sierra research

Report No. SR93-11-01b

Driving Cycle Development Software User's Guide

prepared for:

U.S. Environmental Protection Agency

November 10, 1993

prepared by:

Sierra Research, Inc.
1801 J Street
Sacramento, California 95814
(916) 444-6666

DRIVING CYCLE DEVELOPMENT SOFTWARE USER'S GUIDE

prepared for:

United States Environmental Protection Agency
in response to Work Assignment No. 1-10,
Contract No. 68-C1-0079

November 10, 1993

prepared by:

Thomas R. Carlson
John M. Lee

Sierra Research, Inc.
1801 J Street
Sacramento, CA 95814
(916) 444-6666

DRIVING CYCLE DEVELOPMENT SOFTWARE
USER'S GUIDE

Table of Contents

List of Figures

1. INTRODUCTION

Since the 1972 model year, passenger car and light truck emissions have been measured using a Federal Test Procedure (FTP) that includes a second-by-second driving trace or "driving cycle" called the "LA-4" cycle. Recent evidence indicates that the LA-4 cycle is not representative of the full range of driving conditions under which vehicles are operated (and produce emissions) in metropolitan areas. Using data collected from vehicles in customer service, Sierra Research has developed a set of driving cycles designed to initially supplement, and eventually replace, the current LA-4 test cycle.* This User's Guide describes a series of software programs which were used to generate the driving cycles.

This document is organized as follows.

Section 2 discusses the computer system requirements needed to operate the cycle development software.

Section 3 presents a conceptual overview of the software package. Summaries of the key programs in the package are given and the flow of program and data files are illustrated here.

Section 4 provides more detailed descriptions of each of the programs in the package and the data files they access.

Appendix A contains complete program code listings of all the software.

Appendix B contains several "sample" driving cycle development data files.

###

*T.C. Austin, et al., "Development of Driving Cycles to Represent Light-Duty Vehicle Operation in Urban Areas," Sierra Research, Inc., August 31, 1993.

2. SYSTEM REQUIREMENTS

Almost all the cycle development programs are written in VAX/VMS-FORTRAN77 (Version 6.0). A few short programs are written in SAS and VMS DCL (operating system command language). This software will operate with little initial configuration on a VAX mini-computer running the VMS operating system. The FORTRAN programs can be easily ported to any non-VAX system which has a "VAX-FORTRAN77" compatible compiler. The SAS and DCL programs are few in number and simple in design and can be readily re-coded on a non-VAX system. However, the amount of driving data used under this effort (requiring substantial disk storage) and the computational and memory intensive nature of several key programs severely limit the ability to port this package to a "typical" PC system.

The following system requirements are recommended:

- 32-bit CPU and operating system;
- minimum 24 megabytes (MB) of physical memory (RAM), with 64 MB recommended; and
- disk storage capacity of 2 gigabytes (GB).

###

3. SOFTWARE OVERVIEW

This section provides an overview of the software programs developed and used to construct driving cycles. A series of flow charts are presented to illustrate key programs and data flow. Individual program descriptions for all programs are given in Section 4.

Nearly 40 separate programs were written to perform cycle selection and development. Over 90% of these are written in VAX/VMS-FORTRAN (Version 6.0). The other non-FORTRAN programs are simple, short programs written in either SAS or VMS DCL (VAX operating system command language) that can easily be re-coded in another language on a non-VAX/VMS computer system.

The programs (and their associated data files) were organized into four separate phases:

(1). Data Preparation - Organization, consistency checking, reformatting and combining of Instrumented Vehicle and Chase Car driving databases;

(2). Start Cycle Selection - Evaluation and selection of the most representative initial (i.e., start) segments of each trip in the combined Baltimore/Los Angeles database;

(3). Composite Cycle Selection - Generation and evaluation of candidate "Composite" driving cycles representing all driving; and

(4). Bag 4/Remainder Cycle Selection - Generation and selection of additional candidate "Bag 4" and "Remainder" cycles representing "Outside FTP" and "All Driving minus Bag 4" travel distributions, respectively.

Summaries of the key programs and data flow in each of these four phases are presented in the following sub-sections. A key to symbols and notations used in the flow charts is given below.

Flow Chart Key

- Shaded boxes — represent programs. The language in which the program was written is given in parentheses.

- Circled numbers — indicate FORTRAN logical unit numbers associated with the data file beside which

they appear. Circled asterisks refer to system-default input and output logical units.

3.1 Data Preparation Phase

Two basic datasets were used in the cycle development: (1) Radian's Instrumented Vehicle (3-Parameter) database of driving data collected in Baltimore and Spokane and (2) Sierra's 1992 Los Angeles Chase Car driving database. Both of these databases attempted to capture representative samples of driving in each of the regions in which data were collected. However, due to the nature of each effort, the data cannot be trivially combined. For example, the Instrumented Vehicle (IV) data contain measured vehicle speed/time information from the moment the vehicle is started up. Time idling in a driveway or traveling on "neighborhood" streets is contained in the IV data. Conversely, Chase Car data does not contain this "trip end" travel.

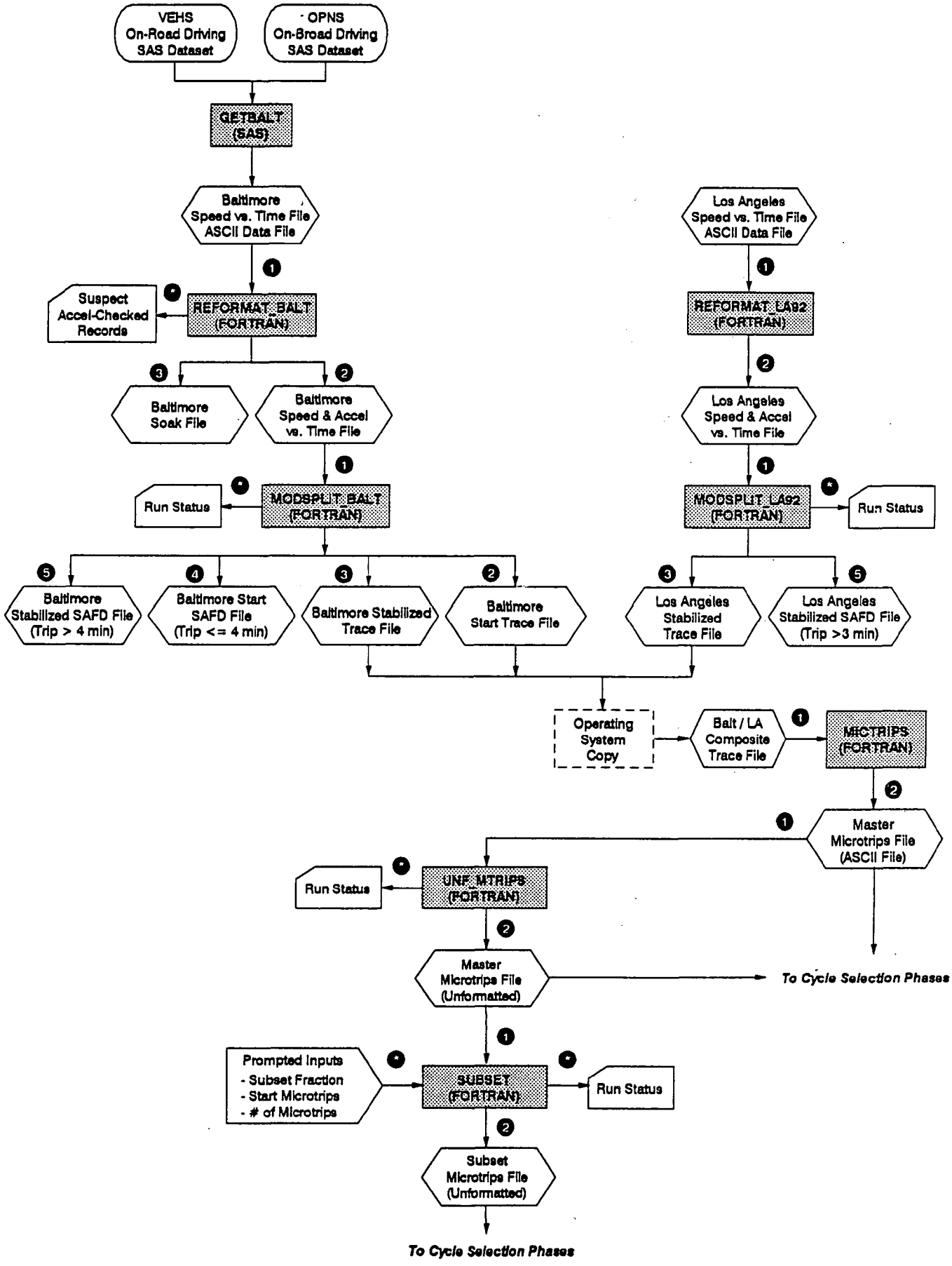
Figure 1 illustrates the programs that were used and the steps employed to produce a composite driving database and to re-organize the data into several forms used in subsequent cycle selection. First, the IV database (contained in SAS-datasets) was converted to a single ASCII file containing Baltimore-only records using the GETBALT program. REFORMAT_BALT was then used to compute second-by-second accelerations from the speeds and reformat the output records, retaining needed fields only. (A similar program, REFORMAT_LA92 was applied to the Los Angeles Chase Car dataset.) Programs titled MODSPLIT_BALT and MODSPLIT_LA92 were then used to break out the second-by-second driving data into "Start" and "Stabilized" files. As discussed above, the Chase Car (Los Angeles) database did not contain "trip end" travel. Based on trip start activity data collected and analyzed by Sierra,* it was estimated that one minute of trip end travel occurred prior to initiated trips in the Chase Car database. As a result, Baltimore data "Start" travel was defined as the first 4 minutes of each trip while Los Angeles data "Start" travel was defined as the initial 3 minutes (4 minus 1).

The driving data were then combined into a single file referred to in Figure 1 as "Balt/LA Composite Trace File". From this file, the speed/time traces for each trip were broken up into a series of "microtrips" using the MICTRIPS program. A microtrip is defined as travel between two points in a driving trace where the vehicle comes to a stop (zero velocity), including the leading idle period, if any. These microtrips, and their speed and acceleration profiles, were stored in the "Master Microtrips File" shown. The UNF_MTRIPS and SUBSET programs were then used to create a binary (i.e., unformatted) version of the Microtrips File and randomly select a user-sized subset of microtrips from it. The Master Microtrips File and it's randomly

*T.C. Austin, et al., "Characterization of Driving Patterns and Emissions from Light-Duty Vehicles in California," Draft Final Report, Sierra Research, Inc., March 5, 1993.

Figure 1

Data Preparation Phase



selected Subset File are then input to several cycle selection programs discussed in the following sub-sections.

3.2 Start Cycle Selection Phase

In generating representative driving cycles, it was recognized that a separate evaluation of the representativeness of the "start" portion of a candidate driving cycle (when cold engine/catalyst emission levels are elevated) was required. As a result, a separate selection process to identify those microtrips occurring at the beginning of a trip which best represented all "start" travel was developed.

Figure 2 shows the "Start Cycle Selection" flow chart. (Only Baltimore data were used to find representative start cycles due to the aforementioned lack of "trip ends" in the Los Angeles data.) The GETSTART program inputs the "All Start Travel" speed/acceleration frequency distribution (SAFD) file created earlier and the Baltimore Start Driving Trace File (second-by-second speed and acceleration) and computes and outputs "DIFFSUMS" for each trip start. DIFFSUM is the statistic of representativeness. DIFFSUM is defined as the summed differences between the speed/accel frequency distribution for an individual driving trace and a target population of driving. In this case, the individual driving traces are the microtrips that make up the first nominal 4 minutes of each trip and the target population is the composite SAFD of the first 4 minutes of all trips. Output DIFFSUMS by trip ID from GETSTART are then sorted by ascending DIFFSUM with the SORT_STARTS program. From this sorted list, the "best" trip ID is identified and the SEL_BESTART and SEL_START programs are used to generate a speed/time "Trace File" and "Start Microtrips File", respectively.

3.3 Composite Cycle Selection Phase

Similar to Start Cycle Selection, the selection of a Composite Cycle requires initially defining a "target" population speed/accel frequency distribution. As shown in Figure 3, the Composite Cycle Selection Phase begins by generating sample-weighted "Composite" and "Stabilized" target SAFD files using the COMPSURF and STABSURF programs. These programs input the city- and mode (start/stabilized)-specific SAFD files generated earlier. These frequency distributions (input as counts) must then be weighted to account for the fact that there are nearly 40 times more data in the Baltimore database than in the Los Angeles database. The COMPSURF and STABSURF programs were set up to achieve the following sample weighting:

- Stabilized travel from each city must be weighted equally (i.e., 50/50); and

- Start travel (in COMPSURF only) is then weighted into the composite with the frequency of "start" vs. "all driving" observations collected in Baltimore.

Figure 2

Start Cycle Selection Phase

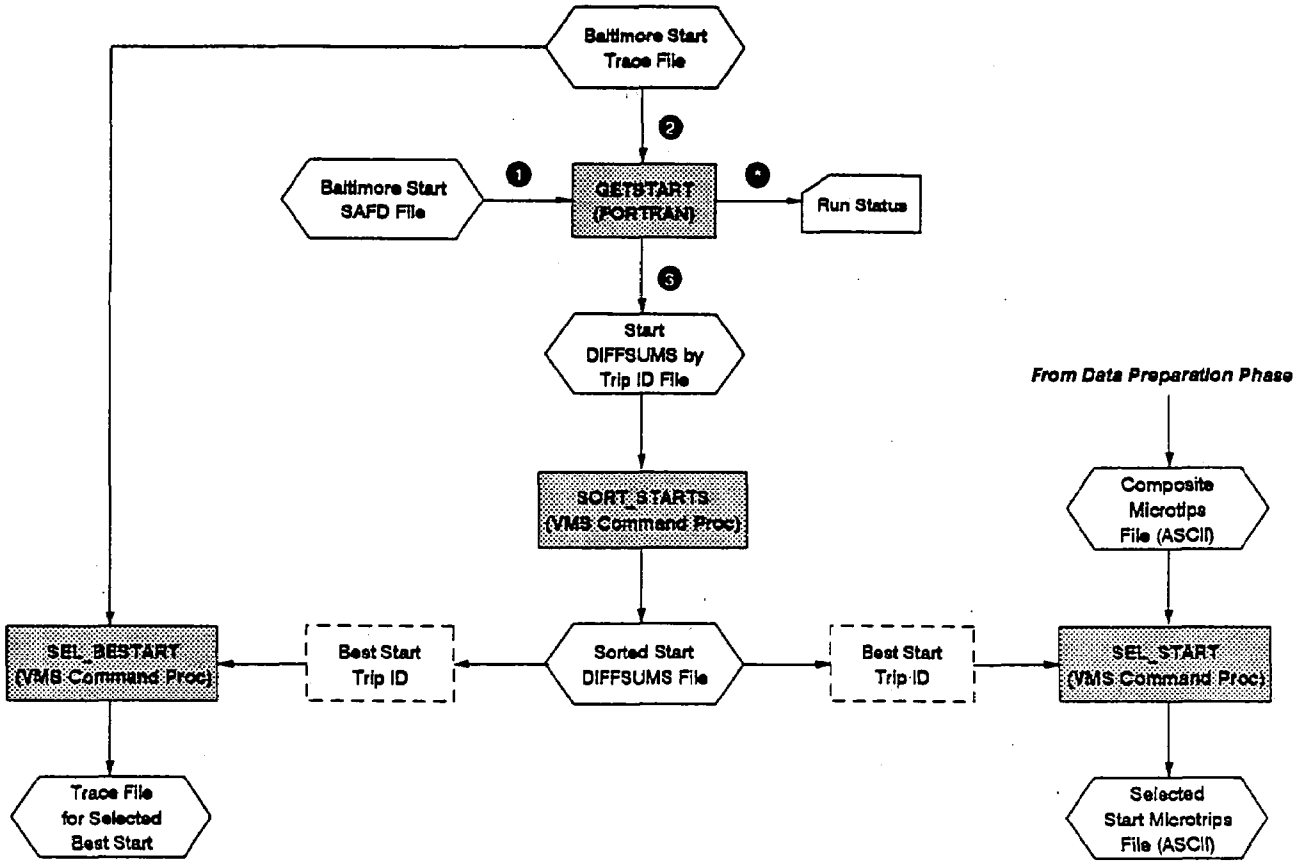
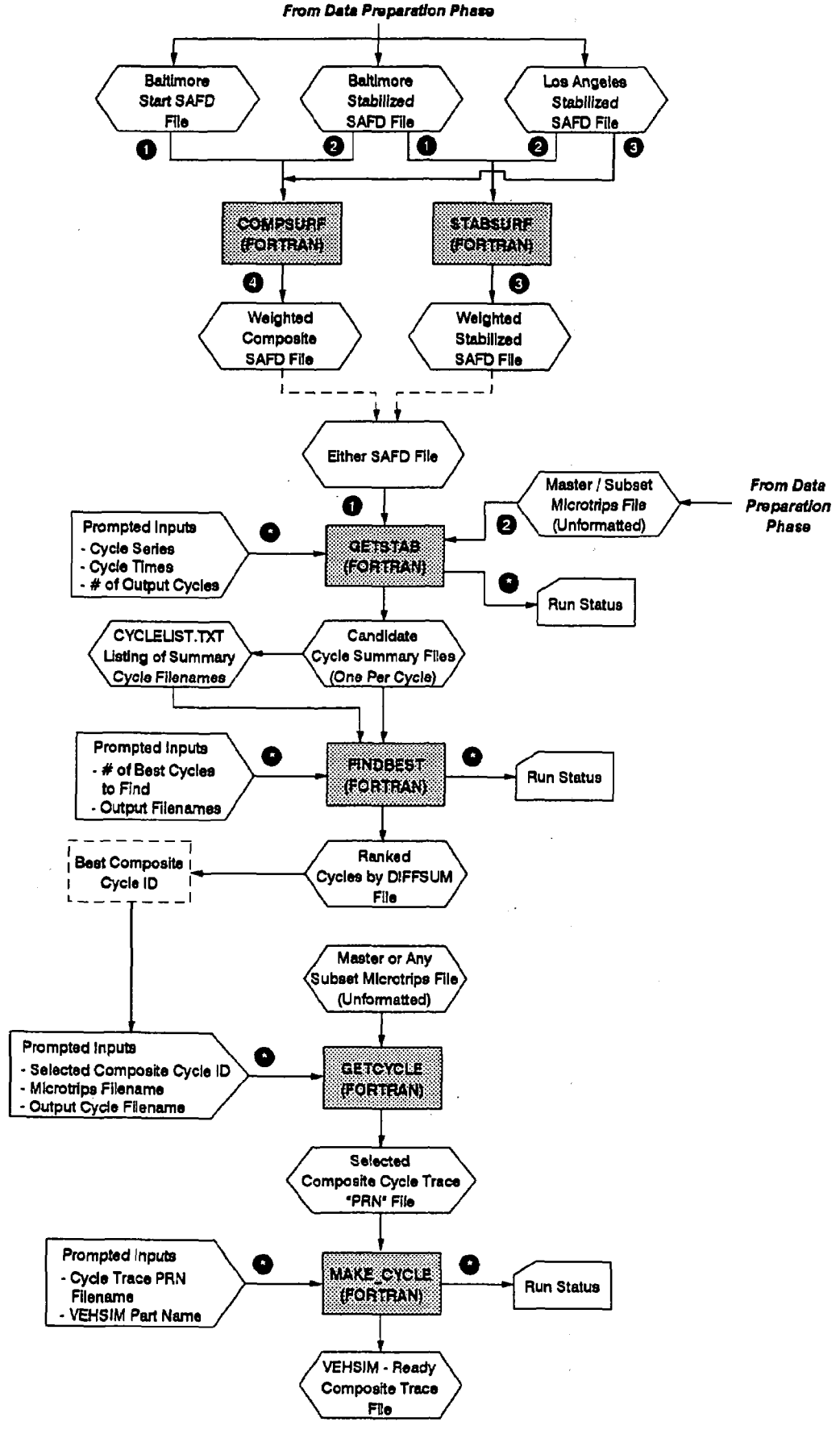


Figure 3

Composite Cycle Selection Phase



As shown, these target SAFD files representing composite (i.e., all driving) and stabilized travel populations are then input to GETSTAB. GETSTAB then generates a series (up to 1,000 per run) of candidate driving cycles and computes DIFFSUMs between each cycle and the target population SAFD. The microtrip IDs and DIFFSUMs for each cycle are output into small "Cycle Summary Files". These files are then scanned by the FINDBEST program, which ranks those cycles with the lowest DIFFSUMS and reports them in the "Ranked Cycles by DIFFSUM" File.

Once the most representative cycles have been found, their ID names are then fed to the GETCYCLE program which inputs the Master (or Subset) Microtrips File and retrieves the speed/time trace for each input cycle and stores them in files referred to as "PRN" files. In turn, the MAKE_CYCLE program converts these PRN files into VEHSIM-ready Driving Schedule Files.

3.4 Bag 4/Remainder Cycle Selection Phase

A number of the elements used to generate the "Bag 4" and "Remainder" cycles are similar to those used in Composite Cycle development. The Bag 4 cycle is generated first.

As illustrated in Figure 4, Bag 4 cycle development begins with the GETOUT75 program. GETOUT75 is a modified version of the GETSTAB program discussed earlier. GETOUT75 individually compares each microtrip in the input Microtrips File to a target speed/accel "Outside FTP" (i.e., travel beyond the bounds of the FTP) distribution. The 75 microtrips with the lowest DIFFSUMS are written to the "Best Outside LA4 Microtrips File". From this list, a single microtrip is selected and input to the GETOUTSEED program. GETOUTSEED performs identically to GETOUT75 program except the input microtrip is used as an initial "seed" microtrip and additional microtrips are incrementally added which, when added, yield the lowest DIFFSUM. These microtrips form the initial "Bag 4" cycle. As shown, GETCYCLE is then run on this collection of microtrips to generate speed/time "Cycle Trace PRN File". Manual editing of individual portions of this trace produced the final "Bag 4" cycle.

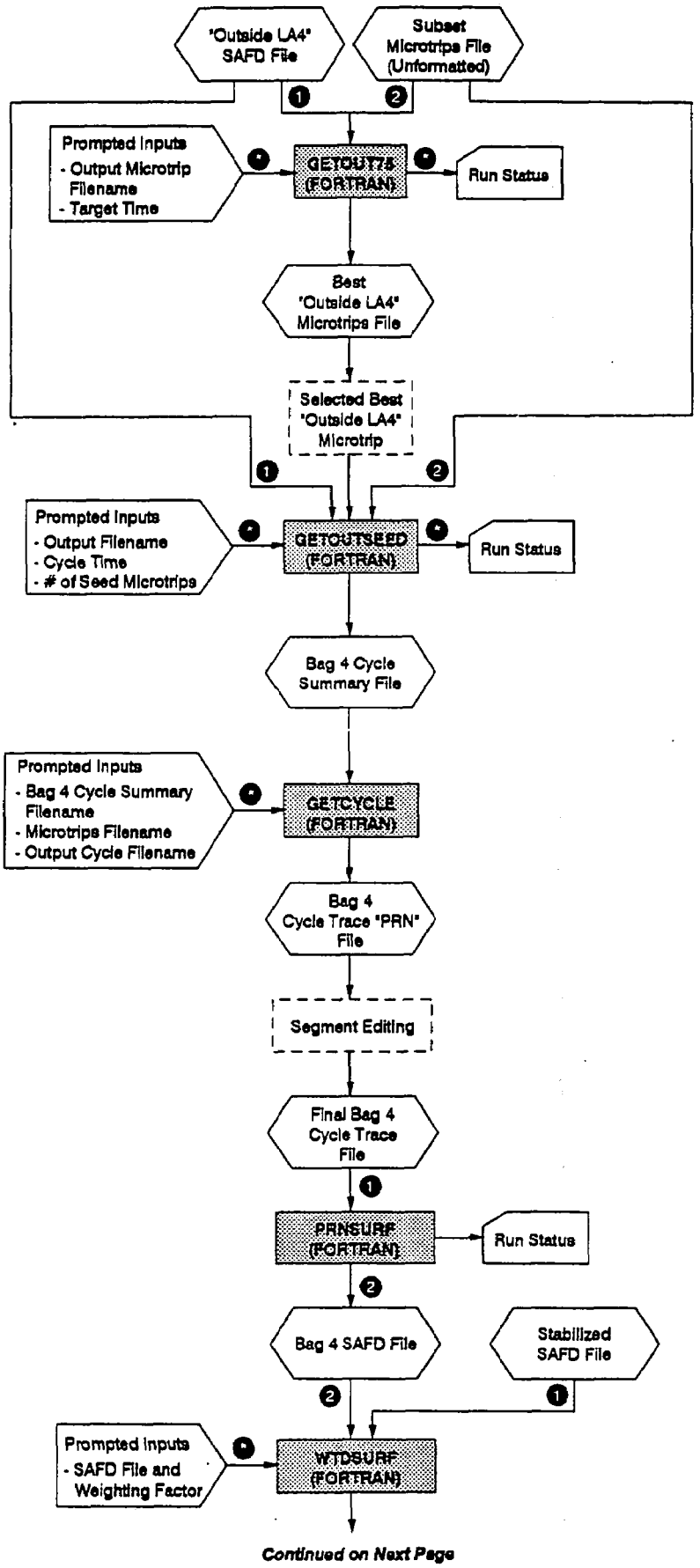
Remainder cycle development then began by converting the Bag 4 driving trace into a SAFD file with the PRNSURF program, as shown. WTDSURF then subtracts the Bag 4 SAFD file (with a weighting factor applied) from the Stabilized SAFD file to produce the Remainder Target SAFD File. The GETRMNDS program (a modified version of GETSTAB) then generates a series of candidate remainder cycles and computes DIFFSUMS based on the Remainder target speed/accel distribution.

Finally, as in the Composite Cycle Selection Phase, the FINDBEST and GETCYCLE programs are then executed to identify the best cycles and generate speed/time traces for them.

###

Figure 4

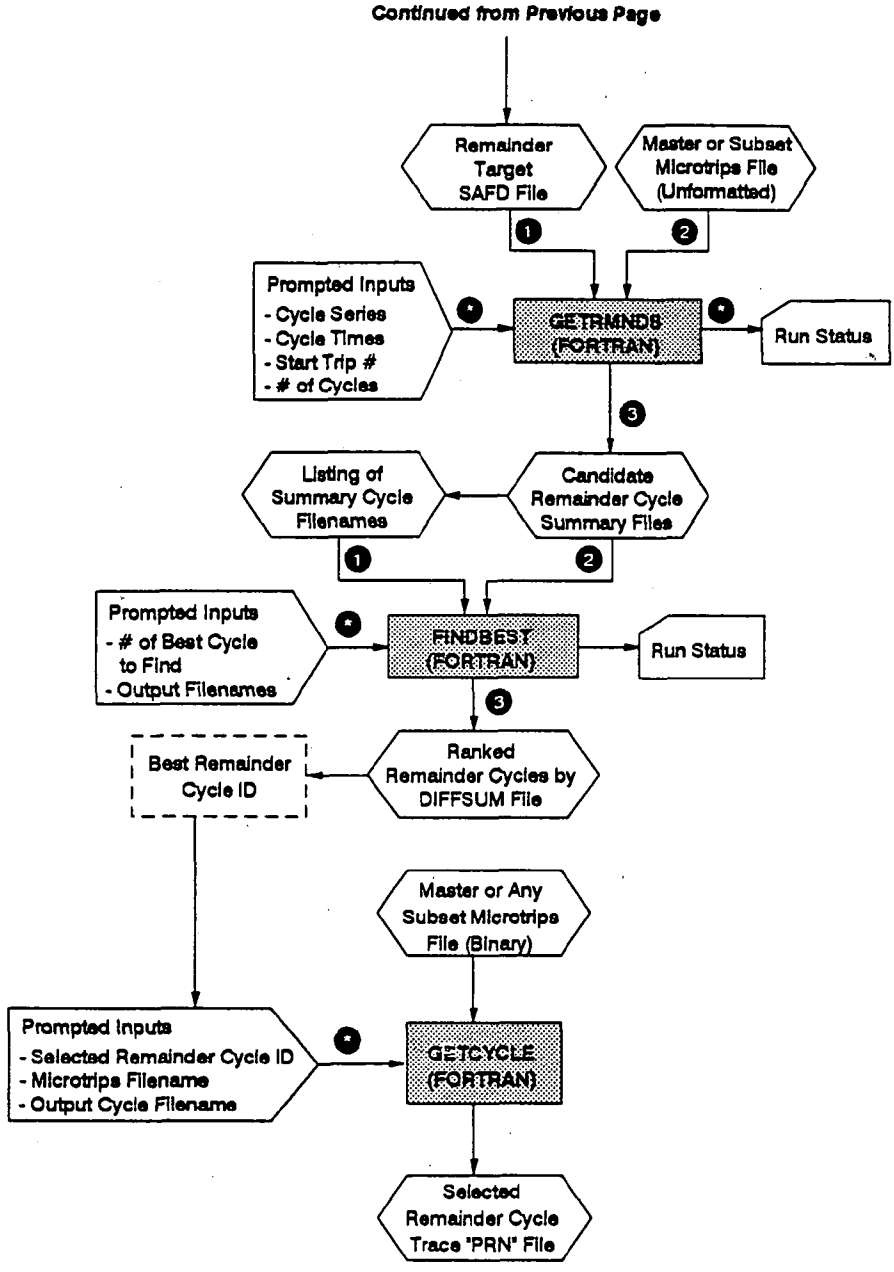
Bag 4 / Remainder Cycle Selection Phase



Continued on Next Page

Figure 4 continued...

Bag 4 / Remainder Cycle Selection Phase (continued)



4. PROGRAM DESCRIPTIONS

This section contains summaries of each of the programs used to generate driving cycles. The program summaries are ordered alphabetically to enable easy lookup. The program language (FORTRAN, SAS, etc.) and the cycle development "phase" in which the program was used (see Section 3) are shown in parenthesis. Several programs which were used but are not shown in the flow charts in Section 3 are described in this section. These programs are noted as "Unlisted". In addition, some of the programs were used in several cycle development phases. These programs are marked as "Utility".

Each summary consists of a description of the program, any user-prompted input and a list of the external input and output files accessed by the program. External files in FORTRAN programs are referenced by logical unit number (LUN). External files accessed by SAS program are listed by file reference name.

COMPARE (FORTRAN, Data Preparation-Unlisted)

This program was used to generate tabulated comparisons of the Baltimore "cold" and "hot" SAFD files and the Baltimore and Los Angeles stabilized SAFD files to determine if the speed/acceleration distributions for Balt-Cold vs. Balt-Hot and Balt-Stabilized vs. LA-Stabilized showed any significant differences. (The comparisons showed close similarity. Stabilized travel was nearly identical in both cities. Hot and cold start distributions were quite similar, except for longer initial idling for cold starts.)

External Files

LUN	I/O?	File Description
1	Input	Baltimore Cold Start SAFD File
2	Input	Baltimore Hot Start SAFD File
3	Input	Baltimore Stabilized SAFD File
4	Input	Los Angeles Stabilized SAFD File
5	Output	File containing tabular comparisons

COMPSURF (FORTRAN, Data Preparation)

COMPSURF produces a weighted composite speed/accel "SAFD" file from the Baltimore and Los Angeles Stabilized SAFD files and the Baltimore Start SAFD file. Weighting factors are necessary since there are roughly 40 times more data collected in Baltimore than Los Angeles. The

"stabilized" portions of the composite speed/accel distribution are generated by a 50/50 weighting of the Baltimore and Los Angeles stabilized driving. The "start" portion of the composite is based on Baltimore data only. The Baltimore start data are weighted into the composite with the same frequency of "start" vs. "total" travel observed in Baltimore.

The output composite speed/accel distribution is used as a "target" to compare against candidate driving cycles. Cycles are selected from a set of candidates based on how well their individual speed/accel distributions match that of the composite "All Driving" distribution.

External Files

LUN	I/O?	File Description
1	Input	Baltimore Start SAFD File
2	Input	Baltimore Stabilized SAFD File
3	Input	Los Angeles Stabilized SAFD File
4	Output	Composite "All Driving" SAFD File

DIFFSURF (FORTRAN, Utility)

The DIFFSURF program generates an output SAFD file from the difference between individual frequencies in two input SAFD files (first minus second). It is used in the cycle selection phases to compare the "fit" of one distribution to another, showing where the second ("Candidate") speed/accel distribution either over- or under-represents the frequencies in the first ("Target").

External Files

LUN	I/O?	File Description
1	Input	Any SAFD "Target Distribution" File
2	Input	Any SAFD "Candidate Distribution" File
3	Output	Candidate - Target SAFD File

DIFFSURFWT (FORTRAN, Utility)

This is a modified version of the DIFFSURF program. DIFFSURF generates an output SAFD file by taking a simple difference between frequencies at each speed/accel interval between two input SAFD files. In the example below, the 0.123 weighting factor represents the fraction of all travel that occurs beyond the boundaries of the LA4. DIFFSURFWT applies a weighting factor to the second input SAFD. Mathematically,

Output SAFD = Input SAFD1 - (Input SAFD2 x WtFac)

where WtFac = 0.123

External Files

LUN	I/O?	File Description
1	Input	Any SAFD "Target Distribution" File
2	Input	Any SAFD "Candidate Distribution" File
3	Output	Candidate - Target SAFD File

FINDBEST (FORTRAN, Data Preparation)

The program FINDBEST reads a directory listing of driving cycle files and finds the best cycles based on the DIFFSUM statistic. (DIFFSUM is the summed absolute difference between the cycle SAFD and target SAFD.) The directory listing file contains the ID names of each candidate drive cycle to be evaluated (one filename per record). From this input cycle ID list, FINDBEST retrieves cycle statistics from the Cycle Summary File for each cycle. The user inputs the number of "best" cycles to be output. The output file is a ranked list (by increasing DIFFSUM) of the top cycles. The program is dimensioned to scan up to 10,000 cycles per run.

External Files

LUN	I/O?	File Description
1	Input	Listing of candidate cycle ID names
2	Input	Candidate Cycle Summary Files
3	Output	Ranked "best" cycles list

Prompted Input

- Number of "best" cycles to list
- Output (ranked cycle list) filename
- Run title (appears at top of output list)

GETBALT (SAS, Data Preparation)

GETBALT is a short SAS program which is used to convert the Instrumented Vehicle (IV) Database (received as SAS-dataset files) into ASCII files. The program accesses 2 of the 4 IV datasets, VEHS and OPNS and filters and outputs Baltimore-only driving records (given by vehicle ID in the VEHS dataset) by VIN-matched merging of VEHS and OPNS records.

External Files

FILEREF	I/O?	File Description
RAD.VEHS	Input	Instr. Vehicle VEHS File (SAS-dataset)
RAD.OPNS	Input	Instr. Vehicle OPNS File (SAS-dataset)
DRIVDAT	Output	Baltimore Driving Database File (ASCII)

GETBIN (FORTRAN, Composite Cycle Selection-Unlisted)

The program GETBIN retrieves individual "Microtrip Profile" records from a Microtrips file which contain travel in speed/accel intervals (bins) specified by the user. If a microtrip contains travel within the input bin(s), GETBIN retrieves the speed vs. time data for that microtrip and outputs it. GETBIN was used to find candidate "high speed" microtrips which were used as "seed" microtrips in generating candidate Composite cycles.

External Files

LUN	I/O?	File Description
1	Input	Balt/LA Composite Master Driving Trace File
2	Input	Input file of speed/accel bin pairs of interest
3	Input	Master Microtrips File
4	Output	Speed/time trace of microtrips containing travel in the user-input speed/accel bins

GETCYCLE (FORTRAN, Composite and Bag 4/Remainder Cycle Selection)

The GETCYCLE program generates a second-by-second speed vs. time driving cycle trace from the list of microtrip ID's contained in a "Cycle Summary" output file. A comma-delimited, "SAFD" speed/accel distribution file for the cycle is also output.

External Files

LUN	I/O?	File Description
1	Input	Balt/LA Composite Master Driving Trace File
2	Input	User-Specified Cycle Summary File
3	Input	Master (or Subset) Microtrips File
9	Output	Speed vs. Time Driving Trace File
10	Output	Cycle SAFD File

GETMTRIP (FORTRAN, Utility-Unlisted)

GETMTRIP retrieves a speed time trace for a single microtrip from the Master Driving Trace File based on user-input start record number and number of seconds.

External Files

LUN	I/O?	File Description
1	Input	Balt/LA Composite Master Driving Trace File
2	Output	Speed vs. Time Driving Trace File

Prompted Input

- Starting record number
- Number of seconds (segments)

GETOUT75 (FORTRAN, Bag 4/Remainder Cycle Selection)

The program GETOUT75 reads a Microtrip file and the target speed/accel "Outside FTP" SAFD file and finds the 75 microtrips that individually "best" match the target speed/accel distribution, using the "Best Incremental" microtrip selection method. The "best" statistic is minimum DIFFSUM.

External Files

LUN	I/O?	File Description
1	Input	"Outside FTP" SAFD File
2	Input	Master or Subset Microtrips File
3	Output	Selected best individual microtrips and DIFFSUMS

GETOUTSEED (FORTRAN, Data Preparation)

GETOUTSEED is a modified version of GETOUT75 except the microtrip selection is "seeded" with the single best microtrip determined from executing GETOUT75 and then microtrips are added using the "Best Incremental" microtrips selection method.

External Files

LUN	I/O?	File Description
1	Input	"Outside FTP" SAFD File
2	Input	Master or Subset Microtrips File
3	Output	Selected best linked microtrips and DIFFSUMS

GETREST (FORTRAN, Composite Cycle Selection)

The program GETREST reads a Master or Subset Microtrips File and the target speed/acceleration composite SAFD file and builds and outputs a series (up to 1000 per run) candidate composite driving cycles using the "Hybrid" microtrip selection method.

GETREST is a modified version of the GETSTAB program described below in that it "seeds" each candidate cycle with user-input best start and best "outside FTP" microtrips determined prior to running GETREST.

External Files

LUN	I/O?	File Description
1	Input	Balt/LA Composite "All Driving" SAFD File
2	Input	Master or Subset Microtrips File
3	Output	Composite Cycle Summary Files (one per candidate cycle)

Prompted Input

- Output cycle series name
- Target cycle total time
- Target cycle "base" (random m-trip selection) time
- Start microtrip ID's
- "Outside FTP" microtrip ID's
- Number of candidate cycles to generate (up to 1,000)

GETRMNDS (FORTRAN, Bag 4/Remainder Cycle Selection)

GETRMNDS is a slightly modified version of the GETSTAB/GETREST programs used create candidate "Remainder" cycles by comparing the selected microtrips to a target Remainder speed/accel distribution. In the example shown below, the 0.2067 factor represents the fraction of stabilized travel outside the LA4 divided by the fraction of Long Bag 4 travel occurring outside the LA4. (This was used to generate a weighted surface for the remnant cycle.) The target Remainder distribution is defined as follows:

Remainder SAFD = Stabilized SAFD - (Long Bag 4 SAFD x WtFac)

where WtFac = 0.2067

External Files

LUN	I/O?	File Description
1	Input	Target Remainder SAFD File
2	Input	Master or Subset Microtrips File
3	Output	Remainder Cycle Summary Files (one per candidate cycle)

Prompted Input

- Output cycle series name
- Target cycle total time
- Target cycle "base" (random m-trip selection) time
- Number of candidate cycles to generate (up to 1,000)

GETSTAB (FORTRAN, Composite Cycle Selection)

The program GETREST reads a Master or Subset Microtrips File and the target speed/acceleration composite SAFD file and builds and outputs a series (up to 1000 per run) candidate composite driving cycle using the "Hybrid" microtrip selection method.

The Hybrid method adds an initial "base" of microtrips to the start "seed" of microtrips using random selection. Additional microtrips are then added with the Best Incremental technique.

The program algorithm is outlined below.

Step 1 - Generate pseudo-random seed value from current system time.

Step 2 - Generate a series of randomly selected microtrips from the population and add these to the start microtrip "seed" until the total linked microtrip time exceeds a user-specified subset TIMERAN of the target time limit TIMELIMIT. Compute match of each randomly added microtrip to the population surface. This set of microtrips forms the "base" sample.

Step 3 - Pick the "best" single microtrip of those remaining. "Best" is defined as the minimum summed difference between the overall surface and the base + incremental microtrips (i.e., DIFFSUM).

Step 4 - Iteratively test the remaining microtrips and add the one which coupled with the previously selected trip(s), best matches the overall surface. This incremental approach is repeated until the total linked microtrip time exceeds target time TIMELIMIT.

External Files

LUN	I/O?	File Description
1	Input	Balt/LA Composite "All Driving" SAFD File
2	Input	Master or Subset Microtrips File
3	Output	Composite Cycle Summary Files (one per candidate cycle)

Prompted Input

- Output cycle series name
- Target cycle total time (TIMELIMIT)
- Target cycle "base" (random m-trip selection) time (TIMERAN)
- Start microtrip IDs
- Number of candidate cycles to generate (up to 1,000)

GETSTART (FORTRAN, Start Cycle Selection)

GETSTART is another revised version of GETSTAB. This program reads the target "Start" SAFD file (the speed/accel distribution compiled from the first 4 minutes of each driving trip) and then compares the speed/accel

distribution of the microtrips at the beginning of each trip to it. Output consists of a file of DIFFSUMS by trip ID which is later sorted to find the best start microtrips. GETSTART rejects start trips that have less than 15 seconds of initial idle or contain long microtrips that terminate after 5 minutes from trip start.

External Files

LUN	I/O?	File Description
1	Input	Start Target SAFD File
2	Input	Start Trace File
3	Output	Start DIFFSUMS by Trip ID File

LA4SURF (FORTRAN, Utility-Unlisted)

This is a quick program used to generate the comma delimited, "LA4" SAFD file from the LA-4 speed vs. time driving trace.

External Files

LUN	I/O?	File Description
1	Input	LA-4 Speed vs. Time Driving Trace File
2	Output	LA4 SAFD File

MAKE_CYCLE (FORTRAN, Composite Cycle Selection)

The MAKE_CYCLE program is a short reformatting program that inputs a speed/time driving cycle trace "PRN" file created by the GETCYCLE program and converts it into a VEHSIM-ready "Driving Schedule" file.

External Files

LUN	I/O?	File Description
1	Input	Speed/Time Cycle Trace "PRN" File
2	Output	VEHSIM-ready Driving Schedule File

Prompted Input

- Driving Cycle Trace PRN filename
- VEHSIM part name
- VEHSIM Driving Schedule filename

MAKETBL (FORTRAN, Utility-Unlisted)

MAKETBL is a short program used to reformat comma-delimited SAFD files into two-way tabular form called a tabular "Watson Plot" (velocity-acceleration probability density function). Speed intervals are by column and acceleration intervals by row. Each element in the output

matrix shows the frequency of travel (expressed in percent) for that speed-acceleration interval combination. Modified versions of MAKETBL called MAKETBL2 and MAKETBL3 show row and column totals and net plus absolute totals.

External Files

LUN	I/O?	File Description
1	Input	Any comma-delimited SAFD File
2	Output	Tabular Watson Plot File

Prompted Input

- Title (appears at top of table)

MICTRIPS (FORTRAN, Data Preparation)

This program reads the Balt/LA Master Driving Trace File (second-by-second data) and develops "speed/acceleration profiles" by microtrip and outputs them into the Master Microtrips File. A microtrip is defined as travel between distinct stops. MICTRIPS scans the Balt/LA driving data and for each identified microtrip, computes the frequency distribution (expressed as number of seconds) of speed/accel for that microtrip.

External Files

LUN	I/O?	File Description
1	Input	Balt/LA Master Driving Trace File
2	Output	Master Microtrips File (ASCII)

MODSPLIT_BALT (FORTRAN, Data Preparation)

The MODSPLIT_BALT program inputs the entire Baltimore Speed & Accel vs. Time File and separates the second-by-second records into either of two output "mode" files: (1) Start and (2) Stabilized. Start mode is defined as the first 4 minutes of driving during each trip; stabilized mode is travel after 4 minutes. Speed and acceleration intervals (which are used extensively in the cycle development software) are computed and also written to the output files. The speed and acceleration interval are defined below.

Spd Int	Spd Range (mph)	Acl Int	Acl Range (mph/s)
0	0	...	...
1	> 0 to 5	-2	-2.5 to > -1.5
2	> 5 to 10	-1	-1.5 to > -0.5
3	>10 to 15	0	-0.5 to 0.5
4	>15 to 20	1	>0.5 to 1.5
5	>20 to 25	2	>1.5 to 2.5
...	...	...	...

MODSPLIT_BALT also generates and outputs speed/accel frequency distribution ("SAFD") files for all start and all stabilized Baltimore driving.

External Files

LUN	I/O?	File Description
1	Input	Baltimore Speed & Accel vs. Time File
2	Output	Baltimore Start Trace File
3	Output	Baltimore Stabilized Trace File
4	Output	Baltimore Start SAFD File
5	Output	Baltimore Stabilized SAFD File

MODSPLIT_LA92 (FORTRAN, Data Preparation)

This is a companion program to MODSPLIT_BALT except the Los Angeles data start time is defined as 3 minutes. (It was assumed that there was 1 minute of "trip start" travel not picked up in the Los Angeles Chase Car dataset.) In addition, only the stabilized portions of LA trips were kept. LA "start" travel was not used.

External Files

LUN	I/O?	File Description
1	Input	Los Angeles Speed & Accel vs. Time File
2	Output	Los Angeles Start Trace File
4	Output	Los Angeles Start SAFD File

OUTFTP (FORTRAN, Bag 4/Remainder Cycle Selection-Unlisted)

OUTFTP is a quick program used to generate an "Outside FTP" spd/accel distribution from the "Composite" and "LA4" input spd/accel distribution files. The output distribution contains only travel beyond the boundaries of the FTP (in speed/accel space) and is re-normalized.

A revised version of OUTFTP called OUTFTPSTAB was also written. The revised version substituted the "Stabilized" distribution file for the Composite "All Driving" input file.

External Files

LUN	I/O?	File Description
1	Input	Composite "All Driving" SAFD File
2	Input	LA4 SAFD File
3	Output	"Outside FTP" SAFD File
4	Output	Tabular "Outside FTP" Watson Plot File

PRNDIFF (FORTRAN, Utility-Unlisted)

PRNDIFF is a modified version of the SHOWDIFF program and inputs a "PRN" Driving Cycle Trace File and any target SAFD file and computes the difference between the input cycle speed/accel distribution and that of the target distribution. Output is in tabular "Watson Plot" form.

External Files

LUN	I/O?	File Description
1	Input	Driving Cycle Trace PRN File
2	Input	Any Target SAFD File
3	Output	Tabular "Watson Plot" of difference between PRN file and SAFD file

PRNSURF (FORTRAN, Bag 4/Remainder Cycle Selection)

PRNSURF is a short program used to generate a speed/accel "SAFD" file from an input Driving Cycle Trace PRN file.

External Files

LUN	I/O?	File Description
1	Input	Any Driving Cycle Trace PRN file
2	Output	Speed/Accel Distribution in "SAFD" comma-delimited format

REFORMAT_BALT (FORTRAN, Data Preparation)

This program reads the Baltimore second-by-second speed vs. time driving data file and reformats the fields and computes second-by-second accelerations. Computed accelerations between two adjacent speeds are assigned to the beginning of the time step.

In addition to the reformatted output driving data file, a second file containing computed soak times prior to each trip is also output.

A companion program, REFORMAT_LA92 performs the same function on the Los Angeles second-by-second driving data (except no "Soak Time" file is created).

External Files

LUN	I/O?	File Description
1	Input	Second-by-second speed vs. time driving data
2	Output	Speed & Accel vs Time File
3	Output	Soak Time by Trip ID File

SEL_BESTART (VAX-VMS Command Proc, Start Cycle Selection)

This is a VMS operating system command language procedure that employs a VAX system sorting utility to retrieve the second-by-second driving trace for the selected "best start" microtrips. The utility simply scans the trip ID field in the Master Driving Trace File and outputs the records for the selected trip that contained the best start microtrips. The output file was then manually edited to retain only the "start" microtrips.

SHOWDIFF (FORTRAN, Utility-Unlisted)

SHOWDIFF computes and outputs a tabular "Watson Plot" showing the speed/accel frequency differences between an input cycle and a target speed/accel distribution. SHOWDIFF generates the cycle SAFD from an input list of microtrips contained in a Cycle Summary file. The speed/accel frequencies for each of the microtrips in the input list are retrieved from the Master or Subset Microtrips File. This cycle SAFD is then compared to the target SAFD.

External Files

LUN	I/O?	File Description
1	Input	Any Cycle Summary File
2	Input	Master or Subset Microtrips File
3	Input	Target SAFD File
4	Output	Tabular "Watson Plot" of difference between cycle and target SAFD's

Prompted Input

- Cycle Summary filename
- Master or Subset Microtrips filename
- Target SAFD filename

SORT_STARTS (VAX-VMS Command Proc, Start Cycle Selection)

SORT_STARTS is a simple VMS command procedure that invokes the VAX system SORT utility to order the Start DIFFSUMS by Trip ID File by ascending DIFFSUM.

STABSURF (FORTRAN, Composite Cycle Selection)

STABSURF is a companion to the COMPSURF program. It inputs Baltimore and Los Angeles Stabilized SAFD files, weights their frequencies evenly (50/50) and outputs a weighted Balt/LA Stabilized SAFD file.

External Files

LUN	I/O?	File Description
1	Input	Baltimore Stabilized SAFD File
2	Input	Los Angeles Stabilized SAFD File
3	Output	Balt/LA Combined Stabilized SAFD File

SUBSET (FORTRAN, Data Preparation)

SUBSET generated a randomly selected subset of microtrips from the Master Microtrips File ("Unformatted" version) and outputs them to a "Subset" Microtrips file (with the same unformatted records structure). (This program was written and run in order to pare down the database of available microtrips for cycle selection to a manageable number based on Sierra's VAX physical memory limitations.) The user has the option to choose the size of the subset. The program also ensures that previously determined "best" start microtrips are not written to the Subset file.

External Files

LUN	I/O?	File Description
1	Input	Master Microtrips File (Unformatted)
2	Output	Subset Microtrips File (Unformatted)

Prompted Input

- Subset size (as a fraction)
- Start microtrip ID

UNF_MTRIPS (FORTRAN, Data Preparation)

This is a simple program that reads and converts the Master Microtrips File from a formatted ASCII file to an "unformatted" (binary) file. (The Master Microtrips File was converted to "unformatted" form for faster retrieval and more efficient disk storage.)

External Files

LUN	I/O?	File Description
1	Input	Formatted Master Microtrips File
2	Output	Unformatted Master Microtrips File

WTDSURF (FORTRAN, Bag 4/Remainder Cycle Selection-Unlisted)

The WTDSURF program computes and outputs a weighted SAFD file from any number of input SAFD files and weighting factors for each input SAFD.

External Files

LUN	I/O?	File Description
1	Input	Input SAFD Files
2	Output	Weighted SAFD File

Prompted Input

- Number of input SAFD files
- Weighting factors for each input SAFD file
- Input SAFD filenames

###

APPENDIX A

Program Code Listings

```
      PROGRAM PRNDIFF
C
C     A modified version of SHOWDIFF program, PRNDIFF inputs a
C     "PRN" cycle file and target surface file and computes
C     a difference spd/acl array in tabular form.
C
C     This program is primarily used to test the effects of editing
C     a cycle file by chopping/adding cruise, accel, etc.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):     0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):    -0.5 to 0.5 mph/s = 0
C                      -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                      0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     I/O:   Unit 1 (input)   - edited PRN cycle file.
C                               selected cycle.
C            Unit 2 (input)   - Speed/accel target surface created from
C                               sec-by-sec drive cycle data.
C            Unit 3 (output)  - Tabular "Watson Plot" of the spd/acl fit
C                               between selected cycle & target surface.
C
C     Author:  Tom Carlson
C     Date:    05/17/93
C-----------------------------------------------------------------------

      CHARACTER    MTYPE(50)
      CHARACTER*14 MTRIPID(50), ITRIPID
      CHARACTER*80 INFILE, TARGFILE, TITLE, OUTFILE


      REAL CFREQ(-20:20,0:20), ACLSUM(2,0:20), PFREQ(-20:20,0:20),
     +     SPDSUM(2,-20:20), PCTDIF(-20:20,0:20)

      DATA IHDR / 7 /
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for input and output filenames.
C
      WRITE (*, 10)
   10 FORMAT (/1X, 'Enter input PRN cycle filename:')
      READ (*, 20) INFILE
   20 FORMAT (A)
      WRITE (*, 30)
   30 FORMAT (/1X, 'Enter target surface filename:')
      READ (*, 20) TARGFILE

C     Name OUTFILE and TITLE based on INFILE and TARGFILE.

      L1 = INDEX (INFILE, ']') + 1
      L2 = INDEX (INFILE(L1:80), '.') + L1 - 2
      OUTFILE = INFILE(L1:L2) // '.DIF'
      K = INDEX (TARGFILE, '.') - 1
      TITLE = 'Difference Between Cycle ' // INFILE(L1:L2) //
     +        ' And Target Surface ' // TARGFILE(1:K)
      LL = INDEX (TITLE, '   ') - 1
      ILEN = 87 - LL/2
C
C     Open all files.
C
      OPEN (UNIT=1, FILE=INFILE, STATUS='OLD')
      OPEN (UNIT=2, FILE=TARGFILE, STATUS='OLD')
      OPEN (UNIT=3, FILE=OUTFILE, STATUS='NEW', RECL=186,
     +      CARRIAGECONTROL='LIST')
C
C-----------------------------------------------------------------------
C     Read target population surface (unit 2).
C-----------------------------------------------------------------------
C
C     Target population surface.
C
  100 READ (2, *, END=200) IACL, ISPD, RBIN, PCT
      PFREQ(IACL,ISPD) = PCT
      IN2 = IN2 + 1
      GO TO 100
C
```

```
C-----------------------------------------------------------------------
C     Get second by second trace from input PRN cycle file (unit 1).
C-----------------------------------------------------------------------
C
  200 READ (1, *, END=300) ITIME, NTRP, SPD, ACL
      N = N + 1
C
C-----------------------------------
C     Determine speed bin.
C-----------------------------------
C
      IF (SPD .LT. 0.1) THEN
      IS = 0
      GO TO 220
      END IF

      DO 210 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD .GT. I1 .AND. SPD .LE. I2) THEN
      IS = ISPD
      GO TO 220
      END IF

  210 CONTINUE
C
C-----------------------------------
C     Determine acceleration bin.
C-----------------------------------
C
  220 IF (ACL .GE. -0.5 .AND. ACL .LE. 0.5) THEN
      IA = 0
      GO TO 250
      END IF

      IF (ACL .GT. 0.0) GO TO 235

      DO 230 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL .LT. A1 .AND. ACL .GE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  230 CONTINUE

  235 DO 240 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL .GT. A1 .AND. ACL .LE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  240 CONTINUE
C
  250 CFREQ(IA,IS) = CFREQ(IA,IS) + 1
C
      GO TO 200
C
C-----------------------------------------------------------------------
C     Total bin count reads finished.  Compute cycle surface frequencies
C     and "difference" surface frequencies and output.
C-----------------------------------------------------------------------
C
  300 DO 310 IACL = -20,20
      DO 310 ISPD = 0,20
      CFREQ(IACL,ISPD) = CFREQ(IACL,ISPD) * 100 / N
      PCTDIF(IACL,ISPD) = CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD)
  310 CONTINUE
C
C     Compute row (across spd) and column (across acl) totals.
C
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      SPDSUM(1,IACL) = SPDSUM(1,IACL) + PCTDIF(IACL,ISPD)
```

```
      SPDSUM(2,IACL) = SPDSUM(2,IACL) + ABS (PCTDIF(IACL,ISPD))
  320 CONTINUE

      DO 325 ISPD = 0,20
      DO 325 IACL = -20,20
      ACLSUM(1,ISPD) = ACLSUM(1,ISPD) + PCTDIF(IACL,ISPD)
      ACLSUM(2,ISPD) = ACLSUM(2,ISPD) + ABS (PCTDIF(IACL,ISPD))
      TOTNET = TOTNET + PCTDIF(IACL,ISPD)
      TOTABS = TOTABS + ABS (PCTDIF(IACL,ISPD))
  325 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      WRITE (3, 330) TITLE(1:LL)
  330 FORMAT (///<ILEN>X,A//68X,'Speed/Accel Frequency Distribution (%)'
     +        ///78X, 'SPD BIN (mph)')
      WRITE (3, 340) (ISPD*5, ISPD = 0,20)
  340 FORMAT (166X,'   Net      Abs'/
     + 'ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), 2('   Totals')/
     + '---------------', 4X, '----', 3X, 20('----',3X), 2('   ------'))
C
      DO 350 IACL = -20,20
      WRITE (3, 345) IACL, (PCTDIF(IACL,ISPD), ISPD = 0,20),
     +               (SPDSUM(I,IACL), I = 1,2)
  345 FORMAT (6X, I4, 6X, 21F7.3, 5X, F7.3, 2X, F7.3)
  350 CONTINUE
C
      WRITE (3, 355) (ACLSUM(1,ISPD), ISPD = 0,20), TOTNET,
     +               (ACLSUM(2,ISPD), ISPD = 0,20), TOTABS
  355 FORMAT (/3X, 'Net Totals', 3X, 21F7.3, 5X, F7.3, 2X/
     +         3X, 'Abs Totals', 3X, 21F7.3, 14X, F7.3)
      WRITE (3, 360) N
  360 FORMAT (/1X, 'Cycle Time (sec): ', I4)
C
      STOP

      END
```

```
      PROGRAM PRNDIFF2
C
C     A modified version of SHOWDIFF program, PRNDIFF2 inputs
C     2 "PRN" cycle files and target surface file and computes
C     a difference spd/acl array in tabular form.
C
C     This program is primarily used to test the effects of editing
C     a cycle file by chopping/adding cruise, accel, etc.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     I/O:   Unit 1 (input)   - edited PRN cycle file.
C                               selected cycle.
C            Unit 2 (input)   - Speed/accel target surface created from
C                               sec-by-sec drive cycle data.
C            Unit 3 (output)  - Tabular "Watson Plot" of the spd/acl fit
C                               between selected cycle & target surface.
C
C     Author:  Tom Carlson
C     Date:    05/17/93
C-----------------------------------------------------------------------

      CHARACTER    MTYPE(50)
      CHARACTER*14 MTRIPID(50), ITRIPID
      CHARACTER*80 INFILE1, INFILE2, TARGFILE, TITLE, OUTFILE


      REAL CFREQ(-20:20,0:20), ACLSUM(2,0:20), PFREQ(-20:20,0:20),
     +     SPDSUM(2,-20:20), PCTDIF(-20:20,0:20)

      DATA IHDR / 7 /
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for input and output filenames.
C
      WRITE (*, 10)
   10 FORMAT (/1X, 'Enter input PRN cycle1 filename:')
      READ (*, 20) INFILE1
      WRITE (*, 15)
   15 FORMAT (/1X, 'Enter input PRN cycle2 filename:')
      READ (*, 20) INFILE2
   20 FORMAT (A)
      WRITE (*, 30)
   30 FORMAT (/1X, 'Enter target surface filename:')
      READ (*, 20) TARGFILE
      WRITE (*, 40)
   40 FORMAT (/1X, 'Enter Cycle 1 weight factor:')
      READ (*, *) WT1

C     Name OUTFILE and TITLE based on INFILE and TARGFILE.

      L1 = INDEX (INFILE1, ']') + 1
      L2 = INDEX (INFILE1(L1:80), '.') + L1 - 2
      L3 = INDEX (INFILE2, ']') + 1
      L4 = INDEX (INFILE2(L1:80), '.') + L1 - 2
      OUTFILE = INFILE1(L1:L2) // '.DF2'
      K = INDEX (TARGFILE, '.') - 1
      TITLE = 'Difference Between Cycles ' // INFILE1(L1:L2) // '/' //
     +        INFILE2(L3:L4) // ' And Target Surface ' // TARGFILE(1:K)
      LL = INDEX (TITLE, '   ') - 1
      ILEN = 87 - LL/2
C
C     Open all files.
C
      OPEN (UNIT=1, FILE=INFILE1, STATUS='OLD')
      OPEN (UNIT=2, FILE=INFILE2, STATUS='OLD')
      OPEN (UNIT=3, FILE=TARGFILE, STATUS='OLD')
      OPEN (UNIT=4, FILE=OUTFILE, STATUS='NEW', RECL=186,
     +      CARRIAGECONTROL='LIST')
C
C-----------------------------------------------------------------------
C     Read target population surface (unit 3).
```

```
C-----------------------------------------------------------------------
C
C     Target population surface.
C
  100 READ (3, *, END=200) IACL, ISPD, RBIN, PCT
      PFREQ(IACL,ISPD) = PCT
      IN3 = IN3 + 1
      GO TO 100
C
C-----------------------------------------------------------------------
C     Get second by second traces from input PRN cycle files (units 1
C     and 2).
C-----------------------------------------------------------------------
C
  200 READ (1, *, END=300) ITIME, NTRP, SPD, ACL
      N1 = N1 + 1
C
C----------------------------------
C     Determine speed bin.
C----------------------------------
C
      IF (SPD .LT. 0.1) THEN
      IS = 0
      GO TO 220
      END IF

      DO 210 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD .GT. I1 .AND. SPD .LE. I2) THEN
      IS = ISPD
      GO TO 220
      END IF

  210 CONTINUE
C
C----------------------------------
C     Determine acceleration bin.
C----------------------------------
C
  220 IF (ACL .GE. -0.5 .AND. ACL .LE. 0.5) THEN
      IA = 0
      GO TO 250
      END IF

      IF (ACL .GT. 0.0) GO TO 235

      DO 230 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL .LT. A1 .AND. ACL .GE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  230 CONTINUE

  235 DO 240 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL .GT. A1 .AND. ACL .LE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  240 CONTINUE
C
  250 CFREQ(IA,IS) = CFREQ(IA,IS) + WT1
C
      GO TO 200

C-----------------------------------------------------------------------

  300 READ (2, *, END=400) ITIME, NTRP, SPD, ACL
      N2 = N2 + 1
C
C----------------------------------
```

```
C     Determine speed bin.
C-----------------------------------
C
      IF (SPD .LT. 0.1) THEN
      IS = 0
      GO TO 320
      END IF

      DO 310 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD .GT. I1 .AND. SPD .LE. I2) THEN
      IS = ISPD
      GO TO 320
      END IF

  310 CONTINUE
C
C-----------------------------------
C     Determine acceleration bin.
C-----------------------------------
C
  320 IF (ACL .GE. -0.5 .AND. ACL .LE. 0.5) THEN
      IA = 0
      GO TO 350
      END IF

      IF (ACL .GT. 0.0) GO TO 335

      DO 330 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL .LT. A1 .AND. ACL .GE. A2) THEN
      IA = IACL
      GO TO 350
      END IF

  330 CONTINUE

  335 DO 340 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL .GT. A1 .AND. ACL .LE. A2) THEN
      IA = IACL
      GO TO 350
      END IF

  340 CONTINUE
C
  350 CFREQ(IA,IS) = CFREQ(IA,IS) + (1-WT1)
C
      GO TO 300
C
C-----------------------------------------------------------------
C     Total bin count reads finished.  Compute cycle surface frequencies
C     and "difference" surface frequencies and output.
C-----------------------------------------------------------------
C
  400 WTN = N1*WT1 + N2*(1-WT1)
      DO 410 IACL = -20,20
      DO 410 ISPD = 0,20
      CFREQ(IACL,ISPD) = CFREQ(IACL,ISPD) * 100 / WTN
      PCTDIF(IACL,ISPD) = CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD)
  410 CONTINUE
C
C     Compute row (across spd) and column (across acl) totals.
C
      DO 420 IACL = -20,20
      DO 420 ISPD = 0,20
      SPDSUM(1,IACL) = SPDSUM(1,IACL) + PCTDIF(IACL,ISPD)
      SPDSUM(2,IACL) = SPDSUM(2,IACL) + ABS (PCTDIF(IACL,ISPD))
  420 CONTINUE

      DO 425 ISPD = 0,20
      DO 425 IACL = -20,20
      ACLSUM(1,ISPD) = ACLSUM(1,ISPD) + PCTDIF(IACL,ISPD)
      ACLSUM(2,ISPD) = ACLSUM(2,ISPD) + ABS (PCTDIF(IACL,ISPD))
```

```
      TOTNET = TOTNET + PCTDIF(IACL,ISPD)
      TOTABS = TOTABS + ABS (PCTDIF(IACL,ISPD))
  425 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      WRITE (4, 430) TITLE(1:LL)
  430 FORMAT (///<ILEN>X,A//68X,'Speed/Accel Frequency Distribution (%)'
     +        ///78X, 'SPD BIN (mph)')
      WRITE (4, 440) (ISPD*5, ISPD = 0,20)
  440 FORMAT (166X,'   Net      Abs'/
     + 'ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), 2('   Totals')/
     + '---------------', 4X, '----', 3X, 20('----',3X), 2('   ------'))
C
      DO 450 IACL = -20,20
      WRITE (4, 445) IACL, (PCTDIF(IACL,ISPD), ISPD = 0,20),
     +               (SPDSUM(I,IACL), I = 1,2)
  445 FORMAT (6X, I4, 6X, 21F7.3, 5X, F7.3, 2X, F7.3)
  450 CONTINUE
C
      WRITE (4, 455) (ACLSUM(1,ISPD), ISPD = 0,20), TOTNET,
     +               (ACLSUM(2,ISPD), ISPD = 0,20), TOTABS
  455 FORMAT (/3X, 'Net Totals', 3X, 21F7.3, 5X, F7.3, 2X/
     +         3X, 'Abs Totals', 3X, 21F7.3, 14X, F7.3)
      NWT = NINT (WTN)
      NN = N1 + N2
      WRITE (4, 460) NWT, WT1, NN, 1-WT1
  460 FORMAT (/2X, 'Weighted Cycle Time (sec): ', I4, 5X,
     +        'Weightings:  Cycle 1 -> ', F5.3/
     +        1X, 'Aggregate Cycle Time (sec): ', I4, 18X,
     +         'Cycle 2 -> ', F5.3)
C
      STOP

      END
```

```
      PROGRAM PRNSURF
C
C     A quick program to generate the speed/accel surface for any "PRN"
C     speed vs time file.
C
C     Author:  Tom Carlson
C     Date:    05/25/93
C-----------------------------------------------------------------------
      INTEGER BINCNT(-20:20,0:20)
      REAL BINPCT(-20:20,0:20)
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW')
C
C     Begin processing.
C
  100 I = I + 1
      READ (1, *, END=110) ITIME, MTRP, SPD, ACL
C
C-----------------------------------
C     Determine speed bin.
C-----------------------------------
C
      IF (SPD .LT. 0.1) THEN
      IS = 0
      GO TO 220
      END IF

      DO 210 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD .GT. I1 .AND. SPD .LE. I2) THEN
      IS = ISPD
      GO TO 220
      END IF

  210 CONTINUE
C
C-----------------------------------
C     Determine acceleration bin.
C-----------------------------------
C
  220 IF (ACL .GE. -0.5 .AND. ACL .LE. 0.5) THEN
      IA = 0
      GO TO 250
      END IF

      IF (ACL .GT. 0.0) GO TO 235

      DO 230 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL .LT. A1 .AND. ACL .GE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  230 CONTINUE

  235 DO 240 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL .GT. A1 .AND. ACL .LE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  240 CONTINUE
C
  250 BINCNT(IA,IS) = BINCNT(IA,IS) + 1
      GO TO 100
C
C     Bin counting complete.  Print spd/acl counts and percentages for
C     each output file (cold, hot, stabilized).
C
  110 N = I - 1
      DO 120 IACL = -20,20
```

```
      DO 120 ISPD = 0,20
      BINPCT(IACL,ISPD) = BINCNT(IACL,ISPD) * 100. / N
      WRITE (2, 115) IACL, ISPD, BINCNT(IACL,ISPD), BINPCT(IACL,ISPD)
  115 FORMAT (2(I3,','), I8, ',', F10.6)
  120 CONTINUE
C
      CLOSE (1)
      CLOSE (2)
C
      WRITE (*, 130) N
  130 FORMAT (/1X, 'Processing completed for ', I5, ' records.'/)
C
      END
```

```
      PROGRAM REFORMAT_BALT
C
C     This program reads the second-by-second Radian Baltimore ASCII
C     driving data file, computes accels on a basis w/adopted ARB/EPA
C     convention and outputs those fields needed for cycle
C     selection (See companion program REFORMAT_LA92).
C
C     I/O:   Unit 1 (input)  - Sec-by-sec Balt ASCII driving data.
C            Unit 2 (output) - EPA_BALT.DAT output file.
C            Unit 3 (output) - BALTSOAK.DAT output file.
C
C     Author:  Tom Carlson
C     Date:    4/13/93
C
C-----------------------------------------------------------------------
      CHARACTER*4 VID, LVID
      INTEGER JTIME(3), JDATE(3), LDATE(3), TRIPFLAG
      REAL LSPD, LACL, NSPD
C
      DATA N /0/, ITRIP /1/, IVEH /1/
C-----------------------------------------------------------------------
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='NEW', CARRIAGECONTROL='LIST')
      OPEN (UNIT=3, STATUS='NEW', CARRIAGECONTROL='LIST')
C-----------------------------------------------------------------------

      JY = JULDAY (1,1,1992) - 1

  100 READ (1, 101, END=200) VID, (JDATE(I), I = 1,3),
     +                       (JTIME(I), I = 1,3), SPD
  101 FORMAT (A4, 19X, 3I2, 1X, 3(I2,1X), F6.2)
      N = N + 1
      IF (SPD .LT. 0.0) SPD = 0.0
C
C     Convert JTIME (HHMMSS) into seconds since midnight.
C
      JD = JULDAY (JDATE(2), JDATE(3), JDATE(1)+1900)
      JDAY = JD - JY
      MSEC = JDAY*86400 + JTIME(1)*3600 + JTIME(2)*60 + JTIME(3)
C
C     Set new trip flag, TRIPFLAG, when discontinuity observed in MSEC.
C
      IF (N .EQ. 1) THEN
      LVID = VID
      SOAK = 70.0
      WRITE (3, 105) VID, (JDATE(I), I = 1,3), ITRIP, SOAK
      GO TO 125
      END IF

      ISOAK = MSEC - LMSEC
      IF (ISOAK .GT. 30) THEN
         TRIPFLAG = 1
         SOAK = ISOAK / 60.
      END IF
C
C     Test if new vehicle or trip.
C
      IF (VID .NE. LVID) GO TO 130
      IF (TRIPFLAG .EQ. 1) GO TO 140
  110 ACL = SPD - LSPD
C
C     Accel computed from current and last speeds.  Check that it's
C     reasonable:
C
C
C
C     If computed accel is not OK.  Then "synthesize" it from an
C     interpolation of last and next speeds.
C
      IF (ABS (ACL) .GT. 12.0) THEN
         READ (1, 111) NSPD
  111    FORMAT (39X, F6.2)
         BACKSPACE 1
         IF (SPD .EQ. 0.0 .AND. NSPD .GT. 0.0) THEN
            XSPD = (LSPD + NSPD) / 2
            XACL = XSPD - LSPD
         WRITE (*, 112) N, LSPD, SPD, NSPD, ACL, XSPD, XACL
  112    FORMAT (I10, 6F8.2)
         SPD = XSPD
         ACL = XACL
```

```
         GO TO 115
         END IF
         WRITE (*, 112) N, LSPD, SPD, NSPD, ACL
      END IF
C
  115 WRITE (2, 120) LVID, (LDATE(I), I = 1,3), ITRIP, LSEC, LSPD, ACL,
     +               LSPD, ACL
  120 FORMAT (A4, 4I2.2, I6, 4F6.2)
  125 LSEC = ISEC
      ISEC = ISEC + 1
      LDATE(1) = JDATE(1)
      LDATE(2) = JDATE(2)
      LDATE(3) = JDATE(3)
      LMSEC = MSEC
      LSPD = SPD
      GO TO 100
C
C     New vehicle (and therefore new trip) section.
C
  130 ACL = 0.0
      WRITE (2, 120) LVID, (LDATE(I), I = 1,3), ITRIP, LSEC, LSPD, ACL,
     +               LSPD, ACL
      ITRIP = 1
      IVEH = IVEH + 1
      LVID = VID
      SOAK = 70.0
      WRITE (3, 105) VID, (JDATE(I), I = 1,3), ITRIP, SOAK
  105 FORMAT (A4, 4I2.2, F6.1)
      GO TO 145
C
C     New trip section.
C
  140 ACL = 0.0
      WRITE (2, 120) LVID, (LDATE(I), I = 1,3), ITRIP, LSEC, LSPD, ACL,
     +               LSPD, ACL
      IF (JDATE(3) .NE. LDATE(3)) THEN
         ITRIP = 1
      ELSE
         ITRIP = ITRIP + 1
      END IF
      WRITE (3, 105) VID, (JDATE(I), I = 1,3), ITRIP, SOAK
  145 TRIPFLAG = 0
      ISEC = 0
      GO TO 125
C
  200 ACL = 0.0
      WRITE (2, 120) VID, (JDATE(I), I = 1,3), ITRIP, ISEC, SPD, ACL,
     +               SPD, ACL
      END

      FUNCTION julday (mm, dd, yyyy)
C
C     This function returns the Julian Day Number beginning at noon of
C     the input calendar date specified by month mm, day dd, and year
C     yyyy (all integer variables).  Positive year indicates A.D.;
C     negative, B.C.  Remember the year after 1 B.C. was 1 A.D.
C
C     Source:  Numerical Recipes (2nd Edition), 1992.
C
      INTEGER julday, mm, dd, yyyy, igreg, ja, jm, jy
      PARAMETER (igreg=15+31*(10+12*1582))
C
      jy = yyyy
      if (jy .eq. 0) pause 'JULDAY:  There is no year zero.'
      if (jy .lt. 0) jy = jy + 1

      if (mm .gt. 2) then
         jm = mm + 1
      else
         jy = jy - 1
         jm = mm + 13
      end if

      julday = int (365.25*jy) + int (30.6001*jm) + dd + 1720995

      if (dd + 31*(mm + 12*yyyy) .ge. igreg) then
         ja = int (0.01*jy)
         julday = julday + 2 - ja + int (0.25*ja)
      end if
```

```
return
END
```

```
      PROGRAM REFORMAT_LA92
C
C     This program reads the second-by-second LA92_SIERRA driving cycle
C     file, re-computes accels to a consistent basis w/adopted ARB/EPA
C     convention and outputs only those fields needed for cycle
C     selection.
C
C     I/O:   Unit 1 (input)  - Sec-by-sec LA92_SIERRA drive cycle data.
C            Unit 2 (output) - EPA_LA92.DAT output file.
C
C     Author:  Tom Carlson
C     Date:    4/13/93
C
C-----------------------------------------------------------------------
      CHARACTER*7  ROUTEID(101000)
      INTEGER ISEC(101000)
      REAL PSPD(101000), CSPD(101000)
C
      DATA N /1/
C-----------------------------------------------------------------------
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='NEW', CARRIAGECONTROL='LIST')
C-----------------------------------------------------------------------

  100 READ (1, 101, END=200) ROUTEID(N), ISEC(N), PSPD(N), CSPD(N)
  101 FORMAT (A7, T11, I6, T27, F6.2, T48, F6.2)
      N = N + 1
      IF (PSPD(N) .LT. 0.0) PSPD(N) = 0.0
      IF (CSPD(N) .LT. 0.0) CSPD(N) = 0.0
      GO TO 100
C
C     All data input. Compute accel (mph/s) and apply to BEGINNING of
C     time interval.  Also detect occurrences of new routes being
C     started to handle accels at route starts - don't incorrectly
C     compute accel from the last obs from previous route and first obs
C     from current route.
C
  200 DO 300 I = 1,N

      IF (ROUTEID(I+1) .NE. ROUTEID(I) .OR. I .EQ. N) THEN
      CACL = 0.0
      PACL = 0.0
      ELSE
      PACL = PSPD(I+1) - PSPD(I)
      CACL = CSPD(I+1) - CSPD(I)
      END IF

      READ (ROUTEID(I)(6:7), FMT='(BN,I2)') ITRP

      WRITE (2, 250) ROUTEID(I)(1:4), ITRP, ISEC(I), PSPD(I), PACL,
     +               CSPD(I), CACL
  250 FORMAT ('L00192', A, I2.2, I6, 4F6.2)

  300 CONTINUE

      END
```

```
      PROGRAM SBAG4
C
C     SBAG4 (Short Bag 4) retrieves the m-trip profile information for
C     a user input cycle and finds the subset of m-trips which best
C     represent non-FTP operation.  This logic is laid out as follows:
C
C        (1).  A target "non-FTP" spd/acl surface is generated from the
C              difference between the composite "All Driving" surface
C              and the LA4 surface (CFRQT.DAT - LA4FRQ.DAT).
C
C        (2).  Two subset arrays are set up:  Non-FTP and Quasi-FTP
C
C        (3).  The m-trips at the beginning of the input cycle which are
C              designated as start m-trips are placed into the
C              Quasi-FTP array.
C
C        (4).  All remaining non-start m-trips are then iteratively
C              added to the Non-FTP subset IN ALL POSSIBLE COMBINATIONS
C              and each combination is tested against the non-FTP target
C              surface. (see subroutine COMBOS).
C
C        (5).  Combinations of all sizes from 1 to N-MS+1 are
C              considered (N=# of cycle m-trips, NS=# of start m-trips).
C
C        (6).  Top NBEST m-trips combinations (based on minimum DIFFSUM)
C              for Non-FTP and Quasi-FTP are saved and output.
C
C     I/O:   Unit 1 (input)  - "All Driving" composite surface.
C            Unit 2 (input)  - LA4 surface.
C            Unit 3 (input)  - Selected M-trips & goodness stats file.
C            Unit 4 (input)  - M-trips profile file (binary).
C            Unit 5 (output) - non-FTP surface.
C            Unit 6 (output) - Listing of the best m-trip combinations
C                              in the Non-FTP subset.
C            Unit 7 (output) - Best Short Bag 4 spd/acl surfaces.
C            Unit 7 (output) - Best Long Bag 4 spd/acl target surfaces.
C
C
C     Author:  Tom Carlson
C     Date:    5/4/93
C
C-----------------------------------------------------------------------
C
C     Set run size.  MM ----> Max # of m-trips allowed in a cycle.
C                    MS ----> Max # of start m-trips allowed in a cycle.
C                    NBST --> # of "best" short bag 4 combos to output.
C                    NMAX --> # of total m-trip combinations allowed.
C
      PARAMETER (MM = 25, MS = 10, NBST = 10, NMAX = 5000)
C
      CHARACTER    TYPE(MM), TYPES(MS)
      CHARACTER*2  CNUM
      CHARACTER*14 MTRIPID(MM), KTRIPID, MTRIPIDS(MS)
      CHARACTER*25 CYCFILE, B4SFILE, B4TFILE

      INTEGER*2 MCNT(MM,-20:20,0:20), KTIME, ITIMES(MS), MTRPS(MS), KK,
     +          ITIME(MM), KCNT(-20:20,0:20), MTRP(MM),
     +          LISTBST(NBST,MM)
      INTEGER*4 IREC(MM), KREC, IRECS(MS), KOMBOS(MM,NMAX)

      REAL SPD_MEAN(MM), SPD_STD(MM), ACL_STD(MM), DIFFBST(NBST),
     +     FFREQ(-20:20,0:20), SPD_MEANS(MS), SPD_MAXS(MS),
     +     SPD_MAX(MM), NFFREQ(-20:20,0:20), TIMEBST(NBST),
     +     TFREQ(-20:20,0:20), B4S(-20:20,0:20), B4L(-20:20,0:20),
     +     B4T(-20:20,0:20), B4SBST(-20:20,0:20,NBST),
     +     B4TBST(-20:20,0:20,NBST)

      DATA DIFFBST / NBST*999. /

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='OLD')
      OPEN (UNIT=4, STATUS='OLD', RECL=1760, FORM='UNFORMATTED')
      OPEN (UNIT=5, STATUS='NEW', CARRIAGECONTROL='LIST')
      OPEN (UNIT=6, STATUS='NEW', CARRIAGECONTROL='LIST')

C-----------------------------------------------------------------------
C    Begin execution -- Input required data.
```

```
C-----------------------------------------------------------------------
C
C     Read in total "All Driving" surface (unit 1).
C
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      TFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in LA4 surface (unit 2).  While reading compute the "non-FTP"
C     surface.
C
   20 READ (2, *, END=30) IACL, ISPD, NFREQ, PCT
      FFREQ(IACL,ISPD) = PCT / 100
      NFFREQ(IACL,ISPD) = TFREQ(IACL,ISPD) - FFREQ(IACL,ISPD)
      IN2 = IN2 + 1
      GO TO 20
C
C     Read in selected cycle stats file (unit 3).
C
   30 READ (3, 31) CYCFILE
   31 FORMAT (//23X, A)
      IE = INDEX (CYCFILE, '.') - 1

      DO 32 ISKIP = 1,4
      READ (3, FMT='()')
   32 CONTINUE

   33 IN3 = IN3 + 1
      READ (3, 34, ERR=35) N, TYPE(N), MTRP(N), MTRIPID(N), IREC(N),
     +                     ITIME(N), SPD_MEAN(N), SPD_MAX(N)
   34 FORMAT (I3, 1X, A1, 1X, I5, 1X, A14, 1X, I7, 1X, I4, 2(1X, F7.2))
      GO TO 33

   35 N = IN3 - 1
C
C     Now detect and move the start m-trips to the back of the pack so
C     the COMBOS routine can be called over the range 1 to N-NS.
C
      DO 36 I = 1,N
      IF (TYPE(I) .NE. 'S') THEN
      NS = I - 1
      GO TO 37
      END IF
      MTRPS(I) = MTRP(I)
      TYPES(I) = TYPE(I)
      MTRIPIDS(I) = MTRIPID(I)
      IRECS(I) = IREC(I)
      ITIMES(I) = ITIME(I)
      SPD_MEANS(I) = SPD_MEAN(I)
      SPD_MAXS(I) = SPD_MAX(I)
   36 CONTINUE
C
   37 M = N - NS
C
      DO 38 I = 1,M
      MTRP(I) = MTRP(I+NS)
      TYPE(I) = TYPE(I+NS)
      MTRIPID(I) = MTRIPID(I+NS)
      IREC(I) = IREC(I+NS)
      ITIME(I) = ITIME(I+NS)
      SPD_MEAN(I) = SPD_MEAN(I+NS)
      SPD_MAX(I) = SPD_MAX(I+NS)
   38 CONTINUE

      DO 39 I = M+1,N
      MTRP(I) = MTRPS(I-M)
      TYPE(I) = TYPES(I-M)
      MTRIPID(I) = MTRIPIDS(I-M)
      IREC(I) = IRECS(I-M)
      ITIME(I) = ITIMES(I-M)
      SPD_MEAN(I) = SPD_MEANS(I-M)
      SPD_MAX(I) = SPD_MAXS(I-M)
   39 CONTINUE
C
C     Read in microtrip profiles binary records one at a time (unit 4).
C
   40 READ (4, END=50) KK, KTRIPID, KREC, KTIME, SPDMEAN, SPDMAX,
     +       SPDSTD, ACLSTD, ((KCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)
      IN4 = IN4 + 1
```

```
C
C     Check the m-trip # of each record & keep only those records which
C     are found in the MTRP list read from unit 3 for the input cycle.
C
      DO 45 I = 1,N

      IF (KK .EQ. MTRP(I)) THEN
      NN = NN + 1
      SPD_STD(I) = SPDSTD
      ACL_STD(I) = ACLSTD
      DO 42 ISPD = 0,20
      DO 42 IACL = -20,20
      MCNT(I,IACL,ISPD) = KCNT(IACL,ISPD)
   42 CONTINUE
      IF (NN .EQ. N) GO TO 50
      GO TO 40
      END IF

   45 CONTINUE

      GO TO 40
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Now call subroutine COMBOS within a loop which
C     controls the number of m-trips to choose (1 to N-NS) to generate
C     all possible combinations of m-trips.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
   50 DO 300 K = 1,M

      NK = 0
      CALL COMBOS (M, K, KOMBOS, NK)
C
C     Loop thru all combinations generated for size K.
C
      DO 200 J = 1,NK
C
C     Initialize sums and counter before each combo is tested.
C
      DIFFS = 0.0
      TIMES = 0.0
      TIMEL = 0.0
      DO 55 IACL = -20,20
      DO 55 ISPD = 0,20
      B4S(IACL,ISPD) = 0.0
      B4L(IACL,ISPD) = 0.0
   55 CONTINUE
C
C     Loop thru and process bin counts for each m-trip in current combo.
C
      DO 60 I = 1,K

      ITRP = KOMBOS(I,J)
      TIMES = TIMES + ITIME(ITRP)

      DO 60 IACL = -20,20
      DO 60 ISPD = 0,20
      B4S(IACL,ISPD) = B4S(IACL,ISPD) + MCNT(ITRP,IACL,ISPD)
   60 CONTINUE
C
C     Compute the summed diff DIFFS for the current combo.
C
      DO 65 IACL = -20,20
      DO 65 ISPD = 0,20
      B4S(IACL,ISPD) = B4S(IACL,ISPD) / TIMES
      DIFFS = DIFFS + ABS (B4S(IACL,ISPD) - NFFREQ(IACL,ISPD))
   65 CONTINUE
C
      IF (K .EQ. M) GO TO 85
C
C     OK, evaluation of non-FTP for current combo complete.  Now get
C     the remaining m-trips in the input PCC which are NOT in the
C     current combo.
C
      DO 80 L = 1,M

      DO 70 I = 1,K
      IF (L .EQ. KOMBOS(I,J)) GO TO 80
```

```
   70 CONTINUE
C
C     IF statement always false -- we've found a leftover m-trip.
C
      ITRP = L
      TIMEL = TIMEL + ITIME(ITRP)

      DO 75 IACL = -20,20
      DO 75 ISPD = 0,20
      B4L(IACL,ISPD) = B4L(IACL,ISPD) + MCNT(ITRP,IACL,ISPD)
   75 CONTINUE

   80 CONTINUE
C
C     All leftovers processed.  Don't forget to include "trailing"
C     start m-trips.
C
   85 DO 90 I = M+1,N

      ITRP = I
      TIMEL = TIMEL + ITIME(ITRP)

      DO 90 IACL = -20,20
      DO 90 ISPD = 0,20
      B4L(IACL,ISPD) = B4L(IACL,ISPD) + MCNT(ITRP,IACL,ISPD)
   90 CONTINUE
C
C     Compute the new target Long Bag 4 surface from leftover m-trips:
C
C     Long Bag4 Target Surface = "All Driving" - "Leftover" Bag 4
C
C          where "Leftover" Bag 4 = PCC - Short Bag 4
C
      DO 95 IACL = -20,20
      DO 95 ISPD = 0,20
      B4L(IACL,ISPD) = B4L(IACL,ISPD) / TIMEL
      B4T(IACL,ISPD) = TFREQ(IACL,ISPD) - B4L(IACL,ISPD)
   95 CONTINUE
C
C     All stats for this m-trip combo (and its leftovers) complete.
C     Now check current DIFFS against "top-NBST" array.  If current
C     DIFFS is smaller than any element in the DIFFBST array, push it
C     in and shift the subsequent elements, moving the last one out.
C
      DO 120 I = 1,NBST

      IF (DIFFS .LT. DIFFBST(I)) THEN
      MOVE = I
C
C     OK, smaller DIFFS found -- do the element shift first.
C
      DO 100 L = NBST,MOVE+1,-1
      TIMEBST(L) = TIMEBST(L-1)
      DIFFBST(L) = DIFFBST(L-1)
      DO 98 LIST = 1,MM
      LISTBST(L,LIST) = LISTBST(L-1,LIST)
   98 CONTINUE
      DO 100 IACL = -20,20
      DO 100 ISPD = 0,20
      B4SBST(IACL,ISPD,L) = B4SBST(IACL,ISPD,L-1)
      B4TBST(IACL,ISPD,L) = B4TBST(IACL,ISPD,L-1)
  100 CONTINUE
C
C     And now the "push".
C
      TIMEBST(MOVE) = TIMES / 60
      DIFFBST(MOVE) = DIFFS
      DO 102 LIST = 1,K
      LISTBST(MOVE,LIST) = KOMBOS(LIST,J) + NS
  102 CONTINUE
      DO 110 IACL = -20,20
      DO 110 ISPD = 0,20
      B4SBST(IACL,ISPD,MOVE) = B4S(IACL,ISPD)
      B4TBST(IACL,ISPD,MOVE) = B4T(IACL,ISPD)
  110 CONTINUE
C
      GO TO 200
C
      END IF
```

```
  120 CONTINUE
C
C     Go back on J loop and try next combination.
C
  200 CONTINUE
C
C     Update total combos counter.
C
      NTOT = NTOT + NK
C
C     Go back on outer loop K (the combo size variable).
C
  300 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Processing completed, output all results and end.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     Write non-FTP surface distribution.
C
      DO 400 IACL = -20,20
      DO 400 ISPD = 0,20
      PCT = NFFREQ(IACL,ISPD) * 100
      WRITE (5, 315) IACL, ISPD, IDUM, PCT
  315 FORMAT (2(I3,','), I8, ',', F10.6)

  400 CONTINUE
C
C     Write top NBST summary table.
C
      WRITE (6, 402) NBST, CYCFILE(1:IE)
  402 FORMAT ('TOP ', I2, ' "SHORT BAG 4" MICROTRIP COMBINATIONS FOR',
     +        ' CYCLE ', A///
     +        '       Non-FTP   Short Bag4'/
     +        'Rank   DIFFSUM   Time (min)   Microtrips (#)'/
     +        '----   -------   ----------   --------------')
C
      DO 410 I = 1, NBST

      DO 405 L = 1, M
      IF (LISTBST(I,L) .EQ. 0) THEN
      K = L - 1
      GO TO 408
      END IF
  405 CONTINUE

  408 WRITE (6, 409) I, DIFFBST(I), TIMEBST(I), (LISTBST(I,LIST),
     +               LIST = 1,K)
  409 FORMAT (I3, 3X, F8.5, 6X, F5.2, 3X, <K>I4)

  410 CONTINUE
C
C     Write Sbag4 and Tbag4 surfaces to units 7 and 8, respectively.
C
      DO 420 I = 1,NBST

      WRITE (CNUM, FMT='(I2.2)') I
      B4SFILE = 'S4FRQ_' // CYCFILE(1:IE) // '_' // CNUM // '.DAT'
      B4TFILE = 'T4FRQ_' // CYCFILE(1:IE) // '_' // CNUM // '.DAT'
      OPEN (UNIT=7, FILE=B4SFILE, STATUS='NEW', CARRIAGECONTROL='LIST')
      OPEN (UNIT=8, FILE=B4TFILE, STATUS='NEW', CARRIAGECONTROL='LIST')

      DO 417 IACL = -20,20
      DO 417 ISPD = 0,20
      PCT = B4SBST(IACL,ISPD,I) * 100
      WRITE (7, 315) IACL ,ISPD, IDUM, PCT
  417 CONTINUE
      CLOSE (7)

      DO 418 IACL = -20,20
      DO 418 ISPD = 0,20
      PCT = B4TBST(IACL,ISPD,I) * 100
      WRITE (8, 315) IACL ,ISPD, IDUM, PCT
  418 CONTINUE
      CLOSE (8)

  420 CONTINUE
C
```

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Print status summary and end.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      WRITE (*, 500) CYCFILE(1:IE), NTOT
  500 FORMAT (/' Short Bag 4 analysis completed for cycle ', A//
     +        10X, I5, ' microtrips combinations evaluated'/)
C
      END

      SUBROUTINE COMBOS (N, K, KOMBOS, NUMBER)
C
C     This routine generates all unique combinations of numbers in the
C     range 1 to N of choosing K items without replacement from N
C     elements.
C
C     Passed Args:
C
C        N ..... the number of elements (and implied range) to
C                choose from
C        K ..... the count or size of each chosen set
C
C
C     Returned Args:
C
C        KOMBOS ..... a two-dimensional integer array (with dimensions
C                     K x NUMBER) containing NUMBER unique combinations
C                     each of size K.
C        NUMBER ..... the total number of unique combinations generated:
C
C                                        N!
C                              NUMBER  = -----------
C                                        K! (N - K)!
C
C     Combinations refer to selected number sets in which the order of
C     selected numbers doesn't matter.  (This differs from permutations
C     in which the order establishes a unique set.)  For example, 1-2-3
C     and 1-3-2 are a the same, single combination.
C
C     ------------------------------------------------------------------
C     NOTE:  Care should be used in calling this routine when N > 10
C            since the # of combinations increases rapidly due to the
C            factorials.  As a result, KOMBOS size can get enormous.
C     ------------------------------------------------------------------
C
C     Author:  Tom Carlson
C     Date     05/05/93
C
C-----------------------------------------------------------------------
C
      PARAMETER (NMAX = 50, MM = 25)
      LOGICAL HOLE
      INTEGER I, J, K, L1, M, N, LEAD, NUMBER, C, KOMBOS
      DIMENSION C(NMAX), KOMBOS(MM,*)
C
C     Begin execution -- initialize the "index" array C with K
C     left-filled elements = 1 and N-K right filled elements = 0.
C
      LEAD = K
      L1 = 0
      LHOLE = .FALSE.

      DO 10 I = 1,N
      IF (I .LE. K) THEN
      C(I) = 1
      ELSE
      C(I) = 0
      END IF
   10 CONTINUE
C
C     Return point from either a finished "Migrate LEAD" sequence or
C     a "Move Elements" operation.  (It's also stepped to from above.)
C     Call KFILL to "write" currently selected indices in C (those set
C     to "1") to KOMBOS.
C
   20 CALL KFILL (K, N, C, KOMBOS, NUMBER)
C
C     Test if "Migrate LEAD" operations have pushed the LEAD pointer
```

```
C     to our range limit given by N.  If so, branch to "Move Elements"
C     section.
C
      IF (LEAD .EQ. N) GO TO 40
C
C     --------------------
C     Migrate LEAD Section
C     --------------------
C
C     We're still within range.  Perform a shift sequence for the LEAD
C     pointer only.  Migrate it until out of range, calling KFILL after
C     each shift.
C
      DO 30 I = LEAD+1,N
      C(I) = 1
      C(I-1) = 0
      CALL KFILL (K, N, C, KOMBOS, NUMBER)
   30 CONTINUE
C
C     Apply special termination condition test when K = 1.
C
      IF (K .EQ. 1) GO TO 100
C
C     Increment LEAD and reset it and its left-most neighbor to "1".
C     Then go back to 20 to do a KFILL and a range test.
C
      C(N) = 0
      C(LEAD+1) = 1
      C(LEAD) = 1
      C(LEAD-1) = 0
      LEAD = LEAD + 1
      GO TO 20
C
C     --------------------
C     Move Elements Section
C     --------------------
C
C     Perform a backward scan of C (i.e., starting at element N) using
C     the loop variable M.  In the scan, detect:
C
C       - the first occurrence of an element = 0 (sets LHOLE to .TRUE.)
C       - with LHOLE .TRUE., the next element = 1 (final M value)
C       - # of elements between N & final M for which C = 1 (final L1)
C
C     When final M is found, the "1" value for the final M element is
C     moved to the right and remaining non-zero elements to the right
C     (indicated by L1) are reset just to the right of the moved final
C     M element.
C
C     The test for subroutine termination is also performed in this
C     section.  Terminate when L1 equals K (all "1" elements will now)
C     be right-justified in the index array C.
C
   40 L1 = 0
      LHOLE = .FALSE.
      M = N

   50 IF (C(M) .EQ. 0) GO TO 70
      L1 = L1 + 1
      IF (.NOT. LHOLE) GO TO 80
      L1 = L1 - 1
      C(M) = 0
      J = M + L1 + 1
      DO 60 I = M+1,J
      C(I) = 1
   60 CONTINUE
      DO 65 I = J+1,N
      C(I) = 0
   65 CONTINUE
      LEAD = J

      GO TO 20
C
C     Subroutine terminate test -- is C now right-justified with "1's"?
C
   70 IF (L1 .EQ. K) GO TO 100
C
C     Zero value detected in backward search thru array C.  This is
C     called a "hole".
C
```

```
      LHOLE = .TRUE.
C
C     Decrement "move" variable M and re-test for element to move.
C     Element to move is the first non-zero element found by backward
C     searching C after a hole has been detected (LHOLE true).
C
   80 M = M - 1
      GO TO 50
C
C     Return condition.
C
  100 RETURN
      END

      SUBROUTINE KFILL (K, N, C, KOMBOS, NUMBER)
C
C     A companion routine called by COMBOS to fill the "current"
C     numbers indexed in array C into the selection array KOMBOS.
C
      PARAMETER (MM = 25)

      INTEGER K, N, NUMBER, C, KOMBOS
      DIMENSION C(*), KOMBOS(MM,*)
C
      J = 0
      NUMBER = NUMBER + 1

      DO 10 I = 1,N
      IF (C(I) .EQ. 1) THEN
      J = J + 1
      KOMBOS(J,NUMBER) = I
      IF (J .EQ. K) GO TO 20
      END IF
   10 CONTINUE
C
   20 RETURN
      END
```

```
      PROGRAM SHORTB4
C
C     SHORTB4 (Short Bag 4) retrieves the m-trip profile information for
C     a user input cycle and finds the subset of m-trips which best
C     represent non-FTP operation.  This logic is laid out as follows:
C
C        (1).  A "target" FTP spd/acl surface is input (LA4FRQ.DAT).
C
C        (2).  Two subset arrays are set up:  FTP & leftover or non-FTP
C
C        (3).  The m-trips at the beginning of the input cycle which are
C              designated as start m-trips are placed into the
C              FTP array.  Then all remaining m-trips are iteratively
C              added to the FTP subset IN ALL POSSIBLE COMBINATIONS and
C              each combination is tested against the FTP target surface
C              (see subroutine COMBOS).
C
C        (4).  Combinations of all sizes from 1 to N-1 are
C              considered (N=# of cycle m-trips).
C
C        (6).  Top NBEST m-trips combinations (based on minimum DIFFSUM)
C              for FTP are then saved and output along with their
C              "leftovers".  These leftover m-trips make up Short Bag 4.
C
C     I/O:   Unit 1 (input)  - "All Driving" composite surface.
C            Unit 2 (input)  - LA4 surface.
C            Unit 3 (input)  - Selected M-trips & goodness stats file.
C            Unit 4 (input)  - M-trips profile file (binary).
C            Unit 6 (output) - Listing of the best m-trip combinations
C                              in the FTP subset and the corresponding
C                              "leftovers" in the Short Bag 4 subset.
C            Unit 7 (output) - Best Short Bag 4 spd/acl surfaces.
C            Unit 8 (output) - Best Long Bag 4 spd/acl target surfaces.
C
C
C     Author:  Tom Carlson
C     Date:    5/4/93
C
C-----------------------------------------------------------------------
C
C     Set run size.  MM ----> Max # of m-trips allowed in a cycle.
C                    MS ----> Max # of start m-trips allowed in a cycle.
C                    NBST --> # of "best" short bag 4 combos to output.
C                    NMAX --> # of total m-trip combinations allowed.
C
      PARAMETER (MM = 25, MS = 10, NBST = 10, NMAX = 5000)
C
      CHARACTER    TYPE(MM), TYPES(MM)
      CHARACTER*2  CNUM
      CHARACTER*14 MTRIPID(MM), KTRIPID, MTRIPIDS(MM)
      CHARACTER*25 CYCFILE, B4TFILE

      INTEGER*2 MCNT(MM,-20:20,0:20), KTIME, KK, LISTBST(NBST,MM),
     +          ITIME(MM), KCNT(-20:20,0:20), MTRP(MM), ITIMES(MS),
     +          MTRPS(MS)
      INTEGER*4 IREC(MM), KREC, KOMBOS(MM,NMAX), IRECS(MS)

      REAL SPD_MEAN(MM), SPD_STD(MM), ACL_MAX(MM), DIFFBST(NBST),
     +     FFREQ(-20:20,0:20), B4F(-20:20,0:20), B4T(-20:20,0:20),
     +     SPD_MAX(MM), TIMEBST(NBST), B4TBST(-20:20,0:20,NBST),
     +     TFREQ(-20:20,0:20), B4S(-20:20,0:20), B4L(-20:20,0:20),
     +     B4FBST(-20:20,0:20,NBST), B4SBST(-20:20,0:20,NBST),
     +     SPD_MAXS(MS), SPD_MEANS(MS)

      DATA DIFFBST / NBST*999. /

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='OLD')
      OPEN (UNIT=4, STATUS='OLD', RECL=1760, FORM='UNFORMATTED')
      OPEN (UNIT=6, STATUS='NEW', CARRIAGECONTROL='LIST')

C-----------------------------------------------------------------------
C    Begin execution -- Input required data.
C-----------------------------------------------------------------------
C
C     Read in total "All Driving" surface (unit 1).
C
```

```
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      TFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in LA4 surface (unit 2).
C
   20 READ (2, *, END=30) IACL, ISPD, NFREQ, PCT
      FFREQ(IACL,ISPD) = PCT / 100
      IN2 = IN2 + 1
      GO TO 20
C
C     Read in selected cycle stats file (unit 3).
C
   30 READ (3, 31) CYCFILE
   31 FORMAT (//23X, A)
      IE = INDEX (CYCFILE, '.') - 1

      DO 32 ISKIP = 1,4
      READ (3, FMT='()')
   32 CONTINUE

   33 IN3 = IN3 + 1
      READ (3, 34, ERR=35) N, TYPE(N), MTRP(N), MTRIPID(N), IREC(N),
     +                     ITIME(N), SPD_MEAN(N), SPD_MAX(N)
   34 FORMAT (I3, 1X, A1, 1X, I5, 1X, A14, 1X, I7, 1X, I4, 2(1X, F7.2))
      GO TO 33

   35 N = IN3 - 1
C
C     Now detect and move the start m-trips to the back of the pack so
C     the COMBOS routine can be called over the range 1 to N-NS.
C
      DO 36 I = 1,N
      IF (TYPE(I) .NE. 'S') THEN
      NS = I - 1
      GO TO 37
      END IF
      MTRPS(I) = MTRP(I)
      TYPES(I) = TYPE(I)
      MTRIPIDS(I) = MTRIPID(I)
      IRECS(I) = IREC(I)
      ITIMES(I) = ITIME(I)
      SPD_MEANS(I) = SPD_MEAN(I)
      SPD_MAXS(I) = SPD_MAX(I)
   36 CONTINUE
C
   37 M = N - NS
C
      DO 38 I = 1,M
      MTRP(I) = MTRP(I+NS)
      TYPE(I) = TYPE(I+NS)
      MTRIPID(I) = MTRIPID(I+NS)
      IREC(I) = IREC(I+NS)
      ITIME(I) = ITIME(I+NS)
      SPD_MEAN(I) = SPD_MEAN(I+NS)
      SPD_MAX(I) = SPD_MAX(I+NS)
   38 CONTINUE

      DO 39 I = M+1,N
      MTRP(I) = MTRPS(I-M)
      TYPE(I) = TYPES(I-M)
      MTRIPID(I) = MTRIPIDS(I-M)
      IREC(I) = IRECS(I-M)
      ITIME(I) = ITIMES(I-M)
      SPD_MEAN(I) = SPD_MEANS(I-M)
      SPD_MAX(I) = SPD_MAXS(I-M)
   39 CONTINUE
C
C
C     Read in microtrip profiles binary records one at a time (unit 4).
C
   40 READ (4, END=50) KK, KTRIPID, KREC, KTIME, SPDMEAN, SPDMAX,
     +      SPDSTD, ACLMAX, ((KCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)
      IN4 = IN4 + 1
C
C     Check the m-trip # of each record & keep only those records which
C     are found in the MTRP list read from unit 3 for the input cycle.
C
      DO 45 I = 1,N
```

```
      IF (KK .EQ. MTRP(I)) THEN
      NN = NN + 1
      SPD_STD(I) = SPDSTD
      ACL_MAX(I) = ACLMAX
      DO 42 ISPD = 0,20
      DO 42 IACL = -20,20
      MCNT(I,IACL,ISPD) = KCNT(IACL,ISPD)
   42 CONTINUE
      IF (NN .EQ. N) GO TO 50
      GO TO 40
      END IF

   45 CONTINUE

      GO TO 40
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Now call subroutine COMBOS within a loop which
C     controls the number of m-trips to choose (1 to N-NS) to generate
C     all possible combinations of m-trips.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
   50 DO 300 K = 0, M-1

      NK = 0
      IF (K .EQ. 0) GO TO 52
      CALL COMBOS (M, K, KOMBOS, NK)
C
C     Loop thru all combinations generated for size K.
C
      DO 200 J = 1,NK
C
C     Initialize sums and counter before each combo is tested.  "Seed"
C     FTP subset with the NS start m-trips.
C
   52 DIFFS = 0.0
      TIMEF = 0.0
      TIMEL = 0.0
      DO 55 IACL = -20,20
      DO 55 ISPD = 0,20
      B4S(IACL,ISPD) = 0.0
      B4L(IACL,ISPD) = 0.0
      B4F(IACL,ISPD) = 0.0
   55 CONTINUE
C
      DO 58 I = M+1,N
      TIMEF = TIMEF + ITIME(I)
      DO 58 IACL = -20,20
      DO 58 ISPD = 0,20
      B4F(IACL,ISPD) = B4F(IACL,ISPD) + MCNT(I,IACL,ISPD)
   58 CONTINUE
C
C     Loop thru and process bin counts for each m-trip in current combo.
C
      IF (K .EQ. 0) GO TO 62

      DO 60 I = 1,K

      ITRP = KOMBOS(I,J)
      TIMEF = TIMEF + ITIME(ITRP)

      DO 60 IACL = -20,20
      DO 60 ISPD = 0,20
      B4F(IACL,ISPD) = B4F(IACL,ISPD) + MCNT(ITRP,IACL,ISPD)
   60 CONTINUE
C
C     Compute the summed diff DIFFS for the current combo.
C
   62 DO 65 IACL = -20,20
      DO 65 ISPD = 0,20
      B4F(IACL,ISPD) = B4F(IACL,ISPD) / TIMEF
      DIFFS = DIFFS + ABS (B4F(IACL,ISPD) - FFREQ(IACL,ISPD))
   65 CONTINUE
C
C     OK, evaluation of FTP for current combo complete.  Now get
C     the remaining m-trips in the input PCC which are NOT in the
C     current combo.  These non-FTP leftovers are "Short Bag 4".
```

```
C
      DO 80 L = 1,M

      DO 70 I = 1,K
      IF (L .EQ. KOMBOS(I,J)) GO TO 80
   70 CONTINUE
C
C     IF statement always false -- we've found a leftover m-trip.
C
      ITRP = L
      TIMEL = TIMEL + ITIME(ITRP)

      DO 75 IACL = -20,20
      DO 75 ISPD = 0,20
      B4L(IACL,ISPD) = B4L(IACL,ISPD) + MCNT(ITRP,IACL,ISPD)
   75 CONTINUE

   80 CONTINUE
C
C     All leftovers detected and processed.  Now, compute the new
C     target Long Bag 4 surface from FTP-matched m-trips:
C
C     Long Bag4 Target Surface = "All Driving" - "FTP" Bag 4
C
C          where "FTP" Bag 4 = PCC - Short Bag 4
C
      DO 95 IACL = -20,20
      DO 95 ISPD = 0,20
      B4F(IACL,ISPD) = B4F(IACL,ISPD) / TIMEF
      B4T(IACL,ISPD) = TFREQ(IACL,ISPD) - B4F(IACL,ISPD)
   95 CONTINUE
C
C     All stats for this m-trip combo (and its leftovers) complete.
C     Now check current DIFFS against "top-NBST" array.  If current
C     DIFFS is smaller than any element in the DIFFBST array, push it
C     in and shift the subsequent elements, moving the last one out.
C
      DO 120 I = 1,NBST

      IF (DIFFS .LT. DIFFBST(I)) THEN
      MOVE = I
C
C     OK, smaller DIFFS found -- do the element shift first.
C
      DO 100 L = NBST,MOVE+1,-1
      TIMEBST(L) = TIMEBST(L-1)
      DIFFBST(L) = DIFFBST(L-1)
      DO 98 LIST = 1,MM
      LISTBST(L,LIST) = LISTBST(L-1,LIST)
   98 CONTINUE
      DO 100 IACL = -20,20
      DO 100 ISPD = 0,20
      B4FBST(IACL,ISPD,L) = B4FBST(IACL,ISPD,L-1)
      B4TBST(IACL,ISPD,L) = B4TBST(IACL,ISPD,L-1)
  100 CONTINUE
C
C     And now the "push".
C
      TIMEBST(MOVE) = TIMEF / 60
      DIFFBST(MOVE) = DIFFS
      LISTBST(MOVE,1) = 1
      LISTBST(MOVE,2) = 2
      DO 102 LIST = 1,K
      LISTBST(MOVE,LIST+NS) = KOMBOS(LIST,J) + NS
  102 CONTINUE
      DO 110 IACL = -20,20
      DO 110 ISPD = 0,20
      B4FBST(IACL,ISPD,MOVE) = B4F(IACL,ISPD)
      B4TBST(IACL,ISPD,MOVE) = B4T(IACL,ISPD)
  110 CONTINUE
C
      GO TO 200
C
      END IF

  120 CONTINUE
C
C     Go back on J loop and try next combination.
C
  200 CONTINUE
```

```
C
C     Update total combos counter.
C
      NTOT = NTOT + NK
C
C     Go back on outer loop K (the combo size variable).
C
  300 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Processing completed, output all results and end.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
CC
C     Write top NBST summary table.
C
      WRITE (6, 402) NBST, CYCFILE(1:IE)
  402 FORMAT ('TOP ', I2, ' "FTP" MICROTRIP COMBINATIONS FOR',
     +        ' CYCLE ', A///
     +        '          FTP         FTP'/
     +        'Rank   DIFFSUM   Time (min)   Microtrips (#)'/
     +        '----   -------   ----------   --------------')
C
      DO 410 I = 1, NBST

      DO 405 L = 1, M
      IF (LISTBST(I,L) .EQ. 0) THEN
      K = L - 1
      GO TO 408
      END IF
  405 CONTINUE

  408 WRITE (6, 409) I, DIFFBST(I), TIMEBST(I), (LISTBST(I,LIST),
     +               LIST = 1,K)
  409 FORMAT (I3, 3X, F8.5, 6X, F5.2, 3X, <K>I4)

  410 CONTINUE
C
C     Write Tbag4 surfaces to unit 7.
C
      DO 420 I = 1,NBST

      WRITE (CNUM, FMT='(I2.2)') I
      B4TFILE = 'T4FRQ_' // CYCFILE(1:IE) // '_' // CNUM // '.DAT'
      OPEN (UNIT=7, FILE=B4TFILE, STATUS='NEW', CARRIAGECONTROL='LIST')

      DO 417 IACL = -20,20
      DO 417 ISPD = 0,20
      PCT = B4TBST(IACL,ISPD,I) * 100
      WRITE (7, 415) IACL ,ISPD, IDUM, PCT
  415 FORMAT (2(I3,','), I8, ',', F10.6)
  417 CONTINUE
      CLOSE (7)

  420 CONTINUE
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Print status summary and end.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
      WRITE (*, 500) CYCFILE(1:IE), NTOT
  500 FORMAT (/' Short Bag 4 analysis completed for cycle ', A//
     +        10X, I5, ' microtrips combinations evaluated'/)
C
      END

      SUBROUTINE COMBOS (N, K, KOMBOS, NUMBER)
C
C     This routine generates all unique combinations of numbers in the
C     range 1 to N of choosing K items without replacement from N
C     elements.
C
C     Passed Args:
C
C        N ..... the number of elements (and implied range) to
C                choose from
C        K ..... the count or size of each chosen set
```

```
C
C
C     Returned Args:
C
C        KOMBOS ..... a two-dimensional integer array (with dimensions
C                     K x NUMBER) containing NUMBER unique combinations
C                     each of size K.
C        NUMBER ..... the total number of unique combinations generated:
C
C                                        N!
C                              NUMBER  = -----------
C                                        K! (N - K)!
C
C     Combinations refer to selected number sets in which the order of
C     selected numbers doesn't matter.  (This differs from permutations
C     in which the order establishes a unique set.)  For example, 1-2-3
C     and 1-3-2 are a the same, single combination.
C
C     ---------------------------------------------------------------
C     NOTE:  Care should be used in calling this routine when N > 10
C            since the # of combinations increases rapidly due to the
C            factorials.  As a result, KOMBOS size can get enormous.
C     ---------------------------------------------------------------
C
C     Author:  Tom Carlson
C     Date     05/05/93
C
C-----------------------------------------------------------------------
C
      PARAMETER (NMAX = 50, MM = 25)
      LOGICAL HOLE
      INTEGER I, J, K, L1, M, N, LEAD, NUMBER, C, KOMBOS
      DIMENSION C(NMAX), KOMBOS(MM,*)
C
C     Begin execution -- initialize the "index" array C with K
C     left-filled elements = 1 and N-K right filled elements = 0.
C
      LEAD = K
      L1 = 0
      LHOLE = .FALSE.

      DO 10 I = 1,N
      IF (I .LE. K) THEN
      C(I) = 1
      ELSE
      C(I) = 0
      END IF
   10 CONTINUE
C
C     Return point from either a finished "Migrate LEAD" sequence or
C     a "Move Elements" operation.  (It's also stepped to from above.)
C     Call KFILL to "write" currently selected indices in C (those set
C     to "1") to KOMBOS.
C
   20 CALL KFILL (K, N, C, KOMBOS, NUMBER)
C
C     Test if "Migrate LEAD" operations have pushed the LEAD pointer
C     to our range limit given by N.  If so, branch to "Move Elements"
C     section.
C
      IF (LEAD .EQ. N) GO TO 40
C
C     -------------------
C     Migrate LEAD Section
C     -------------------
C
C     We're still within range.  Perform a shift sequence for the LEAD
C     pointer only.  Migrate it until out of range, calling KFILL after
C     each shift.
C
      DO 30 I = LEAD+1,N
      C(I) = 1
      C(I-1) = 0
      CALL KFILL (K, N, C, KOMBOS, NUMBER)
   30 CONTINUE
C
C     Apply special termination condition test when K = 1.
C
      IF (K .EQ. 1) GO TO 100
C
```

```
C     Increment LEAD and reset it and its left-most neighbor to "1".
C     Then go back to 20 to do a KFILL and a range test.
C
      C(N) = 0
      C(LEAD+1) = 1
      C(LEAD) = 1
      C(LEAD-1) = 0
      LEAD = LEAD + 1
      GO TO 20
C
C     --------------------
C     Move Elements Section   --
C     --------------------
C
C     Perform a backward scan of C (i.e., starting at element N) using
C     the loop variable M.  In the scan, detect:
C
C       - the first occurrence of an element = 0 (sets LHOLE to .TRUE.)
C       - with LHOLE .TRUE., the next element = 1 (final M value)
C       - # of elements between N & final M for which C = 1 (final L1)
C
C     When final M is found, the "1" value for the final M element is
C     moved to the right and remaining non-zero elements to the right
C     (indicated by L1) are reset just to the right of the moved final
C     M element.
C
C     The test for subroutine termination is also performed in this
C     section.  Terminate when L1 equals K (all "1" elements will now)
C     be right-justified in the index array C.
C
   40 L1 = 0
      LHOLE = .FALSE.
      M = N

   50 IF (C(M) .EQ. 0) GO TO 70
      L1 = L1 + 1
      IF (.NOT. LHOLE) GO TO 80
      L1 = L1 - 1
      C(M) = 0
      J = M + L1 + 1
      DO 60 I = M+1,J
      C(I) = 1
   60 CONTINUE
      DO 65 I = J+1,N
      C(I) = 0
   65 CONTINUE
      LEAD = J

      GO TO 20
C
C     Subroutine terminate test -- is C now right-justified with "1's"?
C
   70 IF (L1 .EQ. K) GO TO 100
C
C     Zero value detected in backward search thru array C.  This is
C     called a "hole".
C
      LHOLE = .TRUE.
C
C     Decrement "move" variable M and re-test for element to move.
C     Element to move is the first non-zero element found by backward
C     searching C after a hole has been detected (LHOLE true).
C
   80 M = M - 1
      GO TO 50
C
C     Return condition.
C
  100 RETURN
      END

      SUBROUTINE KFILL (K, N, C, KOMBOS, NUMBER)
C
C     A companion routine called by COMBOS to fill the "current"
C     numbers indexed in array C into the selection array KOMBOS.
C
      PARAMETER (MM = 25)

      INTEGER K, N, NUMBER, C, KOMBOS
      DIMENSION C(*), KOMBOS(MM,*)
```

```
C
      J = 0
      NUMBER = NUMBER + 1

      DO 10 I = 1,N
      IF (C(I) .EQ. 1) THEN
      J = J + 1
      KOMBOS(J,NUMBER) = I
      IF (J .EQ. K) GO TO 20
      END IF
   10 CONTINUE
C
   20 RETURN
      END
```

```
      PROGRAM SHOWDIFF
C
C     A quick program to generate tabular "Watson plots" from
C     input spd/acl surface files in "comma-delimited" format.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):     0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):    -0.5 to 0.5 mph/s = 0
C                      -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                      0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     I/O:   Unit 1 (input)   - GETSTAB output table of microtrips for a
C                               selected cycle.
C            Unit 2 (input)   - MICTRIPS profile statistics output file.
C            Unit 3 (input)   - Speed/accel target surface created from
C                               sec-by-sec drive cycle data.
C            Unit 4 (output)  - Tabular "Watson Plot" of the spd/acl fit
C                               between selected cycle & target surface.
C
C     Author:  Tom Carlson
C     Date:    05/13/93
C-----------------------------------------------------------------------

      CHARACTER    MTYPE(50)
      CHARACTER*14 MTRIPID(50), ITRIPID
      CHARACTER*80 INFILE, MTRIPFILE, TARGFILE, TITLE, OUTFILE

      INTEGER*2 MTRIPS(50), ITIME(50), MCNT(-20:20,0:20), JTIME, M
      INTEGER*4 IREC(50), IPNT(50), JREC, ITRIP(50)

      REAL CFREQ(-20:20,0:20), MAXSPD,ACLSUM(2,0:20),PFREQ(-20:20,0:20),
     +     SPDSUM(2,-20:20), SPD_MEAN, SPD_MAX, SPD_STD, ACL_MAX,
     +     PCTDIF(-20:20,0:20)

      DATA IHDR / 7 /
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for input and output filenames.
C
      WRITE (*, 10)
   10 FORMAT (/1X, 'Enter input M-Trips summary filename:')
      READ (*, 20) INFILE
      WRITE (*, 15)
   15 FORMAT (/1X, 'Enter unformatted M-Trips Profile filename:')
      READ (*, 20) MTRIPFILE
   20 FORMAT (A)
      WRITE (*, 30)
   30 FORMAT (/1X, 'Enter target surface filename:')
      READ (*, 20) TARGFILE

C     Name OUTFILE and TITLE based on INFILE and TARGFILE.

      L1 = INDEX (INFILE, ']') + 1
      L2 = INDEX (INFILE(L1:80), '.') + L1 - 2
      OUTFILE = INFILE(L1:L2) // '.DIF'
      K = INDEX (TARGFILE, '.') - 1
      TITLE = 'Difference Between Cycle ' // INFILE(L1:L2) //
     +        ' And Target Surface ' // TARGFILE(1:K)
      LL = INDEX (TITLE, '   ') - 1
      ILEN = 87 - LL/2
C
C     Open all files.
C
      OPEN (UNIT=1, FILE=INFILE, STATUS='OLD')
      OPEN (UNIT=2, FILE=MTRIPFILE, STATUS='OLD', FORM='UNFORMATTED',
     +      RECL=1760)
      OPEN (UNIT=3, FILE=TARGFILE, STATUS='OLD')
      OPEN (UNIT=4, FILE=OUTFILE, STATUS='NEW', RECL=186,
     +      CARRIAGECONTROL='LIST')
C
C-----------------------------------------------------------------------
C     Read target population surface (unit 3).
C-----------------------------------------------------------------------
C
C     Target population surface.
C
```

```
  100 READ (3, *, END=190) IACL, ISPD, RBIN, PCT
      PFREQ(IACL,ISPD) = PCT
      IN3 = IN3 + 1
      GO TO 100
C
C-----------------------------------------------------------------------
C     Get list of selected M-trips from INFILE, skipping IHDR header
C     records (unit 1).
C-----------------------------------------------------------------------
C
  190 DO 199 ISKIP = 1,IHDR
      READ (1, FMT='()')
  199 CONTINUE

  200 ITRP = ITRP + 1
      READ (1, 205, ERR=210) ITRIP(ITRP), MTYPE(ITRP), MTRIPS(ITRP),
     +                       MTRIPID(ITRP), IREC(ITRP), ITIME(ITRP)
  205 FORMAT (I3, 1X, A1, 1X, I5, 1X, A14, 1X, I7, 1X, I4)
      GO TO 200
C
C-----------------------------------------------------------------------
C     All M-trips read from INFILE.  Now sort 'em by increasing initial
C     record number IREC.  Use indirect sorting, storing pointer
C     rankings in IPNT.
C-----------------------------------------------------------------------
C
  210 NTRP = ITRP - 1
      DO 220 I = 1, NTRP
      IPNT(I) = I
  220 CONTINUE

      DO 250 I = 1, NTRP-1
      L = IPNT(I+1)

      DO 230 J = 1, I
      K = I + 1 - J
      IF (IREC(L) .GT. IREC(IPNT(K))) GO TO 240
      IPNT(K+1) = IPNT(K)
  230 CONTINUE

      K = 0
  240 IPNT(K+1) = L
  250 CONTINUE
C
C-----------------------------------------------------------------------
C     Pointer sorted.  Now read speed/accel bin frequency counts data
C     for selected M-trips created by the MICTRIPS program from unit 2.
C     After each M-trip is input, increment bin counts for that M-trip.
C-----------------------------------------------------------------------
C
      DO 300 I = 1, NTRP
      J = IPNT(I)

      IF (I .GT. 1) THEN
      NSKIP = MTRIPS(J) - (MTRIPS(J1) + 1)
      ELSE
      NSKIP = MTRIPS(J) - 1
      END IF

C     Skip records until next selected M-trip is reached.

      DO 280 ISKIP = 1, NSKIP
      READ (2) M, ITRIPID, JREC, JTIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((MCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)
  280 CONTINUE

C     Read segment.

      READ (2) M, ITRIPID, JREC, JTIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((MCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)

      J1 = J

C     Now increment bin counts with current M-trip counts.

      DO 290 IACL = -20,20
      DO 290 ISPD = 0,20
      CFREQ(IACL,ISPD) = CFREQ(IACL,ISPD) + MCNT(IACL,ISPD)
  290 CONTINUE
```

```
      TOTTIME = TOTTIME + ITIME(J)
  300 CONTINUE
C
C-----------------------------------------------------------------------
C     Total bin count reads finished.  Compute cycle surface frequencies
C     and "difference" surface frequencies and output.
C-----------------------------------------------------------------------
C
      DO 310 IACL = -20,20
      DO 310 ISPD = 0,20
      CFREQ(IACL,ISPD) = CFREQ(IACL,ISPD) * 100 / TOTTIME
      PCTDIF(IACL,ISPD) = CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD)
  310 CONTINUE
C
C     Compute row (across spd) and column (across acl) totals.
C
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      SPDSUM(1,IACL) = SPDSUM(1,IACL) + PCTDIF(IACL,ISPD)
      SPDSUM(2,IACL) = SPDSUM(2,IACL) + ABS (PCTDIF(IACL,ISPD))
  320 CONTINUE

      DO 325 ISPD = 0,20
      DO 325 IACL = -20,20
      ACLSUM(1,ISPD) = ACLSUM(1,ISPD) + PCTDIF(IACL,ISPD)
      ACLSUM(2,ISPD) = ACLSUM(2,ISPD) + ABS (PCTDIF(IACL,ISPD))
      TOTNET = TOTNET + PCTDIF(IACL,ISPD)
      TOTABS = TOTABS + ABS (PCTDIF(IACL,ISPD))
  325 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      WRITE (4, 330) TITLE(1:LL)
  330 FORMAT (///<ILEN>X,A//68X,'Speed/Accel Frequency Distribution (%)'
     +        ///78X, 'SPD BIN (mph)')
      WRITE (4, 340) (ISPD*5, ISPD = 0,20)
  340 FORMAT (166X,'   Net      Abs'/
     + 'ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), 2('   Totals')/
     + '---------------', 4X, '----', 3X, 20('----',3X), 2('   ------'))
C
      DO 350 IACL = -20,20
      WRITE (4, 345) IACL, (PCTDIF(IACL,ISPD), ISPD = 0,20),
     +               (SPDSUM(I,IACL), I = 1,2)
  345 FORMAT (6X, I4, 6X, 21F7.3, 5X, F7.3, 2X, F7.3)
  350 CONTINUE
C
      WRITE (4, 355) (ACLSUM(1,ISPD), ISPD = 0,20), TOTNET,
     +               (ACLSUM(2,ISPD), ISPD = 0,20), TOTABS
  355 FORMAT (/3X, 'Net Totals', 3X, 21F7.3, 5X, F7.3, 2X/
     +         3X, 'Abs Totals', 3X, 21F7.3, 14X, F7.3)
      JTIME = TOTTIME
      WRITE (4, 360) JTIME
  360 FORMAT (/1X, 'Cycle Time (sec): ', I4)
C
      STOP

      END
```

```
      PROGRAM STABSURF
C
C     A quick program to produce a stabilized spd/acl surface from
C     Baltimore and LA92 mode-specific surface files:
C
C     (1) BFRQ2.DAT (Balt Stabilized)  ---|---> (3) CFRQS.DAT
C     (2) LFRQ2.DAT (LA92 Stabilized)  ---|         (Comp Stabilized)
C
C     Stabilized portions of total surfaces) are generated by a 50/50
C     weighting of the Baltimore and LA92 stabilized driving.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):     0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):    -0.5 to 0.5 mph/s = 0
C                      -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                      0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    07/14/93
C
C-----------------------------------------------------------------------
C
      INTEGER NBIN(2,-20:20,0:20), NREC(2)
      REAL CBIN(-20:20,0:20), WFAC(2)
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, CARRIAGECONTROL='LIST', STATUS='NEW')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Read input mode surface files (units 1-2).
C
      DO 110 IFILE = 1,2
  100 READ (IFILE, *, END=105) IACL, ISPD, NBIN(IFILE,IACL,ISPD)
      GO TO 100
  105 CLOSE (IFILE)
  110 CONTINUE
C
C-----------------------------------------------------------------------
C     Input complete, generate weighting factors for LA92/Balt:
C-----------------------------------------------------------------------
C     Balt Stab  = nrec1
C     LA92 Stab  = nrec2
C
C     For 50/50 weighting of stabilized data, we need the following
C     weightings:
C
C                 wfac1 = 1      ("unit" weighting)
C
C                 wfac2 = nrec1 / nrec2
C
      DO 150 IACL = -20,20
      DO 150 ISPD = 0,20
      DO 150 IFILE = 1,2
      NREC(IFILE) = NREC(IFILE) + NBIN(IFILE,IACL,ISPD)
  150 CONTINUE
C
      WFAC(1) = 1.0
      WFAC(2) = FLOAT (NREC(1)) / FLOAT(NREC(2))
C
C     Generate total weighted composite surface.
C
      DO 200 IACL = -20,20
      DO 200 ISPD = 0,20
      DO 200 I = 1,2
      X = WFAC(I) * NBIN(I,IACL,ISPD)
      CBIN(IACL,ISPD) = CBIN(IACL,ISPD) + X
      CREC = CREC + X
  200 CONTINUE

      DO 220 IACL = -20,20
      DO 220 ISPD = 0,20
      CPCT = CBIN(IACL,ISPD) * 100 / CREC
      WRITE (3, 210) IACL, ISPD, CBIN(IACL,ISPD), CPCT
  210 FORMAT (2(I3,','), F8.0, ',', F10.6)
  220 CONTINUE
```

```
      CLOSE (3)

      WRITE (*, 230) CREC, (WFAC(I), I = 1,2)
  230 FORMAT (/' Processing completed:    Crec  = ', F8.0//
     +        '                          Wfac1 = ', F5.1/
     +        '                          Wfac2 = ', F5.1/)

      END
```

```
      PROGRAM SUBSET
C
C     This program generates a randomly selected subset of the
C     composite LA92 and Baltimore microtrip profiles dataset.
C
C     Author:  Tom Carlson
C     Date:    04/26/93
C
C-----------------------------------------------------------------------
C
      CHARACTER*14 MTRIPID
      CHARACTER*23 CTIME

      INTEGER*2 ITIME, FREQ(-20:20,0:20), K, M, N, NRAN(30000), LTIME,
     +          HH, MM, DD, SS
      INTEGER*4 IREC, ISEED, ISEED0, POSACL(0:20)

      REAL SPD_MEAN, SPD_STD, ACL_MAX, SPD_MAX, AMAX, SMAX

      DATA MTRIPS /29133/
C
      OPEN (UNIT=1, STATUS='OLD', FORM='UNFORMATTED', RECL=1760)
      OPEN (UNIT=2, STATUS='NEW', FORM='UNFORMATTED', RECL=1760)
C
C-----------------------------------------------------------------------
C
C     Prompt user for subset factor (e.g., 0.10 = get 10% of full file).
C
      WRITE (*, 10)
   10 FORMAT (/1X, 'Enter subset fraction:')
      READ (*, *) SFRAC
      NSUB = NINT (MTRIPS * SFRAC)

      WRITE (*, 12)
   12 FORMAT (/1X, 'Enter start M-trip number:')
      READ (*, *) MSTART
      WRITE (*, 14)
   14 FORMAT (/1X, 'Enter # of start M-trips:')
      READ (*, *) NSTART
C
C     Seed selected subset array with the start mtrips.
C
      DO 15 M = MSTART, MSTART+NSTART-1
      I = I + 1
      NRAN(I) = M
   15 CONTINUE

C     Get system time.  Call to LIB$DATE_TIME returns 23-byte character
C     string which contains hour, minute, second and decimal fraction.

      ISTAT = LIB$DATE_TIME (CTIME)
      READ (CTIME, FMT='(11X,4(1X,I2))') HH, MM, SS, DD

C     Create seed value for random series.  Seed ranges from 0 to 10**8.
C     Make seed odd per VAX-FORTRAN recommendations (FORTRAN Language
C     Reference Manual V 5.0, pg. D-49).

      ISEED = - (DD*100 + SS*1.666667)
      IR = MOD (ISEED, 2)
      IF (IR .EQ. 0) ISEED = ISEED + 1
      ISEED0 = ISEED
C
C     Generate NSUB-NSTART unique random numbers.
C
      DO 50 I = NSTART+1, NSUB
   20 XRAN = RAN1 (ISEED)
      IF (XRAN .LT. 1.0) THEN
      M = INT (MTRIPS * XRAN)
      ELSE
      M = MTRIPS
      END IF

      DO 30 J = 1, I-1
      IF (NRAN(J) .EQ. M) GO TO 20
   30 CONTINUE
C
      NRAN(I) = M

   50 CONTINUE
C
```

```
C     Now order the random #'s array NRAN.
C
      CALL QSORT (NSUB, NRAN)
C-----------------------------------------------------------------------
C     OK, you've got the ordered random #'s.  Now retrieve & selectively
C     output the records.
C-----------------------------------------------------------------------
      K = 1
      DO 100 I = 1, MTRIPS

C     Terminate loop early when all NRANs have been processed.

      IF (K .GT. NSUB) GO TO 200

C     Read an m-trips profile record.

   80 READ (1) M, MTRIPID, IREC, ITIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((FREQ(IACL,ISPD), ISPD=0,20), IACL=-20,20)

C     Test it against "current" element K of ordered NRANs. If no match,
C     loop back and read next record.

      IF (I .NE. NRAN(K)) GO TO 100

C     Match found.  Check & "mark" NSTART M-trips starting at MSTART.

      IF (I .EQ. MSTART) KSTART = K

C     Set spd and acl max variables.

      IF (SPD_MAX .GT. SMAX) SMAX = SPD_MAX
      IF (ACL_MAX .GT. AMAX) AMAX = ACL_MAX

      DO 95 ISPD = 0,20
      DO 90 IACL = 20,0,-1
      IF (FREQ(IACL,ISPD) .GT. 0 .AND.
     +    IACL .GT. POSACL(ISPD)) THEN
      POSACL(ISPD) = IACL
      GO TO 95
      END IF
   90 CONTINUE
   95 CONTINUE

C     Output the matched record.

      WRITE (2) K, MTRIPID, IREC, ITIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((FREQ(IACL,ISPD), ISPD=0,20), IACL=-20,20)

      K = K + 1

  100 CONTINUE
C
C     Processing complete.  Print short rec counts summary and end.
C
  200 WRITE (*, 210) ISEED0, MTRIPS, NSUB, KSTART, NSTART
  210 FORMAT (/' M-trip Profile file subsetting complete: '//
     +        '       Initial Seed ....... ', I5/
     +        '       Total Mtrips ....... ', I5/
     +        '       Selected Mtrips .... ', I5/
     +        '       New Start Num ...... ', I5/
     +        '       # Starts ........... ', I5)

      WRITE (*, 220) SMAX, AMAX, (ISPD*5, POSACL(ISPD), ISPD = 0,20)
  220 FORMAT (/' Speed and Accel Extremes:'//
     +        '       Max Speed .......... ', F5.2, ' mph'/
     +        '       Max Accel .......... ', F5.2, ' mph/s'//
     +     21('       Spd --> ', I3, '    Max Pos Acl Bin --> ', I3/))

      END

      SUBROUTINE qsort(n,arr)
      INTEGER n,M,NSTACK
      INTEGER*2 arr(n)
      PARAMETER (M=7,NSTACK=50)
      INTEGER i,ir,j,jstack,k,l,istack(NSTACK)
      INTEGER*2 a,temp
      jstack=0
      l=1
      ir=n
1     if(ir-l.lt.M)then
```

```
        do 12 j=l+1,ir
          a=arr(j)
          do 11 i=j-1,1,-1
            if(arr(i).le.a)goto 2
            arr(i+1)=arr(i)
11        continue
          i=0
2         arr(i+1)=a
12      continue
        if(jstack.eq.0)return
        ir=istack(jstack)
        l=istack(jstack-1)
        jstack=jstack-2
      else
        k=(l+ir)/2
        temp=arr(k)
        arr(k)=arr(l+1)
        arr(l+1)=temp
        if(arr(l+1).gt.arr(ir))then
          temp=arr(l+1)
          arr(l+1)=arr(ir)
          arr(ir)=temp
        endif
        if(arr(l).gt.arr(ir))then
          temp=arr(l)
          arr(l)=arr(ir)
          arr(ir)=temp
        endif
        if(arr(l+1).gt.arr(l))then
          temp=arr(l+1)
          arr(l+1)=arr(l)
          arr(l)=temp
        endif
        i=l+1
        j=ir
        a=arr(l)
3       continue
          i=i+1
        if(arr(i).lt.a)goto 3
4       continue
          j=j-1
        if(arr(j).gt.a)goto 4
        if(j.lt.i)goto 5
        temp=arr(i)
        arr(i)=arr(j)
        arr(j)=temp
        goto 3
5       arr(l)=arr(j)
        arr(j)=a
        jstack=jstack+2
        if(jstack.gt.NSTACK)pause 'NSTACK too small in sort'
        if(ir-i+1.ge.j-l)then
          istack(jstack)=ir
          istack(jstack-1)=i
          ir=j-1
        else
          istack(jstack)=j-1
          istack(jstack-1)=l
          l=i
        endif
      endif
      goto 1
      END
C  (C) Copr. 1986-92 Numerical Recipes Software pLijn>#(.

      FUNCTION ran1(idum)
      INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV
      REAL ran1,AM,EPS,RNMX
      PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,
     *NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
      INTEGER j,k,iv(NTAB),iy
      SAVE iv,iy
      DATA iv /NTAB*0/, iy /0/
      if (idum.le.0.or.iy.eq.0) then
        idum=max(-idum,1)
        do 11 j=NTAB+8,1,-1
          k=idum/IQ
          idum=IA*(idum-k*IQ)-IR*k
          if (idum.lt.0) idum=idum+IM
          if (j.le.NTAB) iv(j)=idum
```

```
11      continue
        iy=iv(1)
      endif
      k=idum/IQ
      idum=IA*(idum-k*IQ)-IR*k
      if (idum.lt.0) idum=idum+IM
      j=1+iy/NDIV
      iy=iv(j)
      iv(j)=idum
      ran1=min(AM*iy,RNMX)
      return
      END
C  (C) Copr. 1986-92 Numerical Recipes Software pLijn>#(.
```

```
      PROGRAM UNF_MTRIPS
C
C     This program inputs a formatted "Microtrips Profile" file and
C     outputs the records to a sequential unformatted file.  This code
C     was written and the GETSTAB/GETHBRD program were modified to
C     accomodate the -50 MB addressing of the Composite Baltimore
C     and LA92 microtrip data.
C
C     Author:  Tom Carlson
C     Date:    04/24/93
C
C-----------------------------------------------------------------------
C
      CHARACTER*14 MTRIPID

      INTEGER*2 ITIME, FREQ(-20:20,0:20), M, N
      INTEGER*4 IREC

      REAL SPD_MEAN, SPD_STD, ACL_MAX, SPD_MAX
C
      OPEN (UNIT=1, STATUS='OLD', RECL=3512)
      OPEN (UNIT=2, STATUS='NEW', FORM='UNFORMATTED', RECL=1760)
C
C-----------------------------------------------------------------------
C
      DO 50 ISKIP = 1,3
      READ (1, FMT='()')
   50 CONTINUE
C
C     Sequential formatted input, one rec at a time.
C
  100 READ (1, 110, END=200) M, MTRIPID, IREC, ITIME, SPD_MEAN,
     +                       SPD_MAX, SPD_STD, ACL_MAX,
     +                     ((FREQ(IACL,ISPD), ISPD = 0,20), IACL=-20,20)
  110 FORMAT (I5, 1X, A14, 1X, I7, 1X, I5, 3(1X,F7.2), 1X, F7.3,
     +        1X, 861(I4))
      N = N + 1
C
      IF (M .NE. N) THEN
      WRITE (*, 120) M, N
  120 FORMAT (/' M-TRIP NUMBER MISMATCH:   M = ', I5, '   N = ', I5/)
      GO TO 300
      END IF
C
      WRITE (2) M, MTRIPID, IREC, ITIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((FREQ(IACL,ISPD), ISPD=0,20), IACL=-20,20)

      GO TO 100
C
C     Sequential input and binary output complete.  Print short
C     rec counts summary and end.
C
  200 WRITE (*, 210) N
  210 FORMAT (/' Binary M-trip Profile file complete: ',I5,
     +         ' records'/)

  300 CONTINUE
      END
```

```
      PROGRAM WTDSURF
C
C     A quick program to produce a weighted spd/acl surface from
C     input FRQ surface files and weighting factors.
C
C      Output Surface = Input Surface 1 * Wt1 + Input Surface 2 * Wt2 +
C                       ..... + Input Surface N * WtN
C
C              where N = is the # of surfaces (bags).
C
C     The output surface is then renormalized to 100%.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    07/22/93
C
C-----------------------------------------------------------------------
C
      PARAMETER (NMAX=10)
C
      CHARACTER*80 INFILE(NMAX)
      INTEGER NREC(NMAX)
      REAL PCT(NMAX,-20:20,0:20), OUTPCT(-20:20,0:20), WFAC(NMAX)
C
      DATA RDUM / 0.0 /
C
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW')
C
      WRITE (*, 10)
   10 FORMAT (/' Enter # of surfaces to be weighted together:')
      READ (*, *) NFAC
      WRITE (*, 20) NFAC
   20 FORMAT (/' Enter the ', I2, ' weighting factors separated by ',
     +        'spaces:')
      READ (*, *) (WFAC(I), I = 1,NFAC)
      WRITE (*, 30) NFAC
   30 FORMAT (/' Enter the ', I2, ' surface filenames (one per line):')
      DO 40 I = 1, NFAC
      READ (*, FMT='(A)') INFILE(I)
   40 CONTINUE
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Loop for each input surface.
C
      DO 150 I = 1,NFAC
C
C     Read input surface file (unit 1).
C
      OPEN (UNIT=1, FILE=INFILE(I), STATUS='OLD')
  100 READ (1, *, END=140) IACL, ISPD, IDUM, PCT(I,IACL,ISPD)
      GO TO 100
  140 CLOSE (1)
C
  150 CONTINUE
C
C-----------------------------------------------------------------------
C     Input complete, generate weighted output surface:
C-----------------------------------------------------------------------
C
C     Generate total weighted composite surface.
C
      DO 200 IACL = -20,20
      DO 200 ISPD = 0,20
      DO 190 I = 1,NFAC
      OUTPCT(IACL,ISPD) = OUTPCT(IACL,ISPD) + WFAC(I) * PCT(I,IACL,ISPD)
  190 CONTINUE
      TOTPCT = TOTPCT + OUTPCT(IACL,ISPD)
  200 CONTINUE
C
C     Re-normalize new weighted surface to 100%.
C
```

```
      DO 205 IACL = -20,20
      DO 205 ISPD = 0,20
      OUTPCT(IACL,ISPD) = OUTPCT(IACL,ISPD) * 100 / TOTPCT
205   CONTINUE


      DO 220 IACL = -20,20
      DO 220 ISPD = 0,20
      WRITE (2, 210) IACL, ISPD, RDUM, OUTPCT(IACL,ISPD)
210   FORMAT (2(I3,','), F8.0, ',', F10.6)
220   CONTINUE

      CLOSE (2)

      END
```

```
      PROGRAM LA4SURF
C
C     A quick program to generate the speed/accel surface for the LA4
C     using "EPA" bins.
C
C     Author:  Tom Carlson
C     Date:    05/04/93
C-----------------------------------------------------------------------
      INTEGER BINCNT(0:20,-20:20), ITIME(1400)
      REAL BINPCT(0:20,-20:20), SPD(1400), ACL(1400)
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW')
C
C     Begin processing.
C
  100 I = I + 1
      READ (1, *, END=110) ITIME(I), SPD(I)
      GO TO 100
C
C     Compute accels (assign accel from t to t+1 to time t).
C
  110 N = I - 1
      DO 120 I = 1,N
      IF (I .LT. N) THEN
      ACL(I) = SPD(I+1) - SPD(I)
      ELSE
      ACL(I) = 0.0
      END IF
  120 CONTINUE
C
      DO 300 I = 1,N
C-----------------------------------
C     Determine speed bin.
C-----------------------------------
C
      IF (SPD(I) .LT. 0.1) THEN
      IS = 0
      GO TO 220
      END IF

      DO 210 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD(I) .GT. I1 .AND. SPD(I) .LE. I2) THEN
      IS = ISPD
      GO TO 220
      END IF

  210 CONTINUE
C
C-----------------------------------
C     Determine acceleration bin.
C-----------------------------------
C
  220 IF (ACL(I) .GE. -0.5 .AND. ACL(I) .LE. 0.5) THEN
      IA = 0
      GO TO 250
      END IF

      IF (ACL(I) .GT. 0.0) GO TO 235

      DO 230 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL(I) .LT. A1 .AND. ACL(I) .GE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  230 CONTINUE

  235 DO 240 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL(I) .GT. A1 .AND. ACL(I) .LE. A2) THEN
      IA = IACL
```

```
      GO TO 250
      END IF

  240 CONTINUE
C
  250 BINCNT(IS,IA) = BINCNT(IS,IA) + 1
C
  300 CONTINUE
C
C     Bin counting complete.  Print spd/acl counts and percentages for
C     each output file (cold, hot, stabilized).
C
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      BINPCT(ISPD,IACL) = BINCNT(ISPD,IACL) * 100. / N
      WRITE (2, 315) IACL, ISPD, BINCNT(ISPD,IACL), BINPCT(ISPD,IACL)
  315 FORMAT (2(I3,','), I8, ',', F10.6)
  320 CONTINUE
C
      CLOSE (1)
      CLOSE (2)

      END
```

```
      PROGRAM MAKE_CYCLE
C
C     A quick program to make a VEHSIM-ready master driving schedule
C     file from the GETCYCLE .PRN files.
C
C     Unit 1 - Input .PRN file from a GETCYCLE run.
C     Unit 2 - VEHSIM-ready output file.
C
C     Author:  Tom Carlson
C     Date:    04/28/93
C-----------------------------------------------------------------------
C
      CHARACTER*4  CITY
      CHARACTER*80 PRNFILE, OUTFILE, HEADER(3), HDR(3), PART*6
C
      DATA CITY /'BTLA'/, ACLAST /0.0/
      DATA HEADER  / '*DRIVING SCHEDULE ',
     +               ' COMPOSITE BALT/LA92 - CYCLE ',
     +               'INITIAL COND   0.0   0.0   0.0   0.0'/
C
      WRITE (*, 10)
   10 FORMAT (/1X, 'Enter input GETCYCLE .PRN file:')
      READ (*, 20) PRNFILE
   20 FORMAT (A)
      WRITE (*, 30)
   30 FORMAT (/1X, 'Enter cycle VEHSIM part name (< 7 chars):')
      READ (*, 35) PART
   35 FORMAT (A6)
      WRITE (*, 38)
   38 FORMAT (/1X, 'Enter output VEHSIM-ready file name:')
      READ (*, 20) OUTFILE

      L = INDEX (PRNFILE, '.') - 1
C
      OPEN (UNIT=1, STATUS='OLD', FILE=PRNFILE)
      OPEN (UNIT=2, STATUS='NEW', FILE=OUTFILE, CARRIAGECONTROL='LIST')
C
      HDR(1) = HEADER(1)(1:18) // PART
      HDR(2) = CITY // HEADER(2)(1:29) // PART
      HDR(3) = HEADER(3)
      DO 60 L = 1,3
      WRITE (2, 55) HDR(L)
   55 FORMAT (A)
   60 CONTINUE
C
  100 READ (1, *, END=200) ISEG, MTRP, SPD, ACL
      N = N + 1

C     Convert ACL from mph/s to ft/s2 & use the t-1 value (EPA places
C     the accel at the beginning of the time interval, VEHSIM expects
C     it at the end).

      VACL = ACLAST * 5280 / 3600.

      WRITE (2, 150) ISEG, VACL, SPD, GRADE
  150 FORMAT ('SEGMENT ', I4, F6.2, T49, '  .1.0', T73, 2F6.2)

      ACLAST = ACL
      GO TO 100

  200 WRITE (*, 210) N
  210 FORMAT (/1X, 'Processing completed:    ', I6, ' seconds'/)

      END
```

```
      PROGRAM MAKETBL
C
C     A quick program to generate tabular "Watson plots" from
C     input spd/acl surface files in "comma-delimited" format.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    05/10/93
C
C-----------------------------------------------------------------------
C
      CHARACTER*80 FNAME

      REAL PCTBIN(-20:20,0:20), SPDSUM(-20:20), ACLSUM(0:20)
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW', RECL=175,
     +              RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Prompt for table title.
C
      WRITE (*, 50)
   50 FORMAT (/1X, 'Enter title for spd/acl surface:')
      READ (*, 60) FNAME
   60 FORMAT (A80)
C
C
C     Read input mode surface file (unit 1).
C
  100 READ (1, *, END=110) IACL, ISPD, DUMMY, PCTBIN(IACL,ISPD)
      TOT = TOT + PCTBIN(IACL,ISPD)
      GO TO 100
C
C     Compute row (across spd) and column (across acl) totals.
C
  110 DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      SPDSUM(IACL) = SPDSUM(IACL) + PCTBIN(IACL,ISPD)
  120 CONTINUE

      DO 125 ISPD = 0,20
      DO 125 IACL = -20,20
      ACLSUM(ISPD) = ACLSUM(ISPD) + PCTBIN(IACL,ISPD)
  125 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      WRITE (2, 130) FNAME
  130 FORMAT (///65X, A80//60X, 'Speed/Accel Frequency Distribution (%)'
     +        ///70X, 'SPD BIN (mph)'/)
      WRITE (2, 140) (ISPD*5, ISPD = 0,20)
  140 FORMAT ('ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), '  Totals'/
     +    '---------------', 4X, '----', 3X, 20('----',3X), '  ------')
C
      DO 150 IACL = -20,20
      WRITE (2, 145) IACL, (PCTBIN(IACL,ISPD), ISPD = 0,20),SPDSUM(IACL)
  145 FORMAT (6X, I4, 6X, 21F7.3, 4X, F7.3)
  150 CONTINUE
C
      WRITE (2, 155) (ACLSUM(ISPD), ISPD = 0,20), TOT
  155 FORMAT (/5X, 'Totals', 5X, 21F7.3, 4X, F7.3)
C
      END
```

```
      PROGRAM MAKETBL2
C
C     A quick program to generate tabular "Watson plots" from
C     input spd/acl surface files in "comma-delimited" format.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    05/10/93
C
C-----------------------------------------------------------------------
C
      CHARACTER*80 FNAME

      INTEGER CNTBIN(-20:20,0:20), SPDSUM(-20:20), ACLSUM(0:20), TOT
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW', RECL=175,
     +              RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Prompt for table title.
C
      WRITE (*, 50)
   50 FORMAT (/1X, 'Enter title for spd/acl surface:')
      READ (*, 60) FNAME
   60 FORMAT (A80)
C
C
C     Read input mode surface file (unit 1).
C
  100 READ (1, *, END=110) IACL, ISPD, RCNT
      CNTBIN(IACL,ISPD) = RCNT
      TOT = TOT + CNTBIN(IACL,ISPD)
      GO TO 100
C
C     Compute row (across spd) and column (across acl) totals.
C
  110 DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      SPDSUM(IACL) = SPDSUM(IACL) + CNTBIN(IACL,ISPD)
  120 CONTINUE

      DO 125 ISPD = 0,20
      DO 125 IACL = -20,20
      ACLSUM(ISPD) = ACLSUM(ISPD) + CNTBIN(IACL,ISPD)
  125 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      WRITE (2, 130) FNAME
  130 FORMAT (///65X, A80//60X, 'Speed/Accel Frequency Distribution (#)'
     +        ///70X, 'SPD BIN (mph)'/)
      WRITE (2, 140) (ISPD*5, ISPD = 0,20)
  140 FORMAT ('ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), '  Totals'/
     +    '---------------', 4X, '----', 3X, 20('----',3X), '  ------')
C
      DO 150 IACL = -20,20
      WRITE (2, 145) IACL, (CNTBIN(IACL,ISPD), ISPD = 0,20),SPDSUM(IACL)
  145 FORMAT (6X, I4, 6X, 21I7, 3X, I8)
  150 CONTINUE
C
      WRITE (2, 155) (ACLSUM(ISPD), ISPD = 0,20), TOT
  155 FORMAT (/5X, 'Totals', 5X, 21I7, 3X, I8)
C
      END
```

```
      PROGRAM MAKETBL3
C
C     A quick program to generate tabular "Watson plots" from
C     input spd/acl surface files in "comma-delimited" format.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    05/10/93
C
C-----------------------------------------------------------------------
C
      CHARACTER*80 FNAME

      REAL PCTBIN(-20:20,0:20), SPDSUM(2,-20:20), ACLSUM(2,0:20)
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW', RECL=186,
     +             RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Prompt for table title.
C
      WRITE (*, 50)
   50 FORMAT (/1X, 'Enter title for spd/acl surface:')
      READ (*, 60) FNAME
   60 FORMAT (A80)
C
C
C     Read input mode surface file (unit 1).
C
  100 READ (1, *, END=110) IACL, ISPD, DUMMY, PCTBIN(IACL,ISPD)
      TOT = TOT + PCTBIN(IACL,ISPD)
      GO TO 100
C
C     Compute row (across spd) and column (across acl) totals.
C
  110 DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      SPDSUM(1,IACL) = SPDSUM(1,IACL) + PCTBIN(IACL,ISPD)
      SPDSUM(2,IACL) = SPDSUM(2,IACL) + ABS (PCTBIN(IACL,ISPD))
  120 CONTINUE

      DO 125 ISPD = 0,20
      DO 125 IACL = -20,20
      ACLSUM(1,ISPD) = ACLSUM(1,ISPD) + PCTBIN(IACL,ISPD)
      ACLSUM(2,ISPD) = ACLSUM(2,ISPD) + ABS (PCTBIN(IACL,ISPD))
      TOTNET = TOTNET + PCTBIN(IACL,ISPD)
      TOTABS = TOTABS + ABS (PCTBIN(IACL,ISPD))
  125 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      WRITE (2, 130) FNAME
  130 FORMAT (///40X, A80//60X, 'Speed/Accel Frequency Distribution (%)'
     +        ///70X, 'SPD BIN (mph)'/)
      WRITE (2, 140) (ISPD*5, ISPD = 0,20)
  140 FORMAT (166X,'   Net      Abs'/
     + 'ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), 2('   Totals')/
     + '---------------', 4X, '----', 3X, 20('----',3X), 2('   ------'))
C
      DO 150 IACL = -20,20
      WRITE (2, 145) IACL, (PCTBIN(IACL,ISPD), ISPD = 0,20),
     +                     (SPDSUM(I,IACL), I = 1,2)
  145 FORMAT (6X, I4, 6X, 21F7.3, 5X, F7.3, 2X, F7.3)
  150 CONTINUE
C
      WRITE (2, 155) (ACLSUM(1,ISPD), ISPD = 0,20), TOTNET,
     +               (ACLSUM(2,ISPD), ISPD = 0,20), TOTABS
  155 FORMAT (/3X, 'Net Totals', 3X, 21F7.3, 5X, F7.3, 2X/
```

```
     +         3X, 'Abs Totals', 3X, 21F7.3, 14X, F7.3)
C
      END
```

```
      PROGRAM MICTRIPS
C
C     This program reads a second-by-second driving cycle for a region
C     and develops speed/acceleration profiles by microtrip.  A
C     microtrip is defined as travel between distinct stops.
C
C     This version operates on PATROL not COMPOSITE data.
C
C     I/O:   Unit 1 (input)  - Sec by sec drive cycle data.
C            Unit 2 (output) - Microtrip profile statistics.
C            Unit 3 (output) - Route start profile statistics.
C
C     Author:  Tom Carlson
C     Date:    4/20/93
C
C-----------------------------------------------------------------------

      CHARACTER*2  CTRIP
      CHARACTER*12 TRIPID, LTRIPID
      CHARACTER*14 MTRIPID

      INTEGER*2 CRUS, MTIME, LCRUS
      INTEGER*4 FREQ(-20:20,0:20)

      REAL SPD_SUM, SPD_SUM2, ACL_SUM2,
     +     SPD_MEAN, SPD_STD, ACL_MAX, SPD_MAX

C-----------------------------------------------------------------------

      OPEN (UNIT=2, STATUS='NEW', RECL=3512, CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE', FORM='FORMATTED')

C-----------------------------------------------------------------------

      WRITE (2, 50)
   50 FORMAT (1X,
     +'MTRP                    INIT   TIME AVG SPD MAX SPD STD SPD ',
     +'MAX ACL   SPD/ACL'/
     +1X,
     +' NO     MTRIP ID        REC    (s)  (mph)   (mph)   (mph)  ',
     +'(mph/s)   FREQ -->'/)


  100 READ (1, 101, END=200) TRIPID, ISEC, SPD, ACL, ISPD, IACL
  101 FORMAT (A12, I6, 2F6.2, 2I3)
      NREC = NREC + 1

      IF (SPD .GT. 0.0) THEN
      CRUS = 1
      IFLAG = 1
      ELSE
      CRUS = 0
      END IF

C     ================================================================
      IF (TRIPID .NE. LTRIPID .OR. CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND.
     +    IFLAG .EQ. 1) THEN
C        ---------------------------------------------------------
         IFLAG = 0
         IF (TRIPID .NE. LTRIPID) THEN

C     If a "trailing idle" microtrip is detected at the end of a route,
C     remove it (re-zero sums and decrement microtrip counter MTRIP).
C           - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
            IF (NREC.GT.1 .AND. SPD_MAX.LE.0.5 .AND. NTRIP.GT.1) THEN
            MTRIP = MTRIP - 1
            ITRP = ITRP + 1
            NTRIP = 0
            GO TO 122
            END IF
C           - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         ITRP = ITRP + 1
         NTRIP = 0
         END IF
C        ---------------------------------------------------------
C
C     New microtrip detected.  Generate statistics for "last" microtrip
C     and output them to unit 2, then zero sums and increment counters.
C
      IF (NREC .EQ. 1) GO TO 130
```

```
C
      IF (MTIME .LT. 2 .OR. SPD_MAX .LE. 0.5) THEN
      MTRIP = MTRIP - 1
      GO TO 122
      END IF
C
      SPD_MEAN = SPD_SUM / MTIME
      SPD_STD = SQRT ((SPD_SUM2 - MTIME * SPD_MEAN**2) / (MTIME - 1))
C
      WRITE (2, 120) MTRIP, MTRIPID, IREC, MTIME, SPD_MEAN, SPD_MAX,
     +   SPD_STD, ACL_MAX, ((FREQ(IA,IS), IS=0,20), IA=-20,20)
  120 FORMAT (I5,1X,A14,1X,I7,1X,I5, 3(1X, F7.2),1X, F7.3,1X, 861(I4))
      NGOOD = NGOOD + MTIME
C
C     The stats for "last" m-trip have been output. Re-initialize
C     sums and counters for current m-trip.
C
  122 SPD_SUM  = 0.0
      SPD_SUM2 = 0.0
      SPD_MAX = 0.0
      ACL_MAX = 0.0
      MTIME = 0
      DO 125 IA = -20,20
      DO 125 IS = 0,20
      FREQ(IA,IS) = 0
  125 CONTINUE
C
  130 MTRIP = MTRIP + 1
      NTRIP = NTRIP + 1
      IREC = NREC
      L = INDEX (TRIPID, ' ') - 1
      IF (L .LE. 0) L = 12
      WRITE (CTRIP, FMT='(I2.2)') NTRIP
      MTRIPID = TRIPID(1:L) // CTRIP
      GO TO 150
      END IF
C     ====================================================================
      LCRUS = CRUS
  150 LTRIPID = TRIPID

C     Increment statistical sums and counters for each microtrip.

      IF (SPD .GT. SPD_MAX) SPD_MAX = SPD
      IF (ACL .GT. ACL_MAX) ACL_MAX = ACL
      SPD_SUM  = SPD_SUM  + SPD
      SPD_SUM2 = SPD_SUM2 + SPD**2
      MTIME = MTIME + 1

C     ---  Increment speed/accel bin  ---

      FREQ(IACL,ISPD) = FREQ(IACL,ISPD) + 1

C     Loop back and process the next record.

      GO TO 100

C-----------------------------------------------------------------------
C
C     Write "hanging" m-trip record, log summary results and end.
C
C     If a "trailing idle" microtrip is detected at the end of a route,
C     remove it (re-zero sums and decrement microtrip counter MTRIP).
C
  200 IF (NREC .GT. 1 .AND. SPD_MAX .LE. 0.5 .AND. NTRIP .GT. 1) THEN
      MTRIP = MTRIP - 1
      GO TO 300
      END IF
C
      SPD_MEAN = SPD_SUM / MTIME
      SPD_STD = SQRT ((SPD_SUM2 - MTIME * SPD_MEAN**2) / (MTIME - 1))
      ACL_STD = SQRT (ACL_SUM2 / (MTIME - 1))

      WRITE (2, 120) MTRIP, MTRIPID, IREC, MTIME, SPD_MEAN, SPD_MAX,
     +   SPD_STD, ACL_MAX, ((FREQ(IA,IS), IS=0,20), IA=-20,20)
      NGOOD = NGOOD + MTIME

  300 WRITE (*, 400) NREC, NGOOD, MTRIP, ITRP
  400 FORMAT (/1X, 'Processing Summary'/
     +         1X, '------------------'/
     +         1X, 'Input records: ', I7/
```

```
      PROGRAM MODSPLIT_BALT
C
C     A quick program to split the Baltimore file (EPABALT.DAT) into
C     two separate "mode" files:  (1) start and (2) stabilized.  Target
C     start (first START_TIME secs) and stabilized start surfaces are
C     also generated.
C
      CHARACTER*12 TRIPID, LTRIPID
      INTEGER START_TIME, NMODE(2), CRUS, BINCNT(2,0:20,-20:20), NST(2)
      REAL BINPCT(2,0:20,-20:20)
C
      DATA START_TIME /240/
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW')
      OPEN (UNIT=3, CARRIAGECONTROL='LIST', STATUS='NEW')
      OPEN (UNIT=4, CARRIAGECONTROL='LIST', STATUS='NEW')
      OPEN (UNIT=5, CARRIAGECONTROL='LIST', STATUS='NEW')
C
C     Begin processing.
C
  100 READ (1, 101, END=300) TRIPID, ITIME, SPD, ACL
  101 FORMAT (A12, I6, 2F6.2)
      I = I + 1
C
C     Check if this record is the beginning of a trip.
C
      IF (ITIME .EQ. 0) THEN
      IWAIT = 0
      IMODE = 1
      END IF
C
C     Detect when microtrip has ended (MFLAG = 1).
C
      MFLAG = 0
C
      IF (SPD .GT. 0.0) THEN
      CRUS = 1
      IFLAG = 1
      ELSE
      CRUS = 0
      END IF

      IF (CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND. IFLAG .EQ. 1) IFLAG = 1

      IF (TRIPID .NE. LTRIPID .OR. CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND.
     +    IFLAG .EQ. 1) THEN
      IFLAG = 0
      MFLAG = 1
      GO TO 105
      END IF
C     ======================================================================
      LCRUS = CRUS
  105 LTRIPID = TRIPID
C
C     Determine emission mode (start/stabilized) as follows:  Is this a
C     start or stabilized record? (IMODE:  1 - Start, 2 - Stabilized).
C
      IF (ITIME .EQ. START_TIME) IWAIT = 1
C
C     Just beyond START_TIME, continue marking the records as starts
C     until the end of the current microtrip is reached (MFLAG = 1).
C
      IF (IWAIT .EQ. 1) THEN
      IMODE = 1
         IF (MFLAG .EQ. 1) THEN
         IWAIT = 0
         IMODE = 2
         END IF
      END IF
C
C-----------------------------------
C     Determine speed bin.
C-----------------------------------
C
      IF (SPD .LT. 0.1) THEN
      IS = 0
      GO TO 220
      END IF
```

```
      DO 210 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD .GT. I1 .AND. SPD .LE. I2) THEN
      IS = ISPD
      GO TO 220
      END IF

  210 CONTINUE
C
C-------------------------------
C     Determine acceleration bin.
C-------------------------------
C
  220 IF (ACL .GE. -0.5 .AND. ACL .LE. 0.5) THEN
      IA = 0
      GO TO 250
      END IF

      IF (ACL .GT. 0.0) GO TO 235

      DO 230 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL .LT. A1 .AND. ACL .GE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  230 CONTINUE

  235 DO 240 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL .GT. A1 .AND. ACL .LE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  240 CONTINUE
C
  250 IF (ITIME .LE. START_TIME) THEN
      BINCNT(1,IS,IA) = BINCNT(1,IS,IA) + 1
      NST(1) = NST(1) + 1
      ELSE
      BINCNT(2,IS,IA) = BINCNT(2,IS,IA) + 1
      NST(2) = NST(2) + 1
      END IF

      NMODE(IMODE) = NMODE(IMODE) + 1
      IOUT = IMODE + 1
      WRITE (IOUT, 270) TRIPID, ITIME, SPD, ACL, IS, IA
  270 FORMAT (A12, I6, 2F6.2, 2I3)
      GO TO 100
C
C     Bin counting complete.  Print spd/acl counts and percentages for
C     each output file (cold, hot, stabilized).
C
  300 DO 320 IST = 1,2
      IOUT = IST + 3
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      BINPCT(IST,ISPD,IACL) = BINCNT(IST,ISPD,IACL) * 100. /
     +                        NST(IST)
      WRITE (IOUT, 315) IACL, ISPD, BINCNT(IST,ISPD,IACL),
     +                  BINPCT(IST,ISPD,IACL)
  315 FORMAT (2(I3,','), I8, ',', F10.6)
  320 CONTINUE

C
      CLOSE (1)
      CLOSE (2)
      CLOSE (3)
      CLOSE (4)
      CLOSE (5)

      WRITE (*, 400) (NMODE(J), NST(J), J = 1,2), I
```

```
400 FORMAT (/' Records Processed:'//
   +' Start Recs       --> ', I7, 5X,'Start Surface Recs       --> ',I7/
   +' Stabilized Recs --> ', I7, 5X,'Stabilized Surface Recs --> ',I7/
   +         ' Total Recs       --> ', I7/)

    END
```

```
      PROGRAM MODSPLIT_LA92
C
C     A quick program to filter the LA92 file (EPALA92.DAT), selecting
C     only stabilized operation.  (Stablized = Start Time - 1 minute).
C     The target stabilized surface is also computed.
C
      CHARACTER*12 TRIPID, LTRIPID
      INTEGER START_TIME, NMODE(2), CRUS, BINCNT(2,0:20,-20:20), NST(2)
      REAL BINPCT(2,0:20,-20:20)
C
      DATA START_TIME /180/
C
      OPEN (UNIT=1, STATUS='OLD')
C      OPEN (UNIT=2, CARRIAGECONTROL='LIST', STATUS='NEW')
      OPEN (UNIT=3, CARRIAGECONTROL='LIST', STATUS='NEW')
C      OPEN (UNIT=4, CARRIAGECONTROL='LIST', STATUS='NEW')
      OPEN (UNIT=5, CARRIAGECONTROL='LIST', STATUS='NEW')
C
C     Begin processing.
C
  100 READ (1, 101, END=300) TRIPID, ITIME, SPD, ACL
  101 FORMAT (A12, I6, 2F6.2)
      I = I + 1
C
C     Check if this record is the beginning of a trip.
C
      IF (ITIME .EQ. 0) THEN
      IWAIT = 0
      IMODE = 1
      END IF
C
C     Detect when microtrip has ended (MFLAG = 1).
C
      MFLAG = 0
C
      IF (SPD .GT. 0.0) THEN
      CRUS = 1
      IFLAG = 1
      ELSE
      CRUS = 0
      END IF

      IF (CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND. IFLAG .EQ. 1) IFLAG = 1

      IF (TRIPID .NE. LTRIPID .OR. CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND.
     +    IFLAG .EQ. 1) THEN
      IFLAG = 0
      MFLAG = 1
      GO TO 105
      END IF
C     ================================================================
      LCRUS = CRUS
  105 LTRIPID = TRIPID
C
C     Determine emission mode (start/stabilized) as follows:  Is this a
C     start or stabilized record? (IMODE:  1 - Start, 2 - Stabilized).
C
      IF (ITIME .EQ. START_TIME) IWAIT = 1
C
C     Just beyond START_TIME, continue marking the records as starts
C     until the end of the current microtrip is reached (MFLAG = 1).
C
      IF (IWAIT .EQ. 1) THEN
      IMODE = 1
         IF (MFLAG .EQ. 1) THEN
         IWAIT = 0
         IMODE = 2
         END IF
      END IF
C
      IF (IMODE .EQ. 1) GO TO 250
C
C-----------------------------------
C     Determine speed bin.
C-----------------------------------
C
      IF (SPD .LT. 0.1) THEN
      IS = 0
      GO TO 220
      END IF
```

```
      DO 210 ISPD = 1,20
      I1 = (ISPD-1) * 5
      I2 = I1 + 5

      IF (SPD .GT. I1 .AND. SPD .LE. I2) THEN
      IS = ISPD
      GO TO 220
      END IF

  210 CONTINUE
C
C-----------------------------------
C     Determine acceleration bin.
C-----------------------------------
C
  220 IF (ACL .GE. -0.5 .AND. ACL .LE. 0.5) THEN
      IA = 0
      GO TO 250
      END IF

      IF (ACL .GT. 0.0) GO TO 235

      DO 230 IACL = -20,-1
      A1 = IACL + 0.5
      A2 = IACL - 0.5

      IF (ACL .LT. A1 .AND. ACL .GE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  230 CONTINUE

  235 DO 240 IACL = 1,20
      A1 = IACL - 0.5
      A2 = IACL + 0.5

      IF (ACL .GT. A1 .AND. ACL .LE. A2) THEN
      IA = IACL
      GO TO 250
      END IF

  240 CONTINUE
C
  250 IF (ITIME .LE. START_TIME) THEN
      BINCNT(1,IS,IA) = BINCNT(1,IS,IA) + 1
      NST(1) = NST(1) + 1
      ELSE
      BINCNT(2,IS,IA) = BINCNT(2,IS,IA) + 1
      NST(2) = NST(2) + 1
      END IF

      NMODE(IMODE) = NMODE(IMODE) + 1
      IF (IMODE .EQ. 2) WRITE (3, 270) TRIPID, ITIME, SPD, ACL, IS, IA
  270 FORMAT (A12, I6, 2F6.2, 2I3)
      GO TO 100
C
C     Bin counting complete.  Print spd/acl counts and percentages for
C     each output file (cold, hot, stabilized).
C
  300 DO 320 IST = 1,2
      IOUT = IST + 3
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      BINPCT(IST,ISPD,IACL) = BINCNT(IST,ISPD,IACL) * 100. /
     +                         NST(IST)
      WRITE (IOUT, 315) IACL, ISPD, BINCNT(IST,ISPD,IACL),
     +                  BINPCT(IST,ISPD,IACL)
  315 FORMAT (2(I3,','), I8, ',', F10.6)
  320 CONTINUE
C
      CLOSE (1)
C      CLOSE (2)
      CLOSE (3)
C      CLOSE (4)
      CLOSE (5)

      WRITE (*, 400) (NMODE(J), NST(J), J = 1,2), I
  400 FORMAT (/' Records Processed:'//
```

```
+' Start Recs      --> ', I7, 5X,'Start Surface Recs      --> ',I7/
+' Stabilized Recs --> ', I7, 5X,'Stabilized Surface Recs --> ',I7/
+         ' Total Recs      --> ', I7/)

 END
```

```
      PROGRAM OUTFTP
C
C     A quick program to generate a "outside of FTP" spd/acl surface
C     from the "Composite" and "LA4" input spd/acl surface files.
C     The new surface contains only travel beyond the boundaries of the
C     FTP (in spd/acl space) and is re-normalized.
C
C     All input and output surfaces are in "comma-delimited" format.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    05/10/93
C
C-----------------------------------------------------------------------
C
      INTEGER IANEG(0:20), IAPOS(0:20)

      REAL PCTBIN(3,-20:20,0:20), SPDSUM(-20:20), ACLSUM(0:20),
     +     BINCNT(3,-20:20,0:20), TOTCNT(3)
C
      OPEN (UNIT=1, FILE='CFRQT.DAT', STATUS='OLD')
      OPEN (UNIT=2, FILE='LA4FRQ.DAT', STATUS='OLD')
      OPEN (UNIT=3, FILE='OUTFRQ.DAT', STATUS='NEW',
     +              CARRIAGECONTROL='LIST')
      OPEN (UNIT=4, FILE='OUTFRQ.TBL', CARRIAGECONTROL='LIST',
     +              STATUS='NEW', RECL=175, RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Read "Composite In-Use" surface file (unit 1).
C
  100 READ (1, *, END=110) IACL, ISPD, BINCNT(1,IACL,ISPD),
     +                     PCTBIN(1, IACL,ISPD)
      TOTCNT(1) = TOTCNT(1) + BINCNT(1,IACL,ISPD)
      GO TO 100
C
C     Read "LA4" surface file (unit 2).
C
  110 READ (2, *, END=120) IACL, ISPD, BINCNT(2,IACL,ISPD),
     +                     PCTBIN(2,IACL,ISPD)
      TOTCNT(2) = TOTCNT(2) + BINCNT(2,IACL,ISPD)
      GO TO 110
C
C-----------------------------------------------------------------------
C     Scan for FTP boundaries.
C-----------------------------------------------------------------------
C
C     Loop across speed bins -- from big to small speed bins.
C
  120 DO 150 ISPD = 20,0,-1
C
C     Check decel region -- scan from big to small negative bins.
C
      DO 130 IACL = -20,-1
      IF (BINCNT(2,IACL,ISPD) .GT. 0.0) THEN
      IANEG(ISPD) = IACL
      GO TO 135
      END IF
  130 CONTINUE

  135 DO 140 IACL = 20,0,-1
      IF (BINCNT(2,IACL,ISPD) .GT. 0.0) THEN
      IAPOS(ISPD) = IACL
      GO TO 145
      END IF
  140 CONTINUE

  145 IF (IS .EQ. 0 .AND. (IANEG(ISPD) .NE. 0 .OR. IAPOS(ISPD) .NE. 0))
     +    IS = ISPD

  150 CONTINUE
C
```

```
C-----------------------------------------------------------------------
C     Bin boundary numbers found -- now generate re-normalized surface
C     from In-Use activity BEYOND these detected boundaries.
C-----------------------------------------------------------------------
C
      DO 190 ISPD = 0,20
C
C     Beyond top FTP speed.
C
      IF (ISPD .GT. IS) THEN
      DO 160 IACL = -20,20
      BINCNT(3,IACL,ISPD) = BINCNT(1,IACL,ISPD)
      TOTCNT(3) = TOTCNT(3) + BINCNT(3,IACL,ISPD)
  160 CONTINUE
      GO TO 190
      END IF
C
C     Within FTP speed range -- decel region.
C
      DO 170 IACL = -20,IANEG(ISPD)-1
      BINCNT(3,IACL,ISPD) = BINCNT(1,IACL,ISPD)
      TOTCNT(3) = TOTCNT(3) + BINCNT(3,IACL,ISPD)
  170 CONTINUE
C
C     Within FTP speed range -- accel region.
C
      DO 180 IACL = IAPOS(ISPD)+1,20
      BINCNT(3,IACL,ISPD)= BINCNT(1,IACL,ISPD)
      TOTCNT(3) = TOTCNT(3) + BINCNT(3,IACL,ISPD)
  180 CONTINUE
C
  190 CONTINUE
C
C-----------------------------------------------------------------------
C     Selected counts (outside the FTP boundaries) applied to new
C     surface array (first element 3) -- now normalize and output.
C-----------------------------------------------------------------------
C
      DO 200 IACL = -20,20
      DO 200 ISPD = 0,20
      PCTBIN(3,IACL,ISPD) = BINCNT(3,IACL,ISPD) * 100 / TOTCNT(3)
      WRITE (3, 195) IACL, ISPD,BINCNT(3,IACL,ISPD), PCTBIN(3,IACL,ISPD)
  195 FORMAT (2(I3,','), F8.0, ',', F10.6)
  200 CONTINUE
C
C     Compute row (across spd) and column (across acl) totals.
C
  310 DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      SPDSUM(IACL) = SPDSUM(IACL) + PCTBIN(3,IACL,ISPD)
  320 CONTINUE

      DO 325 ISPD = 0,20
      DO 325 IACL = -20,20
      ACLSUM(ISPD) = ACLSUM(ISPD) + PCTBIN(3,IACL,ISPD)
      TOT = TOT + PCTBIN(3,IACL,ISPD)
  325 CONTINUE
C
C-----------------------------------------------------------------------
C     Output results in tabular form, echo input surfaces first.
C-----------------------------------------------------------------------
C
      WRITE (4, 330)
  330 FORMAT (///60X, '       "Outside FTP" Driving'
     +         //60X, 'Speed/Accel Frequency Distribution (%)'
     +        ///70X, 'SPD BIN (mph)'/)
      WRITE (4, 340) (ISPD*5, ISPD = 0,20)
  340 FORMAT ('ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), '  Totals'/
     +    '---------------', 4X, '----', 3X, 20('----',3X), '  ------')
C
      DO 350 IACL = -20,20
      WRITE (4, 345) IACL, (PCTBIN(3,IACL,ISPD), ISPD = 0,20),
     +               SPDSUM(IACL)
  345 FORMAT (6X, I4, 6X, 21F7.3, 4X, F7.3)
  350 CONTINUE
C
      WRITE (4, 355) (ACLSUM(ISPD), ISPD = 0,20), TOT
  355 FORMAT (/5X, 'Totals', 5X, 21F7.3, 4X, F7.3)
C
      WRITE (*, 360) (TOTCNT(I), I = 1,3)
```

```
  360 FORMAT (/' Distribution Counts:   Composite -----> ', F8.0/
     +         '                        LA4 -----------> ', F8.0/
     +         '                        Outside FTP ---> ', F8.0/)
C
      END
```

```
      PROGRAM OUTFTPSTAB
C
C     A quick program to generate a "outside of FTP" spd/acl surface
C     from the "Composite" and "LA4" input spd/acl surface files.
C     The new surface contains only travel beyond the boundaries of the
C     FTP (in spd/acl space) and is re-normalized.
C
C     All input and output surfaces are in "comma-delimited" format.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    05/10/93
C
C-----------------------------------------------------------------------
C
      INTEGER IANEG(0:20), IAPOS(0:20)

      REAL PCTBIN(3,-20:20,0:20), SPDSUM(-20:20), ACLSUM(0:20),
     +     BINCNT(3,-20:20,0:20), TOTCNT(3)
C
      OPEN (UNIT=1, FILE='CFRQS.DAT', STATUS='OLD')
      OPEN (UNIT=2, FILE='LA4FRQ.DAT', STATUS='OLD')
C      OPEN (UNIT=3, FILE='OUTFRQ.DAT', STATUS='NEW',
C     +              CARRIAGECONTROL='LIST')
C      OPEN (UNIT=4, FILE='OUTFRQ.TBL', CARRIAGECONTROL='LIST',
C     +              STATUS='NEW', RECL=175, RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Read "Composite In-Use" surface file (unit 1).
C
  100 READ (1, *, END=110) IACL, ISPD, BINCNT(1,IACL,ISPD),
     +                     PCTBIN(1, IACL,ISPD)
      TOTCNT(1) = TOTCNT(1) + BINCNT(1,IACL,ISPD)
      GO TO 100
C
C     Read "LA4" surface file (unit 2).
C
  110 READ (2, *, END=120) IACL, ISPD, BINCNT(2,IACL,ISPD),
     +                     PCTBIN(2,IACL,ISPD)
      TOTCNT(2) = TOTCNT(2) + BINCNT(2,IACL,ISPD)
      GO TO 110
C
C-----------------------------------------------------------------------
C     Scan for FTP boundaries.
C-----------------------------------------------------------------------
C
C     Loop across speed bins -- from big to small speed bins.
C
  120 DO 150 ISPD = 20,0,-1
C
C     Check decel region -- scan from big to small negative bins.
C
      DO 130 IACL = -20,-1
      IF (BINCNT(2,IACL,ISPD) .GT. 0.0) THEN
      IANEG(ISPD) = IACL
      GO TO 135
      END IF
  130 CONTINUE

  135 DO 140 IACL = 20,0,-1
      IF (BINCNT(2,IACL,ISPD) .GT. 0.0) THEN
      IAPOS(ISPD) = IACL
      GO TO 145
      END IF
  140 CONTINUE

  145 IF (IS .EQ. 0 .AND. (IANEG(ISPD) .NE. 0 .OR. IAPOS(ISPD) .NE. 0))
     +    IS = ISPD

  150 CONTINUE
C
```

```
C-----------------------------------------------------------------------
C     Bin boundary numbers found -- now generate re-normalized surface
C     from In-Use activity BEYOND these detected boundaries.
C-----------------------------------------------------------------------
C
      DO 190 ISPD = 0,20
C
C     Beyond top FTP speed.
C
      IF (ISPD .GT. IS) THEN
      DO 160 IACL = -20,20
      BINCNT(3,IACL,ISPD) = BINCNT(1,IACL,ISPD)
      TOTCNT(3) = TOTCNT(3) + BINCNT(3,IACL,ISPD)
  160 CONTINUE
      GO TO 190
      END IF
C
C     Within FTP speed range -- decel region.
C
      DO 170 IACL = -20,IANEG(ISPD)-1
      BINCNT(3,IACL,ISPD) = BINCNT(1,IACL,ISPD)
      TOTCNT(3) = TOTCNT(3) + BINCNT(3,IACL,ISPD)
  170 CONTINUE
C
C     Within FTP speed range -- accel region.
C
      DO 180 IACL = IAPOS(ISPD)+1,20
      BINCNT(3,IACL,ISPD)= BINCNT(1,IACL,ISPD)
      TOTCNT(3) = TOTCNT(3) + BINCNT(3,IACL,ISPD)
  180 CONTINUE
C
  190 CONTINUE
C
C-----------------------------------------------------------------------
C     Selected counts (outside the FTP boundaries) applied to new
C     surface array (first element 3) -- now normalize and output.
C-----------------------------------------------------------------------
C
      DO 200 IACL = -20,20
      DO 200 ISPD = 0,20
      PCTBIN(3,IACL,ISPD) = BINCNT(3,IACL,ISPD) * 100 / TOTCNT(3)
C      WRITE (3, 195) IACL, ISPD,BINCNT(3,IACL,ISPD), PCTBIN(3,IACL,ISPD)
C  195 FORMAT (2(I3,','), F8.0, ',', F10.6)
  200 CONTINUE
C
C     Compute row (across spd) and column (across acl) totals.
C
  310 DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      SPDSUM(IACL) = SPDSUM(IACL) + PCTBIN(3,IACL,ISPD)
  320 CONTINUE

      DO 325 ISPD = 0,20
      DO 325 IACL = -20,20
      ACLSUM(ISPD) = ACLSUM(ISPD) + PCTBIN(3,IACL,ISPD)
      TOT = TOT + PCTBIN(3,IACL,ISPD)
  325 CONTINUE
C
C-----------------------------------------------------------------------
C     Output results in tabular form, echo input surfaces first.
C-----------------------------------------------------------------------
C
C      WRITE (4, 330)
C  330 FORMAT (///60X, '       "Outside FTP" Driving'
C     +         //60X, 'Speed/Accel Frequency Distribution (%)'
C     +        ///70X, 'SPD BIN (mph)'//)
C      WRITE (4, 340) (ISPD*5, ISPD = 0,20)
C  340 FORMAT ('ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X), '  Totals'/
C     +    '---------------', 4X, '----', 3X, 20('----',3X), '  ------')
C
      DO 350 IACL = -20,20
C      WRITE (4, 345) IACL, (PCTBIN(3,IACL,ISPD), ISPD = 0,20),
C     +               SPDSUM(IACL)
C  345 FORMAT (6X, I4, 6X, 21F7.3, 4X, F7.3)
  350 CONTINUE
C
C      WRITE (4, 355) (ACLSUM(ISPD), ISPD = 0,20), TOT
C  355 FORMAT (/5X, 'Totals', 5X, 21F7.3, 4X, F7.3)
C
      WRITE (*, 360) (TOTCNT(I), I = 1,3)
```

```
  360 FORMAT (/' Distribution Counts:   Composite -----> ', F8.0/
     +         '                        LA4 -----------> ', F8.0/
     +         '                        Outside FTP ---> ', F8.0/)
C
      END
```

```
      PROGRAM GETOUT75
C
C     GETOUT reads the "microtrips profiles" files and the target spd/
C     acl "Beyond FTP" surface file for the Baltimore & LA92 data
C     and selects a subset of linked microtrips which best match the
C     target speed/accel surface using the Best Incremental.
C
C     GETOUT is a modified version of the GETSTAB program.
C     The sample from which these additional microtrips are
C     selected consists of all Baltimore and LA92 microtrips, with
C     weighting factors which:
C
C        (1)  allow 50/50 selection probability of stabilized m-trips
C             between Baltimore and LA92; and
C        (2)  maintain initial weighting of start/stabilized m-trips
C             encountered in the Baltimore dataset.
C
C     NOTE:
C     This version does not use m-trips weightings discussed above, only
C     the embedded weightings in the input target surface.
C
C     I/O:   Unit 1 (input)  - Spd/acl target "surface" for sec-by-sec
C                              drive cycle data (Balt/LA92).
C            Unit 2 (input)  - Microtrip profile statistics.
C            Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    5/11/93
C
C-----------------------------------------------------------------------

      PARAMETER (MNUM=7283, NBST=75)

      CHARACTER*3  CLOOP
      CHARACTER*14 MTRIPID(MNUM), IDBST(NBST)
      CHARACTER*80 OUTFILE, OUTNAME

      INTEGER*2 MCNT(MNUM,-20:20,0:20), ITIME(MNUM), M,I, ITIMEBST(NBST)
      INTEGER*4 IREC(MNUM), TRIPLIST(1000), IRECBST(NBST), MTRPBST(NBST)

      REAL SPD_MEAN(MNUM), SPD_STD(MNUM), ACL_MAX(MNUM), DIFF(4),
     +     SPD_MAX(MNUM), CFREQ(-20:20,0:20), TRIPDIFF(1000),
     +     CCNT(-20:20,0:20), PFREQ(-20:20,0:20), TCNT(-20:20,0:20),
     +     ASPDBST(NBST), TSPDBST(NBST), SSPDBST(NBST), TACLBST(NBST),
     +     DIFFBST(NBST,4)

      DATA NCYCLES / 1 /
      DATA DIFFBST / NBST*999., NBST*999., NBST*999., NBST*999. /

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD', RECL=1760, FORM='UNFORMATTED')

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for output filename and cycle control parameters.
C
      WRITE (*, 2)
    2 FORMAT (/1X, 'Enter output (selected trips & stats) filename:')
      READ (*, 3) OUTFILE
    3 FORMAT (A)
      WRITE (*, 4)
    4 FORMAT (/1X, 'Enter target cycle time (min):')
      READ (*, *) RTIME

      TIMELIMIT = RTIME * 60
C
C     Read in target surface (unit 1).
C
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      PFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in microtrip profiles (unit 2).
C
   20 READ (2, END=50) I, MTRIPID(I), IREC(I), ITIME(I),
```

```
     +      SPD_MEAN(I), SPD_MAX(I), SPD_STD(I), ACL_MAX(I),
     +      ((MCNT(I,IACL,ISPD), ISPD=0,20), IACL=-20,20)
C   30 FORMAT (I5,1X,A14,1X,I7,1X,I5, 3(1X, F7.2),1X, F7.3,1X, 861(I4))
      IN2 = IN2 + 1
      GO TO 20
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Begin microtrip selection using Best Incremental
C     logic.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C=======================================================================
C     Top-level loop 500 iterates the selection process to generate
C     a number of cycles (NCYCLES).  Each cycle is written to a
C     separate output file on unit 3.
C=======================================================================
C
   50 MTRIPS = IN2
      DO 500 ILOOP = 1,NCYCLES
C
C     Initialize cycle sums and counter.
C
      NTRP = 0
      TRIPTIME = 0.0
      STIME = 0.0
      CSPD = 0.0
      DO 80 IACL = -20,20
      DO 80 ISPD = 0,20
      TCNT(IACL,ISPD) = 0.0
   80 CONTINUE
C
      IPRD = INDEX (OUTFILE, '.')
      IEND = INDEX (OUTFILE, ' ') - 1
      WRITE (CLOOP, FMT='(I3.3)') ILOOP
      OUTNAME = OUTFILE(1:IPRD-1) // '-' // CLOOP // OUTFILE(IPRD:IEND)
      IOUT = INDEX (OUTNAME, ' ') - 1
C
C     Open "current" filename.
C
      OPEN (UNIT=3, FILE=OUTNAME, STATUS='NEW', CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE')
C
C     Write output file header.
C
      WRITE (3, 90) RTIME, OUTNAME(1:IOUT)
   90 FORMAT (5X, 'BEST INCREMENTAL Trip Selection Method', T54,
     +        'Target Time:  ', F5.2, ' min'//
     +        13X, 'Filename: ', A // 8X,
     +'MTRP                  INIT TIME  AVG SPD MAX SPD STD SPD MAX ACL'
     + ,'   TOTAL    CRUISE     ACCEL     DECEL'/2X, '#     ',
     +' NO     MTRIP ID       REC  (s)   (mph)   (mph)   (mph)  (mph/s)'
     + ,'  DIFFSUM   DIFFSUM   DIFFSUM   DIFFSUM'/)
C
C-----------------------------------------------------------------------
      DIFFMIN = 999.
C
C     Loop thru all microtrips in search of best incremental trip(s).
C
      DO 350 I = 1, MTRIPS
C
C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)
C
      DO 305 II = 1,4
      DIFF(II) = 0.0
  305 CONTINUE
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(I,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(I))
      DIFF(1) = DIFF(1) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      IF (IACL .EQ. 0) THEN
      DIFF(2) = DIFF(2) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      ELSE IF (IACL .GT. 0) THEN
      DIFF(3) = DIFF(3) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      ELSE IF (IACL .LT. 0) THEN
```

```
      DIFF(4) = DIFF(4) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      END IF
  320 CONTINUE
C
C     Now check current DIFF against "top-NBST" array.  If current
C     DIFF is smaller than any element in the DIFFBST array, push it
C     in and shift the subsequent elements, moving the last one out.
C
      DO 120 K = 1,NBST

      IF (DIFF(1) .LT. DIFFBST(K,1)) THEN
      MOVE = K
C
C     OK, smaller DIFFS found -- do the element shift first.
C
      DO 100 L = NBST,MOVE+1,-1
      MTRPBST(L) = MTRPBST(L-1)
      IDBST(L) = IDBST(L-1)
      ITIMEBST(L) = ITIMEBST(L-1)
      IRECBST(L) = IRECBST(L-1)
      ASPDBST(L) = ASPDBST(L-1)
      TSPDBST(L) = TSPDBST(L-1)
      SSPDBST(L) = SSPDBST(L-1)
      TACLBST(L) = TACLBST(L-1)
      DO 100 II = 1,4
      DIFFBST(L,II) = DIFFBST(L-1,II)
  100 CONTINUE
C
C     And now the "push".
C
      MTRPBST(MOVE) = I
      IDBST(MOVE) = MTRIPID(I)
      ITIMEBST(MOVE) = ITIME(I)
      IRECBST(MOVE) = IREC(I)
      ASPDBST(MOVE) = SPD_MEAN(I)
      TSPDBST(MOVE) = SPD_MAX(I)
      SSPDBST(MOVE) = SPD_STD(I)
      TACLBST(MOVE) = ACL_MAX(I)
      DO 119 II = 1,4
      DIFFBST(MOVE,II) = DIFF(II)
  119 CONTINUE
      GO TO 350
      END IF

  120 CONTINUE
C
  350 CONTINUE

C     Print statistics for best individual M-trips.


      DO 400 M = 1, NBST
      WRITE (3, 370) M, MTRPBST(M), IDBST(M), IRECBST(M), ITIMEBST(M),
     +               ASPDBST(M), TSPDBST(M), SSPDBST(M), TACLBST(M),
     +               (DIFFBST(M,II), II = 1,4)
  370 FORMAT (I3, 1X, 'I', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 4(1X,F9.5))
  400 CONTINUE

      CLOSE (3)

C=======================================================================

  500 CONTINUE
C
      WRITE (*, 510) MTRIPS, NCYCLES
  510 FORMAT (/' Cycle generation complete:'/
     +        ' M-Trips Input .............  ', I5/
     +        ' Cycles Generated .........  ', I5/)
C
      END
```

```
      PROGRAM GETOUTSEED
C
C     GETOUT reads the "microtrips profiles" files and the target spd/
C     acl "Beyond FTP" surface file for the Baltimore & LA92 data
C     and selects a subset of linked microtrips which best match the
C     target speed/accel surface using the Best Incremental.
C
C     GETOUT is a modified version of the GETSTAB program.
C     The sample from which these additional microtrips are
C     selected consists of all Baltimore and LA92 microtrips, with
C     weighting factors which:
C
C        (1)  allow 50/50 selection probability of stabilized m-trips
C             between Baltimore and LA92; and
C        (2)  maintain initial weighting of start/stabilized m-trips
C             encountered in the Baltimore dataset.
C
C     NOTE:
C     This version does not use m-trips weightings discussed above, only
C     the embedded weightings in the input target surface.
C
C     I/O:   Unit 1 (input)  - Spd/acl target "surface" for sec-by-sec
C                              drive cycle data (Balt/LA92).
C            Unit 2 (input)  - Microtrip profile statistics.
C            Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    5/11/93
C
C-----------------------------------------------------------------------

      PARAMETER (MNUM=7283, NN=50)

      CHARACTER*3  CLOOP
      CHARACTER*14 MTRIPID(MNUM)
      CHARACTER*80 OUTFILE, OUTNAME

      INTEGER*2 MCNT(MNUM,-20:20,0:20), ITIME(MNUM), M, MO, I
      INTEGER*4 IREC(MNUM), TRIPLIST(NN)

      REAL SPD_MEAN(MNUM), SPD_STD(MNUM), ACL_MAX(MNUM), DIFFMIN(4),
     +     SPD_MAX(MNUM), CFREQ(-20:20,0:20), TRIPDIFF(NN,4), DIFF(4),
     +     CCNT(-20:20,0:20), PFREQ(-20:20,0:20), TCNT(-20:20,0:20)

      DATA NCYCLES / 1 /
      DATA DIFFMIN / 4*999. /

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD', RECL=1760, FORM='UNFORMATTED')

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for output filename and cycle control parameters.
C
      WRITE (*, 2)
    2 FORMAT (/1X, 'Enter output (selected trips & stats) filename:')
      READ (*, 3) OUTFILE
    3 FORMAT (A)
      WRITE (*, 4)
    4 FORMAT (/1X, 'Enter target cycle time (min):')
      READ (*, *) RTIME
      WRITE (*, 5)
    5 FORMAT (/1X, 'Enter "seed" m-trip number:')
      READ (*, *) NSEED
C
      TIMELIMIT = RTIME * 60
C
C     Read in target surface (unit 1).
C
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      PFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in microtrip profiles (unit 2).
C
```

```
   20 READ (2, END=50) I, MTRIPID(I), IREC(I), ITIME(I),
     +      SPD_MEAN(I), SPD_MAX(I), SPD_STD(I), ACL_MAX(I),
     +      ((MCNT(I,IACL,ISPD), ISPD=0,20), IACL=-20,20)
C   30 FORMAT (I5,1X,A14,1X,I7,1X,I5, 3(1X, F7.2),1X, F7.3,1X, 861(I4))
      IN2 = IN2 + 1
      GO TO 20
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Begin microtrip selection using Best Incremental
C     logic.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C=======================================================================
C     Top-level loop 500 iterates the selection process to generate
C     a number of cycles (NCYCLES).  Each cycle is written to a
C     separate output file on unit 3.
C=======================================================================
C
   50 MTRIPS = IN2
      DO 500 ILOOP = 1,NCYCLES
C
C     Initialize cycle sums and counter with "seed" values.
C
      NTRP = 1
      TRIPTIME = 0.0
      CSPD = 0.0
      DO 80 IACL = -20,20
      DO 80 ISPD = 0,20
      TCNT(IACL,ISPD) = MCNT(NSEED,IACL,ISPD)
   80 CONTINUE
C
      IPRD = INDEX (OUTFILE, '.')
      IEND = INDEX (OUTFILE, ' ') - 1
      WRITE (CLOOP, FMT='(I3.3)') ILOOP
      OUTNAME = OUTFILE(1:IPRD-1) // '-' // CLOOP // OUTFILE(IPRD:IEND)
      IOUT = INDEX (OUTNAME, ' ') - 1
C
C     Open "current" filename.
C
      OPEN (UNIT=3, FILE=OUTNAME, STATUS='NEW', CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE')
C
C     Write output file header.
C
      WRITE (3, 90) RTIME, OUTNAME(1:IOUT)
   90 FORMAT (5X, 'BEST INCREMENTAL Trip Selection Method', T54,
     +        'Target Time:  ', F5.2, ' min'//
     +        13X, 'Filename: ', A // 8X,
     +'MTRP                  INIT TIME  AVG SPD MAX SPD STD SPD MAX ACL'
     + ,'   TOTAL    CRUISE     ACCEL     DECEL'/2X, '#     ',
     +' NO     MTRIP ID       REC  (s)   (mph)   (mph)   (mph)  (mph/s)'
     + ,'  DIFFSUM   DIFFSUM   DIFFSUM   DIFFSUM'/)
C

C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)

      DO 115 II = 1,4
      DIFF(II) = 0.0
  115 CONTINUE
      DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(NSEED))
      DIFF(1) = DIFF(1) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      IF (IACL .EQ. 0) THEN
      DIFF(2) = DIFF(2) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      ELSE IF (IACL .GT. 0) THEN
      DIFF(3) = DIFF(3) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      ELSE IF (IACL .LT. 0) THEN
      DIFF(4) = DIFF(4) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      END IF
  120 CONTINUE

      M = NSEED
```

```
      TRIPLIST(NTRP) = M
      TRIPTIME = TRIPTIME + ITIME(M)
      DO 125 II = 1,4
      TRIPDIFF(NTRP,II) = DIFF(II)
  125 CONTINUE

C     Print statistics for current M-trip.

      WRITE (3, 170) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +              SPD_MAX(M), SPD_STD(M), ACL_MAX(M),
     +              (TRIPDIFF(NTRP,II), II = 1,4)
  170 FORMAT (I3, 1X, 'S', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 4(1X, F9.5))

C     Terminate main "trip adding" loop when the desired cycle time
C     TIMELIMIT is reached.

      IF (TRIPTIME .GT. TIMELIMIT) GO TO 400
C
C-----------------------------------------------------------------------
C     Big loop 301 adds "next best" microtrips till the total time
C     exceeds TIMELIMIT.
C-----------------------------------------------------------------------
C
  301 NTRP = NTRP + 1
C
C     Loop thru all microtrips in search of best incremental trip.
C
      DO 350 I = 1, MTRIPS
C
C     Skip microtrips which have already been selected.
C
      DO 310 J = 1,NTRP-1
      IF (I .EQ. TRIPLIST(J)) GO TO 350
  310 CONTINUE

C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)

      DO 315 II = 1,4
      DIFF(II) = 0.0
  315 CONTINUE
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(I,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(I))
      DIFF(1) = DIFF(1) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      IF (IACL .EQ. 0) THEN
      DIFF(2) = DIFF(2) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      ELSE IF (IACL .GT. 0) THEN
      DIFF(3) = DIFF(3) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      ELSE IF (IACL .LT. 0) THEN
      DIFF(4) = DIFF(4) + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      END IF
  320 CONTINUE

C     Test integrated difference DIFF of current incremental M-trip.
C     If smaller than "best" of earlier incremental M-trips, then
C     "mark" (M=I) and replace as new best difference.

      IF (DIFF(1) .LT. DIFFMIN(1)) THEN
      DO 330 II = 1,4
      DIFFMIN(II) = DIFF(II)
  330 CONTINUE
      M = I
      END IF

  350 CONTINUE

      TRIPLIST(NTRP) = M
      TRIPTIME = TRIPTIME + ITIME(M)
      DO 355 II = 1,4
      TRIPDIFF(NTRP,II) = DIFFMIN(II)
  355 CONTINUE

      DO 360 IACL = -20,20
      DO 360 ISPD = 0,20
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
```

```
  360 CONTINUE

C     Print statistics for current M-trip.

      WRITE (3, 370) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M),
     +               (TRIPDIFF(NTRP,II), II = 1,4)
  370 FORMAT (I3, 1X, 'I', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 4(1X, F9.5))

C     Terminate main "trip adding" loop when the desired cycle time
C     TIMELIMIT is reached.

      IF (TRIPTIME .GT. TIMELIMIT) GO TO 400

C     Loop back and find another M-trip.

      GO TO 301

C-------------------------------------------------------------------------
C     Current cycle completed.  Compute cycle summary statistics and
C     print totals.
C
  400 TOTTIME = TRIPTIME / 60
      DO 410 I = 1,NTRP
      M = TRIPLIST(I)
      CSPD = CSPD + SPD_MEAN(M) * ITIME(M) / TRIPTIME
  410 CONTINUE

      WRITE (3, 450) NTRP, TOTTIME, CSPD, DIFFMIN(1)
  450 FORMAT (111('_')//
     + 8X, 'Total Microtrips: ', I3/
     + 8X, ' Total Trip Time: ', F6.2, ' min'//
     + 8X, 'Mean Cycle Speed: ', F6.2,' mph'/
     + 8X, 'Total Difference: ', F9.5/)
C
C     Close "current" filename.
C
      CLOSE (3)

C=========================================================================

  500 CONTINUE
C
      WRITE (*, 510) MTRIPS, NCYCLES
  510 FORMAT (/' Cycle generation complete:'/
     +        ' M-Trips Input .............  ', I5/
     +        ' Cycles Generated .........  ', I5/)
C
      END
```

```
      PROGRAM GETREST
C
C     GETREST reads the "microtrips profiles" files and the target spd/
C     acl composite "surface" file for the Baltimore & LA92 data
C     and selects a subset of linked microtrips which best match the
C     target speed/accel surface using the Hybrid Method.
C
C     GETREST "seeds" the cycle with user-selected start and "outside
C     FTP" m-trips.
C
C     GETREST is a modified version of the earlier GETSTAB program.
C     Prior to execution of this code, another routine, GETSTART was run
C     to determine the "start" set of microtrips which best matched the
C     target start surface.  These pre-determined microtrips become the
C     initial "seed" in this program onto which additional microtrips
C     are added.  The sample from which these additional microtrips are
C     selected consists of all Baltimore and LA92 microtrips, with
C     weighting factors which:
C
C        (1)  allow 50/50 selection probability of stabilized m-trips
C             between Baltimore and LA92; and
C        (2)  maintain initial weighting of start/stabilized m-trips
C             encountered in the Baltimore dataset.
C
C     This version matches microtrip PATROL data with surface
C     COMPOSITE data (for LA92 data only).
C
C     NOTE:
C     This version does not use m-trips weightings discussed above, only
C     the embedded weightings in the composite target surface generated
C     by the COMPSURF program.
C
C     I/O:   Unit 1 (input)  - Spd/acl target composite "surface" for
C                              sec-by-sec drive cycle data (Balt/LA92).
C            Unit 2 (input)  - Microtrip profile statistics.
C            Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    4/23/93
C
C-----------------------------------------------------------------------

      PARAMETER (MNUM=7283)

      CHARACTER*3  CLOOP
      CHARACTER*14 MTRIPID(MNUM)
      CHARACTER*23 CTIME
      CHARACTER*80 OUTFILE, OUTNAME

      INTEGER*2 HH, MM, SS, DD, MCNT(MNUM,-20:20,0:20),
     +          ITIME(MNUM), M, MO, I, MOUT
      INTEGER*4 IREC(MNUM), TRIPLIST(1000)

      REAL SPD_MEAN(MNUM), SPD_STD(MNUM), ACL_MAX(MNUM),
     +     SPD_MAX(MNUM), CFREQ(-20:20,0:20), TRIPDIFF(1000),
     +     CCNT(-20:20,0:20), PFREQ(-20:20,0:20), TCNT(-20:20,0:20)

      DATA START_TIME /4.0/

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD', READONLY)
      OPEN (UNIT=2, STATUS='OLD',RECL=1760,FORM='UNFORMATTED', READONLY)

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for output filename and cycle control parameters.
C
      WRITE (*, 2)
    2 FORMAT (/1X, 'Enter output (selected trips & stats) filename:')
      READ (*, 3) OUTFILE
    3 FORMAT (A)
      WRITE (*, 4)
    4 FORMAT (/1X, 'Enter target cycle time, including start (min):')
      READ (*, *) RTIME
      WRITE (*, 5)
    5 FORMAT (/1X, 'Enter target random "base" cycle time (min):')
      READ (*, *) BTIME
```

```
      TIMERAN = BTIME * 60
      WRITE (*, 6)
    6 FORMAT (/1X, 'Enter "best" start trip number:')
      READ (*, *) MO
      WRITE (*, 7)
    7 FORMAT (/1X, 'Enter its number of M-trips:')
      READ (*, *) MSTART
      WRITE (*, 8)
    8 FORMAT (/1X, 'Enter "Outside FTP" trip number:')
      READ (*, *) MOUT
      WRITE (*, 9)
    9 FORMAT (/1X, 'Enter number of cycles to generate (1-999):')
      READ (*, *) NCYCLES

      TIMELIMIT = RTIME * 60
C
C     Read in total surface (unit 1).
C
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      PFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in microtrip profiles (unit 2).
C
   20 READ (2, END=50) I, MTRIPID(I), IREC(I), ITIME(I),
     +      SPD_MEAN(I), SPD_MAX(I), SPD_STD(I), ACL_MAX(I),
     +      ((MCNT(I,IACL,ISPD), ISPD=0,20), IACL=-20,20)
C   30 FORMAT (I5,1X,A14,1X,I7,1X,I5, 3(1X, F7.2),1X, F7.3,1X, 861(I4))
      IN2 = IN2 + 1
      GO TO 20
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Begin microtrip selection using Hybrid Method.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     This code section performs selection of microtrips using the
C     "Hybrid" approach (combination of Random + Best Incremental).
C     The Hybrid method adds an initial "base" of microtrips to the
C     start "seed" of microtrips using random selection.  Additional
C     microtrips are then added with the Best Incremental technique.
C
C     Step 1 - Generate pseudo-random seed value from current system
C     time.
C
C        SEED  =  DD x 1000000 +  SS x 16667  +  MM x 167  +  HH x 4.17
C
C     where DD = hundredths of seconds
C           SS = seconds
C           MM = minutes
C           HH = hours
C
C     Step 2 - Generate a series of randomly selected microtrips from
C     the population and add these to the start microtrip "seed" until
C     the total linked microtrip time exceeds a user-specified subset
C     TIMERAN of the target time limit TIMELIMIT.  Compute match of each
C     randomly added microtrip to the population surface.  This set of
C     microtrips forms the "base" sample.
C
C     Step 3 - Pick the "best" single microtrip of those remaining.
C     "Best" is defined as the minimum integrated difference between the
C     overall surface and the base + incremental microtrip(s).
C
C     Step 4 - Iteratively test the remaining microtrips and add the one
C     which coupled with the previously selected trip(s), best matches
C     the overall surface.  This incremental approach is repeated until
C     the total linked microtrip time exceeds target time TIMELIMIT.
C
C     I/O:   Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    4/23/93
C
C-----------------------------------------------------------------------

C     Get system time.  Call to LIB$DATE_TIME returns 23-byte character
C     string which contains hour, minute, second and decimal fraction.
```

```
   50 ISTAT = LIB$DATE_TIME (CTIME)
      READ (CTIME, FMT='(11X,4(1X,I2))') HH, MM, SS, DD

C     Create seed value for random series.  Seed ranges from 0 to 10**8.
C     Make seed odd per VAX-FORTRAN recommendations (FORTRAN Language
C     Reference Manual V 5.0, pg. D-49).

      ISEED = DD * 1000000 + SS * 16666.67 + MM * 166.67 + HH * 4.1667
      IR = MOD (ISEED, 2)
      IF (IR .EQ. 0) ISEED = ISEED + 1

C=======================================================================
C     Top-level loop 500 iterates the selection process to generate
C     a number of random cycles (NCYCLES).  Each cycle is written to a
C     separate output file on unit 3.
C=======================================================================
C
      MTRIPS = IN2
      DO 500 ILOOP = 1,NCYCLES
C
C     Initialize cycle sums and counter.
C
      NTRP = 0
      TRIPTIME = 0.0
      STIME = 0.0
      CSPD = 0.0
      DO 80 IACL = -20,20
      DO 80 ISPD = 0,20
      TCNT(IACL,ISPD) = 0.0
   80 CONTINUE
C
      IPRD = INDEX (OUTFILE, '.')
      IEND = INDEX (OUTFILE, ' ') - 1
      WRITE (CLOOP, FMT='(I3.3)') ILOOP
      OUTNAME = OUTFILE(1:IPRD-1) // '-' // CLOOP // OUTFILE(IPRD:IEND)
      IOUT = INDEX (OUTNAME, ' ') - 1
C
C     Open "current" filename.
C
      OPEN (UNIT=3, FILE=OUTNAME, STATUS='NEW', CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE')
C
C     Write output file header.
C
      RTIME = TIMELIMIT / 60.0
      BTIME = TIMERAN / 60.0
      WRITE (3, 90) RTIME, ISEED, BTIME, OUTNAME(1:IOUT), START_TIME
   90 FORMAT (5X, 'HYBRID Trip Selection Method', T54,
     +        'Target Time:  ', F5.2, ' min'/
     +        13X, '1st Seed: ', I9, T54,
     +        '  Base Time:  ', F5.2, ' min'/
     +        13X, 'Filename: ', A, T54,
     +        ' Start Time:  ', F5.2, ' min'//8X,
     +'MTRP                  INIT TIME  AVG SPD MAX SPD STD SPD MAX ACL'
     + /2X, '#     ',
     +' NO     MTRIP ID       REC  (s)   (mph)   (mph)   (mph) (mph/s)'
     + ,'  DIFFSUM'/)

C     ++++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++++
C     +++++   "INITIAL SEED" SECTION   +++++
C     ++++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++++

C-----------------------------------------------------------------------
C     "Start" and "Outside FTP" microtrip(s) seeding section.  Use input
C     "best" start and outside FTP microtrips and use the MO,  MSTART
C     and MOUT "pointers" to extract them.
C-----------------------------------------------------------------------
C
C     START SECTION.
C
C     Big loop 150 adds sequentially selected M-trips till all start
C     M-trips read and processed (NTRP = MSTART).
C
      DO 150 M = MO, MO+MSTART-1
C
C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Then, compute integrated
C     difference between the population surface PFREQ and the best
```

```
C     incremental surface CFREQ.  (Use abs value of the difference.)
C
      DIFF = 0.0
      DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(M))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
  120 CONTINUE
C
      NTRP = NTRP + 1
      TRIPLIST(NTRP) = M
      TRIPDIFF(NTRP) = DIFF
      TRIPTIME = TRIPTIME + ITIME(M)

C     Print statistics for current M-trip.

      WRITE (3, 140) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  140 FORMAT (I3, 1X, 'S', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)
C
  150 CONTINUE
C
      STIME = TRIPTIME / 60
      NST = NTRP
C
C     OUTSIDE FTP SECTION.
C
       M = MOUT
C
C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Then, compute integrated
C     difference between the population surface PFREQ and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)
C
      DIFF = 0.0
      DO 160 IACL = -20,20
      DO 160 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(M))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
  160 CONTINUE
C
      NTRP = NTRP + 1
      TRIPLIST(NTRP) = M
      TRIPDIFF(NTRP) = DIFF
      TRIPTIME = TRIPTIME + ITIME(M)

C     Print statistics for current M-trip.

      WRITE (3, 170) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  170 FORMAT (I3, 1X, 'O', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)
C
      IF (TRIPTIME .GE. TIMERAN) GO TO 300

C-----------------------------------------------------------------------
C     Big loop 200 adds randomly selected microtrips till the total time
C     exceeds TIMERAN.
C-----------------------------------------------------------------------

  200 NTRP = NTRP + 1
C
C     Randomly select a microtrip in the range 1 to MTRIPS.
C
  205 XRAN = RAN (ISEED)
      IF (XRAN .LT. 1.0) THEN
      M = INT (MTRIPS * XRAN) + 1
      ELSE
      M = MTRIPS
      END IF
C
C     Once a M-trip is selected, it's not allowed to be picked again.
C
      DO 210 II = 1, NTRP
      IF (TRIPLIST(II) .EQ. M) GO TO 205
```

```
  210 CONTINUE
C
C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface PFREQ and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)
C
      DIFF = 0.0
      DO 220 IACL = -20,20
      DO 220 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(M))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
  220 CONTINUE
C
      TRIPLIST(NTRP) = M
      TRIPTIME = TRIPTIME + ITIME(M)
      TRIPDIFF(NTRP) = DIFF

C     Print statistics for current M-trip.

      WRITE (3, 270) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  270 FORMAT (I3, 1X, 'R', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)

C     Terminate main "trip adding" loop when the desired cycle time
C     TIMELIMIT is reached.

      IF (TRIPTIME .GE. TIMERAN) GO TO 300

C     Loop back and find another M-trip.

      GO TO 200

C=======================================================================
C
C     "Base" of random M-trips complete.  Add subsequent M-trips using
C      Best Incremental logic.
C
C     ++++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++++
C     +++++  BEST INCREMENTAL SECTION  +++++
C     ++++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++++
C
C-----------------------------------------------------------------------
C     Big loop 301 adds "next best" microtrips till the total time
C     exceeds TIMELIMIT.
C-----------------------------------------------------------------------
  300 BASETIME = TRIPTIME / 60
      IF (TRIPTIME .GE. TIMELIMIT) GO TO 400
  301 NTRP = NTRP + 1
      DIFFMIN = 999.
C
C     Loop thru all microtrips in search of best incremental trip.
C
      DO 350 I = 1, MTRIPS
C
C     Skip microtrips which have already been selected.
C
      DO 310 J = 1,NTRP-1
      IF (I .EQ. TRIPLIST(J)) GO TO 350
  310 CONTINUE

C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)

      DIFF = 0.0
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(I,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(I))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
  320 CONTINUE

C     Test integrated difference DIFF of current incremental M-trip.
```

```
C     If smaller than "best" of earlier incremental M-trips, then
C     "mark" (M=I) and replace as new best difference.

      IF (DIFF .LT. DIFFMIN) THEN
      DIFFMIN = DIFF
      M = I
      END IF

  350 CONTINUE

      TRIPLIST(NTRP) = M
      TRIPTIME = TRIPTIME + ITIME(M)
      TRIPDIFF(NTRP) = DIFFMIN

      DO 360 IACL = -20,20
      DO 360 ISPD = 0,20
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
  360 CONTINUE

C     Print statistics for current M-trip.

C     Print statistics for current M-trip.

      WRITE (3, 370) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  370 FORMAT (I3, 1X, 'I', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)

C     Terminate main "trip adding" loop when the desired cycle time
C     TIMELIMIT is reached.

      IF (TRIPTIME .GT. TIMELIMIT) GO TO 400

C     Loop back and find another M-trip.

      GO TO 301

C-----------------------------------------------------------------------
C     Current cycle completed.  Compute cycle summary statistics and
C     print totals.
C
  400 TOTTIME = TRIPTIME / 60
      CSPD = 0.0
      DO 410 I = 1,NTRP
      M = TRIPLIST(I)
      CSPD = CSPD + SPD_MEAN(M) * ITIME(M) / TRIPTIME
  410 CONTINUE

      WRITE (3, 450) NTRP, NST, TOTTIME, STIME, BASETIME, CSPD, DIFFMIN
  450 FORMAT (81('_')//
     + 8X, 'Total Microtrips: ', I3, T50, 'Start Microtrips: ', I3/
     + 8X, ' Total Trip Time: ', F6.2, ' min', T50,' Start Trip Time: ',
     +                                          F6.2, ' min'/
     + 8X, ' Total Base Time: ', F6.2, ' min'/
     + 8X, 'Mean Cycle Speed: ', F6.2, ' mph'/
     + 8X, 'Total Difference: ', F9.5/)
C
C     Close "current" filename.
C
      CLOSE (3)

C=======================================================================

  500 CONTINUE
C
      WRITE (*, 510) MTRIPS, NCYCLES
  510 FORMAT (/' Cycle generation complete:'/
     +        ' M-Trips Input .....  ', I5/
     +        ' Cycles Generated .....  ', I5/)
C
      END
```

```
      PROGRAM GETRMNDS
C
C     GETRMNDS reads the "microtrips profiles" files and the target spd/
C     acl composite "surface" file for the Baltimore & LA92 data
C     and selects a subset of linked microtrips which best match the
C     target speed/accel surface using the Hybrid Method.
C
C     GETRNMDS is a modified version of the GETRMND program which uses
C     the CFRQR.DAT SAFD file as its target surface.
C
C          CFRQR  =  CFRQ Stabilized - LBag4-4 x WtFac
C                                      ---
C                         where WtFac = 0.2067
C
C     I/O:   Unit 1 (input)  - Spd/acl target composite "surface" for
C                              sec-by-sec drive cycle data (Balt/LA92).
C            Unit 2 (input)  - Microtrip profile statistics.
C            Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    7/14/93
C
C-----------------------------------------------------------------------

      PARAMETER (MNUM=7283)

      CHARACTER*3  CLOOP
      CHARACTER*14 MTRIPID(MNUM)
      CHARACTER*23 CTIME
      CHARACTER*80 OUTFILE, OUTNAME

      INTEGER*2 HH, MM, SS, DD, MCNT(MNUM,-20:20,0:20),
     +          ITIME(MNUM), M, MO, I
      INTEGER*4 IREC(MNUM), TRIPLIST(1000)

      REAL SPD_MEAN(MNUM), SPD_STD(MNUM), ACL_MAX(MNUM),
     +     SPD_MAX(MNUM), CFREQ(-20:20,0:20), TRIPDIFF(1000),
     +     CCNT(-20:20,0:20), PFREQ(-20:20,0:20), TCNT(-20:20,0:20)

      DATA START_TIME /4.0/

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD', RECL=1760, FORM='UNFORMATTED')

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for output filename and cycle control parameters.
C
      WRITE (*, 2)
    2 FORMAT (/1X, 'Enter output (selected trips & stats) filename:')
      READ (*, 3) OUTFILE
    3 FORMAT (A)
      WRITE (*, 4)
    4 FORMAT (/1X, 'Enter target cycle time, including start (min):')
      READ (*, *) RTIME
      WRITE (*, 5)
    5 FORMAT (/1X, 'Enter target random "base" cycle time (min):')
      READ (*, *) BTIME
      TIMERAN = BTIME * 60
      WRITE (*, 6)
    6 FORMAT (/1X, 'Enter "best" start trip number:')
      READ (*, *) MO
      WRITE (*, 7)
    7 FORMAT (/1X, 'Enter its number of M-trips:')
      READ (*, *) MSTART

      WRITE (*, 8)
    8 FORMAT (/1X, 'Enter number of cycles to generate (1-999):')
      READ (*, *) NCYCLES

      TIMELIMIT = RTIME * 60
C
C     Read in total surface (unit 1).
C
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      PFREQ(IACL,ISPD) = PCT / 100
```

```
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in microtrip profiles (unit 2).
C
   20 READ (2, END=50) I, MTRIPID(I), IREC(I), ITIME(I),
     +      SPD_MEAN(I), SPD_MAX(I), SPD_STD(I), ACL_MAX(I),
     +      ((MCNT(I,IACL,ISPD), ISPD=0,20), IACL=-20,20)
C   30 FORMAT (I5,1X,A14,1X,I7,1X,I5, 3(1X, F7.2),1X, F7.3,1X, 861(I4))
      IN2 = IN2 + 1
      GO TO 20
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Begin microtrip selection using Hybrid Method.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     This code section performs selection of microtrips using the
C     "Hybrid" approach (combination of Random + Best Incremental).
C     The Hybrid method adds an initial "base" of microtrips to the
C     start "seed" of microtrips using random selection.  Additional
C     microtrips are then added with the Best Incremental technique.
C
C     Step 1 - Generate pseudo-random seed value from current system
C     time.
C
C        SEED  =  DD x 1000000 +  SS x 16667  +  MM x 167  +  HH x 4.17
C
C     where DD = hundredths of seconds
C           SS = seconds
C           MM = minutes
C           HH = hours
C
C     Step 2 - Generate a series of randomly selected microtrips from
C     the population and add these to the start microtrip "seed" until
C     the total linked microtrip time exceeds a user-specified subset
C     TIMERAN of the target time limit TIMELIMIT.  Compute match of each
C     randomly added microtrip to the population surface.  This set of
C     microtrips forms the "base" sample.
C
C     Step 3 - Pick the "best" single microtrip of those remaining.
C     "Best" is defined as the minimum integrated difference between the
C     overall surface and the base + incremental microtrip(s).
C
C     Step 4 - Iteratively test the remaining microtrips and add the one
C     which coupled with the previously selected trip(s), best matches
C     the overall surface.  This incremental approach is repeated until
C     the total linked microtrip time exceeds target time TIMELIMIT.
C
C     I/O:   Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    4/23/93
C
C-----------------------------------------------------------------------
C=======================================================================
C     Top-level loop 500 iterates the selection process to generate
C     a number of random cycles (NCYCLES).  Each cycle is written to a
C     separate output file on unit 3.
C=======================================================================
C
   50 MTRIPS = IN2
      DO 500 ILOOP = 1,NCYCLES
C
C     Initialize cycle sums and counter.
C
      NTRP = 0
      TRIPTIME = 0.0
      STIME = 0.0
      CSPD = 0.0
      DO 80 IACL = -20,20
      DO 80 ISPD = 0,20
      TCNT(IACL,ISPD) = 0.0
   80 CONTINUE
C
      IPRD = INDEX (OUTFILE, '.')
      IEND = INDEX (OUTFILE, ' ') - 1
      WRITE (CLOOP, FMT='(I3.3)') ILOOP
      OUTNAME = OUTFILE(1:IPRD-1) // '-' // CLOOP // OUTFILE(IPRD:IEND)
```

```
      PROGRAM GETSTAB
C
C     GETSTAB reads the "microtrips profiles" files and the target spd/
C     acl composite "surface" file for the Baltimore & LA92 data
C     and selects a subset of linked microtrips which best match the
C     target speed/accel surface using the Hybrid Method.
C
C     GETSTAB is a modified version of earlier GETRIPS/GETHBRD programs.
C     Prior to execution of this code, another routine, GETSTART was run
C     to determine the "start" set of microtrips which best matched the
C     target start surface.  These pre-determined microtrips become the
C     initial "seed" in this program onto which additional microtrips
C     are added.  The sample from which these additional microtrips are
C     selected consists of all Baltimore and LA92 microtrips, with
C     weighting factors which:
C
C        (1)  allow 50/50 selection probability of stabilized m-trips
C             between Baltimore and LA92; and
C        (2)  maintain initial weighting of start/stabilized m-trips
C             encountered in the Baltimore dataset.
C
C     This version matches microtrip PATROL data with surface
C     COMPOSITE data (for LA92 data only).
C
C     NOTE:
C     This version does not use m-trips weightings discussed above, only
C     the embedded weightings in the composite target surface generated
C     by the COMPSURF program.
C
C     I/O:   Unit 1 (input)  - Spd/acl target composite "surface" for
C                              sec-by-sec drive cycle data (Balt/LA92).
C            Unit 2 (input)  - Microtrip profile statistics.
C            Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    4/23/93
C
C-----------------------------------------------------------------------

      PARAMETER (MNUM=7283)

      CHARACTER*3  CLOOP
      CHARACTER*14 MTRIPID(MNUM)
      CHARACTER*23 CTIME
      CHARACTER*80 OUTFILE, OUTNAME

      INTEGER*2 HH, MM, SS, DD, MCNT(MNUM,-20:20,0:20),
     +          ITIME(MNUM), M, M0, I
      INTEGER*4 IREC(MNUM), TRIPLIST(1000)

      REAL SPD_MEAN(MNUM), SPD_STD(MNUM), ACL_MAX(MNUM),
     +     SPD_MAX(MNUM), CFREQ(-20:20,0:20), TRIPDIFF(1000),
     +     CCNT(-20:20,0:20), PFREQ(-20:20,0:20), TCNT(-20:20,0:20)

      DATA START_TIME /4.0/

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD', RECL=1760, FORM='UNFORMATTED')

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for output filename and cycle control parameters.
C
      WRITE (*, 2)
    2 FORMAT (/1X, 'Enter output (selected trips & stats) filename:')
      READ (*, 3) OUTFILE
    3 FORMAT (A)
      WRITE (*, 4)
    4 FORMAT (/1X, 'Enter target cycle time, including start (min):')
      READ (*, *) RTIME
      WRITE (*, 5)
    5 FORMAT (/1X, 'Enter target random "base" cycle time (min):')
      READ (*, *) BTIME
      TIMERAN = BTIME * 60
      WRITE (*, 6)
    6 FORMAT (/1X, 'Enter "best" start trip number:')
```

```
      READ (*, *) MO
      WRITE (*, 7)
    7 FORMAT (/1X, 'Enter its number of M-trips:')
      READ (*, *) MSTART

      WRITE (*, 8)
    8 FORMAT (/1X, 'Enter number of cycles to generate (1-999):')
      READ (*, *) NCYCLES

      TIMELIMIT = RTIME * 60
C
C     Read in total surface (unit 1).
C
   10 READ (1, *, END=20) IACL, ISPD, NFREQ, PCT
      PFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 10
C
C     Read in microtrip profiles (unit 2).
C
   20 READ (2, END=50) I, MTRIPID(I), IREC(I), ITIME(I),
     +      SPD_MEAN(I), SPD_MAX(I), SPD_STD(I), ACL_MAX(I),
     +      ((MCNT(I,IACL,ISPD), ISPD=0,20), IACL=-20,20)
C   30 FORMAT (I5,1X,A14,1X,I7,1X,I5, 3(1X, F7.2),1X, F7.3,1X, 861(I4))
      IN2 = IN2 + 1
      GO TO 20
C
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     Input completed.  Begin microtrip selection using Hybrid Method.
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C     This code section performs selection of microtrips using the
C     "Hybrid" approach (combination of Random + Best Incremental).
C     The Hybrid method adds an initial "base" of microtrips to the
C     start "seed" of microtrips using random selection.  Additional
C     microtrips are then added with the Best Incremental technique.
C
C     Step 1 - Generate pseudo-random seed value from current system
C     time.
C
C        SEED  =  DD x 1000000 +  SS x 16667  +  MM x 167  +  HH x 4.17
C
C     where DD = hundredths of seconds
C           SS = seconds
C           MM = minutes
C           HH = hours
C
C     Step 2 - Generate a series of randomly selected microtrips from
C     the population and add these to the start microtrip "seed" until
C     the total linked microtrip time exceeds a user-specified subset
C     TIMERAN of the target time limit TIMELIMIT.  Compute match of each
C     randomly added microtrip to the population surface.  This set of
C     microtrips forms the "base" sample.
C
C     Step 3 - Pick the "best" single microtrip of those remaining.
C     "Best" is defined as the minimum integrated difference between the
C     overall surface and the base + incremental microtrip(s).
C
C     Step 4 - Iteratively test the remaining microtrips and add the one
C     which coupled with the previously selected trip(s), best matches
C     the overall surface.  This incremental approach is repeated until
C     the total linked microtrip time exceeds target time TIMELIMIT.
C
C     I/O:   Unit 3 (output) - Selected microtrips and "goodness" stats.
C
C     Author:  Tom Carlson
C     Date:    4/23/93
C
C-----------------------------------------------------------------------
C=======================================================================
C     Top-level loop 500 iterates the selection process to generate
C     a number of random cycles (NCYCLES).  Each cycle is written to a
C     separate output file on unit 3.
C=======================================================================
C
   50 MTRIPS = IN2
      DO 500 ILOOP = 1,NCYCLES
C
```

```
C     Initialize cycle sums and counter.
C
      NTRP = 0
      TRIPTIME = 0.0
      STIME = 0.0
      CSPD = 0.0
      DO 80 IACL = -20,20
      DO 80 ISPD = 0,20
      TCNT(IACL,ISPD) = 0.0
   80 CONTINUE
C
      IPRD = INDEX (OUTFILE, '.')
      IEND = INDEX (OUTFILE, ' ') - 1
      WRITE (CLOOP, FMT='(I3.3)') ILOOP
      OUTNAME = OUTFILE(1:IPRD-1) // '-' // CLOOP // OUTFILE(IPRD:IEND)
      IOUT = INDEX (OUTNAME, ' ') - 1

C     Get system time.  Call to LIB$DATE_TIME returns 23-byte character
C     string which contains hour, minute, second and decimal fraction.

      ISTAT = LIB$DATE_TIME (CTIME)
      READ (CTIME, FMT='(11X,4(1X,I2))') HH, MM, SS, DD

C     Create seed value for random series.  Seed ranges from 0 to 10**8.
C     Make seed odd per VAX-FORTRAN recommendations (FORTRAN Language
C     Reference Manual V 5.0, pg. D-49).

      ISEED = DD * 1000000 + SS * 16666.67 + MM * 166.67 + HH * 4.1667
      IR = MOD (ISEED, 2)
      IF (IR .EQ. 0) ISEED = ISEED + 1
C
C     Open "current" filename.
C
      OPEN (UNIT=3, FILE=OUTNAME, STATUS='NEW', CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE')
C
C     Write output file header.
C
      RTIME = TIMELIMIT / 60.0
      BTIME = TIMERAN / 60.0
      WRITE (3, 90) RTIME, ISEED, BTIME, OUTNAME(1:IOUT), START_TIME
   90 FORMAT (5X, 'HYBRID Trip Selection Method', T54,
     +        'Target Time:  ', F5.2, ' min'/
     +        13X, '1st Seed: ', I9, T54,
     +        ' _Base Time:  ', F5.2, ' min'/
     +        13X, 'Filename: ', A, T54,
     +        ' Start Time:  ', F5.2, ' min'//8X,
     +'NTRP                  INIT TIME  AVG SPD MAX SPD STD SPD MAX ACL'
     + /2X, '#     ',
     +' NO     MTRIP ID       REC  (s)   (mph)   (mph)   (mph)  (mph/s)'
     + ,'  DIFFSUM'/)

C     ++++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++++
C     ++++++   "START SEED" SECTION   ++++++
C     ++++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++++

C-------------------------------------------------------------------------
C     "Start" microtrip(s) seeding section.  Use input "best" start
C      microtrips and use the M0 & MSTART "pointers" to extract them.
C-------------------------------------------------------------------------
C
C     Big loop 150 adds sequentially selected M-trips till all start
C     M-trips read and processed (NTRP = MSTART).
C
      DO 150 M = M0, M0+MSTART-1
C
C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Then, compute integrated
C     difference between the population surface PFREQ and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)
C
      DIFF = 0.0
      DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(M))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
```

```
  120 CONTINUE
C
      NTRP = NTRP + 1
      TRIPLIST(NTRP) = M
      TRIPDIFF(NTRP) = DIFF
      TRIPTIME = TRIPTIME + ITIME(M)

C     Print statistics for current M-trip.

      WRITE (3, 140) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  140 FORMAT (I3, 1X, 'S', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)
C
  150 CONTINUE
C
      STIME = TRIPTIME / 60
      NST = NTRP
      IF (TRIPTIME .GE. TIMERAN) GO TO 300

C-----------------------------------------------------------------------
C     Big loop 200 adds randomly selected microtrips till the total time
C     exceeds TIMERAN.
C-----------------------------------------------------------------------

  200 NTRP = NTRP + 1
C
C     Randomly select a microtrip in the range 1 to MTRIPS.
C
  205 XRAN = RAN (ISEED)
      IF (XRAN .LT. 1.0) THEN
      M = INT (MTRIPS * XRAN) + 1
      ELSE
      M = MTRIPS
      END IF
C
C     Once a M-trip is selected, it's not allowed to be picked again.
C
      DO 210 II = 1, NTRP
      IF (TRIPLIST(II) .EQ. M) GO TO 205
  210 CONTINUE
C
C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface PFREQ and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)
C
      DIFF = 0.0
      DO 220 IACL = -20,20
      DO 220 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(M))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
  220 CONTINUE
C
      TRIPLIST(NTRP) = M
      TRIPTIME = TRIPTIME + ITIME(M)
      TRIPDIFF(NTRP) = DIFF

C     Print statistics for current M-trip.

      WRITE (3, 270) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  270 FORMAT (I3, 1X, 'R', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)

C     Terminate main "trip adding" loop when the desired cycle time
C     TIMELIMIT is reached.

      IF (TRIPTIME .GE. TIMERAN) GO TO 300

C     Loop back and find another M-trip.

      GO TO 200

C=======================================================================
C
C     "Base" of random M-trips complete.  Add subsequent M-trips using
C      Best Incremental logic.
```

```
C
C     ++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++
C     +++++  BEST INCREMENTAL SECTION  +++++
C     ++++++++++++++++++++++++++++++++++++
C     ++++++++++++++++++++++++++++++++++++
C
C-----------------------------------------------------------------------
C     Big loop 301 adds "next best" microtrips till the total time
C     exceeds TIMELIMIT.
C-----------------------------------------------------------------------
  300 BASETIME = TRIPTIME / 60
      IF (TRIPTIME .GE. TIMELIMIT) GO TO 400
  301 NTRP = NTRP + 1
      DIFFMIN = 999.
C
C     Loop thru all microtrips in search of best incremental trip.
C
      DO 350 I = 1, MTRIPS
C
C     Skip microtrips which have already been selected.
C
      DO 310 J = 1,NTRP-1
      IF (I .EQ. TRIPLIST(J)) GO TO 350
  310 CONTINUE

C     Add acl/spd bin counts for current microtrip to "current" count
C     array CCNT and express as frequency.  Compute integrated
C     difference between the population surface and the best
C     incremental surface CFREQ.  (Use abs value of the difference.)

      DIFF = 0.0
      DO 320 IACL = -20,20
      DO 320 ISPD = 0,20
      CCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(I,IACL,ISPD)
      CFREQ(IACL,ISPD) = CCNT(IACL,ISPD) / (TRIPTIME + ITIME(I))
      DIFF = DIFF + ABS (CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD))
  320 CONTINUE

C     Test integrated difference DIFF of current incremental M-trip.
C     If smaller than "best" of earlier incremental M-trips, then
C     "mark" (M=I) and replace as new best difference.

      IF (DIFF .LT. DIFFMIN) THEN
      DIFFMIN = DIFF
      M = I
      END IF

  350 CONTINUE

      TRIPLIST(NTRP) = M
      TRIPTIME = TRIPTIME + ITIME(M)
      TRIPDIFF(NTRP) = DIFFMIN

      DO 360 IACL = -20,20
      DO 360 ISPD = 0,20
      TCNT(IACL,ISPD) = TCNT(IACL,ISPD) + MCNT(M,IACL,ISPD)
  360 CONTINUE

C     Print statistics for current M-trip.

C     Print statistics for current M-trip.

      WRITE (3, 370) NTRP, M, MTRIPID(M), IREC(M), ITIME(M),SPD_MEAN(M),
     +               SPD_MAX(M), SPD_STD(M), ACL_MAX(M), TRIPDIFF(NTRP)
  370 FORMAT (I3, 1X, 'I', 1X, I5, 1X, A14, 1X, I7, 1X, I4,
     +        3(1X, F7.2), 1X, F7.3, 1X, F9.5)

C     Terminate main "trip adding" loop when the desired cycle time
C     TIMELIMIT is reached.

      IF (TRIPTIME .GT. TIMELIMIT) GO TO 400

C     Loop back and find another M-trip.

      GO TO 301

C-----------------------------------------------------------------------
C     Current cycle completed.  Compute cycle summary statistics and
C     print totals.
```

```
C
  400 TOTTIME = TRIPTIME / 60
      CSPD = 0.0
      DO 410 I = 1,NTRP
      M = TRIPLIST(I)
      CSPD = CSPD + SPD_MEAN(M) * ITIME(M) / TRIPTIME
  410 CONTINUE

      WRITE (3, 450) NTRP, NST, TOTTIME, STIME, BASETIME, CSPD, DIFFMIN
  450 FORMAT (81('_')//
     + 8X, 'Total Microtrips: ', I3, T50, 'Start Microtrips: ', I3/
     + 8X, ' Total Trip Time: ', F6.2, ' min', T50,' Start Trip Time: ',
     +                                        F6.2, ' min'/
     + 8X, ' Total Base Time: ', F6.2, ' min'/
     + 8X, 'Mean Cycle Speed: ', F6.2, ' mph'/
     + 8X, 'Total Difference: ', F9.5/)
C
C     Close "current" filename.
C
      CLOSE (3)

C=====================================================================

  500 CONTINUE
C
      WRITE (*, 510) MTRIPS, NCYCLES
  510 FORMAT (/' Cycle generation complete:'/
     +        ' M-Trips Input .....  ', I5/
     +        ' Cycles Generated .....  ', I5/)
C
      END
```

```
      PROGRAM GETSTART
C
C     GETRIPS reads the "microtrips profiles" files and the target spd/
C     acl bins "surface" file for the Baltimore cold and hot trips files
C     and selects the trip start (and its linked microtrips) which best
C     match the target surface.
C
C     I/O:   Unit 1 (input)  - Spd/acl target "surface" for first 4 min
C                              of sec-by-sec Balt driving; includes
C                              % and # bin frequencies.
C            Unit 2 (input)  - Sec-by-sec drive start data.
C            Unit 3 (output) - Listing of "goodness" by trip.
C
C     Author:  Tom Carlson
C     Date:    4/21/93
C
C     4/23/92 - Modified code to reject start trips with either initial
C               idle < 15 seconds or with a trailing microtrip from
C               t=240 sec which continued beyond t=300 sec.  (TRC)
C
C-----------------------------------------------------------------------

      CHARACTER*12 TRIPID, LTRIPID
      CHARACTER*14 MTRIPID

      INTEGER START_TIME, TRIP_TIME, CRUS

      REAL FREQ(-20:20,0:20), TFREQ(-20:20,0:20)

      DATA START_TIME /240/, MOST_TIME /300/, DIFFMIN /9999./,
     +     MIN_IDLE /15/

C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='NEW', CARRIAGECONTROL='LIST')
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Read in target start surface (unit 1).
C
  100 READ (1, *, END=150) IACL, ISPD, NCNT, PCT
      TFREQ(IACL,ISPD) = PCT / 100
      IN1 = IN1 + 1
      GO TO 100

C     Write output file header.

  150 WRITE (3, 160)
  160 FORMAT ('Trip #   Trip ID        Start Diff   Start Time (sec)',
     +        '   Initial Idle (sec)'/
     +        '------   -------        ----------   ----------------',
     +        '   ------------------')

C=======================================================================
C     Main processing loop (200).
C=======================================================================
C
C     Read in sec-by-sec start data file (EPABLT1.DAT).
C
  200 READ (2, 201, END=500) TRIPID, ISEC, SPD, ACL, IS, IA
  201 FORMAT (A12, I6, 2F6.2, 2I3)
      IN2 = IN2 + 1
C
C     First thing to check after a record read is if it is the beginning
C     of new trip (ISEC = 0).  If so, go to "Compute Trip Fit" section.
C
      IF (ISEC .EQ. 0) GO TO 250
C
C     Skip further summing of this record if the running trip time
C     exceeds the defined start time.  However, still increment the
C     nominal time until terminating microtrip detected (MFLAG = 1).
C-----------------------------------------------------------------------
      IF (ISEC .GT. START_TIME .AND. MFLAG .EQ. 0) THEN
C
C     Detect when microtrip has ended (MFLAG = 1).
C
```

```
      NOM_TIME = NOM_TIME + 1
C
      IF (SPD .GT. 0.0) THEN
      CRUS = 1
      IFLAG = 1
      ELSE
      CRUS = 0
      END IF

      IF (CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND. IFLAG .EQ. 1) IFLAG = 1

      IF (TRIPID .NE. LTRIPID .OR. CRUS .EQ. 0 .AND. LCRUS .EQ. 0 .AND.
     +    IFLAG .EQ. 1) THEN
      IFLAG = 0
      MFLAG = 1
      GO TO 220
      END IF
C
      LCRUS = CRUS
  220 LTRIPID = TRIPID
C
      END IF
C-----------------------------------------------------------------------
C
C     Increment sum & counter for current trip, scan for "nominal"
C     termining microtrip and process next record.
C
      IF (TRIP_TIME .LE. START_TIME) THEN
      FREQ(IA,IS) = FREQ(IA,IS) + 1
      TRIP_TIME = TRIP_TIME + 1
      END IF
C
C     Detect time at which vehicle motion begins.  Trips will be
C     later rejected from consideration if initial idle < 15 sec.
C
      IF (IDLE .EQ. 0 .AND. SPD .GT. 0.0) THEN
      IDLE = 1
      IDLE_TIME = ISEC - 1
      END IF

      GO TO 200
C
C     === Compute Trip Fit ===
C
C     Input currently positioned at beginning of a trip.  Compute the
C     spd/acl frequency distribution for "last" trip and determine its
C     "goodness-of-fit" value DIFFSUM.
C
  250 IF (IN2 .EQ. 1) GO TO 305
      DIFFSUM = 0.0
      DO 280 IACL = -20,20
      DO 280 ISPD = 0,20
      FREQ(IACL,ISPD) = FREQ(IACL,ISPD) / TRIP_TIME
      DIFFSUM = DIFFSUM + ABS (FREQ(IACL,ISPD) - TFREQ(IACL,ISPD))
  280 CONTINUE

      TRIP_TIME = TRIP_TIME + NOM_TIME - 1

C     Output results to unit 3.
C
C     Modification (4/23/93 TRC) - Give up on trips whose initial idle
C     time is less than 15 seconds or whose "nominal" start time
C     exceeds MOST_TIME.
C
      IF (IDLE_TIME.LT.MIN_IDLE .OR. TRIP_TIME.GT.MOST_TIME) GO TO 305

      WRITE (3, 300) NTRIP, LTRIPID, DIFFSUM, TRIP_TIME, IDLE_TIME
  300 FORMAT (I5, 4X, A12, 3X, F9.6, 10X, I4, 17X, I4)

C     Check for best fitting start trip.  Reset when better trip found.

      IF (DIFFSUM .LT. DIFFMIN) THEN
      DIFFMIN = DIFFSUM
      NMIN = NTRIP
      NTIME = TRIP_TIME
      END IF

C     Reset sum and counter for "next" (current) trip and loop back to
C     increment them w/ current values.
```

```
  305 TRIP_TIME = 0
      NOM_TIME = 0
      MFLAG = 0
      IDLE = 0
      LTRIPID = TRIPID
      NTRIP = NTRIP + 1

      DO 310 IACL = -20,20
      DO 310 ISPD = 0,20
      FREQ(IACL,ISPD) = 0.0
  310 CONTINUE

      FREQ(IA,IS) = FREQ(IA,IS) + 1
      TRIP_TIME = TRIP_TIME + 1
      GO TO 200
C
C     Process "hanging trip", print best trip and end.
C
  500 DIFFSUM = 0.0
      DO 580 IACL = -20,20
      DO 580 ISPD = 0,20
      FREQ(IACL,ISPD) = FREQ(IACL,ISPD) / TRIP_TIME
      DIFFSUM = DIFFSUM + ABS (FREQ(IACL,ISPD) - TFREQ(IACL,ISPD))
  580 CONTINUE

      TRIP_TIME = TRIP_TIME + NOM_TIME - 1

C     Output results to unit 3.

      WRITE (3, 300) NTRIP, TRIPID, DIFFSUM, TRIP_TIME, IDLE_TIME

C     Check for best fitting start trip.  Reset when better trip found.

      IF (DIFFSUM .LT. DIFFMIN) THEN
      DIFFMIN = DIFFSUM
      NMIN = NTRIP
      NTIME = TRIP_TIME
      END IF
C
C      WRITE (3, 590) NMIN, DIFFMIN, NTIME
C  590 FORMAT (/I5, 4X, 'Best Trip', 6X, F9.6, 10X, I4)
C
      WRITE (*, 600) IN1, IN2
  600 FORMAT (/' Processing completed for ', I7, ' target surface ',
     +         'records'/
     +         '                          ', I7, ' driving records'/)
C
      END
```

```
+         1X, 'Records kept:  ', I7/
+         1X, 'Microtrips:    ', I7/
+         1X, 'Vehicle trips: ', I7//)

 END
```

```
      PROGRAM COMPARE
C
C     A quick program to generate tabulate comparisons of the Balt Cold
C     and Hot surfaces and the Balt and LA92 Stabilized surfaces.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    04/21/93
C
C-----------------------------------------------------------------------
C
      CHARACTER*80 FNAME, OUTNAME(2)
      REAL PCTBIN(4,-20:20,0:20), PCTDIF(2,-20:20,0:20)
C
      DATA OUTNAME /
     +    'SURFACE DIFFERENCE COMPARISON:  BALT HOT - BALT COLD',
     +    'SURFACE DIFFERENCE COMPARISON:  LA92 STAB - BALT STAB'/
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='OLD')
      OPEN (UNIT=4, STATUS='OLD')
      OPEN (UNIT=5, CARRIAGECONTROL='LIST', STATUS='NEW', RECL=164,
     +              RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Read input mode surface files (units 1-4).
C
      DO 110 IFILE = 1,4
  100 READ (IFILE, *, END=110) IACL, ISPD, IDUM, PCTBIN(IFILE,IACL,ISPD)
      GO TO 100
  110 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Generate difference comparisons:  (1)  Hot vs. Cold
C                                       (2)  LA92 vs. Balt Stabilized
C
      DO 120 IACL = -20,20
      DO 120 ISPD = 0,20
      PCTDIF(1,IACL,ISPD) = PCTBIN(2,IACL,ISPD) - PCTBIN(1,IACL,ISPD)
      PCTDIF(2,IACL,ISPD) = PCTBIN(4,IACL,ISPD) - PCTBIN(3,IACL,ISPD)
  120 CONTINUE
C
C-----------------------------------------------------------------------
C
C     Output results in tabular form, echo input surfaces first.
C
      DO 200 IFILE = 1,4
C
      INQUIRE (UNIT=IFILE, NAME=FNAME)
      CLOSE (IFILE)
      WRITE (5, 130) FNAME
  130 FORMAT (///65X, A80//60X, 'Speed/Accel Frequency Distribution (%)'
     +        ///70X, 'SPD BIN (mph)'//)
      WRITE (5, 140) (ISPD*5, ISPD = 0,20)
  140 FORMAT ('ACL BIN (mph/s)', 4X, I4, 3X, 20('<',I3,3X)/
     +        '---------------', 4X, '----', 3X, 20('----',3X))
C
      DO 150 IACL = -20,20
      WRITE (5, 145) IACL, (PCTBIN(IFILE,IACL,ISPD), ISPD = 0,20)
  145 FORMAT (6X, I4, 6X, 21F7.3)
  150 CONTINUE
C
  200 CONTINUE
C
      DO 300 IOUT = 1,2
C
      WRITE (5, 130) OUTNAME(IOUT)
      WRITE (5, 140) (ISPD*5, ISPD = 0,20)
C
```

```
      DO 250 IACL = -20,20
      WRITE (5, 145) IACL, (PCTDIF(IOUT,IACL,ISPD), ISPD = 0,20)
  250 CONTINUE
C
  300 CONTINUE
C
      CLOSE (5)
C
      END
```

```
      PROGRAM COMPSURF
C
C     A quick program to produce a composite spd/acl surface from the
C     Baltimore and LA92 mode-specific surface files:
C
C     (1) BFRQ1.DAT (Balt Start) ------|
C     (2) BFRQ2.DAT (Balt Stabilized)  |---> (4) CFRQT.DAT (Comp Totl)
C     (3) LFRQ2.DAT (LA92 Stab)  ------|
C
C     Start portions of total surfaces based on Baltimore data only.
C     Stabilized portions of total surfaces) are generated by a 50/50
C     weighting of the Baltimore and LA92 stabilized driving.
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     Author:  Tom Carlson
C     Date:    04/21/93
C
C-----------------------------------------------------------------------
C
      INTEGER NBIN(3,-20:20,0:20), NREC(3)
      REAL CBIN(-20:20,0:20), WFAC(3)
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='OLD')
      OPEN (UNIT=4, CARRIAGECONTROL='LIST', STATUS='NEW')
C
C-----------------------------------------------------------------------
C     Begin processing.
C-----------------------------------------------------------------------
C
C     Read input mode surface files (units 1-3).
C
      DO 110 IFILE = 1,3
  100 READ (IFILE, *, END=105) IACL, ISPD, NBIN(IFILE,IACL,ISPD)
      GO TO 100
  105 CLOSE (IFILE)
  110 CONTINUE
C
C-----------------------------------------------------------------------
C     Input complete, generate weighting factors for LA92/Balt:
C-----------------------------------------------------------------------
C     Balt Start = nrec1
C     Balt Stab  = nrec2
C     LA92 Stab  = nrec3
C
C     For 50/50 weighting of stabilized data, we need the following
C     weightings:
C                 wfac1 = 2     (by specification of the 50/50
C                                weighting of nrec2 and nrec3)
C                 wfac2 = 1     ("unit" weighting)
C
C                 wfac3 = nrec2 / nrec3
C
      DO 150 IACL = -20,20
      DO 150 ISPD = 0,20
      DO 150 IFILE = 1,3
      NREC(IFILE) = NREC(IFILE) + NBIN(IFILE,IACL,ISPD)
  150 CONTINUE
C
      WFAC(1) = 2.0
      WFAC(2) = 1.0
      WFAC(3) = FLOAT (NREC(2)) / FLOAT(NREC(3))
C
C     Generate total weighted composite surface.
C
      DO 200 IACL = -20,20
      DO 200 ISPD = 0,20
      DO 200 I = 1,3
      X = WFAC(I) * NBIN(I,IACL,ISPD)
      CBIN(IACL,ISPD) = CBIN(IACL,ISPD) + X
      CREC = CREC + X
  200 CONTINUE
```

```
      DO 220 IACL = -20,20
      DO 220 ISPD = 0,20
      CPCT = CBIN(IACL,ISPD) * 100 / CREC
      WRITE (4, 210) IACL, ISPD, CBIN(IACL,ISPD), CPCT
  210 FORMAT (2(I3,','), F8.0, ',', F10.6)
  220 CONTINUE

      CLOSE (4)

      WRITE (*, 230) CREC, (WFAC(I), I = 1,3)
  230 FORMAT (/'Processing completed:    Crec  = ', F8.0//
     +        '                         Wfac1 = ', F5.1/
     +        '                         Wfac2 = ', F5.1/
     +        '                         Wfac3 = ', F5.1/)

      END
```

```
      PROGRAM DIFFSURF
C
C     A quick program to generate a spd/acl surface file in "comma-
C     delimited" format from the difference between two input spd/acl
C     surface files (first minus second).
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):     0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):    -0.5 to 0.5 mph/s = 0
C                      -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                      0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     I/O:   Unit 1 (input)   - Spd/acl surface file 1.
C            Unit 2 (input)   - Spd/acl surface file 2.
C            Unit 3 (output)  - Spd/acl surface file created from
C                               (file 1 - file 2).
C
C     Author:  Tom Carlson
C     Date:    06/22/93
C-----------------------------------------------------------------------
C
      REAL DFREQ(-20:20,0:20), FREQ1(-20:20,0:20), FREQ2(-20:20,0:20)
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Open all files.
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='NEW', CARRIAGECONTROL='LIST')
C
C-----------------------------------------------------------------------
C     Read input spd/acl surfaces (units 1 and 2).
C-----------------------------------------------------------------------
C
C     File 1.
C
  100 READ (1, *, END=200) IACL, ISPD, RBIN, PCT
      FREQ1(IACL,ISPD) = PCT
      IN1 = IN1 + 1
      GO TO 100
C
C     File 2.
C
  200 READ (2, *, END=300) IACL, ISPD, RBIN, PCT
      FREQ2(IACL,ISPD) = PCT
      IN2 = IN2 + 1
      GO TO 200
C
C-----------------------------------------------------------------------
C     Now compute difference surface:  file1 - file2 and output.
C-----------------------------------------------------------------------
C
  300 DO 400 IACL = -20,20
      DO 400 ISPD = 0,20
      DFREQ(IACL,ISPD) = FREQ1(IACL,ISPD) - FREQ2(IACL,ISPD)
      WRITE (3, 350) IACL, ISPD, IDUM, DFREQ(IACL,ISPD)
  350 FORMAT (2(I3,','), I8, ',', F10.6)
  400 CONTINUE
C
C-----------------------------------------------------------------------
C
      END
```

```
      PROGRAM DIFFSURFWT
C
C     A quick program to generate a spd/acl surface file in "comma-
C     delimited" format from the difference between two input spd/acl
C     surface files (first minus second).  Thos is a modified version
C     of DIFFSURF which applies a weighting factor to the 2nd input
C     surface:
C             Output SAFD  =  Input SAFD1 - (Input SAFD2 x WtFac)
C
C
C     Summary of speed & accel bin definitions:
C
C     Speed (mph):      0, 0 - <=5, 5 - <=10, etc.
C     Accel(mph/s):     -0.5 to 0.5 mph/s = 0
C                       -1.5>= to -0.5 = -1, -2.5>= to -1.5 = -2, etc.
C                       0.5 to <= 1.5 = 1, 1.5 to <=2.5 = 2, etc.
C
C     I/O:   Unit 1 (input)   - Spd/acl surface file 1.
C            Unit 2 (input)   - Spd/acl surface file 2.
C            Unit 3 (output)  - Spd/acl surface file created from
C                               (file 1 - file 2).
C
C     Author:  Tom Carlson
C     Date:    06/22/93
C-----------------------------------------------------------------------
C
      REAL DFREQ(-20:20,0:20), FREQ1(-20:20,0:20), FREQ2(-20:20,0:20)

      DATA WTFAC / 0.123 /
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Open all files.
C
      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='NEW', CARRIAGECONTROL='LIST')
C
C-----------------------------------------------------------------------
C     Read input spd/acl surfaces (units 1 and 2).
C-----------------------------------------------------------------------
C
C     File 1.
C
  100 READ (1, *, END=200) IACL, ISPD, RBIN, PCT
      FREQ1(IACL,ISPD) = PCT
      IN1 = IN1 + 1
      GO TO 100
C
C     File 2.
C
  200 READ (2, *, END=300) IACL, ISPD, RBIN, PCT
      FREQ2(IACL,ISPD) = PCT
      IN2 = IN2 + 1
      GO TO 200
C
C-----------------------------------------------------------------------
C     Now compute difference surface:  file1 - file2 and output.
C-----------------------------------------------------------------------
C
  300 DO 400 IACL = -20,20
      DO 400 ISPD = 0,20
      DFREQ(IACL,ISPD) = FREQ1(IACL,ISPD) - FREQ2(IACL,ISPD)*WTFAC
      WRITE (3, 350) IACL, ISPD, IDUM, DFREQ(IACL,ISPD)
  350 FORMAT (2(I3,','), I8, ',', F10.6)
  400 CONTINUE
C
C-----------------------------------------------------------------------
C
      END
```

```
      PROGRAM FINDBEST
C
C     FINDBEST reads a VMS directory listing of drive cycle files and
C     finds the best cycles based on minimum DIFFSUM.  The user inputs
C     the number of cycles to be listed.  The program can scan up to
C     10,000 files.
C
C     I/O:   Unit 1 (input)  - Text file directory listing of the cycles
C                              to be scanned.  The text file is created
C                              in batch driver procedure FINDBEST.COM.
C            Unit 2 (input)  - The candidate cycle files.
C            Unit 3 (output) - Ranked list of top cycles.  Stops/min
C                              also computed ans printed for top cycles.
C
C     Author:  Tom Carlson
C     Date:    9/23/92
C
C-----------------------------------------------------------------------

      CHARACTER*6 CDUMMY, METHOD
      CHARACTER*20 CYCFILE(10000)
      CHARACTER*80 OUTFILE, TITLE, CYCNAME

      INTEGER*2 MTRIPS(10000), IPNT(10000), STRIPS(10000)

      REAL CYCTIME(10000), CYCSPD(10000), DIFFSUM(10000), STOPS(10000),
     +     STIME(10000),  MAXSPD(10000)

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for number of top cycles to be ranked and printed.
C
      WRITE (*, 10)
   10 FORMAT (/1X, 'Enter number of "best" cycles to find:')
      READ (*, *) NBEST
      WRITE (*, 20)
   20 FORMAT (/1X, 'Enter output filename:')
      READ (*, 30) OUTFILE
   30 FORMAT (A)
      WRITE (*, 40)
   40 FORMAT (/1X, 'Enter run title (up to 80 chars):')
      READ (*, 30) TITLE
C
C     Open CYCLELIST file which contains the directory listing of drive
C     cycles to be processed.  Output file (unit 3) opened here also.
C
      OPEN (UNIT=1, FILE='CYCLELIST.TXT', STATUS='OLD')
      OPEN (UNIT=3, FILE=OUTFILE, STATUS='NEW', CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE')
C
C-----------------------------------------------------------------------
C     Cycle name list loop (100).
C-----------------------------------------------------------------------
C
C     Read CYCLELIST one record (cycle filename) at a time to get the
C     "current" cycle file to be scanned.
C
  100 IFILE = IFILE + 1
      READ (1, 110, END=300, ERR=100) CYCFILE(IFILE)
  110 FORMAT (A)

C     Open current cycle file (unit 2) and read METHOD in first record.

      CYCNAME = '[.CYCLES]' // CYCFILE(IFILE)
      OPEN (UNIT=2, FILE=CYCNAME, STATUS='OLD')
      READ (2, 190) METHOD
  190 FORMAT (5X, A6//////)
C
C     Now read current cycle file, scanning for '___' record.
C     Cycle summary statistics data follow the '___' line.
C

  200 READ (2, 201) CDUMMY
  201 FORMAT (49X, A6)

      IF (CDUMMY .EQ. '______') GO TO 210
      READ (CDUMMY, FMT='(F6.2)') SPDMAX
      IF (SPDMAX .GT. MAXSPD(IFILE)) MAXSPD(IFILE) = SPDMAX
```

```
      GO TO 200
C
C     "Underlines" found, now read stats info for current cycle.
C     (Note:  Embedded record "Base Time" for HYBRID is skipped.)
C
  210 READ (2, 220) MTRIPS(IFILE), STRIPS(IFILE), CYCTIME(IFILE),
     +              STIME(IFILE), CYCSPD(IFILE), DIFFSUM(IFILE)
  220 FORMAT (/26X, I3, T68, I3/26X, F6.2, T68, F6.2//26X, F6.2/
     +         26X, F9.5)
C
C     Scanning completed for current cycle file.  Close current file
C     then loop back to 100 and start on next file.
C
      CLOSE (2)
      GO TO 100

C-----------------------------------------------------------------------
C     Input complete, processing and ranking follow.
C-----------------------------------------------------------------------

  300 CLOSE (1)
      IFILE = IFILE - 1

C     Compute stops/min.

      DO 310 I = 1, IFILE
      STOPS(I) = MTRIPS(I) / CYCTIME(I)
  310 CONTINUE
C
C     Sort arrays by descending difference.  Use indirect
C     sorting algorithm, storing the "pointer" rankings in IPNT.
C
C     Initialize IPNT array with ascending pointers.
C
      DO 320 I = 1, IFILE
      IPNT(I) = I
  320 CONTINUE
C
      DO 350 I = 1, IFILE-1
      L = IPNT(I+1)

      DO 330 J = 1, I
      K = I + 1 - J
      IF (DIFFSUM(L) .GT. DIFFSUM(IPNT(K))) GO TO 340
      IPNT(K+1) = IPNT(K)
  330 CONTINUE

      K = 0
  340 IPNT(K+1) = L
  350 CONTINUE
C
C     Pointer sorted.  Now compute PKE for the NBEST cycles.  Second-by
C     second velocities are retrieved from master file LA92_SIERRA.DAT.
C
C     Print first NBEST values in output arrays.
C
      WRITE (3, 390) TITLE
  390 FORMAT (5X, A80///T30,
     +        '                                TIME    MEAN SPD  MAX SPD'/
     +        'RANK   CYCLE FILE', T30, '   TOTAL DIFF     M-TRIPS   ',
     +        '(min)    (mph)     (mph)   STOPS/MIN'//
     +        '____   _________', T30, '   _________     ______   ',
     +        '_____  _______  ______  ________'/)

      DO 400 I = 1, NBEST
      J = IPNT(I)
      WRITE (3, 395) I, CYCFILE(J), DIFFSUM(J), MTRIPS(J), CYCTIME(J),
     +               CYCSPD(J), MAXSPD(J), STOPS(J)
  395 FORMAT (I3, 4X, A20, T30, F12.5, I11, 3F10.2, F9.3)
  400 CONTINUE

      CLOSE (3)

      STOP

      END
```

```
/*                              *
 *     GETBALT.SAS program      *
 *                              */
options ls=132 ps=80 missing=' ';

libname rad 'fujia:[epa.drivecycle]';

data a;
   set rad.vehs;
   IF (SAMPLE=:'s' or SAMPLE=:'S') then delete;
   keep SAMPLE VIN;
proc sort;
   by VIN;

DATA E;
 rename MPH = SPD;
 MERGE A (IN=I1)
 rad.opns (IN=I2);
 BY VIN;
 IF (I1 AND I2);

   file drivdat;
   put @1 SAMPLE  $  4.
       @6 VIN     $ 17.
      @24 DATE       YYMMDD6.
      @31 HHMMSS     TIME8.
      @40 SPD        6.2 ;
```

```
      PROGRAM GETBIN
C
C     The GETBIN program retrieves microtrips which contain travel in
C     spd/acl bins specified by the user.  If a microtrip contains
C     travel within the input bin(s), GETBIN retrieves the spd vs. time
C     for that m-trip and outputs it.
C
C     I/O:   Unit 1 (input)   - Second-by-second "archive" of all route
C                               data.
C            Unit 2 (input)   - File containing the spd/acl bins pairs
C                               of interest.
C            Unit 3 (input)   - MICTRIPS profile statistics output file.
C            Unit 4 (output)  - Speed vs. time records of microtrips
C                               which contain travel in the input bins.
C
C     Author:  Tom Carlson
C     Date:    05/24/93
C-----------------------------------------------------------------------

      CHARACTER*14 IDLIST(500), ITRIPID

      INTEGER*2 MCNT(-20:20,0:20), JTIME, M
      INTEGER*4 LBIN(50,2), IRECLIST(500), MTRIPLIST(500),
     +          ITIMELIST(500), KOUNT(500,50)

      REAL MSPDLIST(500)
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Open all files.
C
      OPEN (UNIT=1, FILE='EPACOMP.DAT', STATUS='OLD')
      OPEN (UNIT=2, STATUS='OLD')
      OPEN (UNIT=3, STATUS='OLD', FORM='UNFORMATTED', RECL=1760)
      OPEN (UNIT=4, STATUS='NEW')
C
C-----------------------------------------------------------------------
C     Get list of selected bins from unit 2.
C-----------------------------------------------------------------------
C
  100 I = I + 1
      READ (2, *, END=200) (LBIN(I,J), J = 1,2)
      GO TO 100

  200 NB = I - 1
C
C-----------------------------------------------------------------------
C     Now read the unformatted mictrip profiles file (unit 3) and
C     mark those m-trips which contain travel in the input bin list.
C-----------------------------------------------------------------------
C
  205 READ (3, END=300) M, ITRIPID, JREC, JTIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((MCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)
C
      DO 210 I = 1,NB
      IA = LBIN(I,1)
      IS = LBIN(I,2)

      IF (MCNT(IA,IS) .GT. 0) THEN
C     <<< Travel in input bin(s) found >>>
      K = K + 1
      MTRIPLIST(K) = M
      IDLIST(K) = ITRIPID
      IRECLIST(K) = JREC
      ITIMELIST(K) = JTIME
      MSPDLIST(K) = SPD_MAX
      KOUNT(K,I) = MCNT(IA,IS)
      GO TO 220
      END IF
  210 CONTINUE
C
  220 GO TO 205
C
  300 NTRP = K
C
C-----------------------------------------------------------------------
C     Matching m-trips retrieved.  Now lookup second-by-second speed vs.
C     time traces in master archive file ROUTEFILE (unit 1) and output
```

```
C     to unit 4.
C-----------------------------------------------------------------------
C
      DO 350 J = 1, NTRP

      IF (J .GT. 1) THEN
      NSKIP = IRECLIST(J) - (IRECLIST(J-1) + ITIMELIST(J-1))
      ELSE
      NSKIP = IRECLIST(J) - 1
      END IF

C     Skip records until next selected M-trip is reached.

      DO 310 ISKIP = 1, NSKIP
      READ (1, FMT='()')
  310 CONTINUE
C
C     Loop thru next selected M-trip.
C
      DO 330 N = 1, ITIMELIST(J)
C     <<< Read segment >>>
      READ (1, 320) SPD, ACL, ISPD, IACL
  320 FORMAT (18X, 2F6.2, 2I3)
C     <<< Write segment to unit 4 >>>
      WRITE (4, 325) N, J, SPD, ACL, ISPD, IACL
  325 FORMAT (2I9, 2F6.2, 2I3)
  330 CONTINUE

  350 CONTINUE
C
C-----------------------------------------------------------------------
C     All processing and output complete.  Print summary and end.
C-----------------------------------------------------------------------
C
      WRITE (*, 390)
  390 FORMAT (//
     +     1X, ' #   M-Trip No   Microtrip ID     Start Rec   Time (s)',
     +              '   Max Spd (mph)   Bin Count'/
     +     1X, ' - . ---------   ------------     ---------   --------',
     +              '   -------------   ---------')
      DO 400 J = 1,NTRP
      WRITE (*, 395) J, MTRIPLIST(J), IDLIST(J), IRECLIST(J),
     +               ITIMELIST(J), MSPDLIST(J), (KOUNT(J,I), I = 1,NB)
  395 FORMAT (I3, 5X, I5, 5X, A14, 4X, I7, 5X, I4, 7X, F6.2, 9X, 10I5)
  400 CONTINUE

      END
```

```
      PROGRAM GETCYCLE
C
C     The GETCYCLE program generates a second-by-second speed vs. time
C     driving cycle from the list of microtrips contained in a GETSTAB
C     output file.  An accel/speed % distribution file and an
C     "envelope boundary" file which are used later by SURFER to plot
C     the accel/speed surface are also output.
C
C     Trip start and trip end profiles are attached to the
C     output cycle as well.
C
C     I/O:   Unit 1 (input)   - Second-by-second "archive" of all route
C                               data.
C            Unit 2 (input)   - GETSTAB output table of microtrips for a
C                               selected cycle.
C            Unit 3 (input)   - MICTRIPS profile statistics output file.
C            Unit 5 (input)   - SURFER-GRID template command file.
C            Unit 6 (input)   - SURFER-SURF template command file.
C            Unit 7 (input)   - Speed/accel total "surface" created from
C                               sec-by-sec drive cycle data.
C            Unit 8 (input)   - Speed/accel start "surface" for first
C                               120 sec of sec-by-sec drive cycle data.
C            - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C            Unit 9 (output)  - Speed vs. time driving cycle (with trip
C                               ends included).
C            Unit 10 (output) - ASCII acl/spd bin % distribution total
C                               file (later surface plotting w/SURFER).
C            Unit 11 (output) - ASCII acl/spd bin % distribution start
C                               file (later surface plotting w/SURFER).
C            Unit 12 (output) - ASCII acl/spd bin % dist total diff
C                               file (later surface plotting w/SURFER).
C            Unit 13 (output) - ASCII acl/spd bin % dist start diff
C                               file (later surface plotting w/SURFER).
C            Unit 14 (output) - ASCII acl/spd bin total "boundary"
C                               file (later surface plotting w/SURFER).
C            Unit 15 (output) - ASCII acl/spd bin start "boundary"
C                               file (later surface plotting w/SURFER).
C            Unit 16 (output) - SURFER-GRID total surface command file.
C            Unit 17 (output) - SURFER-GRID start surface command file.
C            Unit 18 (output) - SURFER-GRID total diff command file.
C            Unit 19 (output) - SURFER-GRID start diff command file.
C            Unit 20 (output) - SURFER-SURF total surface command file.
C            Unit 21 (output) - SURFER-SURF start surface command file.
C            Unit 22 (output) - SURFER-SURF total diff command file.
C            Unit 23 (output) - SURFER-SURF start diff command file.
C
C     Author:  Tom Carlson
C     Date:    04/27/93
C-----------------------------------------------------------------------

      CHARACTER    CJ, CTYPE(2), MTYPE(50)
      CHARACTER*3  CDUMMY
      CHARACTER*6  METHOD
      CHARACTER*14 MTRIPID(50), ITRIPID
      CHARACTER*23 CTIME
      CHARACTER*25 TITLE(4)
      CHARACTER*40 INFILE, OUTROOT, SURFILE(4), FREQFILE, PRNFILE,
     +             BLNFILE(2), CMDFILE(2,4), GRDFILE(4), PLTFILE(4),
     +             MTRIPFILE
      CHARACTER*55 CMDS(2,200)

      INTEGER*2 MTRIPS(50), ITIME(50), HH, MM, SS, DD, JINDX(50),
     +          SPDBIN, NL(2), MCNT(-20:20,0:20), JTIME, M,
     +          BINS(50,5000), BINA(50,5000)
      INTEGER*4 IREC(50), IPNT(50), MRAN(50), MINDX(50), JREC, ITRIP(50)

      REAL SPD(50,5000), ACL(50,5000), CFREQ(-20:20,0:20), BACL(2,0:20),
     +     MAXSPD, ACLSUM(0:20), PFREQ(-20:20,0:20), SFREQ(-20:20,0:20),
     +     CSFREQ(-20:20,0:20), SPDSUM(0:20), SPD_MEAN, SPD_MAX,
     +     SPD_STD, ACL_MAX

      DATA IHDR / 7 /, CTYPE / 'G', 'S' /, ISTART / 240 /
C
C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for input and output filenames.
C
      WRITE (*, 10)
```

```
   10 FORMAT (/1X, 'Enter input M-Trips summary filename:')
      READ (*, 20) INFILE
      WRITE (*, 15)
   15 FORMAT (/1X, 'Enter unformatted M-Trips Profile filename:')
      READ (*, 20) MTRIPFILE
   20 FORMAT (A)
      WRITE (*, 30)
   30 FORMAT (/1X, 'Enter output cycle "root" filename (< 8 chars):')
      READ (*, 20) OUTROOT
      WRITE (*, 40)
   40 FORMAT (/1X, 'Enter Z-value (in %) for "envelope boundary" file:')
      READ (*, *) Z

C     Name all PRN and SURFER files based on OUTROOT.

      L = INDEX (OUTROOT, ' ') - 1
      PRNFILE = OUTROOT(1:L) // '.PRN'
      BLNFILE(1) = OUTROOT(1:L) // '.BNT'
      BLNFILE(2) = OUTROOT(1:L) // '.BNS'
      DO 70 J = 1,4
      WRITE (CJ, FMT='(I1)') J
      SURFILE(J) = OUTROOT(1:L) // '.DA' // CJ
      GRDFILE(J) = OUTROOT(1:L) // '.GR' // CJ
      PLTFILE(J) = OUTROOT(1:L) // '.PL' // CJ
      DO 70 I = 1,2
      CMDFILE(I,J) = OUTROOT(1:L) // CTYPE(I) // '.CM' // CJ
   70 CONTINUE

C     Set up run title names based on OUTROOT.

      TITLE(1) = '  Total Cycle: ' // OUTROOT(1:L)
      TITLE(2) = '  Start Cycle: ' // OUTROOT(1:L)
      TITLE(3) = 'Total Difference: ' // OUTROOT(1:L)
      TITLE(4) = 'Start Difference: ' // OUTROOT(1:L)

C
C     Open all files.
C
      OPEN (UNIT=1, FILE='EPACOMP.DAT', STATUS='OLD')
      OPEN (UNIT=2, FILE=INFILE, STATUS='OLD')
      OPEN (UNIT=3, FILE=MTRIPFILE, STATUS='OLD', FORM='UNFORMATTED',
     +      RECL=1760)
      OPEN (UNIT=5, FILE='DRIVGRID.CMD', STATUS='OLD')
      OPEN (UNIT=6, FILE='DRIVSURF.CMD', STATUS='OLD')
      OPEN (UNIT=7, FILE='CFRQT.DAT', STATUS='OLD')
      OPEN (UNIT=8, FILE='BFRQ1.DAT', STATUS='OLD')

      OPEN (UNIT=9, FILE=PRNFILE, STATUS='NEW', CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE')
      OPEN (UNIT=14, FILE=BLNFILE(1), CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE', STATUS='NEW')
      OPEN (UNIT=15, FILE=BLNFILE(2), CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE', STATUS='NEW')
      DO 80 J = 1,4
      LUN = J + 9
      OPEN (UNIT=LUN, FILE=SURFILE(J), CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE', STATUS='NEW')
      DO 80 I = 1,2
      LUN = (I - 1) * 4 + J + 15
      OPEN (UNIT=LUN, FILE=CMDFILE(I,J), CARRIAGECONTROL='LIST',
     +      RECORDTYPE='VARIABLE', STATUS='NEW')
   80 CONTINUE
C
C-----------------------------------------------------------------------
C     Input SURFER "template" command files (units 5 & 6).
C-----------------------------------------------------------------------
C
   90 DO 150 K = 1,2
      LINE = 0
      LUN = K + 4

  100 LINE = LINE + 1
      READ (LUN, 105, END=130) CMDS(K,LINE)
  105 FORMAT (A55)
      GO TO 100

  130 NL(K) = LINE - 1
  150 CONTINUE
C
C-----------------------------------------------------------------------
```

```
C     Read total population and start population surfaces (units 7 & 8).
C-----------------------------------------------------------------------
C
C     Total population surface.
C
  170 READ (7, *, END=180) IACL, ISPD, NBIN, PCT
      PFREQ(IACL,ISPD) = PCT
      IN7 = IN7 + 1
      GO TO 170
C
C     Start population surface (first 240 seconds of total surface).
C
  180 READ (8, *, END=190) IACL, ISPD, NBIN, PCT
      SFREQ(IACL,ISPD) = PCT
      IN8 = IN8 + 1
      GO TO 180
C
C-----------------------------------------------------------------------
C     Get list of selected M-trips from INFILE, skipping IHDR header
C     records (unit 2).
C-----------------------------------------------------------------------
C
  190 DO 199 ISKIP = 1,IHDR
      READ (2, FMT='()')
  199 CONTINUE

  200 ITRP = ITRP + 1
      READ (2, 205, ERR=210) ITRIP(ITRP), MTYPE(ITRP), MTRIPS(ITRP),
     +                       MTRIPID(ITRP), IREC(ITRP), ITIME(ITRP)
  205 FORMAT (I3, 1X, A1, 1X, I5, 1X, A14, 1X, I7, 1X, I4)
      IF (MTYPE(ITRP) .EQ. 'S') NST = NST + 1
      GO TO 200
C
C-----------------------------------------------------------------------
C     All M-trips read from INFILE.  Now sort 'em by increasing initial
C     record number IREC.  Use indirect sorting, storing pointer
C     rankings in IPNT.
C-----------------------------------------------------------------------
C
  210 NTRP = ITRP - 1
      DO 220 I = 1, NTRP
      IPNT(I) = I
  220 CONTINUE

      DO 250 I = 1, NTRP-1
      L = IPNT(I+1)

      DO 230 J = 1, I
      K = I + 1 - J
      IF (IREC(L) .GT. IREC(IPNT(K))) GO TO 240
      IPNT(K+1) = IPNT(K)
  230 CONTINUE

      K = 0
  240 IPNT(K+1) = L
  250 CONTINUE
C
C-----------------------------------------------------------------------
C     Pointer sorted.  Now lookup second-by-second speed vs. time
C     traces in master archive file ROUTEFILE (unit 1).
C-----------------------------------------------------------------------
C
      DO 350 ITRP = 1, NTRP

      DO 305 K = 1, NTRP
      IF (IPNT(K) .EQ. ITRP) THEN
      J = K
      GO TO 308
      END IF
  305 CONTINUE

  308 JINDX(ITRP) = J
      J = IPNT(ITRP)
      IF (ITRP .GT. 1) THEN
      NSKIP = IREC(J) - (IREC(J1) + ITIME(J1))
      ELSE
      NSKIP = IREC(J) - 1
      END IF

C     Skip records until next selected M-trip is reached.
```

```
      DO 310 ISKIP = 1, NSKIP
      READ (1, FMT='()')
  310 CONTINUE
C
C     Loop thru next selected M-trip.
C
      DO 330 N = 1, ITIME(J)

C     Read segment.

      READ (1, 320) SPD(J,N), ACL(J,N), ISPD, IACL
  320 FORMAT (18X, 2F6.2, 2I3)
      BINS(J,N) = ISPD
      BINA(J,N) = IACL
      NSEC = NSEC + 1

      IF (NSEC .LE. ISTART) THEN
      CSFREQ(IACL,ISPD) = CSFREQ(IACL,ISPD) + 1
      IF (SPD(J,N) .GT. SPEAKSPD) SPEAKSPD = SPD(J,N)
      END IF

      IF (SPD(J,N) .GT. TPEAKSPD) TPEAKSPD = SPD(J,N)

  330 CONTINUE

      J1 = J
  350 CONTINUE
C
C-----------------------------------------------------------------------
C     Speed & accel vs. time traces retrieved for all selected M-trips.
C     Now re-randomize the "stabilized" M-trips (hills).
C-----------------------------------------------------------------------
C     Random Reshuffle section commented out -- TRC 5/10/93.
C
C     Get system time.  Call to LIB$DATE_TIME returns 23-byte character
C     string which contains hour, minute, second and decimal fraction.
C
C      ISTAT = LIB$DATE_TIME (CTIME)
C      READ (CTIME, FMT='(11X,4(1X,I2))') HH, MM, SS, DD
C
C     Create seed value for random series.  Seed ranges from 0 to 10**8.
C     Make seed odd per VAX-FORTRAN recommendations (FORTRAN Language
C     Reference Manual V 5.0, pg. D-49).
C
C      ISEED = DD * 1000000 + SS * 16666.67 + MM * 166.67 + HH * 4.1667
C      IR = MOD (ISEED, 2)
C      IF (IR .EQ. 0) ISEED = ISEED + 1
C-----------------------------------------------------------------------
C      DO 400 I = 1, NTRP
C      MINDX(I) = I
C  400 CONTINUE
C
C      NS = NTRP - NST
C
C      DO 410 I = 1, NTRP
C
C     Skip re-randomization for start portion (first NST M-trips).
C
C      IF (I .LE. NST) THEN
C      MRAN(I) = I
C      GO TO 410
C      END IF
C
C      L = NS - I + 1
C
C     Randomly select a microtrip in the range 0 to NS - I + 1.
C     (NOTE:  Whenever an M-trip is selected, it's lifted from the
C     "selectable M-trips" sample so it's not picked again.)
C
C      XRAN = RAN (ISEED)
C      IF (XRAN .LT. 1.0) THEN
C      K = INT (L * XRAN) + 1
C      ELSE
C      K = L
C      END IF
C
C     Next line shifts the random number. For example, for 10 M-trips
C     with 2 start M-trips, the first pass thru loop 410 selects a
C     random number in the range 0-8 (10-2-1+1) which represents the
```

```
C     stabilized M-trips 2 thru 10.
C
C      K = K + NST
C
C      MRAN(I) = MINDX(K)
C
C      DO 410 J = K, NTRP-1
C      MINDX(J) = MINDX(J+1)
C  410 CONTINUE
C
C-----------------------------------------------------------------------
C     Write sec-by-sec speed and accel to PRNFILE (unit 9) in
C     re-shuffled order.
C-----------------------------------------------------------------------
C
      DO 450 I = 1, NTRP

      DO 440 N = 1, ITIME(I)

C     Write speed and accel for current segment.

      M = M + 1
      WRITE (9, 435) M, ITRIP(I), SPD(I,N), ACL(I,N), BINS(I,N),
     +               BINA(I,N)
  435 FORMAT (I4, I3, F7.2, F7.3, 2I4)

  440 CONTINUE

  450 CONTINUE
C
C-----------------------------------------------------------------------
C     Sec-by-sec cycle speed profile created and output to unit 9.  Now
C     read speed/accel bin frequency counts data for selected M-trips
C     created by the MICTRIPS program from unit 3.  After each M-trip is
C     input, increment bin counts for that M-trip.
C-----------------------------------------------------------------------
C
      DO 530 I = 1, NTRP
      J = IPNT(I)

      IF (I .GT. 1) THEN
      NSKIP = MTRIPS(J) - (MTRIPS(J1) + 1)
      ELSE
      NSKIP = MTRIPS(J) - 1
      END IF

C     Skip records until next selected M-trip is reached.

      DO 510 ISKIP = 1, NSKIP
      READ (3) M, ITRIPID, JREC, JTIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((MCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)
  510 CONTINUE

C     Read segment.

      READ (3) M, ITRIPID, JREC, JTIME, SPD_MEAN, SPD_MAX,
     +    SPD_STD, ACL_MAX, ((MCNT(IACL,ISPD), ISPD=0,20), IACL=-20,20)

      J1 = J

C     Now increment bin counts with current M-trip counts.

      DO 520 IACL = -20,20
      DO 520 ISPD = 0,20
      CFREQ(IACL,ISPD) = CFREQ(IACL,ISPD) + MCNT(IACL,ISPD)
  520 CONTINUE

      TOTTIME = TOTTIME + ITIME(J)
  530 CONTINUE
C
C-----------------------------------------------------------------------
C     Total bin count reads finished.  Compute cycle total and start
C     frequencies and differences from population surfaces (in %) from
C     the accumulated sums and output to SURFILEs (units 10-13).
C-----------------------------------------------------------------------
C
      DO 540 IACL = -20,20
      ACLBIN = IACL
      DO 540 ISPD = 0,20
      SPDBIN = ISPD * 5
```

```
      CFREQ(IACL,ISPD) = CFREQ(IACL,ISPD) * 100 / TOTTIME
      CSFREQ(IACL,ISPD) = CSFREQ(IACL,ISPD) * 100 / ISTART
      DIFFT = CFREQ(IACL,ISPD) - PFREQ(IACL,ISPD)
      DIFFS = CSFREQ(IACL,ISPD) - SFREQ(IACL,ISPD)
      WRITE (10, 535) ACLBIN, SPDBIN, CFREQ(IACL,ISPD)
      WRITE (11, 535) ACLBIN, SPDBIN, CSFREQ(IACL,ISPD)
      WRITE (12, 535) ACLBIN, SPDBIN, DIFFT
      WRITE (13, 535) ACLBIN, SPDBIN, DIFFS
  535 FORMAT (F7.2, I7, F10.6)
  540 CONTINUE
C
C-----------------------------------------------------------------------
C     Cycle "surface" frequency file output complete.  Determine the
C     "envelope boundary" for the input Z-value and output results to
C     BLNFILEs (units 14 and 15).
C-----------------------------------------------------------------------
C
C     ---------------------
C     Total surface boundary
C     ---------------------
C
      DO 600 ISPD = 0,20
      SPDSUM(ISPD) = CFREQ(0,ISPD)
      ACLSUM(ISPD) = 0.0
      DO 600 IACL = -20,20
      ACLSUM(ISPD) = ACLSUM(ISPD) + CFREQ(IACL,ISPD)
  600 CONTINUE
C
      DO 650 ISPD = 0,20
      SUM1 = 0.0
      SUM2 = 0.0
      SUM1L = 0.0
      SUM2L = 0.0
      I1 = 0
      I2 = 0

      IF (ACLSUM(ISPD) .EQ. 0.0) GO TO 640

C     Check negative accel region.

      DO 610 IACL = -20,0
      SUM1 = CFREQ(IACL,ISPD)
  605 IF (SUM1 .GT. Z) THEN
      I1 = IACL - 1
      GO TO 620
      END IF
      SUM1L = SUM1
  610 CONTINUE

C     Check positive accel region.

  620 DO 630 IACL = 0,20
      SUM2 = CFREQ(IACL,ISPD)
  625 IF (SUM2 .GT. Z) THEN
      I2 = IACL + 1
      GO TO 640
      END IF
      SUM2L = SUM2
  630 CONTINUE

  640 IF (I1 .EQ. 0 .AND. I2 .EQ. 0) THEN
      I0 = ISPD - 1
      GO TO 660
      ELSE

         IF (SUM1 - SUM1L .NE. 0.0) THEN
         BACL(1,ISPD) = I1 + (Z - SUM1L) / (SUM1 - SUM1L)
         ELSE
         BACL(1,ISPD) = I1
         END IF

         IF (SUM2 - SUM2L .NE. 0.0) THEN
         BACL(2,ISPD) = I2 - (Z - SUM2L) / (SUM2 - SUM2L)
         ELSE
         BACL(2,ISPD) = I2
         END IF

      END IF

  650 CONTINUE
```

```
  660 I3 = I0 + 1

      DO 662 ISPD = 20,0,-1
      IF (SPDSUM(ISPD) .GT. Z) THEN
      ISPD0 = ISPD + 1
      GO TO 663
      END IF
  662 CONTINUE

C     Envelope locations determined.  Output to unit 14.

  663 DO 690 IREG = 1,2

      WRITE (14, 665) I3
  665 FORMAT (I4, '   0')

      DO 680 ISPD = 0, I0
      SPDBIN = ISPD * 5
      ACLBIN = BACL(IREG,ISPD)
      WRITE (14, 670) ACLBIN, SPDBIN
  670 FORMAT (F4.1, I4)
  680 CONTINUE

      SPDBIN = 10 * (ISPD0 - 1 - (Z - SPDSUM(ISPD0)) /
     +                (SPDSUM(ISPD0-1) - SPDSUM(ISPD0))) + 0.5
      WRITE (14, 685) SPDBIN
  685 FORMAT (' 0.0', I4)

  690 CONTINUE
C
C     ---------------------
C     Start surface boundary
C     ---------------------
C
      DO 700 ISPD = 0,20
      SPDSUM(ISPD) = CSFREQ(0,ISPD)
      ACLSUM(ISPD) = 0.0
      DO 700 IACL = -20,20
      ACLSUM(ISPD) = ACLSUM(ISPD) + CSFREQ(IACL,ISPD)
  700 CONTINUE
C
      DO 750 ISPD = 0,20
      SUM1 = 0.0
      SUM2 = 0.0
      SUM1L = 0.0
      SUM2L = 0.0
      I1 = 0
      I2 = 0

      IF (ACLSUM(ISPD) .EQ. 0.0) GO TO 740

C     Check negative accel region.

      DO 710 IACL = -20,0
      SUM1 = CSFREQ(IACL,ISPD)
  705 IF (SUM1 .GT. Z) THEN
      I1 = IACL - 1
      GO TO 720
      END IF
      SUM1L = SUM1
  710 CONTINUE

C     Check positive accel region.

  720 DO 730 IACL = 0,20
      SUM2 = CSFREQ(IACL,ISPD)
  725 IF (SUM2 .GT. Z) THEN
      I2 = IACL + 1
      GO TO 740
      END IF
      SUM2L = SUM2
  730 CONTINUE

  740 IF (I1 .EQ. 0 .AND. I2 .EQ. 0) THEN
      I0 = ISPD - 1
      GO TO 760
      ELSE

         IF (SUM1 - SUM1L .NE. 0.0) THEN
```

```
        BACL(1,ISPD) = I1 + (Z - SUM1L) / (SUM1 - SUM1L)
        ELSE
        BACL(1,ISPD) = I1
        END IF

        IF (SUM2 - SUM2L .NE. 0.0) THEN
        BACL(2,ISPD) = I2 - (Z - SUM2L) / (SUM2 - SUM2L)
        ELSE
        BACL(2,ISPD) = I2
        END IF

      END IF

  750 CONTINUE

  760 I3 = I0 + 1

      DO 762 ISPD = 20,0,-1
      IF (SPDSUM(ISPD) .GT. Z) THEN
      ISPD0 = ISPD + 1
      GO TO 763
      END IF
  762 CONTINUE

C     Envelope locations determined.  Output to unit 15.

  763 DO 790 IREG = 1,2

      WRITE (15, 665) I3
  765 FORMAT (I4, '   0')

      DO 780 ISPD = 1, I0
      SPDBIN = ISPD * 5
      ACLBIN = BACL(IREG,ISPD)
      WRITE (15, 770) ACLBIN, SPDBIN
  770 FORMAT (F4.1, I4)
  780 CONTINUE

      SPDBIN = 10 * (ISPD0 - 1 - (Z - SPDSUM(ISPD0)) /
     +                 (SPDSUM(ISPD0-1) - SPDSUM(ISPD0))) + 0.5
      WRITE (15, 785) SPDBIN
  785 FORMAT (' 0.0', I4)

  790 CONTINUE
C
C-----------------------------------------------------------------------
C     Create run-specific SURFER command files from the "templates"
C     input into the CMDS array and output to units 16-23.
C-----------------------------------------------------------------------
C
      DO 850 I = 1,2
      DO 850 J = 1,4
      LUN = 15 + (I - 1) * 4 + J

      GO TO (810, 820) I

C     GRID module command files.

  810 LG = INDEX (GRDFILE(J), ' ') - 1
      LS = INDEX (SURFILE(J), ' ') - 1
      CMDS(I,14) = 'groname="' // GRDFILE(J)(1:LG) // '"'
      CMDS(I,73) = 'griname="' // SURFILE(J)(1:LS) // '"'

      DO 815 LINE = 1,NL(I)
      WRITE (LUN, 812) CMDS(I,LINE)
  812 FORMAT (A55)
  815 CONTINUE

      GO TO 850

C     SURF module command files.

  820 LG = INDEX (GRDFILE(J), ' ') - 1
      CMDS(I,12) = 'inname="' // GRDFILE(J)(1:LG) // '"'
      CMDS(I,35) = 'titstr="' // TITLE(J) // '"'

      IF (J .LE. 2) THEN
      LB = INDEX (BLNFILE(J), ' ') - 1
      CMDS(I,98) = 'xyname="' // BLNFILE(J)(1:LB) // '"'
      ELSE
```

```
      CMDS(I,98) = 'xyname=""'
      END IF

      LS = INDEX (SURFILE(J), ' ') - 1
      LP = INDEX (PLTFILE(J), ' ') - 1
      CMDS(I,100) = 'postname="' // SURFILE(J)(1:LS) // '"'
      CMDS(I,112) = 'outname="' // PLTFILE(J)(1:LP) // '"'

      DO 825 LINE = 1,NL(I)
      WRITE (LUN, 822) CMDS(I,LINE)
  822 FORMAT (A55)
  825 CONTINUE

  850 CONTINUE

C-----------------------------------------------------------------------
C     All processing and output complete.  Close all I/O units and end.
C-----------------------------------------------------------------------
C
      DO 1000 LUN = 1,23
      IF (LUN .EQ. 4) GO TO 1000
      CLOSE (LUN)
 1000 CONTINUE

      STOP

      END
```

```
      PROGRAM GETMTRIP
C
C     GETMTRIP retrieves a speed vs. time trace from the master file
C     EPACOMP.DAT based on a user-input start record number and number
C     of seconds.
C
C     GETMTRIP is a modified version of the GETBEST program.
C
C     I/O:   Unit 1 (input)  - Master file (e.g., EPACOMP.DAT).
C            Unit 2 (output) - Retrieved speed vs time trace.
C
C     Author:  Tom Carlson
C     Date:    5/26/93
C
C-----------------------------------------------------------------------

      OPEN (UNIT=1, STATUS='OLD')
      OPEN (UNIT=2, STATUS='NEW')

C-----------------------------------------------------------------------
C    Begin execution.
C-----------------------------------------------------------------------
C
C     Prompt user for retrieval record numbers.
C
      WRITE (*, 2)
    2 FORMAT (/1X,
     + 'Enter start rec number and # of segments (delimited):')
      READ (*, *) INITREC, NREC
C
C-----------------------------------------------------------------------
C     Retrieve trace.
C-----------------------------------------------------------------------
C
C     Skip records until first selected segment is reached.
C
      DO 310 ISKIP = 1, INITREC-1
      READ (1, FMT='()')
      N = N + 1
  310 CONTINUE
C
C     Loop thru next selected segments, reading unit 1 and writing to
C     unit 2.
C
      DO 330 I =-1, NREC
C     <<<  Read segment  >>>
      READ (1, 320) SPD, ACL, ISPD, IACL
  320 FORMAT (18X, 2F6.2, 2I3)
      N = N + 1
C     <<<  Write segment to unit 2  >>>
      WRITE (2, 325) I, IDUM, SPD, ACL, ISPD, IACL
  325 FORMAT (2I5, 2F6.2, 2I3)
  330 CONTINUE
C
      END
```

APPENDIX B

Sample Cycle Development Files

Sample Cycle Summary File

HYBRID Trip Selection Method
1st Seed: 83900569
Filename: E18-394.OUT

Target Time: 30.00 min
Base Time: 18.00 min
Start Time: 4.00 min

#		MTRP NO	MTRIP ID	INIT REC	TIME (s)	AVG SPD (mph)	MAX SPD (mph)	STD SPD (mph)	MAX ACL (mph/s)	DIFFSUM
1	S	513	b0729203140401	258980	193	18.46	38.55	12.97	3.600	0.75267
2	S	514	b0729203140402	259173	83	17.07	40.97	16.16	5.050	0.62903
3	O	6176	b1639203210102	3008050	444	61.43	74.06	12.86	8.460	0.97822
4	R	3012	b3379203200203	1459018	87	9.64	28.82	10.04	3.560	0.86241
5	R	3741	b1309203120713	1824012	19	0.10	0.96	0.30	0.940	0.83554
6	R	904	b0589203161503	457935	84	32.17	54.79	20.97	8.410	0.77295
7	R	2768	B3659203200401	1347283	114	24.70	39.27	10.89	7.720	0.67577
8	R	4624	B3819203270725	2217353	10	1.73	3.47	1.44	2.100	0.66582
9	R	257	b4829203260404	123706	45	22.24	33.40	12.74	5.720	0.64347
10	R	6029	b2109203170201	2909578	143	25.60	35.84	10.70	3.870	0.64210
11	I	792	B3299203240601	383631	394	8.00	28.73	10.27	2.550	0.41068
12	I	6411	B3959203310102	3116648	183	26.31	46.54	19.93	4.870	0.32079
13	I	2108	b0379203110303	1017680	239	22.01	41.24	14.21	3.580	0.28048

Total Microtrips: 13
Total Trip Time: 33.97 min
Total Base Time: 20.37 min
Mean Cycle Speed: 27.73 mph
Total Difference: 0.28048

Start Microtrips: 2
Start Trip Time: 4.60 min

Master Trace File (excerpt)

B28792031801	0	0.00	0.00	0	0
B28792031801	1	0.00	0.00	0	0
B28792031801	2	0.00	0.00	0	0
B28792031801	3	0.00	0.00	0	0
B28792031801	4	0.00	0.00	0	0
B28792031801	5	0.00	0.00	0	0
B28792031801	6	0.00	0.00	0	0
B28792031801	7	0.00	0.00	0	0
B28792031801	8	0.00	0.00	0	0
B28792031801	9	0.00	0.00	0	0
B28792031801	10	0.00	0.00	0	0
B28792031801	11	0.00	0.00	0	0
B28792031801	12	0.00	0.00	0	0
B28792031801	13	0.00	0.00	0	0
B28792031801	14	0.00	0.00	0	0
B28792031801	15	0.00	0.00	0	0
B28792031801	16	0.00	0.00	0	0
B28792031801	17	0.00	0.00	0	0
B28792031801	18	0.00	0.04	0	0
B28792031801	19	0.04	3.20	0	3
B28792031801	20	3.24	1.55	1	2
B28792031801	21	4.79	1.34	1	1
B28792031801	22	6.13	0.40	2	0
B28792031801	23	6.53	0.50	2	0
B28792031801	24	7.03	0.87	2	1
B28792031801	25	7.90	0.36	2	0
B28792031801	26	8.26	-0.36	2	0
B28792031801	27	7.90	-0.92	2	-1
B28792031801	28	6.98	-0.67	2	-1
B28792031801	29	6.31	-1.48	2	-1
B28792031801	30	4.83	-2.26	1	-2
B28792031801	31	2.57	-0.40	1	0
B28792031801	32	2.17	0.40	1	0
B28792031801	33	2.57	-1.05	1	-1
B28792031801	34	1.52	-0.31	1	0
B28792031801	35	1.21	1.18	1	1
B28792031801	36	2.39	0.52	1	1
B28792031801	37	2.91	0.56	1	1
B28792031801	38	3.47	-0.36	1	0
B28792031801	39	3.11	-0.13	1	0
B28792031801	40	2.98	-0.18	1	0
B28792031801	41	2.80	-0.18	1	0
B28792031801	42	2.62	-0.61	1	-1
B28792031801	43	2.01	-0.91	1	-1
B28792031801	44	1.10	-1.10	1	-1
B28792031801	45	0.00	0.31	0	0
B28792031801	46	0.31	2.73	1	3
B28792031801	47	3.04	1.88	1	2
B28792031801	48	4.92	-0.31	1	0
B28792031801	49	4.61	-1.01	1	-1

Master Microtrips File (excerpt)

MTRP NO	MTRIP ID	INIT REC	TIME (s)	AVG SPD (mph)	MAX SPD (mph)	STD SPD (mph)	MAX ACL (mph/s)	SPD/ACL FREQ -->														
1	B0849203100101	1	32	4.13	10.29	3.33	3.470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	B0849203100201	35	28	2.29	8.99	3.25	2.050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Unsorted Start DIFFSUMS File (excerpt)

Trip #	Trip ID	Start Diff	Start Time (sec)	Initial Idle (sec)
2	B28792031802	0.635317	248	30
4	B28792031901	0.876384	262	112
5	B28792031902	0.866611	241	113
7	B28792031904	0.655594	271	40
8	B28792031905	1.454049	50	40
9	B28792031906	1.454049	241	265
12	B28792031909	0.669565	205	28
14	B28792031911	1.454049	16	15
16	B28792031913	0.667696	286	31
17	B28792031914	1.454049	241	31
19	B28792032002	0.838230	241	80
22	B28792032005	0.835857	241	34
31	B28792032014	0.995160	68	19
35	B28792032103	1.454049	164	78
36	B28792032104	0.996096	122	35
37	B28792032105	0.820255	113	29
38	B28792032106	1.454049	33	29
39	B28792032107	1.454049	69	29
48	B28792032116	0.746645	214	26
51	B28792032119	0.965498	241	33
52	B28792032120	0.624720	261	33
56	B28792032201	0.754790	250	65
57	B28792032202	0.962310	267	142
62	B28792032304	0.702186	257	27
63	B28792032305	0.767728	295	50
65	B28792032307	1.097152	153	111
69	B28792032311	0.695920	241	38
71	B28792032401	0.791174	248	121
72	B28792032402	0.951929	240	38
73	B28792032403	0.954662	95	16
74	B28792032404	0.937454	151	22
76	B28792032406	0.983172	97	15
80	B28792032410	0.871525	241	20
82	B28792032412	0.814550	270	109
91	B28792032506	0.790892	242	102
99	b42092032406	1.065885	265	17
102	b42092032503	0.879350	251	25
105	b42092032601	0.787035	248	124
106	b42092032602	0.797378	245	19
108	b42092032702	0.685250	279	48
109	b42092032703	0.861200	300	18
113	b42092033001	0.917643	241	22
114	b42092033002	0.782114	260	37
115	b42092033101	0.752631	241	68
120	B16492031202	0.573831	253	20
121	B16492031203	0.958284	290	23
123	B16492031205	0.814033	284	15
124	B16492031206	0.787444	217	36

Sorted Start DIFFSUMS File (excerpt)

Trip #	Trip ID	Start Diff	Start Time (sec)	Initial Idle (sec)
778	b07292031404	0.464825	275	36
4402	b14392031201	0.481371	296	24
2369	b09892031101	0.495158	281	28
2404	B46792032805	0.519660	263	30
4826	b12392031801	0.520682	241	37
2865	b06692031601	0.521281	292	28
3740	b20892031401	0.523204	283	63
3321	B31792032801	0.524827	255	38
723	b08692031503	0.530022	234	18
1002	B44192040103	0.534939	259	37
2190	b05092031511	0.537994	290	15
521	b13092031603	0.551134	241	18
3779	b41092032303	0.558891	248	19
3514	b21092031901	0.560766	253	45
969	B44192032402	0.566265	241	17
4883	b24392032001	0.567987	296	15
4595	B32592032704	0.568013	289	64
3057	b03792031405	0.568537	260	68
1481	b31492032105	0.569526	279	34
2924	b37592032601	0.570456	269	20
695	b08692030904	0.571474	243	47
2695	b23692031701	0.571717	277	29
2894	b24792032301	0.572722	262	38
120	B16492031202	0.573831	253	20
3794	b41092032501	0.578166	272	19
3945	B39592032303	0.583963	257	16
2530	b29792031901	0.584501	265	23
2891	b24792031905	0.584670	283	24
4154	b46892032708	0.585328	285	61
3501	b21092031601	0.586425	266	33
2539	b29792032104	0.586927	297	28
2177	b05092031307	0.591083	241	23
1033	b01992030506	0.595122	298	57
4123	b27592032403	0.597135	270	15
128	B16492031402	0.599864	201	31
1810	B33892032602	0.603924	262	68
497	b13092031106	0.605395	241	60
3976	B39592032805	0.606008	222	37
3216	B43692040202	0.606599	271	16
951	b31592032703	0.608864	230	39
3942	B39592032202	0.610940	245	15
4765	b12392031201	0.610959	254	33
2874	b24792031701	0.611120	259	59
3229	b01092030603	0.613501	241	19
2556	b29792032504	0.615276	290	74
1708	B44792032701	0.617385	290	29
1692	b13992031804	0.617564	172	25
3128	B41992032404	0.617992	243	16

Sample SAFD File (excerpt)

```
-1,  0,       0.,  0.000000
-1,  1,   85261.,  1.260847
-1,  2,   45051.,  0.666216
-1,  3,   42617.,  0.630220
-1,  4,   45028.,  0.665885
-1,  5,   54506.,  0.806045
-1,  6,   72928.,  1.078464
-1,  7,   84194.,  1.245075
-1,  8,   72847.,  1.077274
-1,  9,   52070.,  0.770014
-1, 10,   34407.,  0.508811
-1, 11,   28515.,  0.421675
-1, 12,   30927.,  0.457348
-1, 13,   29242.,  0.432428
-1, 14,   15483.,  0.228970
-1, 15,    4045.,  0.059811
-1, 16,     846.,  0.012513
-1, 17,     209.,  0.003095
-1, 18,      10.,  0.000148
-1, 19,       2.,  0.000030
-1, 20,       0.,  0.000000
 0,  0, 1571948., 23.246101
 0,  1,  183311.,  2.710824
 0,  2,   62063.,  0.917789
 0,  3,   68662.,  1.015384
 0,  4,   79177.,  1.170879
 0,  5,  119259.,  1.763607
 0,  6,  202305.,  2.991707
 0,  7,  273327.,  4.041977
 0,  8,  255804.,  3.782844
 0,  9,  192051.,  2.840063
 0, 10,  134994.,  1.996299
 0, 11,  123501.,  1.826343
 0, 12,  179237.,  2.650568
 0, 13,  180888.,  2.674984
 0, 14,   95883.,  1.417932
 0, 15,   28464.,  0.420923
 0, 16,    4235.,  0.062622
 0, 17,     367.,  0.005429
 0, 18,      61.,  0.000902
 0, 19,       4.,  0.000059
 0, 20,       0.,  0.000000
 1,  0,   22694.,  0.335604
 1,  1,   52481.,  0.776101
 1,  2,   42242.,  0.624676
 1,  3,   55760.,  0.824591
 1,  4,   67585.,  0.999460
 1,  5,   92944.,  1.374465
 1,  6,  123790.,  1.830621
 1,  7,  128082.,  1.894091
 1,  8,  100224.,  1.482116
 1,  9,   64130.,  0.948365
 1, 10,   41544.,  0.614357
 1, 11,   33702.,  0.498393
 1, 12,   32319.,  0.477929
 1, 13,   28060.,  0.414948
 1, 14,   12576.,  0.185978
 1, 15,    3351.,  0.049558
 1, 16,     619.,  0.009147
 1, 17,     114.,  0.001682
 1, 18,      12.,  0.000177
 1, 19,       4.,  0.000059
 1, 20,       0.,  0.000000
```

Sample Ranked Cycles by DIFFSUM File

BEST OF SUB25 SERIES E CYCLES

RANK	CYCLE FILE	TOTAL DIFF	M-TRIPS	TIME (min)	MEAN SPD (mph)	MAX SPD (mph)	STOPS/MIN
1	E22-655.OUT;1	0.26886	5	31.17	41.16	74.06	0.160
2	E22-654.OUT;1	0.26886	15	33.08	27.26	74.06	0.453
3	E18-394.OUT;2	0.27601	13	33.97	27.73	74.06	0.383
4	E22-423.OUT;2	0.27675	10	36.72	27.34	74.06	0.272
5	E20-194.OUT;2	0.27822	16	42.50	25.31	74.06	0.376
6	E22-759.OUT;1	0.27868	10	37.28	28.05	74.06	0.268
7	E18-985.OUT;1	0.28068	10	36.88	27.89	74.06	0.271
8	E20-036.OUT;2	0.28072	8	33.40	41.82	74.06	0.240
9	E20-035.OUT;2	0.28072	11	35.68	27.41	74.06	0.308
10	E18-602.OUT;1	0.28245	10	33.88	27.96	74.06	0.295
11	E22-496.OUT;2	0.28251	13	35.37	27.04	74.06	0.368
12	E22-893.OUT;1	0.28297	11	36.08	27.42	74.06	0.305
13	E22-099.OUT;2	0.28476	9	35.43	28.30	74.06	0.254
14	E22-183.OUT;2	0.28482	10	39.63	26.57	74.06	0.252
15	E22-220.OUT;2	0.28504	16	36.43	26.87	74.06	0.439
16	E22-215.OUT;2	0.28511	11	35.02	27.39	74.06	0.314
17	E22-639.OUT;1	0.28514	13	37.55	27.35	74.06	0.346
18	E22-842.OUT;1	0.28579	9	36.20	27.13	74.06	0.249
19	E20-307.OUT;2	0.28658	10	37.83	28.08	74.06	0.264
20	E22-030.OUT;2	0.28688	6	37.25	49.00	74.06	0.161
21	E22-029.OUT;2	0.28688	13	34.57	26.99	74.06	0.376
22	E22-824.OUT;1	0.28694	8	34.60	26.98	74.06	0.231
23	E20-151.OUT;2	0.28797	13	32.72	27.68	74.06	0.397
24	E22-348.OUT;2	0.28830	12	34.75	27.41	74.06	0.345
25	E22-176.OUT;2	0.28855	10	37.43	27.36	74.06	0.267
26	E18-363.OUT;2	0.28891	9	35.50	27.06	74.06	0.254
27	E22-872.OUT;1	0.28892	17	44.00	26.00	74.06	0.386
28	E20-433.OUT;2	0.28901	11	39.30	27.85	74.06	0.280
29	E22-710.OUT;1	0.28909	12	35.97	27.50	74.06	0.334
30	E22-424.OUT;2	0.29040	13	37.28	28.13	74.06	0.349
31	E18-300.OUT;2	0.29060	12	33.35	28.46	74.06	0.360
32	E18-426.OUT;2	0.29070	9	36.53	28.24	74.06	0.246
33	E22-858.OUT;1	0.29135	14	35.30	26.51	74.06	0.397
34	E20-388.OUT;2	0.29197	10	32.50	28.51	74.06	0.308
35	E22-005.OUT;2	0.29205	11	36.80	26.99	74.06	0.299
36	E20-963.OUT;1	0.29227	8	35.20	27.94	74.06	0.227
37	E20-344.OUT;2	0.29268	8	31.75	19.45	74.06	0.252
38	E20-343.OUT;2	0.29268	10	36.58	28.49	74.06	0.273
39	E22-852.OUT;1	0.29295	11	36.53	27.02	74.06	0.301
40	E20-314.OUT;2	0.29321	15	33.50	27.40	74.06	0.448
41	E18-815.OUT;1	0.29351	6	36.35	27.49	74.06	0.165
42	E22-447.OUT;2	0.29356	12	32.03	27.26	74.06	0.375
43	E22-063.OUT;2	0.29372	12	33.53	28.86	74.06	0.358
44	E20-600.OUT;1	0.29412	12	35.93	27.89	74.06	0.334
45	E20-137.OUT;2	0.29414	13	43.45	25.58	74.06	0.299
46	E20-037.OUT;2	0.29458	13	35.60	27.50	74.06	0.365
47	E20-732.OUT;1	0.29465	12	38.15	27.50	74.06	0.315
48	E20-459.OUT;2	0.29488	13	42.30	26.55	74.06	0.307
49	E18-844.OUT;1	0.29523	12	37.60	28.05	74.06	0.319
50	E22-277.OUT;2	0.29560	9	35.83	28.50	74.06	0.251

Speed and Accel Trace PRN File (excerpt)

1	1	0.00	0.000	0	0
2	1	0.00	0.000	0	0
3	1	0.00	0.000	0	0
4	1	0.00	0.000	0	0
5	1	0.00	0.000	0	0
6	1	0.00	0.000	0	0
7	1	0.00	0.000	0	0
8	1	0.00	0.000	0	0
9	1	0.00	0.000	0	0
10	1	0.00	0.000	0	0
11	1	0.00	0.000	0	0
12	1	0.00	0.000	0	0
13	1	0.00	0.000	0	0
14	1	0.00	0.000	0	0
15	1	0.00	0.000	0	0
16	1	0.00	0.000	0	0
17	1	0.00	0.000	0	0
18	1	0.00	0.000	0	0
19	1	0.00	0.000	0	0
20	1	0.00	0.000	0	0
21	1	0.00	0.000	0	0
22	1	0.00	0.000	0	0
23	1	0.00	0.000	0	0
24	1	0.00	0.000	0	0
25	1	0.00	0.000	0	0
26	1	0.00	0.000	0	0
27	1	0.00	0.000	0	0
28	1	0.00	0.000	0	0
29	1	0.00	0.000	0	0
30	1	0.00	0.000	0	0
31	1	0.00	0.000	0	0
32	1	0.00	0.000	0	0
33	1	0.00	0.000	0	0
34	1	0.00	0.000	0	0
35	1	0.00	0.000	0	0
36	1	0.00	0.000	0	0
37	1	0.00	0.020	0	0
38	1	0.02	-0.020	0	0
39	1	0.00	0.400	0	0
40	1	0.40	2.910	1	3
41	1	3.31	2.660	1	3
42	1	5.97	2.000	2	2
43	1	7.97	0.690	2	1
44	1	8.66	1.340	2	1
45	1	10.00	2.440	2	2
46	1	12.44	1.320	3	1
47	1	13.76	0.980	3	1
48	1	14.74	0.090	3	0
49	1	14.83	1.790	3	2
50	1	16.62	1.660	4	2