DATA

0       LTOUR=0
5       LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
        IF(LTOUR .EQ. 0) GO TO 75
7       IF(NP .EQ. 10) GO TO 100
        N=NP-5
        CDAT(LTOUR+N+ACTOFF)=CDAT(LVAL+(IPARM-1)*2+1)
        GO TO 95
00      IBL1=ICDAT(LTRTB(1)+ITYPE-1)
        IBL2=ICDAT(LTRTB(2)+ITYPE-1)
        IBL1=ICDAT(LBLRFL+IBL1-1)
        IF(IBL2 .NE. 0) IBL2=ICDAT(LBLRFL+IBL2-1)
        LBLK1=LDAY+BLDOFF+(IBL1-1)*NWDBL
        IF(IBL2 .NE. 0) LBLK2=LDAY+BLDOFF+(IBL2-1)*NWDBL
        IF(IBL2 .NE. 0) CRM=CRM+CDAT(LBLK2+OCBOFF)
        XPARM=CDAT(LVAL+(IPARM-1)*2+1)
        RATIO=XPARM/CRM
        CDAT(LBLK1+OCBOFF)=CDAT(LBLK1+OCBOFF)*RATIO
        IF(IBL2 .NE. 0) CDAT(LBLK2+OCBOFF)=CDAT(LBLK2+OCBOFF)*RATIO
        GO TO 95
0       CONTINUE

              RE-DERIVE BLOCK VALUES FOR EACH DAY AND CHECK FOR MINIMUM

        DO 135 IDAY=1,NDAYRD
        LDAY=LPCT+DYPOFF+(IDAY-1)*NWDDY
5       CALL DERIVE(LPCT,LDAY)
        GO TO 60
0       TOP=LGETT
        RETURN
        END
```

Subroutine MEET

Subroutine MEET implements the MEET command. Its function is to assign enough cars to all shifts within its scope so that a user-specified set of constraints on selected performance measures is met.

At entry, successive calls to subroutine SCAN get pointers to the list of output measures (LPARM) and the list of constraint values (LVAL). If all of the output measure specifications are valid and the number of constraint values matches the number of output measure specifications, GTDSPC is called to scan the MEET command qualifier. Subroutine SETWFL sets the "work" flags for days and tours and subroutine CKOVR (check overlay) insures that if an overlay tour has been specified in the qualifier, then the overlaid tours have also been specified.

MEET then indexes through all selected precincts, days, and tours. The blocks of each shift thus selected are dealt with independently. If a block has not been within the scope of a previous MEET, ADD, or ALOC command since the last READ command (IDATA(LBLK+CTBOFF) less than 0), enough cars are assigned to the block to keep the utilization of an effective car under 1 in each of its hours.

Starting with either the current assignment or the minimum assignment, the number of cars in a block is increased as necessary to meet each specified constraint in turn. Function KNSTR determines whether or not a particular constraint has been met by a given number of effective cars (KNSTR is not used for minimum manning level--constraint 6).

When constraints have been met for all blocks of all tours of a day, subroutine STRCAR (set tour cars) is called to obtain a feasible allocation of cars to the tours of the day that will result in the required number of cars in each block (see Chapter III). Then SBLACT (set block actual cars) is called to convert this tour allocation to a block allocation (see Chapter III) and SBLEF (set block effective cars) is called to determine the resulting number of effective cars in each block.

MEET returns when all constraints have been met for all blocks of all shifts within the scope of the command.

```
      SUBROUTINE MEET

 DETERMINES CAR REQUIREMENTS TO MEET SPECIFIED
CONSTRAINTS.

      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)

      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL

      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL

      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF

      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT

      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))

      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR

      INTEGER TYPE,VAL
      DIMENSION VAL(2)
      LGETT=TOP
      TYPE=CMD

            GET CONSTRAINT SPECIFICATIONS, CHECK VALIDITY

      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. FSPEC) GO TO 20
      WRITE(SYSOUT,1)
      FORMAT(/' *** INVALID CONSTRAINT SPECIFICATION - REENTER')
      TOP=LGETT
      RETURN
      KEYVAL=VAL(1)
      I=KEYVAL-TYPOFF(FSPEC)
      IF(I .NE. 2) GO TO 10
      CALL SCAN(TYPE,VAL)
      IF(TYPE .NE. NUMLST) GO TO 10
```

```
      NPARM=VAL(1)
      LPARM=VAL(2)
      DO 25 IPARM=1,NPARM
      I=ICDAT(LPARM+(IPARM-1)*2)
      IF(I .GT. 0 .AND. I .LT. 11) GO TO 25
      WRITE(SYSOUT,2) I
      FORMAT(/' *** INVALID CONSTRAINT NUMBER : ',I4,' - REENTER')
      TOP=LGETT
      RETURN
      CONTINUE

             GET CONSTRAINT VALUES

      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. NUMLST) GO TO 30
      WRITE(SYSOUT,3)
      FORMAT(/' *** INVALID CONSTRAINT VALJE(S) - REENTER')
      TOP=LGETT
      RETURN
      NVAL=VAL(1)
      LVAL=VAL(2)
      IF(NVAL .EQ. NPARM) GO TO 40
      WRITE(SYSOUT,4)
      FORMAT(/' *** NUMBER OF VALUES DOES NOT MATCH NUMBER OF CONS',
     1 'TRAINTS - REENTER')
      TOP=LGETT
      RETURN

             SCAN QUALIFIER

      CALL SCAN(TYPE,VAL)
      CALL GTDSPC(TYPE,VAL,ORDER)
      IF(TYPE .NE. ERR) GO TO 50
      TOP=LGETT
      RETURN

             SET WORK FLAGS

      CALL SETWFL(IERR)
      IF(IERR .EQ. 0) GO TO 55
      TOP=LGETT
      RETURN

             INSURE THAT OVERLAY SEGMENT IS COMPLETE OR NOT INCLUDED

      CALL CKOVR(IERR)
      IF(IERR .EQ. 0) GO TO 60
      TOP=LGETT
      RETURN
      LPCT=0
      NTOT=0
      LPCT=NXPCT(LPCT)
      IF(LPCT .NE. 0) GO TO 110
      WRITE(SYSOUT,6) NTOT
      FORMAT(/'  ',I4,' CAR HOURS ALLOCATED.')
      TOP=LGETT
      RETURN
      B1=CDAT(LPCT+B1POFF)
```

```
      B2=CDAT(LPCT+B2POFF)
      LDAY=0
120   LDAY =NXDAY(LPCT,LDAY)
      IF(LDAY .EQ. 0) GO TO 100
130   LTOUR=0
140   LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .NE. 0 .AND. ICDAT(LTOUR+TYTOFF) .NE. 5) GO TO 160
C
C           ASSIGN CARS TO THE TOURS OF A DAY SO THAT BLOCK
C           REQUIREMENTS ARE MET
C
      CALL STRCAR(LDAY,CARHRS)
      NTOT=NTOT+CARHRS
C
C           DETERMINE BLOCK ASSIGNMENTS FROM TOUR ASSIGNMENTS
C
      CALL SBLACT(LPCT,LDAY)
      CALL SBLEF(LPCT,LDAY)
      GO TO 120
160   DO 220 IBLK=1,2
      IBLD=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBLD .EQ. 0) GO TO 220
      IBLR=ICDAT(LBLRFL+IBLD-1)
      LBLK=LDAY+BLDOFF+(IBLR-1)*NWDBL
      AWL=CDAT(LBLK+AWBOFF)
      IF(ICDAT(LBLK+CTBOFF) .LT. 0) GO TO 165
      EF=CDAT(LBLK+EFBOFF)
      ACT=CDAT(LBLK+ACBOFF)
      GO TO 170
C
C           DETERMINE MINIMUM BLOCK REQUIREMENTS
C
165   EF= INT(CDAT(LBLK+RMBOFF)+1.)
      ACT=CEIL((EF+B1*AWL)/(1.-B2))
      CDAT(LBLK+ACBOFF)=ACT
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      CDAT(LBLK+EFBOFF)=EF
      ICDAT(LBLK+CTBOFF)=0
C
C           INSURE THAT ALL CONSTRAINTS ARE MET FOR EACH BLOCK
C
170   DO 200 IPARM=1,NPARM
      IP=ICDAT(LPARM+(IPARM-1)*2)
      CVAL=CDAT(LVAL+(IPARM-1)*2+1)
      IF(IP .NE. 6) GO TO 180
      ACT=CVAL
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      GO TO 190
180   I=KNSTR(IP,CVAL,EF,LPCT,LDAY,LTOUR,LBLK,IBLD)
      IF(I .NE. 0) GO TO 190
      ACT=ACT+1.
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      GO TO 180
190   IF(ACT .LE. CDAT(LBLK+ACBOFF)) GO TO 200
      CDAT(LBLK+ACBOFF)=ACT
      CDAT(LBLK+EFBOFF)=EF
      ICDAT(LBLK+CTBOFF)=IP
200   CONTINUE
```

```
220     CONTINUE
        GO TO 140
        END
```

Subroutine STRCAR

Subroutine STRCAR (set tour cars) determines a feasible allocation of cars to the tours of a day so that the resulting number of cars in each block of the day will be at least as great as the number currently assigned. The number of cars currently assigned to each block of the day is the number required to meet some constraint and is set by MEET or ADDALC. Parameter LDAY is a pointer to the data for the day for which the tour assignment is to be determined. CARHRS, on return, is the total number of car hours that have been assigned to all tours of the day. The algorithm used to generate the assignment of cars to tours is given in Chapter III.

```
      SUBROUTINE STRCAR(LDAY,CARHRS)
C
C  DETERMINES FEASIBLE ALLOCATION OF CARS TO TOURS IN A
C DAY, GIVEN THE CAR REQUIREMENTS IN THE BLOCKS OF A DAY.
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      DIMENSION LBLK(2)
C
      CARHRS=0.
      LTOUR=0
5     LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) RETURN
      ISTART=ICDAT(LTRST+ITYPE-1)
      IEND=ICDAT(LTREND+ITYPE-1)
      TOURLN=IEND-ISTART+1
C
C           GET POINTERS TO BLOCKS
C
      DO 10 IB=1,2
      LBLK(IB)=0
      IBLK=ICDAT(LTRTB(IB)+ITYPE-1)
      IF(IBLK .EQ. 0) GO TO 10
      IBLK=ICDAT(LBLRFL+IBLK-1)
      LBLK(IB)=LDAY+BLDOFF+(IBLK-1)*NWDBL
10     CONTINUE
      ID=ICDAT(LTOUR+TYTOFF)
      GO TO (100,20,30,40,50),ID
C
C           TOUR NOT IN OVERLAY SEGMENT
C
20    I=1
      IF(LBLK(2) .EQ. 0) GO TO 25
      IF(CDAT(LBLK(1)+ACBOFF) .LT. CDAT(LBLK(2)+ACBOFF)) I=2
25    CDAT(LTOUR+ACTOFF)=CDAT(LBLK(I)+ACBOFF)
      CARHRS=CARHRS+TOURLN*CDAT(LTOUR+ACTOFF)
      ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(I)+CTBOFF)
```

```
      GO TO 5
C
C          FIRST OVERLAID TOUR
C
30    X1=CDAT(LBLK(1)+ACBOFF)
      CDAT(LTOUR+ACTOFF)=X1
      CARHRS=CARHRS+TOURLN*X1
      ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(1)+CTBOFF)
      X2=CDAT(LBLK(2)+ACBOFF)
      GO TO 5
C
C          SECOND OVERLAID TOUR
C
40    X3=CDAT(LBLK(1)+ACBOFF)
      X4=CDAT(LBLK(2)+ACBOFF)
      CDAT(LTOUR+ACTOFF)=X4
      CARHRS=CARHRS+X4*TOURLN
      ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(2)+CTBOFF)
      GO TO 5
C
C          OVERLAY TOUR
C
50    CDAT(LTOUR+ACTOFF)=AMAX1(X2-X1,X3-X4,0.)
      CARHRS=CARHRS+CDAT(LTOUR+ACTOFF)*TOURLN
      I=1
      IF(X3-X4 .GT. X2-X1) I=2
      ICDAT(LTOUR+CTTOFF)=ICDAT(LBLK(I)+CTBOFF)
      IF(X2-X1 .LE. 0. .AND. X3-X4 .LE. 0.) ICDAT(LTOUR+CTTOFF)=0
C
C          ADJUST OVERLAY SEGMENT ASSIGNMENTS IF THE NUMBER OF CAR
C          HOURS USED CAN BE REDUCED
C
      LOVTR=LTOUR
      ENOV=CDAT(LOVTR+ACTOFF)
      IF(ENOV .EQ. 0.) RETURN
      DELTA=AMAX1(X2-X1,0.)-AMAX1(X3-X4,0.)
      IF(DELTA .EQ. 0.) RETURN
      ISW=1
      IF(DELTA .LT. 0.) ISW=2
      DELTA=ABS(DELTA)
      ITRRD=IOVTR(ISW)
      ITYPE=ICDAT(LTRWFL+ITRRD-1)
      ILEN=ICDAT(LTREND+ITYPE-1)-ICDAT(LTRST+ITYPE-1)+1
      IOVLN=ICDAT(LTREND+NTRDT-1)-ICDAT(LTRST+NTRDT-1)+1
      IF(ILEN .GE. IOVLN) RETURN
      LTOUR=LDAY+TRDOFF+(ITRRD-1)*NWDTR
      LPCT=((LDAY-LPCTDT)/NWDPCT)*NWDPCT+LPCTDT
      B1=CDAT(LPCT+B1POFF)
      B2=CDAT(LPCT+B2POFF)
      CDAT(LOVTR+ACTOFF)=CDAT(LOVTR+ACTOFF)-DELTA
      CDAT(LTOUR+ACTOFF)=CDAT(LTOUR+ACTOFF)+DELTA
      CARHRS=CARHRS-DELTA*(IOVLN-ILEN)
      IBDT=ICDAT(LTRTB(ISW)+ITYPE-1)
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLOCK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      ACT=CDAT(LBLOCK+ACBOFF)+DELTA
      CDAT(LBLOCK+ACBOFF)=ACT
      AWL=CDAT(LBLOCK+AWBOFF)
```

```
      CDAT(LBLOCK+EFBOFF)=ACT*(1.-((B1*AWL/ACT)+B2))
      INV=2/ISW
      IBDT=ICDAT(LTRTB(INV)+NTRDT-1)
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLOCK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      ACT=CDAT(LBLOCK+ACBOFF)-DELTA
      CDAT(LBLOCK+ACBOFF)=ACT
      AWL=CDAT(LBLOCK+AWBOFF)
      CDAT(LBLOCK+EFBOFF)=ACT*(1.-((B1*AWL/ACT)+B2))
100   RETURN
      END
```

Function KNSTR

Function KNSTR determines whether a given number of effective cars on duty in a block of a day results in a specified constraint on an output measure being met. A function value of one (1) is returned if the constraint is met, otherwise a value of zero (0) is returned. Parameter ICNSTR specifies the output measure whose value, with EF effective cars, is to be tested against constraint value CVAL. The valid values of ICNSTR are the output measure specifications given in the MEET command description in Chapter III of the User's Manual. LPCT, LDAY, LTOUR, and LBLK are pointers to the data for the precinct, day, tour, and block for which the output measure is to be tested. IBLD is the relative position of the block among all blocks in the data base (e.g., the *third* block of a day).

The output measure specified by ICNSTR is evaluated for the block, given EF effective cars. Some output measures are computed directly from available data; others are computed by function references. The resulting measure is tested against CVAL and the value of KNSTR set according to the outcome.

```
      FUNCTION KNSTR(ICNSTR,CVAL,EF,LPCT,LDAY,LTOUR,LBLK,IBLD)
C
C  DETERMINES WHETHER CONSTRAINT CVAL ON PERFORMANCE MEASURE
C  ICNSTR IS MET BY EF EFFECTIVE CARS
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      KNSTR=0
      ISTART=ICDAT(LBLKTB(1)+IBLD-1)
      IEND=ICDAT(LBLKTB(2)+IBLD-1)
      LCR=LDAY+CRDOFF
      LST=LDAY+STDOFF
      BLKLN=IEND-ISTART+1
      GO TO (100,200,300,400,500,600,700,800,900,1000),ICNSTR
C
100   X=CDAT(LBLK+AWBOFF)/EF
      IF(X .LE. CVAL) KNSTR=1
      RETURN
C
200   RV=CDAT(LTOUR+RVTOFF)
      X=AVTT(ISTART,IEND,LPCT,LDAY,RV,EF)
      IF( X .LE. CVAL) KNSTR=1
      RETURN
300   X=EF-CDAT(LBLK+AWBOFF)
      IF(X .GE. CVAL) KNSTR=1
      RETURN
400   X=(EF-CDAT(LBLK+AWBOFF))*BLKLN/CDAT(LBLK+OCBOFF)
      IF(X .GE. CVAL) KNSTR=1
      RETURN
500   X=(EF-CDAT(LBLK+AWBOFF))*CDAT(LTOUR+PVTOFF)/CDAT(LPCT+SMPOFF)
      IF( X .GE. CVAL) KNSTR=1
      RETURN
600   RETURN
700   X=OBJF1(ISTART,IEND,LCR,LST,EF)/CDAT(LBLK+CRBOFF)
      IF(X .LE. CVAL) KNSTR=1
      RETURN
800   N=2
```

```
      GO TO 910
900   N=3
910   LFR=LTOUR+HFTOFF
      X=60.*OBJF2(N,ISTART,IEND,LCR,LST,LFR,EF)/
     1 (CDAT(LBLK+CRBOFF)*CDAT(LFR+N-1))
      IF(X .LE. CVAL) KNSTR=1
      RETURN
1000  RV=CDAT(LTOUR+RVTOFF)
      X=60.*OBJF3(ISTART,IEND,LPCT,LDAY,RV,EF)/CDAT(LBLK+CRBOFF)
      IF(X .LE. CVAL) KNSTR=1
      RETURN
      END
```

Subroutine CKOVR

Subroutine CKOVR (check overlay) is used to insure that if the user has selected an overlay tour in a command qualifier, then he has also selected the overlaid tours. Its parameter IERR is set to zero (0) on return if a valid specification has been made, otherwise it is set to one (1). The determination of validity is based on the "work" flags of the tours involved. See the section on table pointers in Chapter IV for a description of the flags and tables involved.

```
      SUBROUTINE CKOVR(IERR)
C
C CHECKS TO INSURE THAT ALL TOURS IN AN OVERLAY SEGMENT HAVE BEEN
C SELECTED IN A COMMAND OR THAT THEY HAVE ALL BEEN OMITTED
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      IERR=0
      IOV=ICDAT(LTRRFL+NTRDT-1)
      IF(IOVRLY .EQ. 0 .OR. IOV .EQ. 0) RETURN
      IOV=ICDAT(LTRWFL+IOV-1)
      IOV1=ICDAT(LTRWFL+IOVTR(1)-1)
      IOV2=ICDAT(LTRWFL+IOVTR(2)-1)
      IF(IOV+IOV1+IOV2 .EQ. 0) RETURN
      IF(IOV .NE. 0 .AND. IOV1 .NE. 0 .AND. IOV2 .NE. 0) RETURN
      WRITE(SYSOUT,1)
1     FORMAT(/' *** INVALID OVERLAY TOUR SPECIFICATION - REENTER.')
      IERR=1
      RETURN
      END
```

Subroutine ADDALC

Subroutine ADDALC (add and allocate) implements the ADD and ALLOCATE commands. Its parameter ISW determines which command is to be executed. If ISW is less than 2, then the ALOC command is executed; otherwise, the ADD command is executed.

Successive calls to subroutine SCAN get the user's specification of the number of car-hours; subroutine GTDSPC scans the command qualifier; and additional calls to SCAN get the user's objective function specification. The user's specification of the number of car-hours is saved as follows: NHOURS holds the numeric part of any specification; ISTAR is 1, 0, or -1, depending upon whether the user's expression is of the form *-n, n, or n-* (* alone is equivalent to *-0).

If an asterisk appears in the expression giving the number of car-hours, the number of car-hours currently allocated to all selected shifts is determined and the expression is evaluated to give a number of car-hours to be allocated or added.

The program then indexes through all selected precincts and days. If an ALOC command is being executed, each block of each selected tour of each day is assigned just enough cars to handle its cfs workload and subroutines STRCAR, SBLACT, and SBLEF are called to get a feasible allocation of cars to tours and to translate the tour allocation back to a block allocation; this step is skipped for ADD commands. The objective function is evaluated for each selected block of a day via a call to SBLOBJ and for each selected tour of a day via a call to STROBJ. The constraint indicators for each block of selected tours are set to zero. Subroutine ADJUST is called for ALOC commands to insure that the initial allocation results in the minimum objective function value for the number of car-hours assigned in each day.

After the objective function has been evaluated for all shifts, the number of car-hours that remain to be allocated is computed and subroutine ADDCAR is called to allocate that number of car-hours.

```
      SUBROUTINE ADDALC(ISW)
C
C  PERFORMS ADD OR ALLOCATE FUNCTION, DEPENDING ON
C THE VALUE OF 'ISW'.
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))
C
      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR
C
      DIMENSION VAL(2),ORDER(3),EN(4)
      INTEGER TYPE,VAL
C
      INTEGER CHARST,CHARMN
      DATA CHARST/1H*/,CHARMN/1H-/
C
      LGETT=TOP
      TYPE=CMD
      NHOURS=0
      ISTAR=0
C
C          GET EXPRESSION FOR CAR HOURS TO ALLOCATE
C
      CALL SCAN(TYPE,VAL)
      IF(TYPE .EQ. NUMLST) GO TO 20
      IF(TYPE .EQ. NAMLST) GO TO 15
10    WRITE(SYSOUT,1)
1     FORMAT(/' *** INVALID NUMBER OF CAR HOURS TO ALLOCATE - ',
```

```
     1 'REENTER')
      TOP=LGETT
      RETURN
15    IF(ICDAT(VAL(2)) .NE. CHARST) GO TO 10
      ISTAR=1
      CALL SCAN(TYPE,VAL)
      IF(TYPE .NE. NUMLST) GO TO 25
      NHOURS=ICDAT(VAL(2))
      IF(NHOURS .GT. 0) GO TO 10
      CALL SCAN(TYPE,VAL)
      GO TO 25
20    NHOURS=ICDAT(VAL(2))
      CALL SCAN(TYPE,VAL)
      IF(TYPE .NE. NAMLST) GO TO 25
      IF(ICDAT(VAL(2)) .NE. CHARMN .OR. ICDAT(VAL(2)+1) .NE. CHARST)
     1 GO TO 10
      ISTAR=-1
      CALL SCAN(TYPE,VAL)
C
C           SCAN QUALIFIER
C
25    CALL GTDSPC(TYPE,VAL,ORDER)
      IF(TYPE .NE. ERR) GO TO 30
      TOP=LGETT
      RETURN
C
C           SCAN AND VALIDATE OBJECTIVE FUNCTION SPECIFICATION
C
30    IF(TYPE .EQ. FSPEC) GO TO 60
50    WRITE(SYSOUT,2)
2     FORMAT(/' ***INVALID OBJECTIVE FUNCTION - REENTER')
      TOP=LGETT
      RETURN
60    KEYOFF=VAL(1)
      I=KEYOFF-TYPOFF(FSPEC)
      IF(I .NE. 4) GO TO 50
      CALL SCAN(TYPE,VAL)
      IF(TYPE .NE. NUMLST) GO TO 50
      NPARM=VAL(1)
      LPARM=VAL(2)
      IFNCTN=ICDAT(LPARM)
      IF(IFNCTN .LT. 1 .OR. IFNCTN .GT. 3) GO TO 50
      IF(IFNCTN .NE. 2) GO TO 130
      IF(NPARM .GT. 1) GO TO 70
      CALL GETTOP(4,LPARM)
      ICDAT(LPARM)=2
      ICDAT(LPARM+2)=0
      GO TO 130
70    IPRIO=ICDAT(LPARM+2)
      IF(IPRIO .GE. 0 .AND. IPRIO .LE. NPRIO) GO TO 130
      WRITE(SYSOUT,3)
3    FORMAT(/' ***INVALID OBJECTIVE FUNCTION PARAMETER(S) - REENTER')
      TOP=LGETT
      RETURN
C
C           SET WORK FLAGS
C
130   CALL SETWFL(IERR)
```

```
      IF(IERR .EQ. 0) GO TO 132
      TOP=LGETT
      RETURN
C
C           CHECK OVERLAY SPECIFICATION
C
132   CALL CKOVR(IERR)
      IF(IERR .EQ. 0) GO TO 135
      TOP=LGETT
      RETURN
135   NCRHRS=0
C
C ** TO ALLOW THE USER TO SPECIFY CARS TO ALLOCATE INSTEAD OF CAR HOURS
C ** A STATEMENT SHOULD BE ADDED HERE TO MULTIPLY 'NHOURS' BY THE LENGTH
C ** OF A TOUR; E.G. INSERT THE STATEMENT NHOURS=NHOURS*8 IF ALL TOURS
C ** ARE EIGHT HOURS IN LENGTH AND CARS (INSTEAD OF CAR HOURS)
C ** ARE TO BE ALLOCATED.
C
      IF(ISTAR .EQ. 0) GO TO 180
C
C           FIND NUM OF CAR HOURS ALREADY ASSIGNED TO SELECTED SHIFTS
C
      CRHRS=NCRHRS
      LPCT=0
140   LPCT=NXPCT(LPCT)
      IF(LPCT .EQ. 0) GO TO 170
      LDAY=0
150   LDAY=NXDAY(LPCT,LDAY)
      IF(LDAY .EQ. 0) GO TO 140
      LTOUR=0
160   LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) GO TO 150
      TOURLN=ICDAT(LTREND+ITYPE-1)-ICDAT(LTRST+ITYPE-1)+1
      CRHRS=CRHRS+CDAT(LTOUR+ACTOFF)*TOURLN
      GO TO 160
C
C            COMPUTE CAR HOURS AVAILABLE TO ALLOCATE
C
170   NCRHRS=INT(CRHRS+.5)
      NHOURS=NHOURS+ISTAR*NCRHRS
C
C            DETERMINE INITIAL ASSIGNMENT (ALOC ONLY) AND EVALUATE
C            CORRESPONDING OBJECTIVE FUNCTION VALUES FOR ALL SHIFTS
C
180   NTOT=0
      LPCT=0
200   LPCT=NXPCT(LPCT)
      IF(LPCT .EQ. 0) GO TO 300
      B1=CDAT(LPCT+B1POFF)
      B2=CDAT(LPCT+B2POFF)
      LDAY=0
210   LDAY=NXDAY(LPCT,LDAY)
      IF(LDAY .EQ. 0) GO TO 200
      IF(ISW .GT. 1) GO TO 245
C
C            FIND MINIMUM ASSIGNMENT FOR EACH BLOCK
C
      IBL=0
```

```
      LTOUR=0
220   LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) GO TO 240
      IND=ICDAT(LTOUR+TYTOFF)
      IF(IND .EQ. 5) GO TO 240
      DO 230 IBLK=1,2
      IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBDT .LT. 1) GO TO 230
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      AWL=CDAT(LBLK+AWBOFF)
      EF= INT(CDAT(LBLK+RMBOFF)+1.)
      ACT=CEIL((EF+B1*AWL)/(1.-B2))
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      CDAT(LBLK+ACBOFF)=ACT
      CDAT(LBLK+EFBOFF)=EF
      IF(IND .NE. 3 .AND. IND .NE. 4) GO TO 230
      IBL=IBL+1
      EN(IBL)=ACT
230   CONTINUE
      GO TO 220
C
C           FIND MINIMUM FEASIBLE TOUR ASSIGNMENT
C
240   CALL STRCAR(LDAY,CARHRS)
      NTOT=NTOT+CARHRS
      CALL SBLACT(LPCT,LDAY)
      CALL SBLEF(LPCT,LDAY)
245   LTOUR=0
250   LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .NE. 0) GO TO 255
      IF(IOVRLY .EQ. 0 .OR. ICDAT(LDAY+OVDOFF) .EQ. 0 .OR. ISW .GT. 1)
     C GO TO 210
C
C           ADJUST INITIAL TOUR ASSIGNMENT TO MINIMIZE
C           OBJECTIVE FUNCTION
C
      X1=AMAX1(EN(2)-EN(1),0.)
      X2=AMAX1(EN(3)-EN(4),0.)
      DELTA=X1-X2
      CALL ADJUST(LPARM,LPCT,LDAY,DELTA)
      GO TO 210
255   ICDAT(LTOUR+CTTOFF)=0
C
C           COMPUTE INITIAL OBJECTIVE FUNCTION VALUES FOR BLOCKS
C
      DO 260 IBLK=1,2
      IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBDT .LT. 1) GO TO 260
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      ICDAT(LBLK+CTBOFF)=0
      IF(ICDAT(LTOUR+TYTOFF) .EQ. 5) GO TO 260
      CDAT(LBLK+ACBOFF)=CDAT(LBLK+ACBOFF)-2.
      CALL SBLOBJ(LPARM,LPCT,LDAY,LTOUR,LBLK,IBDT)
      CALL SBLOBJ(LPARM,LPCT,LDAY,LTOUR,LBLK,IBDT)
260   CONTINUE
      CALL STROBJ(LDAY,LTOUR,ITYPE)
```

```
      GO TO 250
C
C           ALLOCATE REMAINING CAR HOURS
C
300   IF(ISW .LT. 2) GO TO 305
      NLEFT=NHOURS
      IF(NLEFT .GT. 0) GO TO 310
      RETURN
305   NLEFT=NHOURS-NTOT
      IF(NLEFT .GE. 0) GO TO 310
      WRITE(SYSOUT,4) NTOT
4     FORMAT(/' *** ',I5,' CAR HOURS ALLOCATED.')
      TOP=LGETT
      RETURN
310   CALL ADDCAR(NLEFT,LPARM)
      TOP=LGETT
      RETURN
      END
```

Subroutine SBLOBJ

Subroutine SBLOBJ increases the number of cars assigned to a block by one and evaluates a specified objective function with one car more than the new assignment. Both the new current objective function value and the objective function value with an additional car are saved.

LPARM is a pointer to the parameter list that specifies the objective function to be evaluated. LPCT, LDAY, LTOUR, and LBLK are pointers to the precinct, day, tour, and block to be operated upon. IBDT is the relative position of the type of block among blocks in the data base.

```
      SUBROUTINE SBLOBJ(LPARM,LPCT,LDAY,LTOUR,LBLK,IBDT)
C
C DETERMINES THE OBJECTIVE FUNCTION FOR A BLOCK WITH ONE
C MORE ACTUAL CAR THAN CURRENTLY ALLOCATED.
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      CDAT(LBLK+QOBOFF)=CDAT(LBLK+QNBOFF)
      B1=CDAT(LPCT+B1POFF)
      B2=CDAT(LPCT+B2POFF)
      ISTART=ICDAT(LBLKTB(1)+IBDT-1)
      IEND=ICDAT(LBLKTB(2)+IBDT-1)
      ACT=CDAT(LBLK+ACBOFF)+1.
      CDAT(LBLK+ACBOFF)=ACT
      AWL=CDAT(LBLK+AWBOFF)
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      CDAT(LBLK+EFBOFF)=EF
      ACT=ACT+1.
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      CDAT(LBLK+QNBOFF)=OBJFUN(LPARM,ISTART,IEND,LPCT,LDAY,LTOUR,EF)
      RETURN
      END
```

Subroutine STRDF

Subroutine STRDF determines the change in objective function value per car-hour that would be realized by making an incremental change in the assignment of cars to a shift. LDAY and LTOUR are pointers to the day and shift. ITYPE is the relative position of the tour among all tours in the data base.

For any shift, a difference is computed by summing the contribution to the objective function of its blocks with the current number of cars and with one additional car assigned, and dividing by the length of the tour. For overlay shifts an additional difference is obtained, summing the current and proposed objective function values of the first block of the first overlaid shift and the second block of the second overlaid shift, subtracting the sums, and dividing by the difference between the sum of the lengths of the overlaid tours and the length of the overlay tour.

```
      SUBROUTINE STROF(LDAY,LTOUR,ITYPE)
C
C SUBROUTINE TO DETERMINE THE EFFECT ON THE OBJECTIVE FUNCTION OF
C ADDING A CAR TO A TOUR OR TAKING A CAR AWAY FROM AN OVERLAY TOUR
C AND ADDING A CAR TO EACH OF THE OVERLAID TOURS.
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      ISTART=ICDAT(LTRST+ITYPE-1)
      IEND=ICDAT(LTREND+ITYPE-1)
      TOURLN=IEND-ISTART+1
     CDAT(LTOUR+QDTOFF)=(CDAT(LTOUR+QOTOFF)-CDAT(LTOUR+QNTOFF))/TOURLN
      IF(ICDAT(LTOUR+TYTOFF) .NE. 5) RETURN
      IF(CDAT(LTOUR+ACTOFF) .GE. 1.) GO TO 10
      CDAT(LTOUR+QXTOFF)=-1.
      RETURN
10    TOTLEN=0.
      QOLD=0.
      QNEW=0.
      DO 20 I=1,2
      ITRRD=IOVTR(I)
      ITP=ICDAT(LTRWFL+ITRRD-1)
      IBDT=ICDAT(LTRTB(I)+ITP-1)
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      QOLD=QOLD+CDAT(LBLK+QOBOFF)
      QNEW=QNEW+CDAT(LBLK+QNBOFF)
      ISTART=ICDAT(LTRST+ITP-1)
      IEND=ICDAT(LTREND+ITP-1)
      TOTLEN=TOTLEN+(IEND-ISTART+1)
20    CONTINUE
      CDAT(LTOUR+QXTOFF)=(QOLD-QNEW)/(TOTLEN-TOURLN)
      RETURN
      END
```

Subroutine ADJUST

Subroutine ADJUST insures that the initial assignment of cars to shifts for an ALOC command results in the lowest possible objective function value. LPARM is a pointer to a parameter list that specifies the objective function. LPCT and LDAY are pointers to the data for the precinct and day. The absolute value of XDELT is the maximum number of cars that can be moved from an overlay shift to an overlaid shift to reduce the objective function value.

The sign of XDELT indicates whether cars can be moved to the first overlaid shift (positive) or the second overlaid shift (negative). No cars can be shifted if XDELT is zero or no cars are assigned to the overlay shift. Up to ABS(XDELT) cars are moved from the overlay shift to the appropriate overlaid shift. The process terminates when moving another car would increase the objective function value.

```
      SUBROUTINE ADJUST(LPARM,LPCT,LDAY,XDELT)
C
C SUBROUTINE TO EXAMINE ALTERNATIVE INITIAL ALLOCATIONS FOR THE
C ALOC COMMAND TO FIND THE INITIAL ALLOCATION WITH THE BEST
C OBJECTIVE FUNCTION VALUE
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
C          FIND BLOCKS WHOSE ASSIGNMENTS CAN BE CHANGED
C
      LOVTR=LDAY+TRDOFF+(NTRRD-1)*NWDTR
      ENOV=CDAT(LOVTR+ACTOFF)
      IF(ENOV .LE. 0.) RETURN
      ISW=1
      IF(XDELT .LT. 0.) ISW=2
      ITRRD=IOVTR(ISW)
      ITYPE=ICDAT(LTRWFL+ITRRD-1)
      ILEN=ICDAT(LTREND+ITYPE-1)-ICDAT(LTRST+ITYPE-1)+1
      IOVLN=ICDAT(LTREND+NTRDT-1)-ICDAT(LTRST+NTRDT-1)+1
      IF(ILEN .GT. IOVLN) RETURN
      DELTA=ABS(XDELT)
      IF(DELTA .LT. .9999) RETURN
      INV=2/ISW
      IBDT2=ICDAT(LTRTB(INV)+NTRDT-1)
      IBRD=ICDAT(LBLRFL+IBDT2-1)
      LBLK2=LDAY+BLDOFF+(IBRD-1)*NWDBL
      ISTART=ICDAT(LBLKTB(1)+IBDT2-1)
      IEND=ICDAT(LBLKTB(2)+IBDT2-1)
      IBDT1=IBDT2+2*(-1)**ISW
      IBRD=ICDAT(LBLRFL+IBDT1-1)
      LBLK1=LDAY+BLDOFF+(IBRD-1)*NWDBL
      LTOUR=LDAY+TRDOFF+(ITRRD-1)*NWDTR
      ITRRD=IOVTR(INV)
      LTTOUR=LDAY+TRDOFF+(ITRRD-1)*NWDTR
      ITTYPE=ICDAT(LTRWFL+ITRRD-1)
      B1=CDAT(LPCT+B1POFF)
      B2=CDAT(LPCT+B2POFF)
```

```
      AWL=CDAT(LBLK2+AWBOFF)
C
C           ADJUST BLOCK AND TOUR ASSIGNMENTS TO MINIMIZE OBJECTIVE
C           FUNCTION VALUE
C
10    QOLD=CDAT(LBLK1+QOBOFF)+CDAT(LBLK2+QOBOFF)
      ACT=CDAT(LBLK2+ACBOFF)-1.
      EF=ACT*(1.-((B1*AWL/ACT)+B2))
      QTEST=OBJFUN(LPARM,ISTART,IEND,LPCT,LDAY,LTTOUR,EF)
      QNEW=CDAT(LBLK1+QNBOFF)+QTEST
      IF(QOLD .LT. QNEW) GO TO 100
      CALL SBLOBJ(LPARM,LPCT,LDAY,LTOUR,LBLK1,IBDT1)
      CDAT(LTOUR+ACTOFF)=CDAT(LTOUR+ACTOFF)+1.
      CDAT(LOVTR+ACTOFF)=CDAT(LOVTR+ACTOFF)-1.
      CDAT(LBLK2+QNBOFF)=CDAT(LBLK2+QOBOFF)
      CDAT(LBLK2+ACBOFF)=ACT
      CDAT(LBLK2+EFBOFF)=EF
      CDAT(LBLK2+QOBOFF)=QTEST
      DELTA=DELTA-1.
      IF(DELTA .GT. 0.) GO TO 10
100   CALL STROBJ(LDAY,LOVTR,NTRDT)
      CALL STROBJ(LDAY,LTOUR,ITYPE)
      CALL STROBJ(LDAY,LTTOUR,ITTYPE)
      RETURN
      END
```

Subroutine STROBJ

Subroutine STROBJ determines the contribution of one shift to the objective function value and the difference in its contribution per car-hour if an additional car were assigned to the shift. LDAY and LTOUR are pointers to the data for the day and shift. ITYPE is the tour to which the shift belongs.

The objective function contributions of the shift are determined by summing the contributions of its blocks. Subroutine STRDF is called to determine the improvement per car-hour that would be realized if one car were added to the shift. (STRDF also determines the improvement per car-hour that would be realized by removing a car from an overlay shift and adding one car to each of the shifts that it overlays.)

```
      SUBROUTINE STROBJ(LDAY,LTOUR,ITYPE)
C
C  EVALUATES A WEIGHTED OBJECTIVE FUNCTION FOR ONE SHIFT.
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      QOLD=0.
      QNEW=0.
      DO 10 IBLK=1,2
      IBDT=ICDAT(LTRTB(IBLK)+ITYPE-1)
      IF(IBDT .LT. 1) GO TO 10
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLK=LDAY+BLDOFF+(IBRD-1)*NWDBL
      QOLD=QOLD+CDAT(LBLK+QOBOFF)
      QNEW=QNEW+CDAT(LBLK+QNBOFF)
10    CONTINUE
      CDAT(LTOUR+QOTOFF)=QOLD
      CDAT(LTOUR+QNTOFF)=QNEW
      CALL STRDF(LDAY,LTOUR,ITYPE)
      RETURN
      END
```

Function OBJFUN

Function OBJFUN (objective function) evaluates an objective function over a span of hours of a day. Parameter LPARM is a pointer to a number list that specifies the objective function to be evaluated and any associated parameters. ISTART and IEND specify the span of hours over which the function is to be evaluated. LPCT, LDAY, and LTOUR are pointers to the precinct, day, and tour in which the span of hours occurs. EF is the number of effective cars for which the function is to be evaluated.

OBJFUN selects the correct function subprogram to evaluate the objective function specified by LPARM. For objective function 2, the second element of LPARM is the priority for which it is to be evaluated.

```
      FUNCTION OBJFUN(LPARM,ISTART,IEND,LPCT,LDAY,LTOUR,EF)
C
C  EVALUATES AN OBJECTIVE FUNCTION OVER THE SPAN OF HOURS
C  FROM ISTART TO IEND
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      IFNCTN=ICDAT(LPARM)
      LCR=LDAY+CRDOFF
      LST=LDAY+STDOFF
      LFR=LTOUR+HFTOFF
      GO TO (10,20,30),IFNCTN
C
10    OBJFUN=OBJF1(ISTART,IEND,LCR,LST,EF)
      GO TO 100
20    N=ICDAT(LPARM+2)
      OBJFUN=OBJF2(N,ISTART,IEND,LCR,LST,LFR,EF)
      GO TO 100
30    RV=CDAT(LTOUR+RVTOFF)
      OBJFUN=OBJF3(ISTART,IEND,LPCT,LDAY,RV,EF)
100   RETURN
      END
```

Subroutine ADDCAR

Subroutine ADDCAR (add cars) adds cars to a set of shifts so that the average value of a specified objective function is minimized. Parameter LPARM is a pointer to a number list that specifies the function to be evaluated. NCARHR is the number of car-hours available for allocation.

The allocation algorithm used is described in Appendix B of the User's Manual.

```
      SUBROUTINE ADDCAR(NCARHR,LPARM)
C
C  ADDS CARS TO A SET OF SHIFTS SO THAT THE AVERAGE VALUE
C  OF A SPECIFIED OBJECTIVE FUNCTION IS MINIMIZED
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
C
      NLEFT=NCARHR
C
C           FIND SHIFT WITH GREATEST IMPROVEMENT PER CAR HOUR IN
C           OBJECTIVE FUNCTION VALUE IF ALLOCATION IS CHANGED
C           INCREMENTALLY.
C
310   LBPCT=NXPCT(0)
      LBDAY=NXDAY(LBPCT,0)
      LBTOUR=NXTOUR(LBDAY,0,IBTYPE)
      QBIG=CDAT(LBTOUR+QDTOFF)
      LPCT=0
320   LPCT=NXPCT(LPCT)
      IF(LPCT .EQ. 0) GO TO 350
      LDAY=0
330   LDAY=NXDAY(LPCT,LDAY)
      IF(LDAY .EQ. 0) GO TO 320
      LTOUR=0
340   LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
      IF(LTOUR .EQ. 0) GO TO 330
      QDIF=AMAX1(CDAT(LTOUR+QDTOFF),CDAT(LTOUR+QXTOFF))
      IF(QDIF .LE. QBIG) GO TO 340
      QBIG=QDIF
      IBTYPE=ITYPE
      LBTOUR=LTOUR
      LBDAY=LDAY
      LBPCT=LPCT
      GO TO 340
```

```
C
350     IF(LBTOUR .NE. 0) GO TO 360
        WRITE(SYSOUT,5)
5       FORMAT(/' *** NO SHIFTS SELECTED - REENTER')
        RETURN
360     IF(ICDAT(LBTOUR+TYTOFF) .EQ. 5 .AND. CDAT(LBTOUR+QXTOFF)
     C        .GT. CDAT(LBTOUR+QDTOFF)) GO TO 500
        ILEN=ICDAT(LTREND+IBTYPE-1)-ICDAT(LTRST+IBTYPE-1)+1
        IF(ILEN .GT. NLEFT) RETURN
C
C            ADD A CAR TO SELECTED SHIFT AND COMPUTE NEW OBJECTIVE
C            FUNCTION VALUE
C
        CDAT(LBTOUR+ACTOFF)=CDAT(LBTOUR+ACTOFF)+1.
        NLEFT=NLEFT-ILEN
        CDAT(LBTOUR+QOTOFF)=CDAT(LBTOUR+QNTOFF)
        CDAT(LBTOUR+QNTOFF)=0.
        DO 370 IB=1,2
        IBDT=ICDAT(LTRTB(IB)+IBTYPE-1)
        IF(IBDT .LT. 1) GO TO 370
        IBRD=ICDAT(LBLRFL+IBDT-1)
        LBLK=LBDAY+BLDOFF+(IBRD-1)*NWDBL
        LTTOUR=LBTOUR
        IF(ICDAT(LBTOUR+TYTOFF) .EQ. 5)
     1   LTTOUR=LBDAY+TRDOFF+(IOVTR(IB)-1)*NWDTR
        CALL SBLOBJ(LPARM,LBPCT,LBDAY,LTTOUR,LBLK,IBDT)
        CDAT(LBTOUR+QNTOFF)=CDAT(LBTOUR+QNTOFF)+CDAT(LBLK+QNBOFF)
370     CONTINUE
        CALL STRDF(LBDAY,LBTOUR,IBTYPE)
        ID=ICDAT(LBTOUR+TYTOFF)-1
        GO TO (310,390,390,410),ID
C
C            ADJUST OBJECTIVE FUNCTION DIFFERENCES FOR SHIFTS IN
C            OVERLAY SEGMENTS
C
390     ITRRD=ICDAT(LTRRFL+NTRDT-1)
        LTTOUR=LBDAY+TRDOFF+(ITRRD-1)*NWDTR
        CALL STROBJ(LBDAY,LTTOUR,NTRDT)
        GO TO 310
C
410     DO 420 I=1,2
        ITRRD=IOVTR(I)
        ITYPE=ICDAT(LTRWFL+ITRRD-1)
        LTTOUR=LBDAY+TRDOFF+(ITRRD-1)*NWDTR
420     CALL STROBJ(LBDAY,LTTOUR,ITYPE)
        GO TO 310
C
C            DECREASE OVERLAY SHIFT ASSIGNMENT AND INCREASE ASSIGNMENTS
C            TO OVERLAID SHIFTS
C
500     ITOT=0
        DO 510 I=1,2
        ITRRD=IOVTR(I)
        ITP=ICDAT(LTRWFL+ITRRD-1)
        ISTART=ICDAT(LTRST+ITP-1)
        IEND=ICDAT(LTREND+ITP-1)
510     ITOT=ITOT+IEND-ISTART+1
        ILEN=ITOT-(ICDAT(LTREND+IBTYPE-1)-ICDAT(LTRST+IBTYPE-1)+1)
```

```
      IF(ILEN .GT. NLEFT) RETURN
      NLEFT=NLEFT-ILEN
      CDAT(LBTOUR+ACTOFF)=CDAT(LBTOUR+ACTOFF)-1.
      DO 520 I=1,2
      ITRRD=IOVTR(I)
      ITP=ICDAT(LTRWFL+ITRRD-1)
      IBDT=ICDAT(LTRTB(I)+ITP-1)
      IBRD=ICDAT(LBLRFL+IBDT-1)
      LBLK=LBDAY+BLDOFF+(IBRD-1)*NWDBL
      LTTOUR=LBDAY+TRDOFF+(ITRRD-1)*NWDTR
      CDAT(LTTOUR+ACTOFF)=CDAT(LTTOUR+ACTOFF)+1.
      CALL SBLOBJ(LPARM,LBPCT,LBDAY,LTTOUR,LBLK,IBDT)
      CALL STROBJ(LBDAY,LTTOUR,ITP)
520   CONTINUE
      CALL STRDF(LBDAY,LBTOUR,IBTYPE)
      GO TO 310
      END
```

Subroutine WRITE

Subroutine WRITE implements the WRITE command. It writes a file on a user-specified unit number which can later be used as a DATABASE file. Data are written only for precincts, days, and tours specified in the command qualifier.

```
      SUBROUTINE WRITE
C
C SUBROUTINE IMPLEMENTS WRITE COMMAND
C
      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
C
      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR
C
      INTEGER TYPE,VAL
      DIMENSION VAL(2),ORDER(3)
      DATA BLANK/1H /
C
      LGETT=TOP
      TYPE=CMD
      CALL SCAN(TYPE,VAL)
C
C          GET UNIT NUMBER
C
      IF(TYPE .EQ. NUMLST) GO TO 20
10    WRITE(SYSOUT,9010)
9010  FORMAT('0*** INVALID UNIT SPECIFICATION - REENTER.')
      TOP=LGETT
      RETURN
20    NUNIT=ICDAT(VAL(2))
      IF(NUNIT .EQ. SYSIN .OR. NUNIT .EQ. SYSOUT .OR. NUNIT .EQ. IFILE
     1 .OR. NUNIT .LT. 1 .OR. NUNIT .GT. 99) GO TO 10
C
```

```
C             SCAN QUALIFIER
C
        CALL SCAN(TYPE,VAL)
        CALL GTDSPC(TYPE,VAL,ORDER)
        IF(TYPE .NE. ERR) GO TO 30
        TOP=LGETT
        RETURN
C
C             SET WORK FLAGS
C
30       CALL SETWFL(IERR)
        IF(IERR .EQ. 0)GO TO 35
        TOP=LGETT
        RETURN
C
C             CHECK OVERLAY SPECIFICATION
C
35      CALL CKOVR(IERR)
        IF(IERR .EQ. 0) GO TO 40
        TOP=LGETT
        RETURN
C
C             DETERMINE NUMBER OF DAY, TOURS AND PRECINCTS
C             IN NEW DATA BASE
C
40      NDAY=NNAMES(1)
        IF(NDAY .LT. 1) NDAY=NDAYRD
        NTOUR=NNAMES(2)
        IF(NTOUR .LT. 1) NTOUR =NTRRD
        NPCT=0
        LPCT=0
50      LPCT=NXPCT(LPCT)
        IF(LPCT .LE. 0) GO TO 55
        NPCT=NPCT+1
        GO TO 50
55      CONTINUE
        IOVT=IOVRLY
        IF(ICDAT(LTRRFL+NTRDT-1) .EQ. 0 .OR. (ICDAT(LTRRFL+NTRDT-1)
     1  .EQ. 1 .AND. ICDAT(LTRWFL+NTRRD-1) .EQ. 0)) IOVT=0
C
C             WRITE CONTROL RECORD
C
        WRITE(NUNIT,1) DCLSNM,PCLSNM,TOURNM,NDIVRD,NPCT,NDAY,NBLDT,
     1 NTOUR,IOVT
        CALL GETTOP(80,LREC)
        I=0
        K=LREC-1
        DO 60 IDAY=1,NDAYRD
        ID=ICDAT(LDYWFL+IDAY-1)
        IF(ID .LT. 1) GO TO 60
        LNM=(ID-1)*8+LDAYNM
        CALL MOVE(ICDAT(LNM),ICDAT(LREC+I),8)
        I=I+8
        IF(I .LT. 80) GO TO 60
C
C             WRITE DAY NAMES
C
        WRITE(NUNIT,2)(ICDAT(K+J),J=1,80)
```

```
      I=0
60    CONTINUE
      IF(I .GT. 0) WRITE(NUNIT,2) (ICDAT(K+J),J=1,I)
      K=LBLKTB(2)-1
C
C          WRITE BLOCK DESCRIPTOR RECORDS
C
      WRITE(NUNIT,3) (ICDAT(K+I),I=1,NBLDT)
      DO 70 ITOUR=1,NTRRD
      IT=ICDAT(LTRWFL+ITOUR-1)
      IF(IT .LT. 1) GO TO 70
      IT=IT-1
      LNM=IT*8+LTRNM-1
C
C          WRITE TOUR DESCRIPTOR RECORDS
C
      WRITE(NUNIT,4) (ICDAT(LNM+I),I=1,8),ICDAT(LTRTB(1)+IT),
     1 ICDAT(LTRTB(2)+IT)
70    CONTINUE
C
      LPCT=0
100   LPCT=NXPCT(LPCT)
      IF(LPCT .NE. 0) GO TO 110
      ENDFILE NUNIT
      TOP=LGETT
      RETURN
110   IDIV=ICDAT(LPCT+DVPOFF)-1
      LPCTNM=LPCT+NMPOFF-1
      LDVNM=LDIVNM+IDIV*8-1
C
C          WRITE PRECINCT HEADER
C
      WRITE(NUNIT,5) (ICDAT(LPCTNM+I),I=1,8),(ICDAT(LDVNM+I),I=1,8),
     1 (CDAT(LPCT+I),I=ARPOFF,B2POFF)
C
      LDAY=0
200   LDAY=NXDAY(LPCT,LDAY)
      IF(LDAY .LT. 1) GO TO 100
      CPARM=CDAT(LDAY+CPDOFF)
      SPARM=CDAT(LDAY+SPDOFF)
C
C          WRITE DAY DETAIL RECORDS
C
      WRITE(NUNIT,6) CPARM,SPARM,ICDAT(LDAY+DVDOFF)
      LCRO=LREC-1
      LSTO=LREC+23
      LCRI=LDAY+CRDOFF-1
      LSTI=LDAY+STDOFF-1
      SPARM=SPARM/60.
      DO 210 I=1,24
      ICDAT(LCRO+I)=100.*(.005+CDAT(LCRI+I)/CPARM)
210   ICDAT(LSTO+I)=100.*(.005+CDAT(LSTI+I)/SPARM)
      WRITE(NUNIT,7) (ICDAT(LCRO+I),I=1,48)
      DO 220 I=1,24
220   ICDAT(LCRO+I)=0
C
      LTOUR=0
300   LTOUR=NXTOUR(LDAY,LTOUR,ITYPE)
```

```
      IF(LTOUR .NE. 0) GO TO 320
      IF(IOVT .NE. 0 .AND. ICDAT(LDAY+OVDOFF) .EQ. 0)
     1 WRITE(NUNIT,2) BLANK
      DO 310 I=1,NBLDT
      IBLK=ICDAT(LBLRFL+I-1)
      IF(IBLK .LE. 0) GO TO 310
      LBLK=LDAY+BLDOFF+(IBLK-1)*NWDBL
      CDAT(LCRO+I)=CDAT(LBLK+DCBOFF)
310   CONTINUE
C
C           WRITE BLOCK DETAIL RECORD
C
      WRITE(NUNIT,8) (CDAT(LCRO+I),I=1,NBLDT)
      GO TO 200
C
C           WRITE SHIFT DETAIL RECORD
C
320   WRITE(NUNIT,9) (CDAT(LTOUR+I),I=ACTOFF,MFTOFF)
      GO TO 300
1     FORMAT(2(8A1,2X),8A1,1X,I2,1X,I3,1X,I3,1X,2(I2,1X),I1)
2     FORMAT(80A1)
3     FORMAT(24(I2,1X))
4     FORMAT(8A1,1X,I2,1X,I2)
5     FORMAT(8A1,1X,8A1,2X,F5.2,1X,F5.1,2(1X,F5.3))
6     FORMAT(2(F5.2,1X),I1)
7     FORMAT(24I3)
8     FORMAT(24F3.1)
9     FORMAT(3(F5.1,1X),2(F5.4,1X))
      END
```

Subroutine SKIP

Subroutine SKIP is called to skip past N records in the data file that are not needed by the calling routine.

```
      SUBROUTINE SKIP(IFILE,N)
C
C  SKIPS N RECORDS IN FILE IFILE
C
      IF( N.LT.1) RETURN
      DO 1 I=1,N
1     READ(IFILE,2) J
2     FORMAT(A1)
      RETURN
      END
```

Function CEIL

CEIL(X) is the least integer that is greater than or equal to X.

```
      FUNCTION CEIL(X)
C
C  LEAST INTEGER GREATER THAN OR EQUAL TO X
C
      ICEIL=X
      CEIL=ICEIL
      IF(X.GT.CEIL)CEIL=CEIL+1.
      RETURN
      END
```

BLOCK DATA

BLOCK DATA establishes the initial numerical values of variables in COMMON blocks used by the program's subroutines and functions.

The COMMON blocks are as follows:

COMMON/PNTRS/	Pointers. See Chapter IV.
COMMON/OFFSET/	Offsets. See Chapter IV.
COMMON/STORE/	Parameters related to run-time storage requirements.
COMMON/SYSTEM/	Input and output unit numbers.
COMMON/KEYWDS/	Keywords and word types.
COMMON/LCODES/	Lexical types returned by GETTKN.
COMMON/SCODES/	Syntactic types returned by SCAN.
COMMON/STATS/	Statistics and output orders.

```
      BLOCK DATA
C
      COMMON/PNTRS/IOVRLY,IOVTR(2),
     1NPCTDT,NPCTRD,LPCTDT,LNMLST(4),NNAMES(4),NDAYDT,LDAYNM,
     2LDYRFL,NDAYRD,LDYWFL,NTRDT,LTRTB(2),LTRST,LTREND,LTRRFL,LTRNM,
     3NTRRD,LTRWFL,NBLDT,LBLKTB(2),LBLRFL,NBLRD,LBLWFL,NDIVDT,NDIVRD,
     4LDIVNM,LDIVFL
      DATA NDAYRD/0/,NTRRD/0/
C
      COMMON/OFFSET/NMPOFF,DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,
     1NWDPCT,CPDOFF,SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,NWDDY,
     2QDTOFF,QXTOFF,CRTOFF,QOTOFF,QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,
     3PVTOFF,HFTOFF,MFTOFF,LFTOFF,NPRIO,NWDTR,BLDOFF,QOBOFF,QNBOFF,
     4EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,NWDBL
C
      INTEGER DVPOFF,ARPOFF,SMPOFF,B1POFF,B2POFF,DYPOFF,CPDOFF,
     1SPDOFF,OVDOFF,CRDOFF,STDOFF,TRDOFF,QDTOFF,QXTOFF,CRTOFF,QOTOFF,
     2QNTOFF,CTTOFF,TYTOFF,ACTOFF,RVTOFF,PVTOFF,HFTOFF,BLDOFF,
     3EFBOFF,ACBOFF,AWBOFF,CRBOFF,RMBOFF,OCBOFF,CTBOFF,QOBOFF,QNBOFF
C
      DATA NMPOFF/0/,DVPOFF/8/,ARPOFF/9/,SMPOFF/10/,B1POFF/11/,
     1B2POFF/12/,DYPOFF/13/,CPDOFF/0/,SPDOFF/1/,DVPOFF/2/,CRDOFF/3/,
     2STDOFF/27/,TRDOFF/51/,QDTOFF/0/,QXTOFF/1/,CRTOFF/2/,
     3QOTOFF/3/,QNTOFF/4/,CTTOFF/5/,TYTOFF/6/,ACTOFF/7/,RVTOFF/8/,
     4PVTOFF/9/,HFTOFF/10/,MFTOFF/11/,LFTOFF/12/,NPRIO/3/,NWDTR/13/,
     4EFBOFF/0/,ACBOFF/1/,AWBOFF/2/,CRBOFF/3/,RMBOFF/4/,OCBOFF/5/,
     5CTBOFF/6/,QOBOFF/7/,QNBOFF/8/,NWDBL/9/
C
      COMMON/STORE/TOP,BOT,RDBOT,MAXBOT,NWORDS,CDAT(11000)
      INTEGER TOP,BOT,RDBOT
      DIMENSION ICDAT(11000)
      EQUIVALENCE(ICDAT,CDAT)
      DATA BOT/1/,NWORDS/11000/,MAXBOT/0/
C
      COMMON/SYSTEM/SYSIN,SYSOUT,IFILE,LIT
      INTEGER SYSIN,SYSOUT
      DATA SYSIN/4/,SYSOUT/5/,IFILE/19/,LIT/20/
C
      COMMON/KEYWDS/NKYWD,NTYPES,TYPOFF(4),KEYWD(8,30),WDTYPE(30)
      INTEGER TYPOFF,WDTYPE
      DIMENSION PCLSNM(8),DCLSNM(8),TOURNM(8)
      EQUIVALENCE (PCLSNM,KEYWD(1,4)),(DCLSNM,KEYWD(1,3)),
     1(TOURNM,KEYWD(1,2))
      DATA NTYPES/4/,NKYWD/26/,TYPOFF(1)/4/,TYPOFF(2)/4/,TYPOFF(3)/9/,
     C TYPOFF(4)/9/
C
      COMMON/LCODES/LEND,WORD,NUM,LP,RP
      INTEGER WORD,RP
      DATA LEND/1/,WORD/2/,NUM/3/,LP/4/,RP/5/
C
      COMMON/SCODES/SEND,CMD,NUMLST,NAMLST,FSPEC,DSPEC,DUM,ERR
      INTEGER SEND,CMD,FSPEC,DSPEC,DUM,ERR
      DATA SEND/5/,CMD/3/,NUMLST/6/,NAMLST/7/,FSPEC/2/,DSPEC/1/,
     1DUM/4/,ERR/8/
```

```
C
      COMMON/STATS/T(4,8),S(4,8),PORDER(3),RORDER(3),CIND(8)
      INTEGER PORDER,RORDER
      DATA PORDER(1)/3/,PORDER(2)/2/,PORDER(3)/1/
      END
```

Appendix A

DEMONSTRATION DATA BASE

```
DIVISION  PRECINCT  TOUR     02 005 007 05 04 1
SUN-MON MON-TUE TUE-WED WED-THU THU-FRI FRI-SAT SAT-SUN
08 11 16 19 24
MIDDAY    01
PM        02 03
AM        04 05
FOURTH    03 04
EAST      HIGHLAND  34.88 503.4 -.824 .677
7.71  40.23 1
.22.74.84.811.11.1.77.841.01.41.61.91.51.21.21.71.81.71.21.1.94.45.58.29
.91.91.91.91.91.91.91.91.81.81.81.81.81.81.81.81.90.90.90.90.90.90.90.90
9.8   15.0  7.5   .065  .84
9.4   15.0  7.5   .065  .84
8.8   25.0  7.5   .065  .84
4.4   15.0  7.5   .065  .84
4.52.57.01.51.3
7.71  45.08 1
.75.971.1.94.78.811.11.01.2.881.31.21.71.41.11.11.21.3.71.68.45.36.19.23
1.01.01.01.01.01.01.01.0.93.93.93.93.93.93.93.931.11.11.11.11.11.11.11.1
12.9  15.0  7.5   .065  .84
9.4   15.0  7.5   .065  .84
9.3   25.0  7.5   .065  .84
3.8   15.0  7.5   .065  .84
4.02.54.52.81.0
7.71  43.91 1
.88.62.65.841.1.97.911.11.71.11.31.41.31.7.941.0.84.81.45.52.39.29.19.49
1.11.11.11.11.11.11.11.1.97.97.97.97.97.97.97.971.21.21.21.21.21.21.21.2
13.4  15.0  7.5   .065  .84
11.5  15.0  7.5   .065  .84
9.5   25.0  7.5   .065  .84
5.4   15.0  7.5   .065  .84
3.52.84.02.01.3
7.71  45.62 1
1.2.84.58.78.831.3.681.0.941.21.31.51.31.61.11.2.97.55.65.42.23.06.23.36
1.01.01.01.01.01.01.01.0.77.77.77.77.77.77.77.771.11.11.11.11.11.11.11.1
14.2  15.0  7.5   .065  .84
11.5  15.0  7.5   .065  .84
9.8   25.0  7.5   .065  .84
4.1   15.0  7.5   .065  .84
4.03.83.31.5.50
7.71  41.19 1
1.1.97.97.971.2.941.11.11.31.91.51.41.41.31.51.2.97.97.97.68.26.36.32.68
1.11.11.11.11.11.11.11.1.93.93.93.93.93.93.93.931.11.11.11.11.11.11.11.1
12.9  15.0  7.5   .065  .84
11.6  15.0  7.5   .065  .84
9.8   25.0  7.5   .065  .84
5.4   15.0  7.5   .065  .84
4.82.53.52.5.50
7.71  43.92 1
.71.58.65.71.75.781.01.2.911.71.21.51.42.12.21.2.91.75.75.49.32.29.36.42
.97.97.97.97.97.97.97.97.93.93.93.93.93.93.93.931.21.21.21.21.21.21.21.2
12.1  15.0  7.5   .065  .84
11.6  15.0  7.5   .065  .84
10.6  25.0  7.5   .065  .84
5.5   15.0  7.5   .065  .84
5.72.37.02.0.50
7.71  37.91 1
1.0.911.01.01.51.41.21.31.61.61.51.81.51.51.61.51.91.51.2.63.32.48.55.62
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
11.8  15.0  7.5   .065  .84
```

(continued)

```
10.6   15.0   7.5    .065   .84
8.8    25.0   7.5    .065   .84
5.8    15.0   7.5    .065   .84
4.05.33.01.81.5
WEST       LOWLAND     51.92 678.5 -.476 .637
6.65   37.88 1
.38.60.94.53.831.11.31.31.51.71.41.41.71.81.51.92.31.5.86.90.49.53.23.38
0.90.90.90.90.90.90.90.90.90.90.90.90.90.90.90.91.11.11.11.11.11.11.11.1
10.1   15.0   7.5    .061   .81
9.4    15.0   7.5    .061   .81
8.6    25.0   7.5    .061   .81
5.3    15.0   7.5    .061   .81
5.33.32.32.81.5
6.65   39.51 1
.98.94.79.98.98.86.86.681.41.31.31.51.21.51.5.861.3.60.75.41.15.41.34.60
1.21.21.21.21.21.21.21.20.90.90.90.90.90.90.90.91.01.01.01.01.01.01.01.0
11.1   15.0   7.5    .061   .81
11.6   15.0   7.5    .061   .81
7.1    25.0   7.5    .061   .81
4.9    15.0   7.5    .061   .81
4.32.33.31.50.8
6.65   45.13 1
.711.2.68.49.641.41.11.51.21.41.41.61.41.11.51.3.981.1.26.23.23.15.53.64
0.80.80.80.80.80.80.80.81.11.11.11.11.11.11.11.11.11.11.11.11.11.11.11.1
10.7   15.0   7.5    .061   .81
10.3   15.0   7.5    .061   .81
9.1    25.0   7.5    .061   .81
4.9    15.0   7.5    .061   .81
4.52.83.01.32.0
6.65   37.41 1
.83.68.68.94.991.21.0.791.41.51.51.31.41.21.41.4.79.53.53.53.23.34.53.64
0.90.90.90.90.90.90.90.91.01.01.01.01.01.01.01.01.21.21.21.21.21.21.21.2
10.8   15.0   7.5    .061   .81
10.1   15.0   7.5    .061   .81
8.8    25.0   7.5    .061   .81
4.9    15.0   7.5    .061   .81
3.00.31.82.00.5
6.65   38.01 1
.411.11.0.681.2.90.981.01.21.31.61.61.21.61.31.1.64.60.34.71.38.26.38.64
1.01.01.01.01.01.01.01.00.90.90.90.90.90.90.90.91.31.31.31.31.31.31.31.3
12.1   15.0   7.5    .061   .81
11.1   15.0   7.5    .061   .81
8.0    25.0   7.5    .061   .81
5.9    15.0   7.5    .061   .81
3.02.03.80.80.3
6.65   42.21 1
.711.1.94.64.641.1.79.71.931.41.21.31.71.82.31.7.79.68.45.49.41.34.23.34
1.01.01.01.01.01.01.01.00.90.90.90.90.90.90.90.91.31.31.31.31.31.31.31.3
11.1   15.0   7.5    .061   .81
11.0   15.0   7.5    .061   .81
7.6    25.0   7.5    .061   .81
5.9    15.0   7.5    .061   .81
7.03.02.30.80.3
6.65   36.18 1
1.1.831.1.86.[illegible].791.4.791.21.61.71.12.0[illegible] 02 41.72.41.81.5.64.45.49.26.45
1.21.21.21.21.21.21.21.20.90.90.90.90.90.90.90.91.01.01.01.01.01.01.01.0
9.2    15.0   7.5    .061   .81
10.5   15.0   7.5    .061   .81
7.5    25.0   7.5    .061   .81
5.7    15.0   7.5    .061   .81
2.52.34.32.31.0
```

(continued)

```
NORTH    HIGHLAND  24.08 409.1 -.997 .806
5.32  32.34 1
.94.77.851.31.31.01.3.991.61.41.51.81.61.41.11.72.01.01.3.42.61.47.19.61
1.11.11.11.11.11.11.11.1.83.83.83.83.83.83.83.831.11.11.11.11.11.11.11.1
7.5    15.0  7.5   .061  .85
6.3    15.0  7.5   .061  .85
6.5    25.0  7.5   .061  .85
3.1    15.0  7.5   .061  .85
2.81.3.75.30.30
5.32  35.94 1
1.31.21.3.801.5.94.991.01.11.51.61.0.901.11.11.51.1.71.56.52.42.24.38.85
1.11.11.11.11.11.11.11.1.90.90.90.90.90.90.90.901.01.01.01.01.01.01.01.0
10.3   15.0  7.5   .061  .85
7.8    15.0  7.5   .061  .85
6.0    25.0  7.5   .061  .85
3.0    15.0  7.5   .061  .85
2.8.752.01.01.3
5.32  34.26 1
.85.951.0.91.94.95.95.941.01.41.21.61.01.31.41.31.3.80.42.33.38.52.28.61
1.21.21.21.21.21.21.21.2.89.89.89.89.89.89.89.891.01.01.01.01.01.01.01.0
11.9   15.0  7.5   .061  .85
7.8    15.0  7.5   .061  .85
7.1    25.0  7.5   .061  .85
3.0    15.0  7.5   .061  .85
2.5.753.51.0.50
5.32  40.83 1
.75.94.94.66.71.861.31.31.21.11.31.31.51.01.41.6.801.2.66.38.28.28.33.71
1.01.01.01.01.01.01.01.0.90.90.90.90.90.90.90.901.21.21.21.21.21.21.21.2
11.3   15.0  7.5   .061  .85
6.5    15.0  7.5   .061  .85
7.8    25.0  7.5   .061  .85
4.1    15.0  7.5   .061  .85
2.5.503.31.81.3
5.32  34.99 1
1.31.6.71.89.89.85.891.41.31.41.51.61.71.1.85.89.71.85.52.66.19.19.33.52
1.11.11.11.11.11.11.11.11.31.31.31.31.31.31.31.31.01.01.01.01.01.01.01.0
12.0   15.0  7.5   .061  .85
7.1    15.0  7.5   .061  .85
6.5    25.0  7.5   .061  .85
3.8    15.0  7.5   .061  .85
3.01.51.8.75.25
5.32  37.10 1
.801.0.86.76.66.99.94.991.31.61.21.41.81.92.1.94.66.80.89.28.28.24.38.38
1.11.11.11.11.11.11.11.1.91.91.91.91.91.91.91.911.21.21.21.21.21.21.21.2
11.3   15.0  7.5   .061  .85
7.8    15.0  7.5   .061  .85
6.6    25.0  7.5   .061  .85
4.8    15.0  7.5   .061  .85
33.5.502.8.250.3
5.32  35.28 1
.661.11.2.85.85.80.661.01.51.01.01.7.941.61.72.32.21.21.6.90.91.28.28.28
1.21.21.21.21.21.21.21.2.89.89.89.89.89.89.89.89.97.97.97.97.97.97.97.97
7.5    15.0  7.5   .061  .85
7.9    15.0  7.5   .061  .85
6.3    25.0  7.5   .061  .85
4.8    15.0  7.5   .061  .85
3.31.52.31.3.50
SOUTH    LOWLAND   61.16 571.3 -.916 .773
5.50  38.85 1
.50.68.681.21.51.41.11.41.51.41.91.51.41.52.01.71.71.6.911.1.45.36.36.68
1.81.81.81.81.81.81.81.8.71.71.71.71.71.71.71.711.21.21.21.21.21.21.21.2
```

(continued)

```
10.6   15.0  7.5    .067  .84
7.6    15.0  7.5    .067  .84
7.3    25.0  7.5    .067  .84
2.3    15.0  7.5    .067  .84
3.02.35.33.00.8
5.50   47.43 1
1.2.641.2.821.11.01.0.771.11.41.61.21.61.4.96.96.86.86.41.32.32.32.32.64
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
11.0   15.0  7.5    .067  .84
7.6    15.0  7.5    .067  .84
7.1    25.0  7.5    .067  .84
1.3    15.0  7.5    .067  .84
5.52.85.31.01.8
5.50   45.71 1
.59.77.86.91.91.961.11.41.61.61.91.71.21.51.81.1.91.36.18.18.05.18.32.59
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
12.6   15.0  7.5    .067  .84
6.9    15.0  7.5    .067  .84
6.8    25.0  7.5    .067  .84
1.8    15.0  7.5    .067  .84
4.02.03.01.00.3
5.50   44.77 1
.86.96.50.36.50.45.821.11.21.71.51.51.51.91.6.86.73.64.36.32.05.14.27.64
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
11.8   15.0  7.5    .067  .84
8.3    15.0  7.5    .067  .84
6.6    25.0  7.5    .067  .84
1.8    15.0  7.5    .067  .84
2.52.02.00.30.8
5.50   41.55 1
1.0.911.1.771.2.73.96.961.31.31.51.7.961.41.0.68.82.41.27.32.09.18.32.68
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
12.3   15.0  7.5    .067  .84
8.3    15.0  7.5    .067  .84
7.2    15.0  7.5    .067  .84
2.1    15.0  7.5    .067  .84
3.30.81.51.30.8
5.50   40.69 1
1.1.73.91.77.73.861.01.01.11.51.01.72.11.62.01.5.96.55.36.14.23.18.41.59
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
11.1   15.0  7.5    .067  .84
8.8    15.0  7.5    .067  .84
7.6    25.0  7.5    .067  .84
4.3    15.0  7.5    .067  .84
4.31.34.02.51.0
5.50   40.79 1
1.1.77.911.11.01.31.51.11.71.61.61.61.81.62.02.02.02.01.4.73.68.36.18.50
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
9.8    15.0  7.5    .067  .84
8.3    15.0  7.5    .067  .84
8.6    25.0  7.5    .067  .84
4.3    15.0  7.5    .067  .84
2.02.54.442.51.5
CENTRAL  LOWLAND    47.27 498.3 -.746 .658
3.66   37.31 1
.21.96.55.96.751.01.41.22.11.41.81.41.7.[illegible].12.32.31.4.681.0.62.41.[illegible]
1.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
6.3    15.0  7.5    .067  .81
5.4    15.0  7.5    .067  .81
3.8    25.0  7.5    .067  .81
2.3    15.0  7.5    .067  .81
```

(continued)

```
0.52.52.00.80.3
3.66   39.90 1
.68.681.51.21.3.891.0.89.961.71.51.21.61.21.11.3.96.55.55.34.20.21.62.27
0.90.90.90.90.90.90.90.91.01.01.01.01.01.01.01.01.01.01.01.01.01.0
6.8     15.0   7.5    .067   .81
5.0     15.0   7.5    .067   .81
3.5     25.0   7.5    .067   .81
2.8     15.0   7.5    .067   .81
2.32.01.50.50.3
3.66   40.20 1
.75.751.0.34.62.481.31.11.21.81.3.821.11.61.21.0.68.68.34.75.34.41.62.62
1.11.11.11.11.11.11.11.10.90.90.90.90.90.90.90.90.90.90.90.90.90.90.9
7.6     15.0   7.5    .067   .81
5.8     15.0   7.5    .067   .81
4.3     25.0   7.5    .067   .81
3.5     15.0   7.5    .067   .81
2.31.31.50.30.3
3.66   37.44 1
1.41.3.68.891.4.82.961.11.51.51.51.81.21.22.01.0.82.48.82.21.27.34.07.34
1.11.11.11.11.11.11.11.10.90.90.90.90.90.90.91.21.21.21.21.21.21.21.2
8.3     15.0   7.5    .067   .81
5.6     15.0   7.5    .067   .81
4.3     25.0   7.5    .067   .81
3.3     15.0   7.5    .067   .81
3.01.31.80.50.3
3.66   43.19 1
.621.2.68.89.68.75.621.11.31.41.42.11.62.01.1.82.21.41.27.48.14.27.21.41
1.01.01.01.01.01.01.01.0.90.90.90.90.90.90.90.901.41.41.41.41.41.41.41.4
6.6     15.0   7.5    .067   .81
5.3     15.0   7.5    .067   .81
3.8     25.0   7.5    .067   .81
3.1     15.0   7.5    .067   .81
1.30.31.800.50.5
3.66   39.66 1
.62.68.82.55.89.98.891.31.81.61.11.62.11.81.61.7.62.41.41.62.27.07.41.55
1.01.01.01.01.01.01.01.0.90.90.90.90.90.90.901.31.31.31.31.31.31.31.3
7.1     15.0   7.5    .067   .81
6.9     15.0   7.5    .067   .81
3.5     25.0   7.5    .067   .81
4.1     15.0   7.5    .067   .81
3.31.31.50.30.5
3.66   38.60 1
.27.821.31.3.681.21.0.891.21.71.1.961.62.32.02.52.31.81.41.1.96.20.34.14
1.11.11.11.11.11.11.11.11.01.01.01.01.01.01.01.01.01.01.01.01.01.01.0
6.6     15.0   7.5    .067   .81
6.3     15.0   7.5    .067   .81
3.8     25.0   7.5    .067   .81
4.1     15.0   7.5    .067   .81
1.81.51.82.01.5
```

Appendix B

PROGRAM FOR ESTIMATING THE RELATIONSHIP BETWEEN CFS UNAVAILABILITIES AND NON-CFS UNAVAILABILITIES*

INTRODUCTION

Research conducted by students at UCLA on the allocation process for the Los Angeles Police Department** found that unavailabilities of patrol cars for reasons other than calls for service (due to traffic enforcement, meal times, and the like) vary according to the cfs workload of the cars. These unavailabilities, which are an important component of patrol car activity in all police departments, reduce the effective number of cars available to service calls. Scheduled unavailabilities such as meals, and unscheduled but predictable activities such as automobile service stops, can be expected to occur independent of the call-for-service workload. However, discretionary officer-initiated activity might increase during slack periods and decrease during overloaded periods of the day, or vice versa.

In Los Angeles, the variation was found to be rather complicated. Data collected at the dispatch center showed that more time was spent per car on non-cfs unavailabilities when the cfs workload was high than when it was low. An apparent explanation for this is that the times of day in which many emergencies are reported to the police by telephone also have many activities visible from the street that require police intervention.

However, the queuing delays experienced before a car could be dispatched to a call in Los Angeles indicated that more cars were unavailable at times of low cfs workload than were reported unavailable. Moreover, it sometimes happened at times of high cfs workload that more cars were actually available than were reported available. This is because extra patrol units, such as traffic cars and sergeants' cars, could be used to handle cfs work if absolutely necessary.

---

*This appendix was coauthored by David J. Jaquette, who also wrote the program described here.

***An Analysis of the Patrol Car Deployment Methods of the Los Angeles Police Department*, Engineering School report by Public Systems Analysis class, University of California at Los Angeles, 1975.

Since the equations in a patrol allocation program should be designed to predict queuing delays as they will actually occur, it is appropriate to estimate the effective number of patrol cars present in the field from data giving the number of calls delayed, and not from data telling how many cars were fielded and their reported unavailabilities.

Thus if NEFF denotes the effective number of patrol cars which, according to queuing formulas, would cause the observed fraction of calls delayed, our estimate of the fraction of time each car is unavailable on non-cfs activity is

$$\mathrm{UNAVL} = 1 - \frac{\mathrm{NEFF}}{\mathrm{CARS}} , \tag{B.1}$$

where CARS is the number of cars fielded.

Then, the effective fraction of time unavailable (UNAVL) is modeled to be linearly related to the fraction of time the average car spends on calls for service, C:

$$\mathrm{UNAVL} = \mathrm{B1} \cdot \mathrm{C} + \mathrm{B2} , \tag{B.2}$$

where B1 and B2 are coefficients specific to each precinct but assumed to be time homogeneous. This relationship was found to explain the relationship between effective and fielded cars in Los Angeles, as evidenced by the increase in the number of calls delayed during slack periods and the decrease during periods of heavy demand in calls for service. (See Fig. 3 in the User's Manual.)

Given this relationship, the only input data related to non-cfs unavailabilities needed by PCAM is the pair of unavailability parameters B1 and B2 for each precinct. The computer program listed and annotated here was written originally as an aid to the LAPD Automated Deployment of Available Manpower (ADAM) project in their attempt to implement a version of PCAM. It can be used to construct estimates of the unavailability parameters.

INPUT DATA

The program takes raw data which were available from LAPD records and converts them into numbers usable in a standard linear regression. Each data point read in on one data card represents a number of weeks (NWEEKS) of aggregated data for one shift. For example, one line of printout in the data summary available to the LAPD would describe the activity of patrol cars in the Van Nuys Area during the tour from midnight to 3 a.m. on Mondays, over a four-week period; thus NWEEKS = 4. Each input card contains the total number of actual car hours (AVLHR), which in this example would be 4x3 times the average number of cars fielded; the number of hours in the shift (NHOURS), which in the example is 3; actual hours spent on calls-for-service work (CFSWRK); total number of delayed calls (NDELAY); and total calls for service (NTCFS).

CALCULATIONS IN THE PROGRAM

The fraction of each car's time spent on calls for service, which is the independent variable of the regression, is immediately found as the calls-for-service workload divided by the total actual car-hours,

$$C = \frac{CFSWRK}{AVLHR}.$$

This is calculated for each shift.

For each shift, dividing the number of calls delayed by the total number of calls gives the fraction delayed, which is an estimate of the probability of delay. If there are N effective cars on duty, and the number of cfs work hours per hour is ρ, the formula for an M/M/N queue shows that the probability of a call being delayed is

$$P(\text{delay}|N) = \frac{\rho^N/(N!(1-\rho/N))}{1+\rho+\rho^2/2!+\ldots+\rho^{(N-1)}/(N-1)!+\rho^N/N!(1-\rho/N)}. \qquad (B.3)$$

A maximum-likelihood and unbiased estimate of the number of *effective* cars during a given shift can be made by solving N in the relationship

$$P(\text{delay}|N) = \text{actual fraction of calls delayed.}$$

The value of ρ needed in the above calculation is found as the actual call-for-service workload hours (CFSWRK) divided by the number of total hours contained in the data for that shift (NWEEKS x NHOURS):

$$\rho = \frac{CFSWRK}{NWEEKS \cdot NHOURS}.$$

Once ρ and the fraction delayed are estimated, N can be determined by evaluating the expression above for a K such that

$$P(\text{delay}|K) > \text{actual fraction of calls delayed}$$

and

$$P(\text{delay}|K+1) < \text{actual fraction of calls delayed}.$$

Linear interpolation between K and K+1 is used to estimate N. The ratio of N to the actual number of cars fielded, CARS, gives an estimate of the fraction of time unavailable, UNAVL, as shown in Eq. B.1, (CARS = AVLHRS/NWEEKS · NHOURS).

Once UNAVL and C have been calculated for each shift in a precinct, the usual formulas for a regression fit are used to estimate B1 and B2 for that precinct:

$$B1 = \frac{n \Sigma \, UNAVL_i C_i - \Sigma \, UNAVL_i \, \Sigma \, C_i}{n \Sigma \, C_i^2 - (\Sigma \, C_i)^2},$$

where n is the number of observations, and

$$B2 = \Sigma (UNAVL_i - B1 \cdot C_i)/n.$$

INPUT DATA FORMAT FOR PROGRAM TO CALCULATE B1 AND B2

The format instructions may be clarified by the sample data file that follows.

1. Control card. Enter the number of precincts for which data are provided in columns 1-2, format I2.

2. Cards for each precinct

   a. Precinct name. Enter precinct name on one card, left justified.

   b. Number of data cards for this precinct. Enter on one card in columns 1-2, format I2.

   c. Data cards. One for each shift.

Position	Format	Description
1-10	F10.1	AVLHR Number of actual car-hours fielded in the shift
11-12	I2	NHOURS Number of hours in the shift
13-22	F10.2	CFSWRK Number of car-hours of cfs work
23-25	I3	NDELAY Number of calls delayed
26-28	I3	NTCFS Total number of calls for service

SAMPLE DATA FILE FOR PROGRAM TO CALCULATE B1 AND B2

Note that an error has been purposely introduced for the first shift in WEST precinct. The number of calls delayed (64) exceeds the total number of calls (63). This data card (observation 1 in WEST precinct) will be ignored by the program.

```
Column 1
 |
 v
 3
WEST
 5
111.         3 85.2         64 63
140.         5 31.           4 23
312.         8 151.4        67113
123.         3 71.9         46 54
240.         5 104.         40 78
DOWNTOWN
 5
105.         3 42.3         12 31
140.         5 30.5          7 25
336.         8 189.7       110142
126.         3 106.3        82 95
245.         5 130.9        65 98
NORTH
 7
       122   3 38.3         11 34
       152   5 25.3          4 28
       362   8 195.4       103155
       138   3 102.4        84 92
       210   5 118.2        58 95
       128   3  42.3        13 36
       158   5  32.4         5 30
```

OUTPUT FROM RUNNING PROGRAM WITH SAMPLE DATA FILE

```
ERROR IN DELAY DATA FOR OBS    1 IN WEST          PRECINCT
 FOR WEST           PRECINCT  B1=     -0.3476 B2=      0.5876
 FOR DOWNTOWN       PRECINCT  B1=     -0.7170 B2=      0.7476
 FOR NORTH          PRECINCT  B1=     -0.6336 B2=      0.7015
```

LISTING OF PROGRAM TO CALCULATE UNAVAILABILITY

PARAMETERS B1 AND B2

```
      DIMENSION PROB(20)
      INTEGER FACTN(41)
      DATA NWEEKS/4/
C
C
C    CALCULATE FACTORIAL(I) AS FACTN(I+1)
C
      FACTN(1) = 1.
      DO 1 I=2,40
1       FACTN(I) = FACTN(I-1)*(I-1)
C
      READ (5,101) NDIST
101   FORMAT(I2)
      DO 30 IJ=1,NDIST
       READ (5,102) PCTNM1,PCTNM2,PCTNM3
102    FORMAT(3A4)
       READ (5,103) NOBSV
103    FORMAT (I2)
       SUMY=0.0
       SUMYSQ=0.0
       SUMC=0.0
       SUMCSQ=0.0
       SUMYC=0.0
       NOBS = 0
       DO 20 IK=1,NOBSV
        READ (5,104) AVLHR,NHOURS,CFSWRK,NDELAY,NTCFS
        CARS= AVLHR/(NWEEKS*NHOURS)
        NCARS = CARS + .99999999
104     FORMAT(F10.1,1I2,F10.2,2I3)
        RHO= CFSWRK/(NHOURS*NWEEKS)
        DELAYP = NDELAY
        DELAYP=DELAYP/NTCFS
        IF (DELAYP.GT.1.0) GO TO 19
C
C
C   CALCULATE INTEGER N-EFFECTIVE FROM QUEUING FORMULA
C
        NEFF = 1
        LOWCAR= RHO + 1
        DO 5 I=LOWCAR,NCARS
         DENSUM= 1.
         ILESS1= I-1
         DO 4 IL= 1,ILESS1
4         DENSUM= DENSUM + RHO**IL/FACTN(IL+1)
         XNUM = RHO**I/((1.-RHO/I)*FACTN(I+1))
         PROB(I) = XNUM/(DENSUM+XNUM)
         NEFF = I
         IF (PROB(I).LE.DELAYP) GO TO 11
5       CONTINUE
C
C
C     THE FOLLOWING IS AN INTERPOLATION FOR EFFECTIVE N
C     IF THE CLOSEST NEFFECTIVE CARS IS GREATER THAN OR EQUAL TO THE
C     THE ACTUAL NUMBER OF CARS, THEN THE INTERPOLATION IS BYPASSED,AND
C     THE TIME SPENT ON NON-CFS WORK IS SET TO ZERO
C
11      AEFF= NEFF
        IF (AEFF.GE.CARS)  GO TO 7
        JK= NEFF - 1
        IF (NEFF.GT.LOWCAR) EFFN=(PROB(NEFF)-DELAYP)/
```

```
1      (PROB(JK)-PROB(NEFF)) +AEFF
       IF (NEFF.EQ.LOWCAR) EFFN=(1.0-DELAYP)/
1      (1.0-PROB(NEFF)) + RHO
       UNAVL = AMAX1(0.0,1-EFFN/CARS)
       GO TO 8
       UNAVL = 0.0
       EFFN= CARS


 ACCUMULATE TERMS FOR REGRESSION COEFFICIENTS

       C =  RHO/CARS
      SUMY= SUMY + UNAVL
      SUMYSQ= SUMYSQ + UNAVL*UNAVL
      SUMC = SUMC + C
      SUMCSQ = SUMCSQ + C*C
      SUMYC= SUMYC + UNAVL*C
      NOBS = NOBS+1
      GO TO 20

      WRITE(6,123) IK,PCTNM1,PCTNM2,PCTNM3
3     FORMAT(' ERROR IN DELAY DATA FOR OBS',I4,
1     ' IN ',3A4,' PRECINCT')
      CONTINUE


CALCULATE REGRESSION COEFFICIENTS

       YC= NOBS*SUMYC-SUMY*SUMC
       CC= NOBS*SUMCSQ-SUMC*SUMC
      B1= YC/CC
      B2= SUMY/NOBS - B1* SUMC/NOBS

      WRITE(6,106)  PCTNM1,PCTNM2,PCTNM3,B1,B2
)    FORMAT('0 FOR ',3A4,' PRECINCT  B1= ',F10.4,' B2= ',
1    F10.4)
      CONTINUE
    CALL EXIT
    END
```

Appendix C

PROGRAM CROSS-REFERENCE TABLE

Symbol	Defined in	Referenced in
ADDALC	ADDALC	MAIN
ADDCAR	ADDCAR	ADDALC
ADJUST	ADJUST	ADDALC
AVTT	AVTT	COMPTB,KNSTR
CEIL	CEIL	DERIVE,MEET,ADDALC
CKOVR	CKOVR	MEET,ADDALC,WRITE
COMPTB	COMPTB	DSPPDT,DSPDTP
DERIVE	DERIVE	READ,SET
DISP	DISP	MAIN
DSPDTP	DSPDTP	DISP
DSPPDT	DSPPDT	DISP
GETBOT	GETBOT	INIT,READ
GETTKN	GETTKN	SCAN
GETTOP	GETTOP	SCAN,ADDALC,WRITE
GTDSPC	GTDSPC	READ,LIST,DISP,SET,MEET,ADDALC,WRITE
INIT	INIT	MAIN
KEYWDS	DATA	MAIN,INIT,SCAN,READ,GTDSPC,DERIVE,LIST,SETWFL, DISP,DSPPDT,DSPDTP,SET,MEET,ADDALC,WRITE
KNSTR	KNSTR	MEET
LCODES	DATA	SCAN,GETTKN
LIST	LIST	MAIN
LKP1	LKP1	GETTKN,READ,GTDSPC,MRGORD
LKP8	LKP8	SCAN,READ,SETWFL,NXPCT
MEET	MEET	MAIN
MOVE	MOVE	SCAN,READ,MRGORD,WRITE
MRGORD	MRGORD	READ,DISP
NXDAY	NXDAY	LIST,DSPPDT,DSPDTP,SET,MEET,ADDALC,ADDCAR,WRITE
NXPCT	NXPCT	LIST,DSPPDT,DSPDTP,SET,MEET,ADDALC,ADDCAR,WRITE
NXTOUR	NXTOUR	LIST,DSPPDT,DSPDTP,SET,MEET,STRCAR,ADDALC, ADDCAR,WRITE
OBJFUN	OBJFUN	SBLOBJ,ADJUST
OBJF1	OBJF1	COMPTB,KNSTR,OBJFUN
OBJF2	OBJF2	COMPTB,KNSTR,OBJFUN
OBJF3	OBJF3	COMPTB,KNSTR,OBJFUN
OFFSET	DATA	READ,DERIVE,SBLACT,SBLEF,LIST,NXPCT,NXDAY, NXTOUR,DSPPDT,DSPDTP,COMPTB,AVTT,OBJF2, OBJF3,SET,MEET,STRCAR,KNSTR,ADDALC,SBLOBJ, STRDF,ADJUST,STROBJ,OBJFUN,ADDCAR,WRITE
PNTRS	DATA	INIT,READ,GTDSPC,DERIVE,SBLACT,SBLEF,LIST, SETWFL,NXPCT,NXDAY,NXTOUR,DSPPDT,DSPDTP, COMPTB,SET,MEET,STRCAR,KNSTR,CKOVR,ADDALC, SBLOBJ,STRDF,ADJUST,STROBJ,OBJFUN,ADDCAR, WRITE
PQUEUE	PQUEUE	OBJF1,OBJF2,OBJF3

Symbol	Defined in	Referenced in
PRTBL	PRTBL	DSPPDT,DSPDTP,TOTAL
READ	READ	MAIN
SBLACT	SBLACT	DERIVE,MEET,ADDALC
SBLEF	SBLEF	DERIVE,MEET,ADDALC
SBLOBJ	SBLOBJ	ADDALC,ADJUST,ADDCAR
SCAN	SCAN	MAIN,READ,GTDSPC,LIST,DISP,SET,MEET,ADDALC, WRITE
SCODES	DATA	MAIN,SCAN,READ,GTDSPC,LIST,DISP,SET,MEET, ADDALC,WRITE
SET	SET	MAIN
SETWFL	SETWFL	LIST,DISP,SET,MEET,ADDALC,WRITE
SKIP	SKIP	READ
STATS	DATA	READ,DISP,ZERO,COMPTB,PRTBL,TOTAL
STORE	DATA	MAIN,INIT,GETBOT,SCAN,GETTOP,READ,DERIVE, SBLACT,SBLEF,LIST,SETWFL,NXPCT,NXDAY,NXTOUR, DISP,DSPPDT,DSPDTP,COMPTB,AVTT,OBJF1,OBJF2, OBJF3,SET,MEET,STRCAR,KNSTR,CKOVR,ADDALC, SBLOBJ,STRDF,ADJUST,STROBJ,OBJFUN,ADDCAR, WRITE
STRCAR	STRCAR	MEET,ADDALC
STRDF	STRDF	STROBJ,ADDCAR
STROBJ	STROBJ	ADDALC,ADJUST,ADDCAR
SYSTEM	DATA	MAIN,INIT,GETBOT,SCAN,GETTKN,GETTOP,READ, GTDSPC,DERIVE,LIST,SETWFL,DISP,DSPPDT, DSPDTP,TITLE,PRTBL,TOTAL,OBJF2,SET,MEET, CKOVR,ADDALC,ADDCAR,WRITE
TITLE	TITLE	DSPPDT,DSPDTP
TOTAL	TOTAL	DSPPDT,DSPDTP
WRITE	WRITE	MAIN
ZERO	ZERO	DSPPDT,DSPDTP

Appendix D

ADDRESSES FOR FURTHER INFORMATION

1. For copies of the PCAM program on card or tape, answers to questions about the program, and information about related emergency service deployment models:

Dr. Jan M. Chaiken
The Rand Corporation
1700 Main Street
Santa Monica, California 90406
(213) 393-0411

2. Research sponsors:

U.S. Department of Housing and Urban Development

Alan Siegel, Director
Hartley Campbell Fitts, Program Manager

Office of Policy Development and Research
Community Development and Management
Research Division
451 Seventh Street, S.W.
Washington, D.C. 20410
(202) 755-6970

National Institute of Law Enforcement and Criminal Justice

Dr. Richard Linster
Law Enforcement Assistance Administration
National Institute of Law Enforcement and
Criminal Justice
Office of Evaluation
633 Indiana Avenue, N.W.
Washington, D.C. 20530
(202) 376-3933