EPA-AA-SDSB-87-08

Technical Report

Operational Characteristics Study
Columbus, Ohio Program

By

Glenn Thompson
Robin McCoy
Mark Hoekwater
Lisa Snapp
Alan Schuler

August 1987

NOTICE

Technical Reports do not necessarily represent final EPA decisions or positions. They are intended to present technical analysis of issues using data which are currently available. The purpose in the release of such reports is to facilitate the exchange of technical information and to inform the public of technical developments which may form the basis for a final EPA decision, position or regulatory action.

Standards Development and Support Branch
Emission Control Technology Division
Office of Mobile Sources
Office of Air and Radiation
U. S. Environmental Protection Agency

Table of Contents

List of Figures

List of Tables

Abstract

The Columbus, Ohio program of the Operational Characteristics Study (OCS) monitored the driving patterns of 47 private citizens in Columbus, Ohio over a nine month period from April 1983 to January 1984. These citizens were loaned one of four vehicles equipped with time, speed, distance, and temperature recording devices to gather a total of 251 days of in-use data. This report describes the details of this program, the results, and a comparison of the average trip statistics to the Federal Test Procedure used in emissions testing.

I. Introduction

The Federal Test Procedure (FTP) is the urban dynamometer driving cycle used for vehicle exhaust emission certification and fuel economy measurement. The cycle is based on a Los Angeles road route, designated the LA-4, which was selected by Los Angeles County and California State personnel in 1965 or 1966.[1] The FTP was developed about 1970, and consists of the LA-4, a 10 minute engine off period (soak), and a repeat of the first section (bag) of the LA-4.[2]

Many notable changes have occurred in the U.S. during the twenty years since the LA-4 was developed. Demographically, the population distributions have shifted toward increased urbanization. Particularly notable is the recent trend toward growth of small urban areas.[3] The gasoline shortage of the 1974-75 years initiated or accelerated the trend toward smaller vehicles. In addition, the Energy Conservation and Policy Act introduced Corporate Average Fuel Economy Standards for vehicle manufacturers, increasing the incentive for the production of fuel efficient vehicles.[4]

The accumulative effect of all of these changes on vehicle usage is unknown and was the primary objective of this program. Specifically, the program was designed to study the operational characteristics of vehicles while also comparing current average vehicle operation with the LA-4 (and hence the FTP).

II. Report Layout

The report will first summarize the main ideas and results from this program. The main body will then proceed with the program design, data collection and preparation, data analysis, results, and finally with some conclusions. The appendexes contain more voluminous and detailed materials including listings of several computer programs used to analyse the data, histograms of trip characteristics, graphs of the average trip statistics versus trip starting time, a summary of the questionnaire given to the participants, and an engine temperature analysis.

III. Program Design

In any program of this type the goal is to obtain as much vehicle use data as possible, without interference from the data collection technique. Past approaches have included operator surveys, mechanical recording devices, human observors in the monitored vehicle and "chase cars".

Questionnaire surveys of vehicle owner-operators can obtain large quantities of data at very low cost, but the accuracy and the detail of data can be very poor. This approach has primarily been used for, and is best suited for, collecting information such as annual miles traveled, commuting distances, and ratios of the number of hot start versus cold start trips.

Greater confidence and more detailed data may be obtained by using a mechanical recording device to provide a direct record of the vehicle operation. However, mechanical recorders have limited resolution and usually small data capacity. These devices have primarily been used to measure time in various speed regions or to sense and record maximum vehicle or engine speeds.

The chase car approach is notably different from the recorder approach because in this case the recording instrumentation is placed in a vehicle which follows or "chases" the actual vehicle of interest. Since a specially instrumented vehicle is used, the instrumentation can be quite extensive. Also, it can be controlled by the driver or a passenger in the chase car. The weakness of this approach is that the use data is really data on the use of the chase car, not directly the vehicle of interest. There is always some uncertainty about how well the chase vehicle mimics the driving characteristics of the followed vehicle. Sampling is also a problem because the chase car data must be weighted to reflect rush hour traffic densities or the operation of the vehicle must be adjusted to reflect these traffic patterns. Several chase car studies have been conducted to investigate the representativeness of the EPA test cycles.[5]

The final historical approach which has been used is installation of chase car type instrumentation directly in the vehicle of interest. In this case, an observer was sent in the vehicle to operate the instrumentation. This approach was used in an EPA study to develop driving cycles for heavy-duty trucks.[6] While this approach may be suitable for commercial vehicles in delivery use, it is at the very least inconvenient for passenger car use. In either case there is the potential problem that the presence of the observer may alter the behavior of the vehicle operator.

The approach chosen for this study was a logical extension of this previous technique. An instrumentation system was developed with sufficient capacity and intelligence so that an attendant operator was not required. Ideally this system would have been installed in the vehicle owned by the study participant, however, early experience with the system demonstrated that it was quite time intensive to install and to adequately test prior to use. Consequently it was decided to install the instrumentation in several vehicles which were then loaned to the study participants. These vehicles were similar in size and type to the vehicle owned by the study participant. This similarity of the vehicles would hopefully lead the participant to operate the vehicle as he/she would operate his/her own and therefore minimize any changes from "normal" vehicle use.

IV. Equipment

The program design required that the data recording equipment operate unattended in a vehicle for a minimum of several days. However, during most of this time the vehicle would probably be inactive and, therefore, the most important data, the vehicle speed information would be unnecessary. Therefore, the data acquisition system required the ability to adjust as a result of the system signals. A microprocessor provided this instrumentation control, replacing the human instrumentation operator necessary in previous programs.

The instrumentation system was designed and constructed by MB Associates (MBA) of San Ramon, California. The system was based on an RCA development board using the 1802 CMOS microprocessor. This microprocessor was chosen because of its low power requirements, typical of CMOS integrated circuits. In 1975 when the system was first designed, this was one of the very few CMOS microprocessors commercially available. The microprocessor controlled the collection of data from the speed, temperature, and other sensors, applied calibration factors to these data and monitored the recording of the data on a cassette tape.

The vehicle speed signal was generated by an optical encoder connected to the speedometer cable. The encoder pulses were sent to the interrupt request line of the microprocesor and were subsequently counted by the interrupt request subroutine. The microprocessor divided the number of pulses counted each second by a calibration factor to obtain the vehicle speed in mi/hr. This value was then modified by the data encoding system before being queued for the cassette tape.

Eight temperature channels were provided. Six of these channels were used in the program. Each sensor was a thermistor which was multiplexed to a single analog to digital converter. The temperature sensors were installed at the following locations: 1) behind or adjacent to the rear bumper, to measure the ambient temperature; 2) at the water pump, to measure the engine coolant temperature; 3) on the underside of the oil pan, to measure the engine lubricant temperature; 4) at or near the air cleaner intake, to measure the engine compartment temperature; 5) on the underside of the fuel tank, to measure the fuel tank temperature; and 6) at one other nonspecific location under the hood. Temperatures were recorded every minute during the time that the vehicles were running and throughout most of the time when the vehicles sat idle. When the vehicle had not been running for over twelve hours, the temperatures were recorded every twenty minutes.

A few things should be noted about these temperature sensor locations. A sensor attached to the rear bumper of a vehicle in operation will not provide a very accurate measurement of ambient temperature. The bumper is subject to solar heating on sunny days and rapid cooling from puddle splashes on rainy days. Therefore, the "bumper temperature" was not included in the temperature analysis. Improved fuel tank temperatures could have been obtained by inserting the thermistor inside the fuel tank. Unfortunately, this was not feasible. Therefore, an attempt was made at obtaining an approximate temperature profile by placing the sensor underneath the fuel tank, although this too is subject to road splash. The sensor on the underside of the oil pan may be in a similar situation, but due to the confined area where it is placed and the relatively large changes experienced in oil temperature it will not be as affected by outside elements as the sensor on the fuel tank is. Also, the "underhood" temperature may not be comparable between vehicles as the sensor location is not specifically defined.

One additional data input was used in the OCS program, a switch which was operated by the vehicle driver to indicate urban versus rural operation of the vehicle.

V. Data Collection

The general program approach was to instrument vehicles and then provide these vehicles as loaner cars to the participants of the study. Since vehicle size might effect the operational characteristics of the vehicle, the loan vehicles were selected to represent the diverse sizes of vehicles in use. The participants were selected randomly, but were always provided with a vehicle of approximately the same size as their personally owned vehicle.

The data collection phase of the program was conducted by Automotive Testing Laboratories (ATL) of East Liberty, Ohio under an EPA contract.[7]

A. Vehicle Selection

Full size, intermediate, compact and subcompact vehicles were initially planned for the following number of loans.

Vehicle Size	Number of Planned Loans
Full size	12
Intermediate	25
Compact	5
Subcompact	30
Total	72

The vehicle size classifications are those used in the DOE/EPA vehicle mileage guides.[8] The number of loans planned for each category was proportional to the total number of vehicles registered in that category. Therefore, the vehicle sampling process was designed to be a representative sample, by vehicle size, of the total in-use fleet.

During the progress of the program occasional difficulties occurred, primarily either intermittant electronic problems with one of the recorders or loss of battery power due to premature battery drainage. This caused a reduction in the total amount of data which could be obtained within the data collection contract funds. The following number of instrumented vehicle loans were actually used in the final analysis as valid loans.

Vehicle	Size	Number of Successful Instrumented Loans
Chevrolet Impala	Full size	12
Ford Fairmont	Intermediate	13
Chevrolet Chevette	Subcompact	22
Total		47

The loan period was between April 12, 1983 and January 11, 1984. Two different Chevettes were equipped, which resulted in a total of four vehicles used in this study.

B. Participant Selection

The participants in the study were selected randomly subject to the constraint that each participant receive a loan vehicle of approximately the same size as the vehicle owned by the participant. Lists were obtained for 1979 model year vehicles registered in all postal zip codes located within ten miles of zip code 43211. This was essentially Columbus, Ohio. From this list 2000 vehicle owners were sent a letter asking if they were interested in participating in an EPA sponsored test program. All those responding affirmatively, approximately 20 percent, were listed and numbered sequentially. A random number generator was then used to select numbers from this list which were then sequentially tabulated according to vehicle size.

C. The Vehicle Loan

Immediately preceding each loan the battery packs in the vehicle were replaced with freshly charged packs or they were recharged in the vehicle. The vehicle speed sensor was calibrated by driving the vehicle for one mile on a chassis dynamometer. Finally, the vehicle was driven over the first segment of the EPA urban test cycle.

After these preliminary steps the vehicle was delivered to the participant. The time of delivery was noted so that any data recorded while the vehicle was operated by the ATL employee could be separated in the analysis process.

The study participant was instructed to operate the vehicle in the same manner as the participant's own vehicle would be used. Only two additional requests were made; the operator was asked to maintain a fuel purchase log and to flip a switch to a rural position whenever the vehicle was driven in a rural area. For this study an urban area was defined by the Ohio Department of Transportation 1975 urbanized area limits. These urbanized area limits are developed in conjunction with U.S. Department of Transportation guidelines. All other regions were considered rural.

The distinction between an urban and rural area is often not clear and, therefore, several precautions were taken to attempt to improve the urban/rural switch data. First, the participant was given a map of the Columbus area with the boundary of the urbanized area clearly outlined on this map. If the participant's typical daily commute crossed the boundary, the participant was asked to note the boundary point on the map and to try to change the switch position at the precise boundary point. It was also pointed out to the participant that the boundary was approximated by the I-270 freeway circumscribing Columbus. The participant was asked to change the switch position at this freeway whenever the participant was entering or leaving the urban area but was not certain about the exact boundary. While there is always uncertainty about the accuracy of the switch data, Columbus was an ideal metropolitan area for participant accuracy. It is a nearly circular metropolitan area surrounded by rural farmland. The beltway is a good, easily recognized approximate boundary between the urban and rural regions. Finally, there are no long corridors of urbanization along any geographical boundaries.

D. Quality Control

Several steps were taken to maintain quality control, therefore minimizing the collection of data which might later prove to be useless. The initial system test verified that the instrumentation was correctly installed, monthly quality control tapes ensured the long term accuracy of the equipment and data inspection after each loan provided a frequent test that the instrumentation continued to function.

1. Initial System Test

After the instrumentation was first installed in a loan vehicle, the vehicle was operated by ATL personnel for about one week. This data tape was reviewed to ensure that the data appeared reasonable. Only after it was verified that the data appeared reasonable and it was demonstrated that the instrument and battery would survive approximately a week of unattended operation was the vehicle loaned to a participant.

2. Monthly Quality Control Test

The initial system test relied on the ability of ATL personnel to ascertain if the data appeared reasonable. Since the actual use of the vehicle was not accurately known, there was the possibility that some instrumentation drift or other problem might occur and might not be detected. To ensure that any problems of this nature would be detected a quality control data tape was made after the initial installation test and each subsequent month. The quality control data tape consisted of a driving record of the instrumented vehicle over the federal test procedure (FTP) cycle used for exhaust emissions measurement. This data tape was forwarded to EPA for analysis along with all other data. Since the FTP cycle is accurately known, comparison of the driven cycle to the theoretical cycle would detect even minute problems and would provide a method to correct the field data for any calibration drift should this be necessary. For all cases in this study, reasonable comparisons were found.

3. Data Inspection

Immediately after each vehicle loan, random segments of the data were displayed using an Apple microcomputer. These data were visually inspected by the ATL Project Manager to ensure that the instrumentation system continued to function properly. If the data appeared reasonable, the Apple was used to transcribe the data from the cassette tape used in the vehicle recorder onto floppy diskettes. The diskette and the original cassette tape were then transmitted to the EPA Motor Vehicle Emission Laboratory.

VI. Data Preparation

The purpose of the data preparation phase was to detect and correct data problems. Two types of problems were possible: a malfunctioning instrument might result in total loss of usable data, or relatively infrequent random errors might cause extensive data analysis difficulties. The data preparation phase attempted to remove or correct any data problems prior to the analysis of the data.

Analysis of data from an early pilot program indicated that the most significant data problem was the occasional loss of one byte of the recorded data. This problem was never clearly identified but is believed to result from a real time programming or hardware error in the data collection system.

After any cleaning or correction of the raw data, the information was converted into standard engineering units. Once converted, tests of the data, such as the ranges of parameter values or the rate of change of the parameters could be made to assess the reasonableness of the data. Infrequent errors could be corrected or if blocks of data were missing the entire data set could be rejected.

A. Initial Cleaning - The Missing Bytes Problem

The missing data byte problem occurs infrequently, about 0.01 percent of the data are missing. Yet it is potentially a very severe problem because one lost byte could result in many subsequent data being erroneously interpreted, causing a much higher effective data error rate. For example, a single error in a one minute string of speed data could cause all of the data for the remainder of the minute to be analyzed as speeds of several hundred miles per hour. Simply using such obviously illogical data would be undesirable and eliminating significant blocks of these data would seriously compromise the analysis.

The greatest obstacle in correcting the missing data bytes was that no error detection/correction code had been used in the original data collection. The only data decoding asset provided by the collection system was that a unique synchronization mark, an "FD", was provided at the beginning of each minute of recorded data. This "FD" was used as the first step toward an automated data correction system.

The data were reformatted so that all "FDs" were at the beginning of the lines of the data files. This reformatting was done with the Apple using a program written in BASIC. A copy of the program is included in Appendix A.

Reformatting emphasized the data patterns which facilitated recognition of blocks of data which contained one or more missing bytes. Each "FD" was followed by six bytes of identifier information followed by six temperature bytes. After this there might or might not be 60 seconds of speed data in the form of one speed byte pair each second. If any speeds appeared there should always be 60 seconds of data unless the vehicle engine stopped, in which case this would be indicated by an "FE" code. Consequently, by reformatting the data in lines of twelve bytes per line and forcing any "FDs", "FEs", or "FC" calibration codes to start a new line, any missing data would result in a short line and would be immediately obvious.

Unfortunately, the end of a diskette data file could also result in a break in the data which would not correspond to any of the identifier flags. This data break problem was resolved by transferring the data to the main computer used by EPA, the University of Michigan MTS system. This system can accommodate much larger files than the Apple. Therefore, all files of each vehicle loan were concatenated into a single large data file. The transfer of data to MTS was done by configuring the Apple as a data terminal using a binary communications program supplied by MTS.

Initially the reformatted MTS data files were printed and then scanned by data analysis personnel to determine the specific locations of missing data. This location could be unambiguously determined in virtually all cases. For example, in the case of sequential temperature data a dislocation of the data columns was usually apparent:

```
°°°   5C    41    57          4A

°°°   5C    42    4B             -----short line

°°°   5C    43    57          4C
```

The 57 is missing in the middle line and the 4B is shifted.

The identity of a missing speed datum was more subtle but also unambiguous. For example:

```
°°°   01 27   01 28           01 29

°°°   01 29  *29 01           30 01

°°°   30 01   31 01           29    -----short line
```

The "01" is missing at point * and all subsequent data are shifted.

Once the system evolved to this stage where unambiguous human editing could be based on pattern recognitions, than it was obvious that automated computer editing was also possible. Two programs were written, OCS.S-TMPFIX, and OCS.S-SPDFIX to correct temperature and speed missing data, respectively. This automation greatly increased the speed and accuracy of the data editing process.

TMPFIX searches for a line with a missing temperature datum which is both preceded and followed by a correct line. When this condition is found the two correct lines are interpolated to predict values for all temperatures in the line with the missing datam. These predicted temperatures are compared with the observed temperatures and the difference or "errors" are squared and then summed over all temperatures in the line. The last temperature datum is then shifted one position and the process is repeated. In this manner an array of the sums of the squares of the errors is computed for all possible positions of the blank or missing datum. The minimum of this array is then computed and the location of this minimum selects the location of the missing datum. Once the location is known, the preceding and following values are simply averaged to obtain the value which is inserted in the location of the missing datum. A copy of this program and the mathematics of the algorithm are given in Appendix A.

The temperature errors were randomly distributed and infrequent, therefore, the condition that the line with the error lie between two correct lines was generally met. If this condition was not met, the program placed flags in the margin of the output file but did not attempt a correction.

The speed corrections program used a very similar approach to the temperature correction program. Each speed datum required two bytes, the high order or most significant byte (MSB) and the lower order or least significant byte (LSB). Because of the scaling the MSB always had a value of 04 or less while the LSB could have any value. Consequently when low-high pairs appeared in consecutive odd-even positions, respectively, the data stream was known to be correct. Once a high-low pair occurred in a consecutive odd-even position, data loss must have occurred before this point. This logic was used to locate the datum known to be correct before any data loss and the first datum known to be shifted after data loss. Once these reference points were located there was usually only one intermediate point which was the obvious location of the missing datum. In this case the average of the two known reference points was inserted as the missing value. In some instances where the missing datum could occur at more than one point the average of the preceding and following data were tested as the possible missing datum at each point. The configuration which gave the minimum sum of the squares of the acceleration, that is the smoothest transition, was selected as the missing data point. If a clear minimum was not apparent, then the editing program did not introduce any change but wrote an error message to a separate file for later reviewing. A copy of the OCS.S-SPDFIX program is given in Appendix A.

After both editing programs were run, results of the programs were investigated by running the MTS-supplied software *APC (All Purpose Compare) which detected all differences between the original and the edited file. Whenever a change was made, the data before and after the modifications were reviewed to ascertain that the editing programs were functioning in a logical manner. Also, any problems that the programs could not correct were reviewed to see if a logical unambiguous resolution was possible.

The general philosophy toward all data correction was very conservative. Only those corrections that were obvious and unambiguous were made or accepted. This was generally possible because the errors were infrequent and most were completely missing single bytes in a known data sequence. Great care was taken not to introduce any biasing of the data by making frequent or extensive change.

B. Data Conversion

The conversion of the data into engineering units was done by the Fortran program OCS.S-PREPARE run on the MTS system. A copy of the program is given in Appendix A.

Conversion of the data into engineering units is slightly confusing because of the diverse encoding systems used by the recorder systems. Consequently each decoding system is discussed in conjunction with the type of data it was used to decode.

1. Identifier Line

A segment of identifier data appeared each minute unless the vehicle had not been used in the last eight hours, in which case the identifier segment appeared every twenty minutes. This segment always contained an "FD mark" followed by the date, time, and loan number. For example:

```
FD          38     13     46      01      00
"mark"      day     time          loan number
            38      1:46              100
```

All of the parameters on the FD line were recorded as binary encoded decimal, therefore, it was not necessary to perform any change in the numerical system or units.

The date encoding was the only frequent problem in this field. We intended to use the last two digits of the Julian calendar date so that the starting date of all loans, or at least all loans on any one vehicle, would have unique dates. Unfortunately, the software of the recording system was written to subtract 30 from the date at each hour change if the date value was greater than 30. Consequently, Julian 68 would be modified to 38 and then 08. Since this was considered to be a potential cause for confusion, these dates were corrected back to the correct Julian Values. This correction was made whenever necessary by the data analysis personnel using the general MTS Supplied EDIT programs.

2. Temperatures

Six temperatures always followed the identifier block. The information supplied by MBA with the recording units stated that the conversion to degrees Celsius was given by:

C = (H-57)

where H is the decimal value of the recorded hexidecimal (base 16) data. An example of the temperature decoding is:

Recorded temperature value	39	64	85	86	61	6D	6F
Conversion to decimal	57	100	133	134	97	109	111
Subtraction of 57	0	43	76	77	40	52	54

Calibration of the temperature sensors in a hot water bath indicated that the MBA conversion was not adequate for correct Celcius temperatures. Therefore, an additional empirically derived correction was applied. This correction was:

Correction = M * Sensor + B

where, when the sensor value was less than or equal to 90:
M = 0.82473
B = 0.4184
or, when the sensor value was greater than 90:
M = 0.51254
B = 28.847

3. Speed Data

The speed data always appeared in two byte pairs. The conversion, supplied by MBA was:

Convert both hexadecimal bytes to binary.

Append the three least significant bits of the most significant byte to bits 2 through 6 of the least significant byte. The two least significant bits of the least significant byte, bits 0 and 1 are the decimal fraction of the speed.

Convert the resulting binary number to decimal.

An example calculation follows:

	Most Sign. Byte	Least Sign. Byte
Original Hex Data:	01	27
Binary Notation:	0000 0001	0010 0111
Decoded Binary:	00101001.11	
Decimal mi/hr:	41.75	

4. Other Codes

In addition to the identifier, temperature, and speed data, an FC code could appear once near the beginning of the data and several FE codes could appear anywhere in the speed data. The FC code indicated that a calibration had occurred and the following speed data should be valid. This would logically occur only once in each loan file. Each file was checked to ensure that one and only one FC was found. All data before the FC were ignored.

The FE codes indicated an engine on/off change or a change in the urban/rural switch. The interpretation of these are:

FE	OA	ignition on urban
FE	OB	ignition off urban
FE	02	ignition on rural
FE	03	ignition off rural

The FE codes were somewhat troublesome, primarily because repetitive or oscillating values occasionally occurred. The source of these problems were never clearly identified but are believed to be switch bounce or are related to an unsuccessful attempt by MBA to include software to sense vehicle gear changes into the FE code.

C. Data Testing

Once the data were converted into engineering units each value was tested for reasonableness by the FORTRAN program, OCS.S-SCAN. The day, hour, and minute were checked to assure that no data were missing. A message was written to a separate file whenever the date and time did not follow sequentially. The loan identification number was checked to assure it was identical throughout the data. Another message was written to the error file when discrepancies were found. The temperature data were flagged if a change of greater than 10 degrees/minute occurred. Any speed point exhibiting an acceleration or deceleration greater than 7 mph/sec from the preceding point was flagged.

The program OCS.S-SCAN also tested for missing temperatures and speeds. If any temperatures were missing at this point in the analysis, the last (sixth) temperature would be read as a zero. Therefore, a check for a zero valued sixth temperature by the scanning program discovered any missing temperatures. A similar test was made to detect the presence of less than 60 speeds in each minute in which any speeds occurred. A Copy of OCS.S-SCAN is given in Appendix A.

In a typical loan file, approximately 100 possible data problems were observed. Each of these possible problems were reviewed by a data analyst. A few problems were traced to obvious errors which had escaped the initial editing. These data were corrected if the correct data could be unambiguously determined. In most cases the data values appeared unusual but not obviously erroneous. These values were not changed because strong emphasis was placed on not biasing the data toward the expectations of the reviewer. For example, an acceleration rate might seem higher than expected but could be plausible, particularly at low speed. In addition, temperatures often changed abruptly near engine on and engine off segments. The number and percentage of these instances were small, typically between .005 and .01 percent. This very small number should not have any observable effect on the final analysis. Any significant sequences of questionable data were noted or "flagged" so that if any unusual statistics occurred from these data the source could be identified and later traced if desired.

In a few cases, significant amounts of data were missing and, of course, these could not be reconstructed from the information in the file. In this case the interval of missing data was determined and the original cassette data tape was scanned to attempt to locate the data. Generally the information was found. It was then processed through all of the analysis steps as a "partial loan" and then inserted into the original loan data. These losses probably occurred in the original transcription of the data from cassette to the diskettes or from a marginal quality region of the diskette.

In one instance, the data were clearly illogical and could not be reconstructed. In this case the speed signal often changed rapidly from zero to speeds of about 50 mph and then back to zero. It was decided that the problem was most likely a poor connection in the speed sensing area. The entire loan was voided. A check of subsequent loans showed that the speed sensor completely failed on the next loan and was then replaced. A second instance also occurred when the speed sensor apparently failed. This entire loan was also voided.

Upon reduction of the data, some loans were voided because of apparent battery failure. In some instances, battery failure occurred at the end of the loan period, while other times it occurred fairly early in the week. It was decided to void all tests in which the battery failed within three days. If the battery failure occurred after three or more days, the loan was not voided, but any data occuring on and after the day of failure was disregarded.

VII. Data Reduction

The reduction phase of the data analysis reduced the voluminous basic data files into a much more manageable volume of descriptive parameters which could then be statistically analyzed. A "trip" was chosen as the primary concept of the reduced data and therefore most of the reduced data parameters described "trip" characteristics.

The analysis was done with the FORTRAN program OCS.S-STATS.FOR, and the FORTRAN program OCS-CMP.FOR, which are given in Appendix B.

A. Trip Parameters

The following parameters were computed to describe each trip and the results were stored in the output files XXX.TRIPLOG (XXX represents the loan identification number).

Trip Parameters	
SAVDAY	Day number at beginning of trip
TRIPCT	Sequential number of trip
DAYTRP	Sequential number of trip during day
SAVTIM	Time at beginning of trip
TYPE	1 = urban, 2 = combination, 3 = rural
START	1 = hot, 2 = cold
OFFCT	Time car is off before trip (minutes)
AVESPD	Average speed (total distance/total time)
DISTNC	Distance traveled during a trip (miles)
STOPCT	Number of stops
AVEDBS	Average distance between stops
ONCT	Length of trip (seconds)
TOTSS	Total stopped seconds, speed < 4 mph
GOSC	Total going seconds, speed $\geq$ 4 mph

Originally a trip was defined simply as the interval between engine "on" and engine "off." Preliminary analysis of the data showed that this definition of a trip may not reflect actual conceptions of trip driving characterstics. In the original case, a car could be driven for a series of "drives" with short engine off times between the "drives" and have each "drive" count as a separate trip. In reality, if the engine off times are short enough, it might be more appropriate that the whole series of "drives" should be counted as one total trip. For example, a person who drives to the post office, stops only long enough to buy a roll of stamps and then drives on to work, would most likely consider this as one trip with a "quick" stop rather than two separate trips.

In light of this, the definition of a trip was generalized. A program was written that revised the trip data file by combining sequential trips with very short engine off times between trips. The engine off time was a variable in the program, allowing for analysis and comparison using different trip definitions. The program combines trips in such a manner that a trip with an engine off time less than or equal to the desired time, is combined with the previous trip to form one new trip. The new total trip time and stop time will both include the engine off time, since this time is now considered as part of the trip. In this study, engine off times of 0 (original data), 5, 10, 15, and 20 minutes were analyzed. A comparison of the number of trips per day, time of trip, average speed, number of stops, and hot to cold start ratios using these five trip definitions is included in the results section. Engine off times longer than ten minutes may entail significant hot soak emissions, which should not be overlooked when calculating trip statistics for comparison to the urban driving cycle. Ten minutes was also considered as a reasonable upper time limit for a "quick stop." Therefore, unless otherwise noted, the rest of this report assumes engine off periods exceeding ten minutes as the delineator between trips.

B. Stop Parameters

Knowing the number of stops made during a trip, along with the total time spent while stopped, is useful for developing and verifying dynamometer test cycles. Obviously, the end of each trip was considered as a stop. However, just adding the number of zero mile per hour (mph) occurances to this "stop count" could cause inaccuracies. For example, some drivers make "rolling stops" without ever coming to a complete stop. Also, in a congested traffic situation, drivers will frequently stop and then "creep-up" to the car in front of them. Occasionally, this movement will even entail a small, short burst of speed. Counting all of these physical stops within a nominal stop condition would be misleading. Therefore, a stop was defined as any time the vehicle slowed from above 10 to below 4 mph, and then returned to a speed greater than or equal to 10 miles per hour. The end of a trip was also counted as a stop.

A noticeable amount of time and distance can be covered while in these stop conditions. Therefore, bursts of speed between 4 and 10 mph contributed towards "go time" and "go distance", although the vehicle was still considered to be in the same stop. "Stop time" (stop distance) was counted any time the vehicle speed went below 4 mph. Figure 1 gives an example of how these speeds and times interact within one stop condition.

The following data were computed for the intervals between each stop and stored in the output files XXX.STOPLOG.

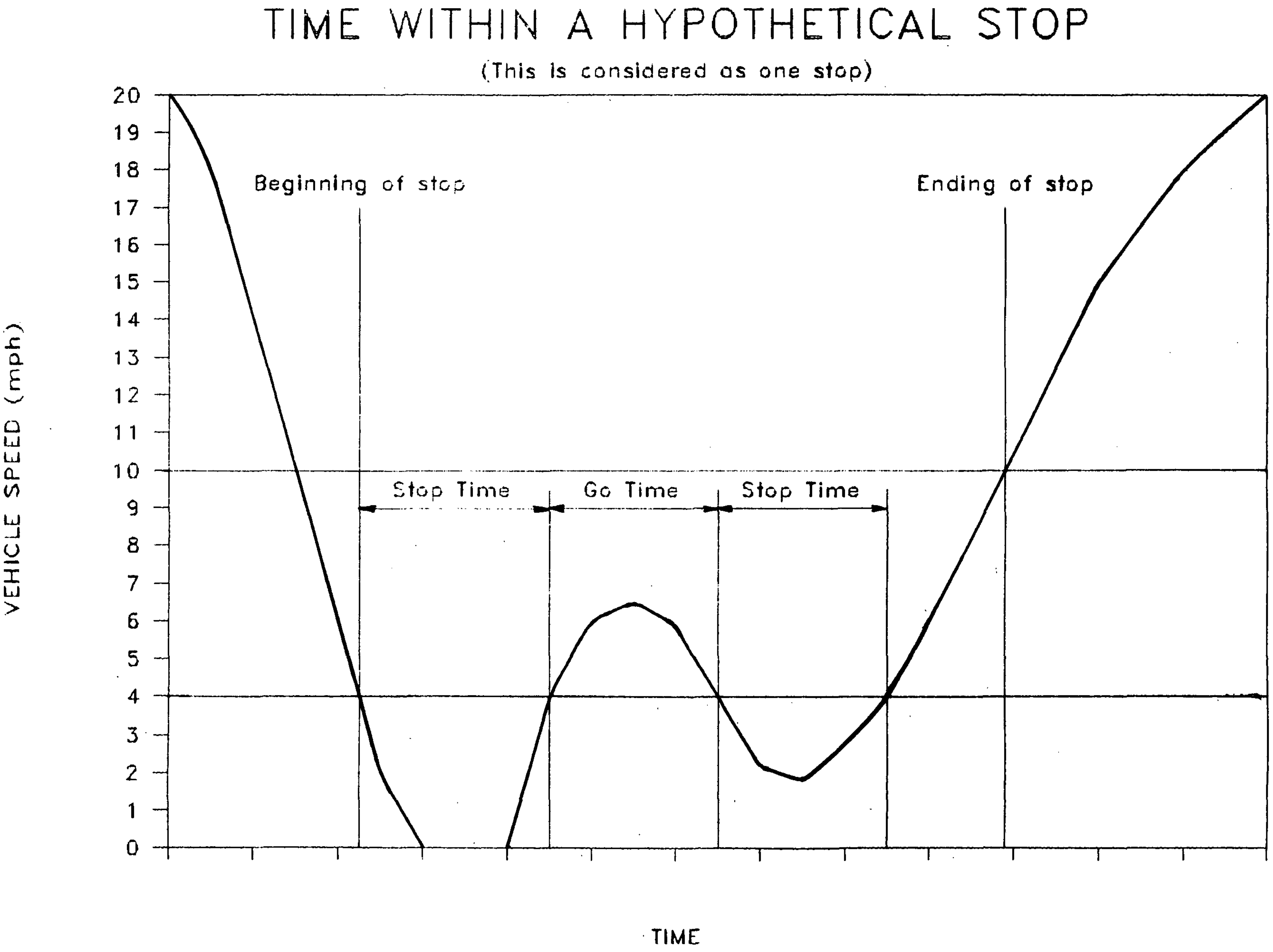

TIME WITHIN A HYPOTHETICAL STOP
(This is considered as one stop)
Beginning of stop
Ending of stop
Stop Time
Go Time
Stop Time
VEHICLE SPEED (mph)
TIME
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

Figure 1

Stop
Parameters

SAVDAY	Day number
TRIPCT	Sequential number of trip
DAYTRP	Sequential number of trip per day
TIME	Time at end of stop
STOPCT	Sequential number of stop during trip
STOPSC	Length of stop (seconds)
STOPDI	Distance travelled when "stopped," speed < 4 mph
GODI	Distance travelled when "moving," speed ≥ 4 mph

C. Final Data Verification

A key purpose of this study was to compare current trip parameters to the LA-4 test cycle used in exhaust emission certification. Since it would be undesirable to report differences between the OCS data and the LA-4 that are due to instrumentation error, an additional data check was made for each vehicle loan. This check was made after generating the vehicle triplog, but required an additional step to be taken before each vehicle loan.

Before the vehicle was presented to each participant, the vehicle was operated over the first 505 seconds of the LA-4. The final data verification was a comparison between the analysis of the results from these dynamometer test periods to the results of applying the data analysis programs to the equivalent interval of the theoretical test cycle. For example, the trip statistics for the first 505 seconds of the LA-4 are:

Average Speed (mi/hr)	Trip Dist (mi)	Number of Stops	Avg. Dist. Between stops (mi)	Total Trip time (sec)	Time In Stops (sec)	Time In Motion (sec)
25.6	3.59	6	0.72	505	109	396

Each vehicle loan was checked to verify that it contained a "signature" of statistics very similar to these at approximately the time the data sheet supplied by ATL stated that the dynamometer test was conducted.

D. Trip Statistics File

After final verification, the trip statistics from each loan were added to the aggregate trip statistics file, OCS.D-TRIPS, using the Fortran program, OCS-CMP.FOR. At this time several additional columns were created in this file to facilitate subsequent analysis. The most important of these was a loan number column that retained the identity of loan and a column which indicated whether the data in the file were computed from a vehicle trip which occurred when the vehicle was operated by the participant, whether it came from a road trip when the vehicle was operated by ATL personnel, or if it came from the dynamometer test. Columns relating to the combining of short trips were also added so that the operator could check for the reasonableness of the trip combining methodology. This collection of the data into a single file with multiple labeling columns was an ideal format for multiple analyses using the MTS statistics program package MIDAS.

E. Description of Analysis

The total collected data base was very extensive. Therefore, it was essential to extract simple statistics from the data which described the important parameters of the data. A logical method of organizing the data was to consider the characteristics of observed vehicle trips. These trip parameters can then be related to each other and used for different analysis as suits each individual, such as comparing the OCS results with the parameters of the LA-4. The following trip parameters were investigated: trip type (urban, urban and/or rural), trip distance, trip duration (including time moving, and time stopped), time since last trip, trip speed, number of trips per day, number of stops during trip, and the number of observed hot to cold trip starts (assumed engine temperature at start of the trip).

Two trip speeds were calculated in this analysis: the average trip speed, and the average driving speed. The average trip speed is defined as the total distance traveled during the trips of interest, divided by the total duration of these trips. Since this speed includes the effects of stop time, it is more descriptive of the entire trip condition (i.e., trips with many long stops will have lower average trip speeds than trips with only a few short stops). The average driving speed describes the speeds seen during actual vehicle movement. It is defined as the total distance traveled during the trips of interest, divided by the total time moving (excludes total time stopped). By comparing these speeds together, the effects of stop duration can be seen.

All of the analysis described here considers only those data obtained when the vehicle was being operated by the loan participant. Furthermore, since receiving or surrendering the instrumented vehicle in the middle of the day could influence the number or type of trips made with the vehicle, only the data from days in which the participant had the vehicle for the entire day are included. This resulted in 251 days of in-use vehicle operation data from the 47 different loans. Statistics were calculated for urban trips only, and for urban and/or rural (UAR) trips, to characterize driving habits in both cases. The urban only criteria limited the number of acceptable data to 42 vehicle loans. The UAR analysis utilized all 47 vehicle loans.

As previously mentioned, the temperature data may not be highly accurate. However, a temperature analysis was performed to develop representative time-temperature histories for five of the six vehicle temperatures monitored. The goal was to develop characteristic patterns for vehicles beginning at ambient conditions, going through a full warmup, and experiencing a hot soak followed by cooldown to ambient again. The trips chosen for analysis therefore were those that were felt to be isolated from the possible carryover effects from previous trips. The trips needed to be long enough for the measured temperatures to reach a somewhat steady level. The soak after the trip needed to be long enough for the temperatures to return to their pre-trip levels. Therefore the criteria for choosing a trip for temperature analysis were: 1) an uninterrupted soak of at least six hours prior to the trip, 2) a trip length of at least 15 minutes, and 3) an uninterrupted soak of at least six hours after the trip. Twenty-nine trips were found meeting these conditions, of which ten trips were chosen at random. These are outlined in Table 1.

The lengths of the ten trips chosen ranged from 15 to 34 minutes. For the first 15 minutes, where data for all trips existed, the temperatures measured were averaged across all ten trips to obtain values for a typical trip. For the remainder of the times, the average change in temperatures from minute to minute for the remaining trips were averaged, and this change was applied to the temperature for the previous minute. In this iterative fashion, the temperatures for the typical trip were determined. This continued until the trip reached a length of 29 minutes after which ony two of the original ten trips remained.

VIII. Trip Statistics Results

The analysis of the trip data are broken down into four areas: discussion of the general trip statistics and how they vary by changing the trip definition; comparison to the LA-4 trip statistics; discussions of the distribution of the various trip parameters; and comparison of the trip statistics for different trip starting times. A summary of the results from the temperature analysis is in Section IX. A more detailed reporting of the temperature analysis is included in Appendix H.

A. General Trip Statistics vs. Trip Definition

Tables 2A and 2B show the statistics for trips separated by engine off times exceeding 0, 5, 10, 15, and 20 minutes. Table 2A is for urban trips only, while Table 2B is for UAR trips (urban and/or rural). Both tables include hot trip start to cold trip start ratios for cold start definitions of at least 1, 4, 6, 8, and 12 hour engine off times.

Table 1

Trips Selected for Temperature Analysis

Vehicle	Loan #	Date	Time of Trip	Length Prior to to Trip (hrs/min)	Length of Trip (min)	Length of Soak After Trip (hrs/min)
83 Chevy Impala	211	5/24/83	7:19	8:03	20	11:06
83 Chevy Impala	230	7/28/83	7:38	11:12	29	8:40
83 Chevy Impala	234	8/18/83	7:20	14:28	15	8:16
83 Chevy Chevette	315	6/7/83	7:20	11:31	28	8:47
83 Chevy Chevette	344	10/3/83	7:20	18:54	34	8:55
83 Chevy Chevette	361	12/5/83	7:40	62:57	33	8:32
83 Ford Fairmont	443	10/3/83	7:40	15:04	26	8:48
83 Ford Fairmont	468	12/26/83	21:03	7:56	17	10:46
83 Ford Fairmont	468	12/28/83	8:05	10:49	23	9:11
84 Chevy Chevette	559	12/1/83	8:49	11:07	15	6:53

TABLE 2A
MEAN VALUES FOR URBAN ONLY TRIPS

		Engine Off Time*				
CHARACTERISTIC		0 MIN.	5 MIN.	10 MIN.	15 MIN.	20 MIN.
1.	# of trips/day**	6.6	5.4	4.7	4.3	3.9
2.	Time since last trip (hrs)	3.3	4.0	4.6	5.1	5.5
3.	Average trip speed (mph)	21.1	20.1	18.5	17.1	15.9
4.	Ave. driving speed (mph)	28.8	28.8	28.8	28.8	28.8
5.	Distance of trip (miles)	3.3	4.0	4.6	5.1	5.6
6.	# of stops during trip	6.0	7.4	8.5	9.4	10.2
7.	Distance btwn stops (miles)	0.55	0.55	0.55	0.55	0.55
8.	Total time of trip (min)	9.3	12.0	15.1	18.0	21.0
9.	Total time stopped (min)	2.5	3.6	5.4	7.3	9.4
10.	Total time moving (min)	6.8	8.4	9.7	10.7	11.6
11.	# of trips analyzed	1434	1158	1000	900	828
12.	Hot start/cold start ratios					
	a. 1 hr min for cold start	1.4	0.94	0.69	0.52	0.40
	b. 4 hr min for cold start	3.5	2.7	2.2	1.9	1.7
	c. 6 hr min for cold start	4.8	3.7	3.1	2.7	2.4
	d. 8 hr min for cold start	5.2	4.1	3.4	3.0	2.7
	e. 12 hr min for cold start	11.7	9.2	8.0	7.1	6.5

TABLE 2B
MEAN VALUES FOR URBAN AND/OR RURAL TRIPS

		Engine Off Time*				
CHARACTERISTIC		0 MIN.	5 MIN.	10 MIN.	15 MIN.	20 MIN.
1.	# of trips/day	7.2	5.8	5.0	4.5	4.2
2.	Time since last trip (hrs)	3.2	3.9	4.5	5.0	5.3
3.	Average trip speed (mph)	25.1	23.8	22.2	20.6	19.3
4.	Ave. driving speed (mph)	32.7	32.7	32.7	32.7	32.7
5.	Distance of trip (miles)	4.4	5.5	6.3	7.0	7.5
6.	# of stops during trip	6.0	7.5	8.6	9.5	10.3
7.	Distance btwn stops (miles)	0.73	0.73	0.73	0.73	0.73
8.	Total time of trip (min)	10.5	13.8	17.0	20.3	23.4
9.	Total time stopped (min)	2.4	3.7	5.5	7.5	9.5
10.	Total time moving (min)	8.1	10.0	11.5	12.8	13.8
11.	# trips analyzed	1795	1443	1255	1131	1047
12.	Hot start/cold start ratios					
	a. 1 hr min for cold start	1.4	0.93	0.68	0.52	0.40
	b. 4 hr min for cold start	3.7	2.7	2.3	1.9	1.7
	c. 6 hr min for cold start	5.0	3.8	3.2	2.8	2.5
	d. 8 hr min for cold start	5.4	4.1	3.5	3.0	2.7
	e. 12 hr min for cold start	11.9	9.4	8.0	7.1	6.5

---

* Engine Off Time = the minimum time an engine off period must exceed before a trip is considered ended. Sequential operations with engine off periods less than or equal to these amounts were included in the same trip.

** For calculating this parameter, we have excluded all days where any non-urban trip occurred. (The 251 days was therefore reduced to 188 urban days.)

A comparison between trip definitions shows the expected trends. As the allowed engine off time within a trip increases, the mean values for trip duration, total stopped time, total time moving, distance travelled, and time since last trip, all increase. Also with increased engine off time, the number of trips made per day, and the average trip speed, decrease. The decrease in average trip speed is expected since this speed includes stopped time.

The hot to cold start ratios also decrease with the increasing engine off time within a trip. This decrease reflects the combining of trips that were previously considered as hot start, into longer trips. The ratio of hot trip starts to cold trip starts shown in Tables 2A and 2B vary by a factor of approximately 3.6 depending on the cold trip start definition, and a factor of 2.0 depending on the trip definition. There is little difference between the UAR data set and the urban only data set.

Evaluating trips separated by a 10 minute minimum engine off time, participants made an average of 4.7 urban (5.0 UAR) trips per day, with a typical trip taking 15 (17) minutes and covering 4.6 (6.3) miles. These trips included 8.5 (8.6) stops, which consumed 36 (32) percent of the total trip time.

B. Distribution of Trip Statistics

Histograms of the individual trip statistics for the urban only trips and UAR trips are included in Appendixes C and D, respectively. The histograms are for the 10 minute minimum engine off time between trips, and show the average speeds, distances traveled, number of stops, and total, moving, and stop times. Histograms of the average trip speed, trip distance, time since last trip, and the number of trips per day are reproduced in Figures 2, 3, 4 and 5, respectively (urban trips only). As seen in these figures, the distributions are skewed. In the cases shown, most participants made slower trips with shorter distances travelled, than indicated by the average values in Table 2A. Most participants also made fewer trips per day with less time between trips, than indicated by the average values.

The number of short distances travelled during a trip are higher than what may be expected. Figure 3 shows that ten percent of the trips were less than half a mile in length, and the data in Appendix C shows that 4.6 percent were less than 0.1 miles. A closer analysis indicated that 2.3 percent of the trips went less than 0.01 miles (53 feet). Although no follow through was made to the loan participants, these extremely short trips likely represent such things as driving from one store to another at a shopping mall or switching car position on a driveway. Most (81 percent) of the 0.1 to 0.5 mile trips were part of "longer trips", where the preceeding and/or following engine off stops were less than one hour in duration.

AVERAGE TRIP SPEED (0-64 mph) **each X = 2 trips**

URBAN TRIPS

```
LEFT-END     TOT%  COUNT

 0.           4.4    44 +XXXXXXXXXXXXXXXXXXXXXX
 2.0000       2.2    22 +XXXXXXXXXXX
 4.0000       3.7    37 +XXXXXXXXXXXXXXXXXXX
 6.0000       4.2    42 +XXXXXXXXXXXXXXXXXXXXX
 8.0000       5.4    54 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000       6.0    60 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000       9.3    93 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 14.000      10.4   104 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 16.000       9.7    97 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18.000       7.9    79 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 20.000      10.3   103 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 22.000       7.7    77 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 24.000       5.6    56 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 26.000       3.5    35 +XXXXXXXXXXXXXXXXXX
 28.000       2.5    25 +XXXXXXXXXXXXX
 30.000       1.9    19 +XXXXXXXXXX
 32.000       1.5    15 +XXXXXXXX
 34.000       1.4    14 +XXXXXXX
 36.000        .5     5 +XXX
 38.000        .3     3 +XX
 40.000        .3     3 +XX
 42.000        .3     3 +XX
 44.000        .3     3 +XX
 46.000        .2     2 +X
 48.000        .1     1 +X
 50.000        .2     2 +X
 52.000      0.       0 +
 54.000      0.       0 +
 56.000      0.       0 +
 58.000      0.       0 +
 60.000        .1     1 +X
 62.000        .1     1 +X

 TOTAL      100.0  1000  (INTERVAL WIDTH= 2.0000)
```

Figure 2

TRIP DISTANCE (0-25 miles) **each X = 2 trips**

URBAN TRIPS

```
 LEFT-END     TOT%  COUNT

  0.          10.5   105 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  .50000       8.5    85 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  1.0000       8.1    81 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  1.5000       7.6    76 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2.0000       5.2    52 +XXXXXXXXXXXXXXXXXXXXXXXXXX
  2.5000       5.3    53 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
  3.0000       6.0    60 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3.5000       4.0    40 +XXXXXXXXXXXXXXXXXXXX
  4.0000       6.5    65 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4.5000       4.2    42 +XXXXXXXXXXXXXXXXXXXXX
  5.0000       2.8    28 +XXXXXXXXXXXXXX
  5.5000       4.5    45 +XXXXXXXXXXXXXXXXXXXXXXX
  6.0000       3.6    36 +XXXXXXXXXXXXXXXXXX
  6.5000       2.3    23 +XXXXXXXXXXXX
  7.0000       2.0    20 +XXXXXXXXXX
  7.5000       1.5    15 +XXXXXXXX
  8.0000       1.1    11 +XXXXXX
  8.5000       1.5    15 +XXXXXXXX
  9.0000       1.3    13 +XXXXXXX
  9.5000       1.1    11 +XXXXXX
  10.000       1.9    19 +XXXXXXXXXX
  10.500        .9     9 +XXXXX
  11.000       1.3    13 +XXXXXXX
  11.500       1.3    13 +XXXXXXX
  12.000       1.2    12 +XXXXXX
  12.500        .9     9 +XXXXX
  13.000        .8     8 +XXXX
  13.500        .3     3 +XX
  14.000        .3     3 +XX
  14.500        .4     4 +XX
  15.000        .5     5 +XXX
  15.500        .2     2 +X
  16.000        .3     3 +XX
  16.500        .2     2 +X
  17.000        .2     2 +X
  17.500        .2     2 +X
  18.000      0.       0 +
  18.500        .2     2 +X
  19.000        .4     4 +XX
  19.500        .1     1 +X
  20.000        .1     1 +X
  20.500        .1     1 +X
  21.000        .1     1 +X
  21.500      0.       0 +
  22.000        .1     1 +X
  22.500      0.       0 +
  23.000      0.       0 +
  23.500      0.       0 +
  24.000      0.       0 +
  24.500      0.       0 +

                .4     4 > 25.000
  TOTAL      100.0  1000  (INTERVAL WIDTH= .50000)
```

Figure 3

TIME SINCE LAST TRIP (0-24 hours)

each X = 3 trips

URBAN TRIPS

```
LEFT-END      TOT%   COUNT

 0.           25.0    250 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000       15.7    157 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000        8.6     86 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000        7.2     72 +XXXXXXXXXXXXXXXXXXXXXXXX
 2.0000        3.6     36 +XXXXXXXXXXXX
 2.5000        3.4     34 +XXXXXXXXXXXX
 3.0000        3.1     31 +XXXXXXXXXXX
 3.5000        2.2     22 +XXXXXXXX
 4.0000        2.1     21 +XXXXXXX
 4.5000        2.7     27 +XXXXXXXXX
 5.0000        1.1     11 +XXXX
 5.5000        1.0     10 +XXXX
 6.0000         .5      5 +XX
 6.5000         .5      5 +XX
 7.0000         .2      2 +X
 7.5000         .4      4 +XX
 8.0000        1.6     16 +XXXXXX
 8.5000        2.1     21 +XXXXXXX
 9.0000        1.1     11 +XXXX
 9.5000        1.6     16 +XXXXXX
 10.000        1.7     17 +XXXXXX
 10.500        1.1     11 +XXXX
 11.000        1.1     11 +XXXX
 11.500        1.3     13 +XXXXX
 12.000         .9      9 +XXX
 12.500         .9      9 +XXX
 13.000         .6      6 +XX
 13.500         .8      8 +XXX
 14.000        1.3     13 +XXXXX
 14.500         .4      4 +XX
 15.000         .2      2 +X
 15.500         .2      2 +X
 16.000         .4      4 +XX
 16.500         .3      3 +X
 17.000         .2      2 +X
 17.500         .5      5 +XX
 18.000         .1      1 +X
 18.500         .1      1 +X
 19.000         .4      4 +XX
 19.500         .4      4 +XX
 20.000         .1      1 +X
 20.500         .1      1 +X
 21.000         .1      1 +X
 21.500         .4      4 +XX
 22.000         .1      1 +X
 22.500         .3      3 +X
 23.000         .4      4 +XX
 23.500         .1      1 +X

               1.8     18 > 24.000
 TOTAL       100.0   1000  (INTERVAL WIDTH= .50000)
```

12 [illegible] 93 [illegible]
24 hrs [illegible] 14

Figure 4

NUMBER OF TRIPS MADE PER DAY (0-17)

each x = 1 day

URBAN DAYS

```
MIDPOINT   HIST%  COUNT

   0.0       9.0    17 +XXXXXXXXXXXXXXXXX
   1.0       6.4    12 +XXXXXXXXXXXX
   2.0      11.2    21 +XXXXXXXXXXXXXXXXXXXXX
   3.0      14.4    27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.0      10.1    19 +XXXXXXXXXXXXXXXXXXX
   5.0      14.4    27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
   6.0       8.0    15 +XXXXXXXXXXXXXXX
   7.0       8.0    15 +XXXXXXXXXXXXXXX
   8.0       8.0    15 +XXXXXXXXXXXXXXX
   9.0       4.3     8 +XXXXXXXX
  10.0       2.7     5 +XXXXX
  11.0       1.1     2 +XX
  12.0       0.0     0 +
  13.0       0.5     1 +X
  14.0       0.0     0 +
  15.0       0.5     1 +X
  16.0       1.1     2 +XX
  17.0       0.5     1 +X

  TOTAL            188  (INTERVAL WIDTH= 1.0000)
```

Figure 5

C. Comparison to the LA-4

Table 3 shows the trip statistics from the LA-4 along with the mean and median trip statistics for the 1,000 Columbus urban trips (using the ten minute off time trip definition). Comparing the mean and median Columbus values first, in all cases the median values are smaller than the mean values. This is because the distributions are skewed, with a majority of the participants making fewer, shorter (in both time and length), slower speed trips than indicated by the mean, (as shown in section B).

In general, the Columbus trips were substantially shorter than the LA-4. Considering the mean values they went 62 percent of the distance in 66 percent of the time with 45 percent of the number of stops. However, the average speeds are approximately equal, 18.5 versus 19.6 mph, and the average Columbus trip went further between stops, 0.55 miles compared to 0.41 miles. Considering the large urban spread with many connecting freeways found in L.A., the longer trip distances used in the LA-4 can be expected. The LA-4 is also based on morning rush-hour traffic [2] which is usually characterized by shorter distances between stops. Therefore, the differences between the LA-4 and the Columbus statistics appear reasonable.

The Code of Federal Regulations weights the cold start and hot start exhaust emission values by 0.43 and 0.57, respectively, to obtain a composite value.[9] This gives a hot to cold start ratio of 1.3, or 1.3 trips starting with the engine hot for every trip starting with a cold engine. As shown in Table 3, the hot start to cold start ratios from the Columbus data are 0.69 and 2.2 for a one hour and four hour cold start definition, respectively. Interpolating between these values, the LA-4 cold start definition would correspond to 2.2 hours of off time. Based upon the temperature results to be discussed in Section IX, this amount of time would allow peak vehicle temperatures (above ambient) to decline by 60-75 percent.

D. Trip Statistics by Trip Starting Time

Figures 6 and 7 (urban only and UAR trips, respectively), show the number of trips starting within given hour increments. The hour values shown represent the beginning of the hour period. Morning and afternoon rush hours can be observed centered at the 7-8 a.m. and 4-5 p.m. periods. As might be expected, the number of trips made after 1 a.m. are very few. While the trip statistics for these early morning hours are included in the graphs in Appendixes E and F, the values may not be representative of average driving habits for these hours.

Graphs of the average distance travelled, the number of stops made, the average trip speed, the time since last trip,

TABLE 3

OCS DATA OVERVIEW -- URBAN TRIPS*

CHARACTERISTIC	MEAN	MEDIAN	LA-4
1. # of trips/day**	4.7	5.0	N/A
2. Average speed (mph)	18.5	16.6	19.6
3. Distance of trip (miles)	4.6	3.4	7.4
4. Distance btwn stops (miles)	0.55	0.48	0.41
5. # of stops during trip	8.5	7.0	19.0
6. Time since last trip (hrs)	4.6	1.5	N/A
7. Total time of trip (min)	15.1	12.4	22.9
8. Total time stopped (min)	5.4	2.8	5.3
9. Total time moving (min)	9.7	8.1	17.5
10. Hot start/cold start ratio			1.3
a. 1 hr min. for cold start	0.69		
b. 4 hr min. for cold start	2.2		

---

* Trips can include an engine off period lasting up to and including 10 minutes. Engine off periods longer than 10 minutes imply the ending of the original trip.

** For calculating the number of trips per day, we have excluded all days where any non-urban trip occurred.

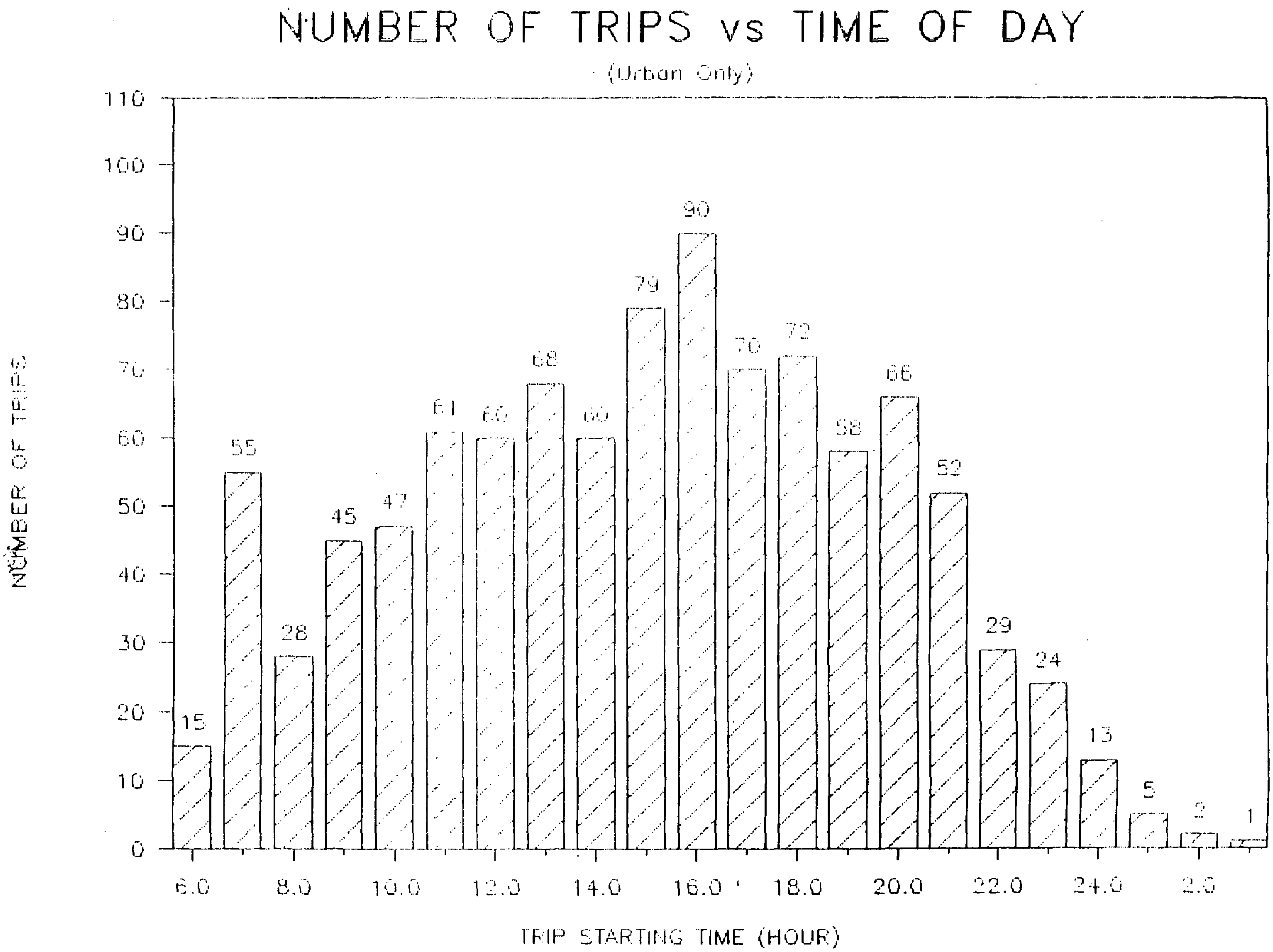

NUMBER OF TRIPS vs TIME OF DAY
(Urban Only)
NUMBER OF TRIPS
0
10
20
30
40
50
60
70
80
90
100
110
15
55
28
45
47
61
60
68
60
79
90
70
72
58
66
52
29
24
13
5
2
1
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
2.0
TRIP STARTING TIME (HOUR)

Figure 6

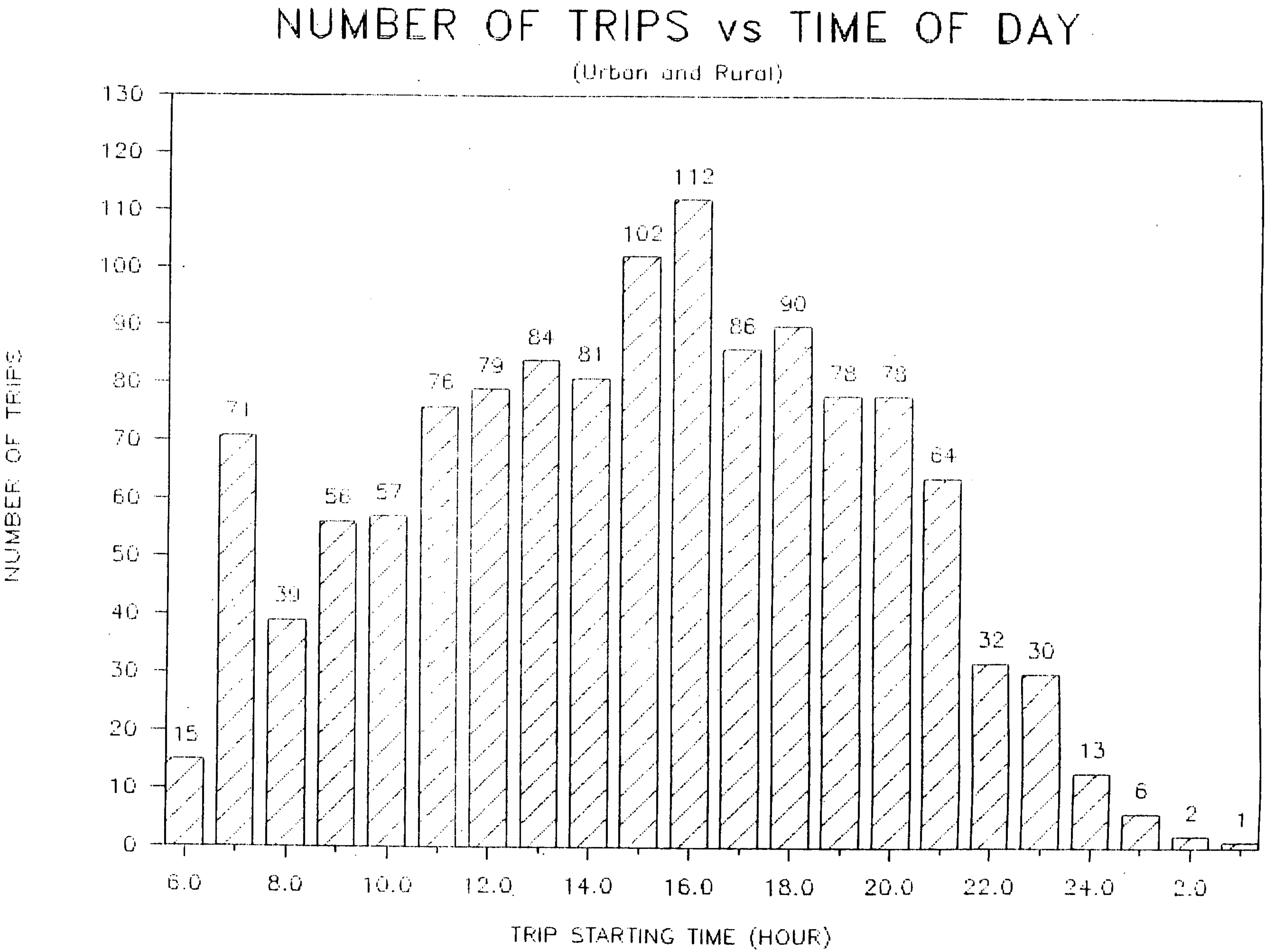
 NUMBER OF TRIPS vs TIME OF DAY
(Urban and Rural)
NUMBER OF TRIPS
TRIP STARTING TIME (HOUR)
0
10
20
30
40
50
60
70
80
90
100
110
120
130
6.0
8.0
10.0
12.0
14.0
16.0
18.0
20.0
22.0
24.0
2.0
15
71
39
56
57
76
79
84
81
102
112
86
90
78
78
64
32
30
13
6
2
1

Figure 7

and the trip duration are shown in Figures 8, 9, 10, 11, and 12, respectively, for urban only trips (these graphs can also be found in Appendix E). The distance travelled, trip speed, and time since the last trip show pronounced morning time effects. The value of 6.5 miles travelled during the morning rush hour (7-8 a.m.) is fairly close to the 7.4 miles travelled in the LA-4 (which simulates rush hour conditions). The average trip speed drops from 25 mph during the 6 a.m. hour to 13.6 mph during the 9 a.m. hour. This drop in speed is probably due to an increase in traffic. The time since the last trip is large (10-12 hours) for the morning trips and decreases from there. However, this is to be expected.

IX. Summary of Temperature Analysis

This section summarizes the temperature analysis. A more detailed analysis including temperature profiles for trips taken during a sumer day, winter night, and winter day is included in Appendix H.

Figure 13 shows the temperature profile for the typical trip. The water pump has the most rapid increase in temperature, approaching its peak temperature of 45°C (113°F) in 15 minutes. The oil pan also reaches this temperature seven minutes later. The air cleaner and fuel tank temperatures barely change. In all the data analyzed the fuel tank temperature, measured by a skin sensor on the bottom of the fuel tank, never appears to appreciably deviate from ambient temperatures. Given the inconsistency of this result with other in-use fuel tank temperature measurements, it appears that the skin sensor did not adequately measure internal fuel tank temperature. Therefore, the only use made of the fuel tank temperature data has been to consider it a surrogate for ambient temperature against which to compare the other temperature data during vehicle cool down.

Figure 14 shows the temperature profile for the first 45 minutes after engine shut-off. The water pump temperature has the largest increase, jumping 20°C to a peak level of nearly 70°C (158°F) in the first ten minutes of the soak. The oil pan reaches a peak temperature of 50°C (122°F). The temperature at the air cleaner rises markedly, almost 30°C.

The temperature profile of the complete soak is shown in Figure 15. The water pump, oil pan, air cleaner, and underhood temperatures all exhibit decay down to ambient temperature. The fuel tank temperature rises slowly through the soak period, apparently tracking a rise in the ambient temperature (all but one of the ten trips occured in the morning).

Beginning at twenty minutes after the soak begins, exponential decay curves were fit to the water pump, oil pan,

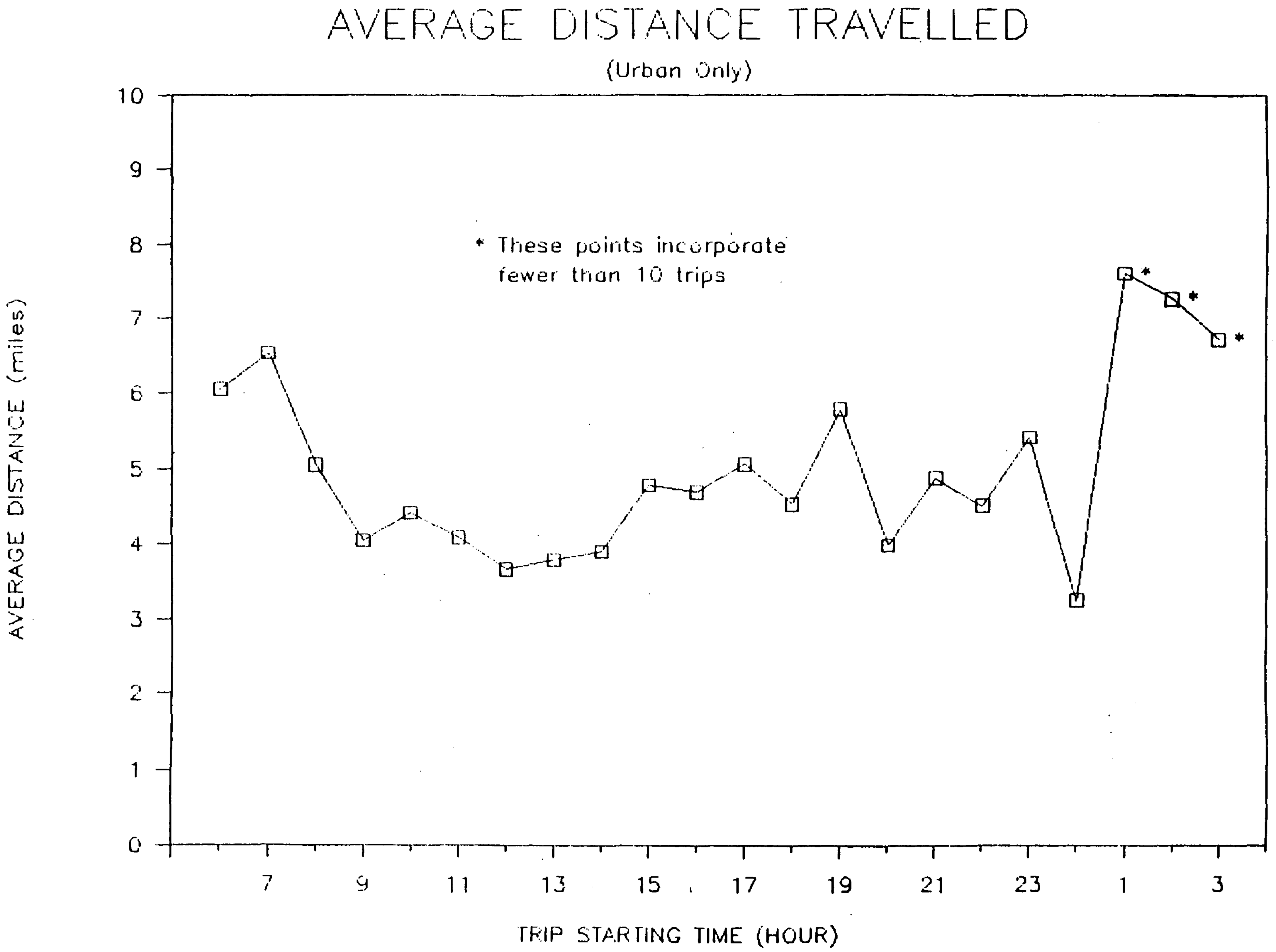

AVERAGE DISTANCE TRAVELLED
(Urban Only)
* These points incorporate fewer than 10 trips
AVERAGE DISTANCE (miles)
0
1
2
3
4
5
6
7
8
9
10
7
9
11
13
15
17
19
21
23
1
3
TRIP STARTING TIME (HOUR)
*
*
*

Figure 8

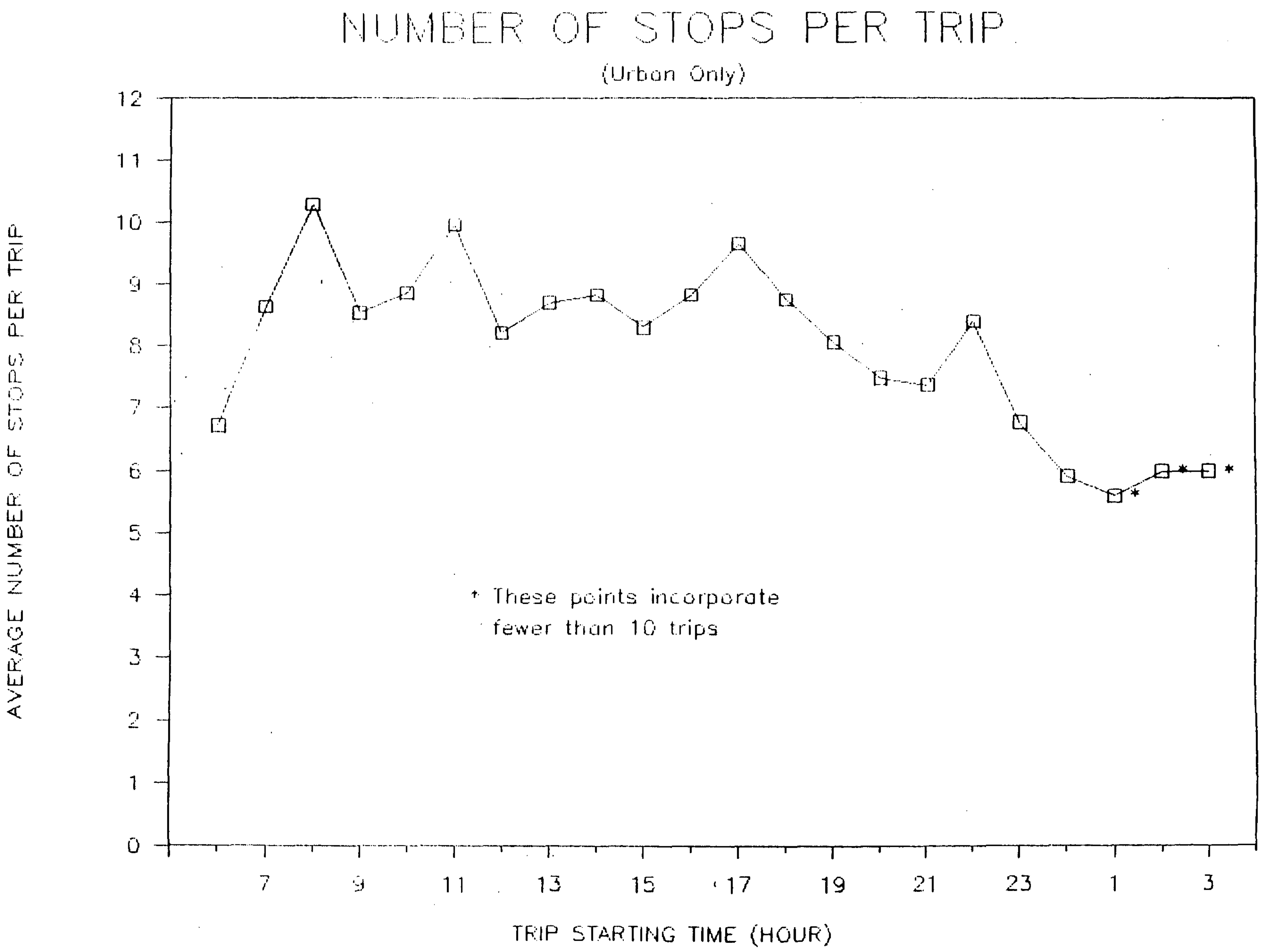
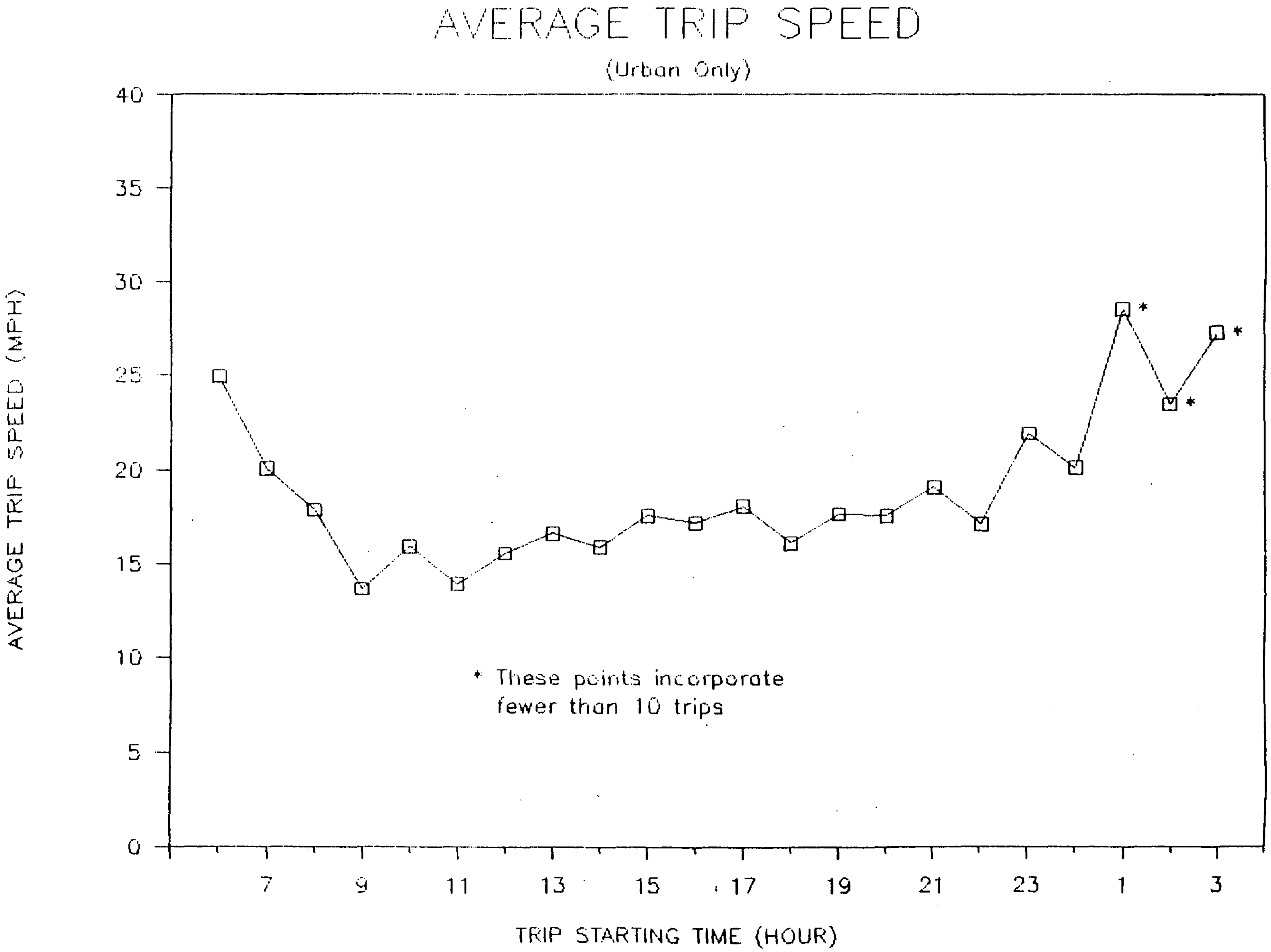

NUMBER OF STOPS PER TRIP
(Urban Only)
AVERAGE NUMBER OF STOPS PER TRIP
0
1
2
3
4
5
6
7
8
9
10
11
12
7
9
11
13
15
17
19
21
23
1
3
TRIP STARTING TIME (HOUR)
* These points incorporate fewer than 10 trips
*
*
*

Figure 9


AVERAGE TRIP SPEED

(Urban Only)

Figure 10

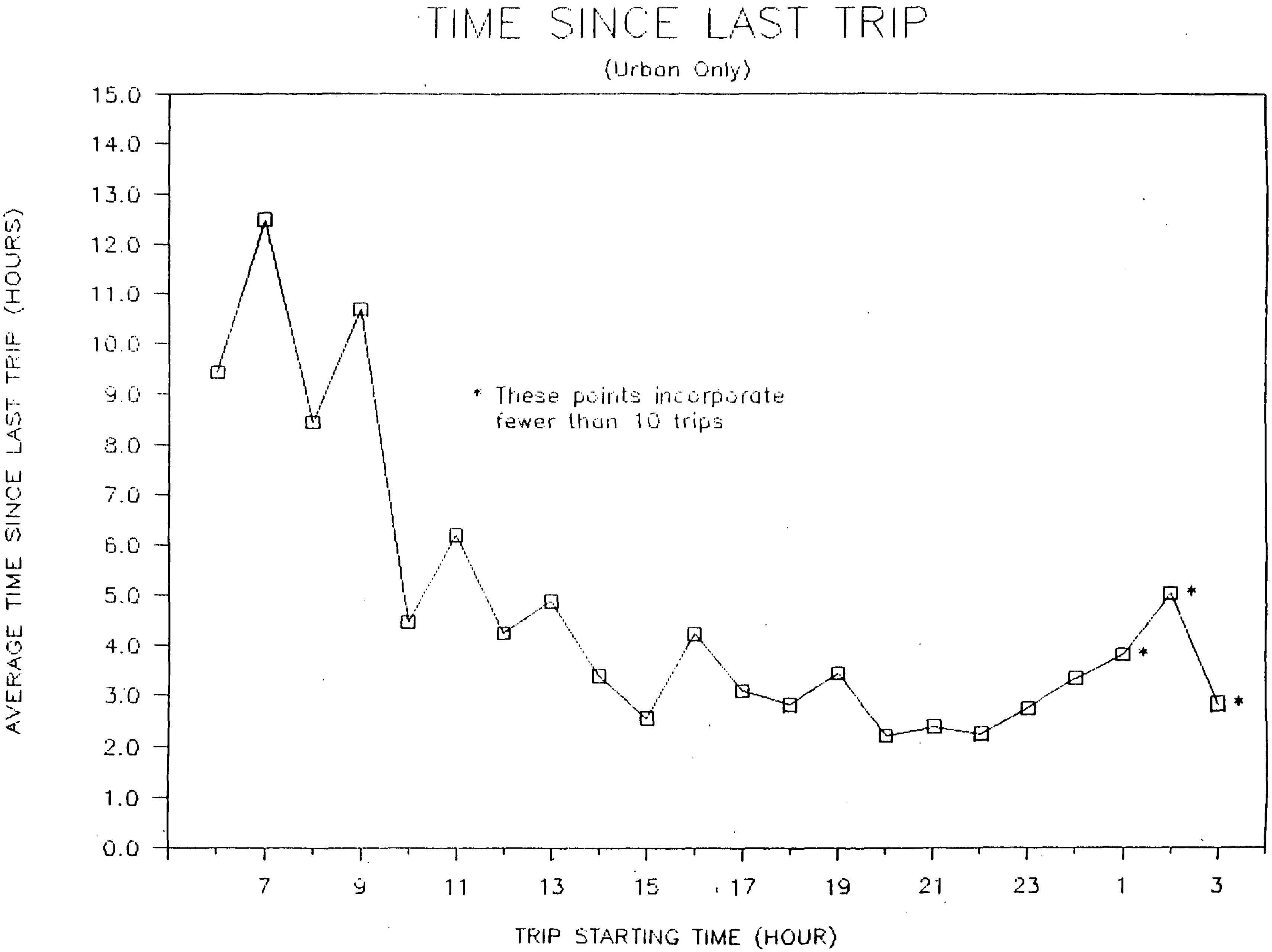

TIME SINCE LAST TRIP
(Urban Only)
AVERAGE TIME SINCE LAST TRIP (HOURS)
15.0
14.0
13.0
12.0
11.0
10.0
9.0
8.0
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
7
9
11
13
15
17
19
21
23
1
3
TRIP STARTING TIME (HOUR)
* These points incorporate fewer than 10 trips
*
*
*

Figure 11

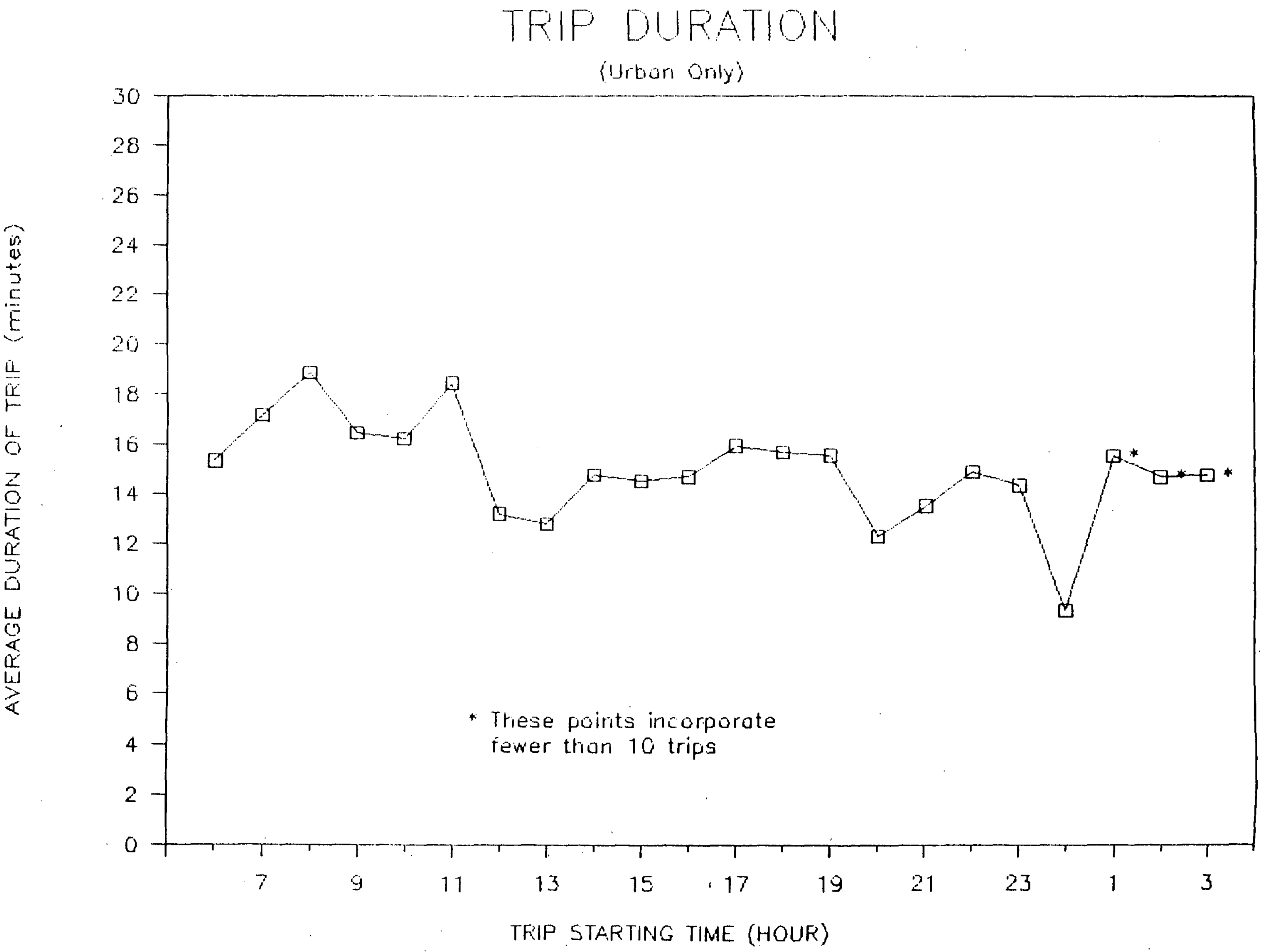

TRIP DURATION
(Urban Only)
AVERAGE DURATION OF TRIP (minutes)
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
7
9
11
13
15
17
19
21
23
1
3
TRIP STARTING TIME (HOUR)
* These points incorporate fewer than 10 trips
*
*
*

Figure 12

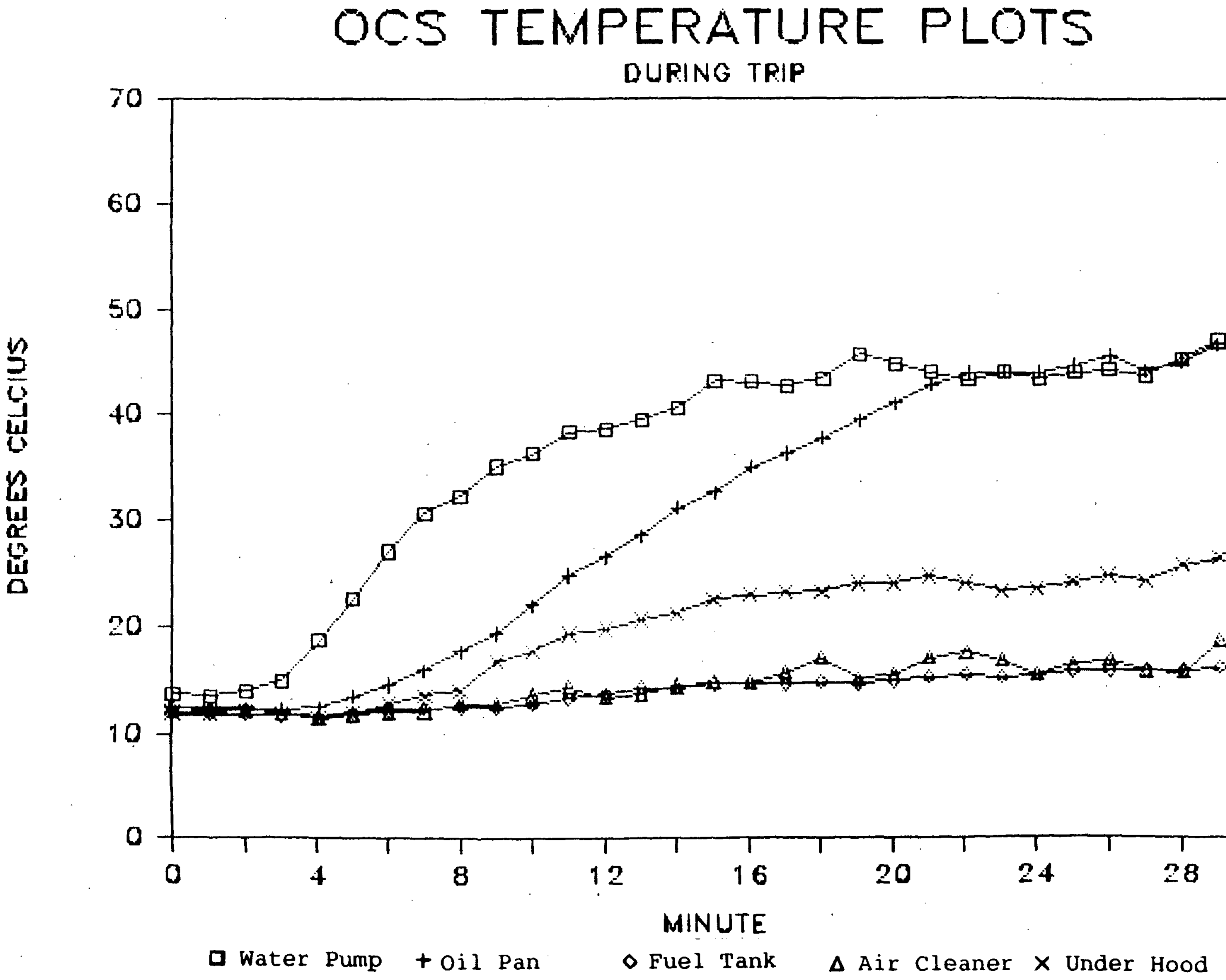

OCS TEMPERATURE PLOTS
DURING TRIP
DEGREES CELCIUS
70
60
50
40
30
20
10
0
0
4
8
12
16
20
24
28
MINUTE
□ Water Pump
+ Oil Pan
◇ Fuel Tank
△ Air Cleaner
× Under Hood

Figure 13

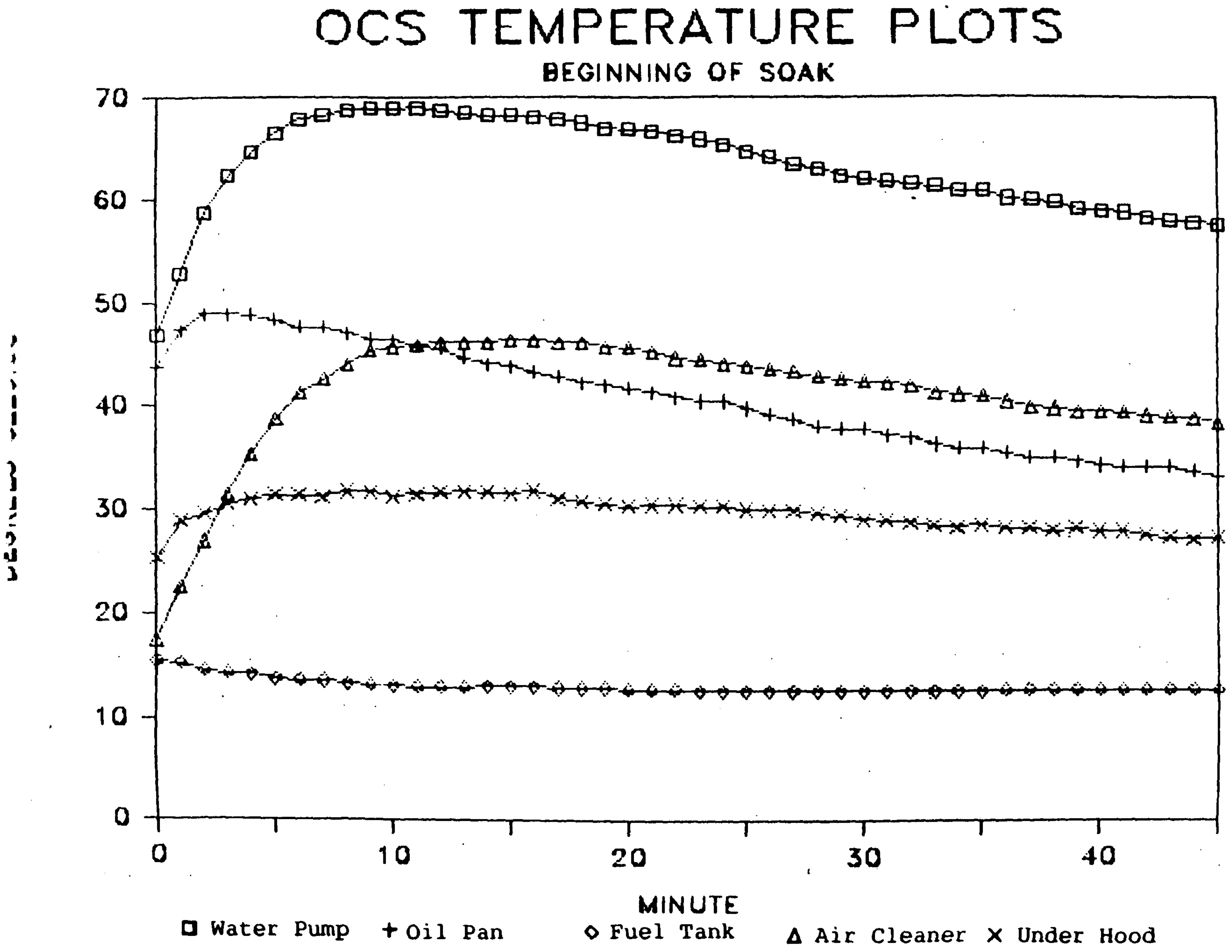

OCS TEMPERATURE PLOTS
BEGINNING OF SOAK
70
60
50
40
30
20
10
0
0
10
20
30
40
MINUTE
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure 14

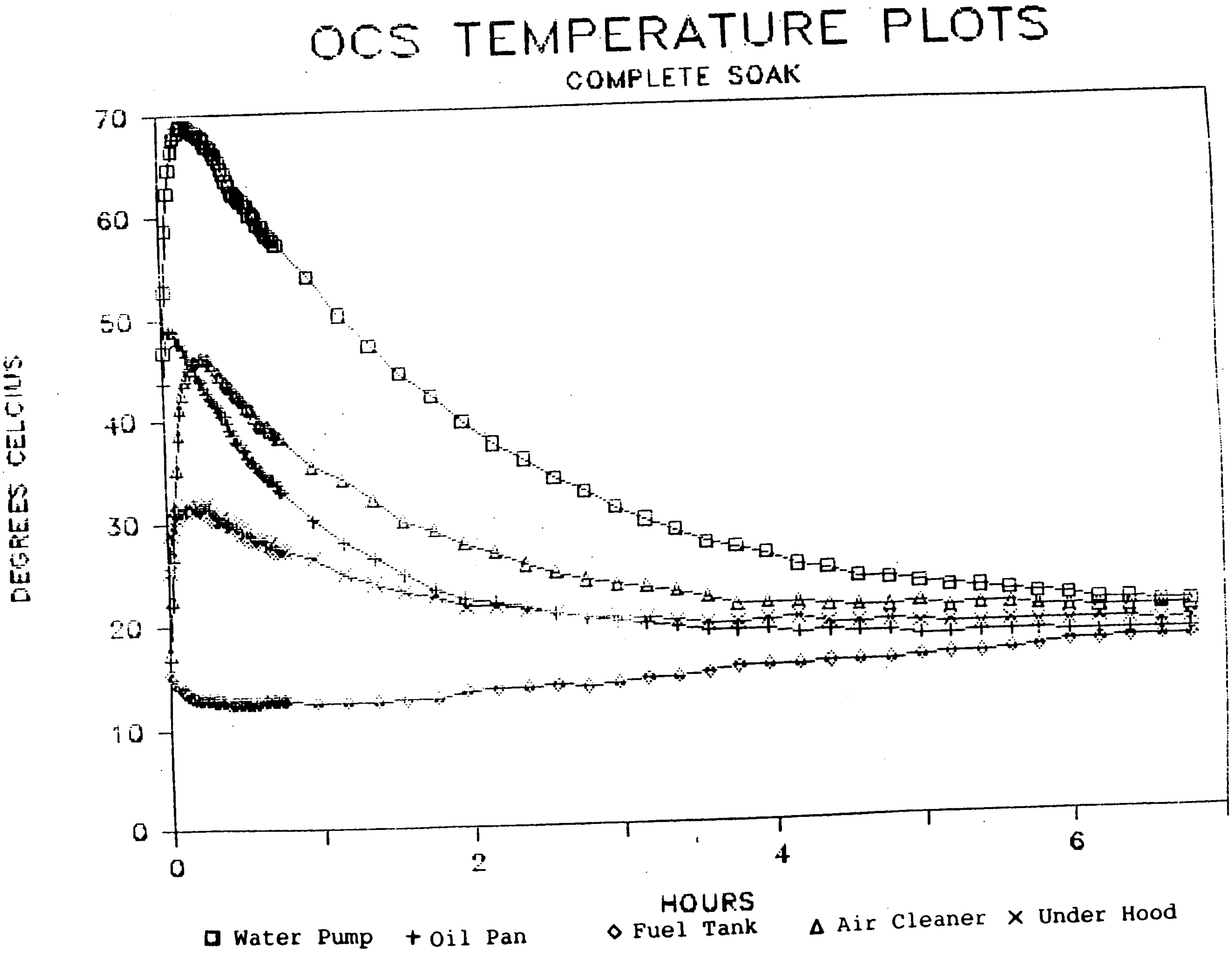

OCS TEMPERATURE PLOTS
COMPLETE SOAK
DEGREES CELCIUS
70
60
50
40
30
20
10
0
0
2
4
6
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure 15

air cleaner, and underhood temperatures. These were first adjusted by subtracting the fuel tank temperature, in order to model decay to ambient temperature. The resulting decay constants are .007, .011, .008, and .007 (1/minutes) for the water pump, oil pan, air cleaner, and underhood temperatures, respectively. To put these values in perspective, Table 4 shows the time required for the difference between the peak soak temperatures and ambient temperatures to be reduced to 50 percent, 20 percent, 10 percent, 5 percent, and 1 percent of their original values for the decay constants of .007 and .011 (1/minutes). These may be useful in determining when a vehicle is experienceing a "hot start" versus a "cold-start".

X. Conclusions

Two hundred and fifty-one days worth of in-use vehicle speed, time, distance, and temperature data were monitored and analyzed from participants driving in the Columbus, Ohio area. The data were catagorized by urban, and Urban and/or Rural (UAR) trips, with an engine-off period exceeding 10 minutes separating consecutive trips. The average participant made 4.7 urban (5.0 UAR) trips per day, going 4.6 (6.3) miles in 15 (17) minutes. The majority of urban trips (indicated by median values) went a smaller distance in less time than indicated by the respective mean values.

The urban trips were considerably shorter and less time consuming than the EPA urban dynamometer driving cycle (LA-4), although average speeds are approximately the same. The differences may occur because the current test cycle was only intended to represent a morning commute (in Los Angeles), not average vehicle use throughout a day. Also, the large urban sprawl with connecting freeways found in L.A. is very different from the smaller, more compact setting found in Columbus, which would cause different driving habits.

Temperature data collected during the vehicle loans were also analyzed to determine characteristic temperature profiles. The results showed that most parameters require approximately 15-20 minutes to reach stable warm-up levels. Soak temperatures rise very rapidly following engine shutdown, reaching peak values within 10 to 20 minutes. After shut-off, approximately half of the total temperature rise is dissipated in about two hours, while 4.1 to 5.8 hours are required to dissipate 80 to 90 percent of the peak hot soak temperature.

Table 4

Hot Soak Times

Decay Constant (1/minutes)	Time Required to Reduce Temperature Difference to Indicated Percentage of Original (hrs:min)				
	50%	20%	10%	5%	1%
.007	1:39	3:50	5:29	7:08	10:58
.011	1:03	2:26	3:29	4:32	6:59

References

1. "Laboratory Simulation of Driving Conditions in the los Angeles Area," G. C. Hass, M. P. Sweeney, Society of Automobive Engineers, Paper 660546.

2. "Development of the Federal Urban Driving Schedule," R. E. Kruse and T. A. Huls, Society of Automotive Engineers, Paper 730553.

3. "The Changing Rural Village in America: Demographic and Economic Trends Since 1950," H. Johansen and G. Fuguitt, Harper and Rowe.

4. Title III of the Energy Policy and Conservation Act of 1975 amending the Motor Vehicle Information and Cost Savings.

5. "Vehicle Operations Survey," prepared by Scott Research Laboratories Inc., for the Coordinating Research Council and the U. S. Environmental Protection Agency, December 17, 1971.

6. "Truck Driving Pattern and Use Survey Phase II-Final Report, Part 1," prepared by Wilber Smith and Associate for the U. S. Environmental Protection Agency, June 1977.

7. Memorandum and Final Report for Work Assignment No. 3, Contract 68-03-3157, Myron, Gallogly, ATL February 24, 1984.

8. 1979 Gas Mileage Guide compiled and prepared by the U. S. Environmental Protection Agency and published and distributed by the U. S. Department of Energy, 1979.

9. Derived from the Code of Federal Regulations, Title 40, Section 86.144-78, "Calculations; exhaust emissions."

APPENDIXES

Operational Characteristics Study
Columbus, Ohio Program

Appendix A

Data Preparation Programs

OCS.REFMT.SRC

```
100  REM  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
110  REM  *                                                       *
120  REM  *    PROGRAM NAME:  OCS.REFMT.SRC                       *
130  REM  *      PROGRAMMER:  ALVARO M. CHPAMAN (SDSB)            *
140  REM  *    YEAR WRITTEN:  1983                                *
150  REM  *                                                       *
160  REM  *    PURPOSE:  THIS PROGRAM WAS DESIGNED TO REFORMAT    *
170  REM  *              RAW OCS DATABASE FILES.  IT UNPACKS THE  *
180  REM  *    ORIGINAL DATA INTO A FROM WHICH LEAVES EACH NEW    *
190  REM  *    STARTING WITH AN 'FD' IN COLUMNS 1&2.  THEREAFTER, *
200  REM  *    HEXADICMAL SPEED BYTES MAY OR MAY NOT OCCUR.       *
210  REM  *    CAR-ON AND CAR-OFF CODES ARE PRECEDED BY AN 'FE'   *
220  REM  *    CODE. THERE ARE AT PRESENT ONLY FOUR RECOGNIZABLE  *
230  REM  *    ON/OFF CODES, THESE ARE:                           *
240  REM  *                                                       *
250  REM  *    FE 02 =                                            *
260  REM  *    FE 03 =                                            *
270  REM  *    FE 0A =                                            *
280  REM  *    FE 0B =                                            *
290  REM  *                                                       *
300  REM  *    A CALIBRATION MODE CODE IS 'FC' AND ONLY APPEARS   *
310  REM  *    ONCE THROUGHOUT A FILE. IF MORE THAN ONE APPEARS,  *
320  REM  *    DATA RECORDED PAST THE LAST OCCURANCE SHOULD BE    *
330  REM  *    CONSIDERED VALID.                                  *
340  REM  *                                                       *
350  REM  *    A NORMAL DATA BLOCK SHOULD CONSIST OF ONE FD-LINE, *
360  REM  *    WITH NO MORE THAN TEN LINES OF SPEED BYTES         *
370  REM  *    IMMEDIATELY FOLLOWING AN 'FD'.  LESS THAN TEN      *
380  REM  *    LINES ARE ACCEPTABLE PROVIDED THAT SEEMS TO BE NO  *
390  REM  *    MISSING DATA WHICH WOULD CAUSE LATER ANALYSIS      *
400  REM  *    ERRORS.                                            *
410  REM  *                                                       *
420  REM  *    DATA DICTIONARY:                                   *
430  REM  *                                                       *
440  REM  *    VARIABLE NAME    TYPE    DESCRIPTION               *
450  REM  *    -------------    ----    ------------------------- *
460  REM  *        B$            A      CTRL-G (BELL)             *
470  REM  *        D$            A      CTRL-D (DOS)              *
480  REM  *        UF$           A      ARRAY CONTIANING NAMES OF *
490  REM  *                             UNFORMATTED FILES.        *
500  REM  *        FF$           A      ARRAY CONTIANING NAMES OF *
510  REM  *                             FORMATTED FILES.          *
520  REM  *        IX$           A      INTERMEDIATE STRING USED  *
530  REM  *                             PROCESSING DATA FILES.    *
540  REM  *        A$            A      INTERMEDIATE STRING USED  *
550  REM  *                             TO EXTRACT CTRL-J'S.      *
560  REM  *        CH$           A      INTERMEDIATE STRING USED  *
570  REM  *                             VIEW DATA BYTES.          *
580  REM  *        NF            I      INTEGER SCALAR INDICATES  *
590  REM  *                             NUMBER OF FILES TO FORMAT.*
600  REM  *        ND            I      INTEGER SCALAR INDICATES  *
610  REM  *                             NUMBER OF DATA LINES IN A *
620  REM  *                             FILE.                     *
630  REM  *        CHPTR         I      INTEGER SCALAR INDICATES  *
640  REM  *                             CURRENT CHARACTER         *
650  REM  *                             POSITION IN FILE.         *
660  REM  *        SD            I      INTEGER SCALAR INDICATES  *
670  REM  *                             SOURCE DRIVE.             *
680  REM  *        DD            I      INTEGER SCALAR INDICATES  *
690  REM  *                             DESTINATION DRIVE.        *
700  REM  *        DC            I      INTEGER SCALAR INDICATES  *
710  REM  *                             DATA COUNT OF BUFFER.     *
720  REM  *        PTR           I      INTEGER SCALAR INDICATES  *
730  REM  *                             CURRENT POSITION WITHIN   *
```

```
740  REM  *                                        THE BUFFER.                  *
750  REM  *          EMPTY$          A4     CONSTANT STRING USED WHEN *
760  REM  *                                 A BLANK DATA LINE IS READ *
770  REM  *                                 IN.                       *
780  REM  *                                                           *
790  REM  * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
800  REM
810 D$ =  CHR$ (4): REM   CTRL-D
820 B$ =  CHR$ (7): REM   CTRL-G
830 EMPTY$ = "MI SS IN G* DA TA MI SS IN G* DA TA MI SS IN G* "
840  DIM BUFF$(100)
850  REM
860  REM     CALL SUBROUTINES TO INITIATE FILE PROCESSING
870  REM
1000  GOSUB 11000: REM     SUBROUTINE DISK INFORMATION
1010  GOSUB 12000: REM     SUBROUTINE FILE INFORMATION
1020  REM
1030  REM     TOP OF OUTER-LOOP CONTROLLING FILE PROCESSING.
1040  REM
1050  FOR OL = 1 TO NF
1060 TX$ = "": REM     RE-INITIALIZE TX$ TO NOTHING
1070  REM
1080  REM     OPEN BOTH INPUT & OUTPUT FILES.
1090  REM
1100  PRINT D$;"OPEN ";UF$(OL);",D";SD
1110  PRINT D$;"READ ";UF$(OL)
1120  INPUT ND
1130  PRINT D$;"OPEN ";FF$(OL);",D";DD
1140  PRINT D$;"WRITE ";FF$(OL)
1150  PRINT FF$(OL)
1160  GOSUB 13000: REM     SUBROUTINE INPUT DATA
1170  REM
1180  REM     IF LENGTH OF TX$ IS GREATER THAN 72 CHARACTERS, THEN
1190  REM     START PROCESSING.  OTHERWISE, GO BACK FOR MORE DATA.
1200  REM
1210  IF  LEN (TX$) > 72 THEN 2000
1220 DC = DC - 1: REM     DECREMENT DATA COUNT
1220 PTR = PTR + 1: REM     INCREMENT BUFFER POINTER
1240 TMP$ = BUFF$(PTR): REM     ASSIGN CURRENT DATA LINE TO TMP$
1250  REM
1260  REM     IMMEDIATELY CHECK IF A BLANK LINE WAS READ.  IF IT
1270  REM  .  HAS BEEN, THEN INSERT THE ERROR MESSAGE STRING EMPTY$.
1280  REM
1290  IF  LEN (TMP$) < 1 THEN TMP$ = EMPTY$
1300  REM
1310  REM     SCAN THE TEXT LINE FOR ANY CTRL-J'S.  THIS SPECIAL
1320  REM     CHARACTER CAUSES A LINE FEED AND LATER CAUSES MANY
1330  REM     REFORMATTING ERRORS IF NOT EXTRACTED NOW.
1340  REM
1350 A$ = ""
1360  FOR DL = 1 TO  LEN (TMP$)
1370  IF  MID$ (TMP$,DL,1) =  CHR$ (10) THEN 1400
1380  IF  MID$ (TMP$,DL,2) = "  " THEN 1400
1390 A$ = A$ +  MID$ (TMP$,DL,1)
1400  NEXT DL
1410 TMP$ = A$
1420 TX$ = TX$ + TMP$
1430  REM
1440  REM     SEE IF WE HAVE REACHED THE END OF OUR BUFFER.  IF SO,
1450  REM     THEN GO GET MORE DATA TO PROCESS.
1460  REM
1470  REM     NEXT SEE IF WE HAVE REACHED THE END OF THE FILE.  IF SO,
1480  REM  .  THEN JUMP TO LINE [illegible] WHERE WE DUMP REMAINING DATA AND
1490  REM     CLOSE BOTH FILES.
1500  REM
```

```
1530  GOTO 1210
2000  REM  * * * * * * * * * *
2010  REM
2020  REM     CHECK 1ST CHARACTER POSITION TO BE SURE IT ISN'T A
2030  REM     BLANK.
2040  REM
2050  IF  MID$ (TX$,1,1) <  > " " THEN 3000
2060 TX$ =  MID$ (TX$,2, LEN (TX$))
3000  REM  * * * * * * * * * *
3010  REM
3020  REM     SCAN THROUGH TX$ UNTIL WE FALL UPON ONE THE THESE
3030  REM     CHARACTER REPRESENTATIONS AND JUMP TO THE INDICATED
3040  REM     INSTRUCTION BLOCKS.
3050  REM
3060  FOR CHPTR = 1 TO 36 STEP 3
3070 LN = 0
3072 CH$ =  MID$ (TX$,CHPTR,2)
3080  IF CH$ = "FD" THEN 4000
3090  IF CH$ = "FE" THEN LN = 6: GOTO 7000
3100  IF CH$ = "FC" THEN LN = 3: GOTO 7000
3110  NEXT CHPTR
3120  REM
3130  REM     IF HERE, WE ASSUME WE EITHER HAVE A COMPLETE FD-LINE
3140  REM     OR THERE ARE JUST SPEED BYTES TO BE PRINTED.
3150  REM
3160  PRINT  MID$ (TX$,1,36)
3170 TX$ =  MID$ (TX$,37, LEN (TX$))
3180  GOTO 1210
4000  REM  * * * * * * * * * *
4010  REM
4020  REM     IF HERE, WE HAVE FOUND AN 'FD'.  SCAN THROUGH THE REST
4030  REM     OF THE LINE TO BE SURE NOTHING ELSE DOES NOT STAY WITH
4040  REM     THE TEXT WHICH DOESN'T BELONG.
4050  REM
4060  REM     HOWEVER, IF THE POINTER IS POINTING TO THE 1ST CHARACTER
4070  REM     POSITION JUMP TO ANOTHER BLOCK OF INSTRUCTIONS.
4080  REM
4090  IF CHPTR > 1 THEN 6000
4100  FOR DL = 19 TO 36 STEP 3
4110 CH$ =  MID$ (TX$,DL,2)
4120  IF CH$ = "FD" THEN 5000
4130  IF CH$ = "FE" THEN 5000
4140  IF CH$ = "00" THEN 5000
4150  IF CH$ = "01" THEN 5000
4160  IF CH$ = "02" THEN 5000
4170  IF CH$ = "03" THEN 5000
4180  IF CH$ = "04" THEN 5000
4190  NEXT DL
4200  REM
4210  REM     IF HERE, WE ASSUME WE HAVE A COMPLETE FD-LINE.
4220  REM
4230  PRINT  MID$ (TX$,1,36)
4240 TX$ =  MID$ (TX$,37, LEN (TX$))
4250  GOTO 1210
5000  REM  * * * * * * * * * *
5010  REM
5020  REM     IF HERE, WE HAVE FOUND ADDITIONAL INFORMATION ON
5030  REM     AN FD-LINE.  MOST OF THE TIME IT IS EITHER ANOTHER
5040  REM     FD-LINE OR SPEEDS BEGIN TO APPEAR HERE.
5050  REM
5060  PRINT  MID$ (TX$,1,DL - 1)
5070 TX$ =  MID$ (TX$,DL, LEN (TX$))
5080  GOTO 1210
6000  REM  * * * * * * * * * *
6010  REM
```

```
6030  REM      THE DATA WE WISH TO LOOK AT.  WE MUST DUMP DATA
604[illegible] [illegible]      UP TO THE POSITION WHICH OUR POINTER POINTS TO.
6050  REM
6060  PRINT  MID$ (TX$,1,CHPTR - 1)
60   TX$ =  MID$ (TX$,CHPTR, LEN (TX$))
60    GOTO 3000
7000  REM  * * * * * * * * * *
7010  REM
7020  REM     IF HERE, WE HAVE COME ACROSS EITHER AN 'FE'OR 'FC'.
7030  REM
7040  IF CHPTR > 1 THEN 8000
7050  PRINT  MID$ (TX$,1,LN)
7060 TX$ =  MID$ (TX$,LN + 1, LEN (TX$))
7070  GOTO 1210
8000  REM  * * * * * * * * * *
8010  REM
8020  REM     IF HERE, WE HAVE COME ACROSS EITHER AN 'FE' OR 'FC'
8030  REM     BEYOND COLUMN ONE.  DUMP THAT INFORMATION WHICH LIES
8040  REM     BEFORE IT.
8050  REM
8060  PRINT  MID$ (TX$,1,CHPTR - 1)
8070  PRINT  MID$ (TX$,CHPTR,LN)
8080 TX$ =  MID$ (TX$,LN + CHPTR, LEN (TX$))
8090  GOTO 1210
9000  REM  * * * * * * * * * *
9010  REM
9020  REM     IF HERE, WE HAVE REACHED THE END OF THE INPUT FILE.
9030  REM     DATA STILL REMAINS IN TX$, SO GET RID OF IT AND CLOSE
9040  REM     BOTH FILES.
9050  REM
9   ) LCH =  LEN (TX$): REM     LAST CHARACTER POSITION
9   )  IF LCH < 36 THEN 9120
9080  PRINT  MID$ (TX$,1,36)
9090 LCH = LCH - 36
9100 TX$ =  MID$ (TX$,37, LEN (TX$))
9110  GOTO 9060
9120  PRINT  MID$ (TX$,1, LEN (TX$))
9130  REM
9140  REM     NOW CLOSE FILES.
9150  REM
9160  PRINT D$;"CLOSE ";UF$(OL)
9170  PRINT D$;"CLOSE ";FF$(OL)
9172  VTAB (13 + OL): HTAB 20: PRINT "DONE.";D$
9174  VTAB (14 + NF):
9180  NEXT OL
9182  PRINT
9190  PRINT : INVERSE
9200  PRINT "TO CONTINUE REFORMATTING"
9210  PRINT "TYPE:  CALL 6064 <CR>"
9220  PRINT : NORMAL : END
11000  REM  * * * * * * * * * * * * * * * * * * * * * * * * * *
11010  REM  *                                                  *
11020  REM  *               S U B R O U T I N E                *
11030  REM  *         D I S K    I N F O R M A T I O N         *
11040  REM  *                                                  *
11050  REM  * * * * * * * * * * * * * * * * * * * * * * * * * *
1   60  REM
1   70  HOME : VTAB 1: HTAB 10: INVERSE
11080  PRINT "DISKETTE INFORMATION";D$: NORMAL
11090  REM
11100  REM     GET SOURCE DRIVE
11110  REM
11120  VTAB 3: CALL  - 958: INPUT "SOURCE DRIVE: ";SD
11130  IF SD >  = 1 AND SD <  = 2 THEN 11160
11140  VTAB 4: PRINT "(RANGE ERROR)";D$
```

```
11160  REM
11170  REM      GET DESTINATION DRIVE
11180  REM
11190  VTAB 5: CALL  - 958: INPUT "DESTINATION DRIVE: ";DD
11   0  IF DD >  = 1 AND DD <  = 2 THEN 11230
11   0  VTAB 6: PRINT "(RANGE ERROR)";B$
11220  FOR DL = 1 TO 1500: NEXT DL: GOTO 11190
11230  REM
11240  REM      MAKE SURE THEY ARE NOT THE SAME DISK DRIVES.
11250  REM
11260  IF SD <  > DD THEN 11330
11270  PRINT : PRINT "ILLEGAL STATEMENT";B$
11280  INVERSE : PRINT "(PROGRAM REQUIRES 2 DISK DRIVES)": NORMAL
11290  PRINT : PRINT "DOES THIS SYSTEM HAVE 2 DISK DRIVES"
11300  INPUT "(Y/N) ?";YN$
11310  IF YN$ = "N" THEN  PRINT "(PROGRAM TERMINATED)";B$: END
11320  GOTO 11120
11330  REM
11340  REM      HAVE USER REMOVE AND INSERT PROPER DISKETTES AND SEE
11350  REM      IF THE DISKETTE IN THE DESTINATION DRIVE IS TO BE
11360  REM      INITIALIZED.
11370  REM
11380  VTAB 11: INVERSE : PRINT "REMOVE": NORMAL
11390  PRINT "- OCS SYSTEM MASTER DISKETTE"
11400  PRINT
11410  INVERSE : PRINT "INSERT": NORMAL
11420  PRINT "- OCS DATA DISKETTE INTO DRIVE (";SD;")"
11430  PRINT "- CLEAN DATA DISKETTE INTO DRIVE (";DD;")"
11440  PRINT
11450  PRINT "INITIALIZE DISKETTE IN DRIVE (";DD;")";
1   60  INPUT YN$
1.  70  IF YN$ = "N" THEN 11510
11480  PRINT D$;"INIT TEMP,D";DD
11490  PRINT D$;"DELETE TEMP"
11492  PRINT
11500  PRINT "INITIALIZATION COMPLETE";B$
11510  VTAB 23: INVERSE
11520  PRINT "(DEPRESS ANY KEY TO CONTINUE)";
11530  GET YN$: NORMAL
11540  RETURN
12000  REM  * * * * * * * * * * * * * * * * * * * * * * * * * * *
12010  REM  *                                                   *
12020  REM  *                 S U B R O U T I N E               *
12030  REM  *          F I L E   I N F O R M A T I O N          *
12040  REM  *                                                   *
12050  REM  * * * * * * * * * * * * * * * * * * * * * * * * * * *
12060  REM
12070  HOME : VTAB 1: HTAB 10: INVERSE
12072  PRINT "FILE INFORMATION": NORMAL
12080  VTAB 2: PRINT D$;"CATALOG,D";SD
12090  VTAB 14: PRINT "(ENTER    TO END PROGRAM)"
12100  INPUT "NUMBER OF FILES TO REFORMAT: ";NF
12110  IF NF < 1 THEN  PRINT "PROGRAM TERMINATED";B$: END
12130  FOR DL = 1 TO NF
12140  PRINT : VTAB (13 + DL): CALL  - 958
12150  PRINT "FILE >";DL;" ";
1   60  INPUT UF$(DL)
1   70  PRINT "SPELLED CORRECTLY (Y/N): ";: INVERSE
12172  PRINT UF$(DL);: NORMAL : PRINT " ";
12180  INPUT YN$
12190  IF YN$ = "N" THEN 12140
12192  VTAB (13 + DL): HTAB 20: CALL  - 958
12200  NEXT DL
12210  REM
12220  REM      CREATE NEW FILE NAMES FOR OUTPUT FILES.
```

```
12230  REM
12240  FOR DL = 1 TO NF
12250 FF$(DL) = UF$(DL) + "*"
12260  NEXT DL
12270  RETURN
13000  REM  * * * * * * * * * * * * * * * * * * * * * * * * * *
13010  REM  *                                                 *
13020  REM  *               S U B R O U T I N E               *
13030  REM  *              D A T A     I N P U T              *
13040  REM  *                                                 *
13050  REM  * * * * * * * * * * * * * * * * * * * * * * * * * *
13060  REM
13070 PTR = 0: REM      RE-INITIALIZE BUFFER POINTER
14000  REM
14010  IF ND < 100 THEN 14080
14020 ND =  ABS (ND - 100)
14030 DC = 100
14032  PRINT D$;"READ ";UF$(DL)
14040  FOR DL = 1 TO DC
14050 BUFF$(DL) = "": INPUT BUFF$(DL)
14060  NEXT DL
14070  GOTO 14130
14080 DC = ND
14090 ND = 0
14092  IF DC = 0 THEN 14130
14094  PRINT D$;"READ ";UF$(DL)
14100  FOR DL = 1 TO DC
14110 BUFF$(DL) = "": INPUT BUFF$(DL)
14120  NEXT DL
14130  PRINT D$;"WRITE ";FF$(DL)
14140  RETURN

]
```

OCS.S-TMPFIX

```
C********************************************************************
C*                                                                  *
C*    PROGRAM NAME:  OCS.S-TMPFIX                                   *
C*    PROGRAMMER:    ALVARO M. CHAPMAN (SDSB)                       *
C*                                                                  *
C*    BRANCH:        STANDARD DEVELOPMENT & SUPPORT BRANCH (SDSB)   *
C*    PROJECT:       OPERATIONAL CHARACTERISTICS STUDY (OCS)        *
C*                                                                  *
C*    =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =    *
C*                                                                  *
C*    PROGRAM DESCRIPTION:                                          *
C*                                                                  *
C*    THIS PROGRAM IS DESIGNED TO CORRECT FD-LINES WITHIN THE       *
C*    OCS DATABASE FILES.  AN FD-LINE CONSIST OF THE FOLLOWING:     *
C*                                                                  *
C*    FD DD HH MM ID ID T1 T2 T3 T4 T5 T6                           *
C*                                                                  *
C*    WHERE:  FD - INDICATES BEGINNING OF NEW MINUTE BLOCK          *
C*            DD - INDICATES THE JULIAN DAY OF THE YEAR             *
C*            HH - INDICATES THE HOUR OF THE DAY IN MILITARY TIME   *
C*            MM - INDICATES THE MINUTE DURING THE 'HH' HOUR        *
C*            ID - INDICATES THE VEHICLE LOAN ID NUMBER, THIS       *
C*                 ID NUMBER CONSIST OF TWO HEX PAIRS.              *
C*                 (I.E. 04 20 WOULD MEAN LOAN ID 420)              *
C*    T1 THRU T6 - INDICATES THE SIX DIFFERENT TEMPERATURES FOR     *
C*                 VARIOUS PURPOSES.                                *
C*                                                                  *
C*    A TYPICAL FD-LINE WILL APPEAR AS SUCH:                        *
C*                                                                  *
C*    FD 99 14 02 03 99 5A 5B 5C 5D 5E 5F                           *
C*                                                                  *
C*    THIS PROGRAM IS DESIGNED TO ONLY CORRECT THE FOLLOWING:       *
C*                                                                  *
C*    JULIAN DAY - IT WILL ONLY BE ALTERED WHEN THE ID IS NOT       *
C*                 IN PLACE.                                        *
C*                                                                  *
C*    ITEMS TO THE LEFT OF THE ID WILL ONLY BE ALTERED IF THE       *
C*    ID IS EITHER MISSING OR SHIFTED TO THE LEFT ONE SPACE OR      *
C*    MORE.                                                         *
C*                                                                  *
C*    ITEMS TO THE RIGHT OF THE ID WILL ONLY BE ALTERED IF THE      *
C*    ID IS FOUND TO BE IN ITS CORRECT SPACE. IF THE ID IS FOUND    *
C*    IN ITS CORRECT SPACE AND THERE APPEARS TO BE MORE THAN ONE    *
C*    TEMPERATURE MISSING. THIS DATA LINE WILL NOT BE ALTERED DUE   *
C*    TO THE FACT THAT THIS IS NOT POSSIBLE AT THIS POINT.          *
C*                                                                  *
C*    =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =  =    *
C*                                                                  *
C*    UNIT FILE DESCRIPTIONS:                                       *
C*                                                                  *
C*    UNIT #   FILE NAME  FILE DESCRIPTION                          *
C*    ------   ---------  ----------------------------------------- *
C*       4      ERRORS    OUTPUT DEVICE FOR ERROR MESSAGES          *
C*       5      INPUT     INPUT  DEVICE FOR OCS HEX DATA            *
C*       6      OUTPUT    OUTPUT DEVICE FOR CORRECTED OCS DATA      *
C*       7      SCREEN    OUTPUT DEVICE FOR TERMINAL USE            *
C*                                                                  *
C*                                                                  *



:
```

```
C*    VARIABLE DESCRIPTIONS:                                          *
C*                                                                    *
C*    TYPE   NAME     DESCRIPTION                                     *
C*    ----   ------   ---------------------------------------------   *
C*    I      FDFLAG   ARRAY CONTAINING STATUS OF FDBUFF.  CONSIST     *
C*                    OF ZEROS AND ONES.                              *
C*    I      FDPTR    POINTS TO POSITION IN BUFFER WHERE AN FD-LINE   *
C*                    OCCURS.                                         *
C*    I      BUFCNT   INDICATES CURRENT NUMBER OF ELEMENTS WITHIN     *
C*                    BUFFER.                                         *
C*    I      COUNT    INTERMEDIATE VARIABLE USED WHEN SCROLLING THE   *
C*                    BUFFER.                                         *
C*    I      FDCNT    INCREMENT COUNTER INDICATING NUMBER OF          *
C*                    FD-LINES READ INTO THE BUFFER.                  *
C*    I      J        WILD CARD VARIABLE USED FOR MEANINGLESS LOOPS.  *
C*    I      PTRCNT   INTERMEDIATE VARIABLE USED WHEN SCROLLING THE   *
C*                    BUFFER.                                         *
C*    I      ERRFIX   CONTAINS NUMBER OF ERRORS CORRECTED BY PROGRAM*
C*                    DURING A RUN.                                   *
C*    I      ERRCNT   CONTAINS NUMBER OF ERRORS WHICH OCCURED DURING*
C*                    A RUN.                                          *
C*                                                                    *
C*    A      FDBUFF   ARRAY CONTAINING DUPLICATES OF FD-LINES FROM    *
C*                    WITHIN THE BUFFER.                              *
C*    A      BUFFER   ARRAY CONTAINING DATA READ IN FROM THE INPUT    *
C*                    FILE.                                           *
C*    A      LINE     INTERMEDIATE VARIABLE USED TO FILL THE BUFFER   *
C*                    ARRAY.                                          *
C*    A      DATAFL   STRING VARIABLE USED ONLY FOR ECHOING.          *
C*    A      DATE     STRING VARIABLE USED FOR ECHOING DATE TO THE    *
C*                    SCREEN.                                         *
C*    A      DATAFL   STRING VARIABLE USED FOR ECHOING THE NAME OF    *
C*                    A DATA FILE DURING A RUN.                       *
C*    A      OPNAME   STRING VARIABLE USED FOR ECHOING THE NAME OF    *
C*                    THE ACTUAL USER DURING A RUN.                   *
C*    A      TMPLIN   INTERMEDIATE VARIABLE USED TO DISPLAY ORIGINAL*
C*                    DATA TO ERROR MESSAGE FILE BEFORE IT HAS BEEN  *
C*                    CORRECTED.                                      *
C*    A      ID1ST    1ST PART OF THE VEHICLE LOAN ID NUMBER.         *
C*    A      ID2ND    2ND PART OF THE VEHICLE LOAN ID NUMBER.         *
C*                                                                    *
C*    B      FIXED    BOOLEAN VARIABLE USED TO INDICATE WHETHER A     *
C*                    DATA LINE HAS BEEN CORRECTED.                   *
C*                                                                    *
C*********************************************************************
C
```

```
      INTEGER      FDFLAG(3), FDPTR(3), BUFCNT, PTRCNT, ERRFIX
      INTEGER      ERRCNT, COUNT, FDCNT, J
      INTEGER      ERRORS, INPUT, OUTPUT, SCREEN
C
      CHARACTER*50 FDBUFF(3), BUFFER(100), LINE, TMPLIN
      CHARACTER*6  ID1ST, ID2ND, OPNAME
      CHARACTER    DATAFL*12, DATE*8
C
C
      LOGICAL      FIXED
C
      DATA         ERRORS, INPUT, OUTPUT, SCREEN /4,5,6,7/
      DATA         ERRCNT, ERRFIX, BUFCNT, FDCNT /0,0,0,0/
C
C******************************************************************
C*                                                                *
C*    USING A DO-LOOP, CLEAR ANY PREVIOUS TEXT FROM THE TERMINAL  *
C*    SCREEN, THEN PROMPT THE USER FOR THE FOLLOWING INFORMATION  *
C*    PERTAINING TO EACH INDIVIDUAL RUN.                          *
C*                                                                *
C******************************************************************
C
      FIXED = .FALSE.
C
      DO 10 J=1,22
         WRITE(SCREEN,*)
 10   CONTINUE
C
      WRITE(SCREEN,100) 'OPERATIONAL CHARACTERISTICS STUDY'
      WRITE(SCREEN,100) '    TEMPERATURE FIXING PROGRAM'
      WRITE(SCREEN,*)
C
      WRITE(SCREEN,100) 'INPUT TODAYS DATE (MM/DD/YY)...'
      READ (SCREEN,100)  DATE
C
      WRITE(SCREEN,100) 'INPUT INITIALS OR FIRST NAME..:'
      READ (SCREEN,100)  OPNAME
C
      WRITE(SCREEN,100) 'INPUT OCS DATA FILE NAME...'
      READ (SCREEN,100)  DATAFL
C
      WRITE(SCREEN,*)
      WRITE(SCREEN,120)  DATE,OPNAME,DATAFL
      WRITE(SCREEN,*)
C
      WRITE(SCREEN,*) 'PROCESSING DATA SET...'
C
C
```

```
C******************************************************************
C*                                                                *
C*    READ IN ONE DATA LINE AT A TIME, PUT IT INTO THE WORKING    *
C*    BUFFER, THEN CHECK TO SEE IF A CHARACTER REPRESENTATION     *
C*    OF 'FD' APPEARS IN COLUMNS 1 & 2.                           *
C*                                                                *
C*    IF AN 'FD' APPEARS IN COLUMNS 1 & 2, COPY THIS DATA LINE    *
C*    INTO THE FD-BUFFER, THEN CHECK ITS LENGTH TO SEE WHETHER    *
C*    IT HAS AN ERROR.  IF THE LENGTH OF THE FD-LINE IS LESS      *
C*    THAN 36 UNITS, THE LINE IS BAD SO INDICATE SO BY SETTING    *
C*    ITS FLAG ARRAY ELEMENT (THE ELEMENT IS DETERMINED BY THE    *
C*    SEQUENCE THIS PARTICULAR FD-LINE WAS FOUND) TO 1.           *
C*                                                                *
C*    FINALLY, CHECK TO SEE IF WE HAVE FILLED OUR FD-BUFFER,      *
C*    IF NOT, THEN RESUME READING DATA UNTIL WE DO.               *
C*                                                                *
C******************************************************************
C
 20   READ(INPUT,100,END=80) LINE
      BUFCNT = BUFCNT + 1
      BUFFER (BUFCNT)  = LINE
C
      IF (LINE(1:2).EQ.'FD') THEN
         FDCNT = FDCNT + 1
         FDFLAG (FDCNT)  = 0
         FDPTR (FDCNT)   = BUFCNT
         FDBUFF (FDCNT)  = LINE
C
         IF ( INDEX(LINE,'  ').LT.35 ) THEN
            FDFLAG (FDCNT)  = 1
            ERRCNT = ERRCNT + 1
         ENDIF
      ENDIF
C
      IF (FDCNT.LT.3) GOTO 20
C

:
```

```
C******************************************************************
C*                                                                *
C*    HERE IS WHERE MOST OF THE DECISION MAKING GOES ON.  WE     *
C*    MUST DETERMINE WHAT EXAXCTLY NEEDS TO BE FIXED ON THE 2ND   *
C*    FD-LINE WITHIN THE FD-BUFFER.  HOWEVER, FIRST WE MUST       *
C*    BE SURE THAT WE HAVE A DESIRED FD-BUFFER TO WORK WITH.      *
C*                                                                *
C*    THE FOLLOWING WILL DEMONSTRATE WHAT TYPE OF PATTERN WE      *
C*    WISH TO WORK WITH & NOT WORK WITH:                          *
C*                                                                *
C*    ONE WE CAN WORK WITH                                        *
C*    --------------------                                        *
C*                                                                *
C*    FD 99 14 02 03 39 5A 5B 5C 5D 5E 5F  = 0 (GOOD FD-LINE)     *
C*    FD 99 14 02 03 39 5A 5B 5C 5D 5E     = 1 (BAD  FD-LINE)     *
C*    FD 99 14 04 03 39 5A 5B 5C 5D 5E 5F  = 0 (GOOD FD-LINE)     *
C*                                                                *
C*    ONE WE CAN'T WORK WITH:                                     *
C*    ----------------------                                      *
C*                                                                *
C*    FD 99 14 02 03 39 5A 5B 5C 5D 5E     = 1 (BAD  FD-LINE)     *
C*    FD 99 14 03 03 39 5A 5B 5C 5D 5E 5F  = 0 (GOOD FD-LINE)     *
C*    FD 99 14 04 03 39 5A 5B 5C 5D 5E     = 1 (BAD  FD-LINE)     *
C*                                                                *
C*    SEE, THE TRICK IS TO WORK WITH THE BAD FD-LINE IN THE       *
C*    MIDDLE SO THAT WE CAN AVERAGE OUT THE TEMPERATURES TO       *
C*    REPLACE THE MISSING ONE.  A SPECIAL SUBROUTINE WILL DO      *
C*    THE ACTUAL CALCULATIONS IN FINDING WHICH ONE IS MISSING,    *
C*    BUT FIRST WE MUST SEND IT A VALID FD-BUFFER TO WORK WITH.   *
C*                                                                *
C******************************************************************
C
      IF (FDFLAG(1).EQ.1) GOTO 40
      IF (FDFLAG(2).EQ.0) GOTO 40
      IF (FDFLAG(3).EQ.1) GOTO 40
C
      IF ( INDEX(FDBUFF(2),'  ').LE.30 ) GOTO 40
C
      ID1ST = FDBUFF(1)(13:17)
      ID2ND = FDBUFF(2)(13:17)
      TMPLIN = FDBUFF(2)
C
C******************************************************************
C*                                                                *
C*    TO BE AT THIS POINT. WE MUST HAVE AN FD-BUFFER WHICH TO     *
C*    WORK WITH.  THE ONLY PROBLEM IS THAT WE DON'T KNOW WHAT     *
C*    TO FIX - DATA TO THE LEFT OF THE ID OR DATA TO THE RIGHT    *
C*    OF THE ID.  IF ID1ST DOES NOT EQUAL ID2ND. THEN WE KNOW     *
C*    THE CORRECTION TO BE MADE MUST BE TO THE LEFT OF THE ID.    *
C*                                                                *
C******************************************************************
C
      IF (ID2ND.NE.ID1ST) THEN
         CALL IDFIX( FDBUFF,FIXED )
         IF (.NOT. FIXED) GOTO 30
         FDFLAG(2) = 0
      ELSE
         CALL TMPFIX( FDBUFF )
         FDFLAG(2) = 0
      ENDIF
C
      ERRFIX = ERRFIX + 1
 30   BUFFER( FDPTR(2) ) = FDBUFF(2)
```

```
C******************************************************************
C*                                                                *
C*    NOW THAT WE HAVE MADE THE NEEDED CORRECTIONS, WRITE OUT THE *
C*    LINES WHICH WERE AFFECTED TO THE ERROR MESSAGE FILE FOR     *
C*    USER ANALYSIS AND WRITE OUT THE BUFFER UP TO THE 2ND        *
C*    FD-LINE TO THE OUTPUT FILE.  THIS IS SO THAT WE CAN SCROLL  *
C*    THE BUFFER UP NEXT.                                         *
C*                                                                *
C******************************************************************
C
      WRITE(ERRORS,110) FDBUFF(1)
      WRITE(ERRORS,130) TMPLIN, FDBUFF(2)
      WRITE(ERRORS,110) FDBUFF(3)
      WRITE(ERRORS,*)
C
 40   IF (FDFLAG(1).EQ.1) BUFFER( FDPTR(1) )(40:46) = 'ERROR'
C
      DO 50 J = 1,FDPTR(2)-1
         WRITE(OUTPUT,110) BUFFER(J)
 50   CONTINUE
C
C******************************************************************
C*                                                                *
C*    NOW, AT THIS POINT WE MUST DO THE ACTUAL SCROLLING OF THE   *
C*    BUFFER.  ALL WE DO IS MOVE EVERYTHING FROM THE 2ND FD-LINE  *
C*    UP TO THE LAST LINE IN THE BUFFER UP TO THE TOP OF THE      *
C*    BUFFER.  WE ALSO NEED TO KEEP INTACK WHERE THE POINTERS ARE *
C*    POINTING.                                                   *
C*                                                                *
C******************************************************************
C
      COUNT = 0
      PTRCNT = 0
C
      DO 60 J = FDPTR(2).FDPTR(3)
         COUNT = COUNT + 1
         BUFFER( COUNT ) = BUFFER(J)
C
         IF (BUFFER(COUNT)(1:2).EQ.'FD') THEN
            PTRCNT = PTRCNT + 1
            FDPTR( PTRCNT ) = COUNT
         ENDIF
C
 60   CONTINUE
C
      DO 70 J = 1,2
         FDFLAG(J) = FDFLAG(J+1)
         FDBUFF(J) = FDBUFF(J+1)
 70   CONTINUE
C
      FDCNT = 2
      BUFCNT = COUNT
      GOTO 20
C


:
```

```
C******************************************************************
C*                                                                *
C*    TO HAVE ARRIEVED HERE, WE MUST HAVE ENCOUNTERED THE EOF.    *
C*    SINCE THERE IS NO FURTHER DATA TO COMPARE ANYTHING WITH,    *
C*    WE WILL LEAVE THE REMAINDER OF THE BUFFER UNTOUCHED AND     *
C*    SIMPLY DUMP IT TO THE OUTPUT FILE.  ALSO, WE WILL DISPLAY   *
C*    THE NUMBER OF ERRORS FOUND AND THOSE WHICH HAVE BEEN        *
C*    CORRECTED.                                                  *
C*                                                                *
C******************************************************************
C
 80   DO 90 J = 1,BUFCNT
         WRITE(OUTPUT,110) BUFFER(J)
 90   CONTINUE
C
      WRITE(SCREEN,*) 'PROCESSING COMPLETE!'
C
      WRITE(ERRORS,*) 'ERRORS FOUND = ',ERRCNT
      WRITE(ERRORS,*) 'ERRORS FIXED = ',ERRFIX
C
      WRITE(SCREEN,*) 'ERRORS FOUND = ',ERRCNT
      WRITE(SCREEN,*) 'ERRORS FIXED = ',ERRFIX
C
C     FORMAT STATEMENTS
C
 100  FORMAT(A)
 110  FORMAT(' ',A)
 120  FORMAT(' ','DATE: ',A8,'    OPERATOR: ',A8,'   FILE: ',A12)
 130  FORMAT(' ',A,'<--->',A)
C
      STOP
      END
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                     @
C@                S U B R O U T I N E    F N D V A L                   @
C@                                                                     @
C@    THIS SUBROUTINE IS DESIGNED TO CALCULATE THE MISSING TEMP-      @
C@    ERATURES WHEN CALLED UPON BY THE SUBROUTINE TMPFIX.  IT         @
C@    DOES THIS BY CALCULATING THE LEAST SQUARE ERROR DIFFERENCE      @
C@    OF A 3 BY 6 MATRIX.                                             @
C@                                                                     @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      SUBROUTINE    FNDVAL( TEMPS )
C
      INTEGER       TEMPS(3.6), TMPVAL(6), TRUTMP(6), ERROR(6)
      INTEGER       ERRSUM(6), MINSUM, CNT, I, J, K
C
C
      MINSUM = 10000
C
      DO 20 I = 1,6
         ERRSUM(I) = 0
C
         DO 10 J = 1,6
            IF (J.LT.I) TMPVAL(J) = TEMPS(2,J)
            IF (J.GT.I) TMPVAL(J) = TEMPS(2,J-1)
            IF (J.EQ.I) GOTO 10
C
            ERROR(J) = ( TEMPS(1,J) + TEMPS(3,J) - 2*TMPVAL(J) )
            ERRSUM(I) = ERRSUM(I) + ERROR(J)**2
 10      CONTINUE
C
         IF (ERRSUM(I).GT.MINSUM) GOTO 20
         MINSUM = ERRSUM(I)
         CNT = I
 20   CONTINUE
C
C
      DO 30 K = 1.6
         IF (K.LT.CNT) TRUTMP(K) = TEMPS(2.K)
         IF (K.GT.CNT) TRUTMP(K) = TEMPS(2,K-1)
         IF (K.EQ.CNT) TRUTMP(K) = ( TEMPS(1.K) + TEMPS(3,K) )/2
 30   CONTINUE
C
      DO 40 K = 1,6
         TEMPS(2,K) = TRUTMP(K)
 40   CONTINUE
C
      RETURN
      END
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                @
C@              S U B R O U T I N E    T M P F I X                @
C@                                                                @
C@    THIS SUBROUTINE IS DESIGNED TO REPAIR THE MISSING TEMP-     @
C@    ERATURE ONCE IT HAS BEEN SUPPLIED BY THE FNDVAL SUBROUTINE. @
C@                                                                @
C@    IT SIMPLY TRANSLATES THE ENGINEER UNIT NUMBER INTO ITS      @
C@    EQUIVALENT HEXADECIMAL CHARACTER VALUE USING THE SUBROUTINE @
C@    HEXDEC (TRANSLATES FROM HEXS TO DECIMAL) AND THE SUBROUTINE @
C@    DECHEX (TRANSLATES FROM DECIMAL TO HEXS).                   @
C@                                                                @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      SUBROUTINE    TMPFIX( TMPBUF )
C
      INTEGER      DECVAL(3,6), HEXVAL, CNT, I, J
      CHARACTER    TMPBUF(3)*50, MATRIX(3,6)*3, BLANK*3
C
C     NULL OUT THE CHARACTER MATRIX
C
      BLANK = '   '
      DO 10 I = 1,3
         DO 10 J = 1,6
            MATRIX(I,J) = BLANK
 10      CONTINUE
 20   CONTINUE
C
C     CONVERT THE HEX VECTOR INTO A DECIMAL MATRIX
C
      DO 40 I = 1,3
         CNT = 0
         DO 30 J = 19,36,3
            CNT = CNT + 1
            MATRIX(I,CNT) = TMPBUF(I)(J:J+2)
            CALL HEXDEC( MATRIX(I,CNT), DECVAL(I.CNT) )
 30      CONTINUE
 40   CONTINUE
C
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                @
C@    CALL UPON SUBROUTINE FNDVAL AND RETURN WITH MISSING         @
C@    TEMPERATURE.                                                @
C@                                                                @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      CALL FNDVAL( DECVAL )
C
      DO 50 J = 1,6
         CALL DECHEX( DECVAL(2,J), MATRIX(2,J) )
 50   CONTINUE
C
      TMPBUF(2)(19:36) = MATRIX(2,1)//MATRIX(2,2)//MATRIX(2,
     *                   3)//MATRIX(2,4)//MATRIX(2,5)//MATRIX(2,6)
C
      RETURN
      END
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                 @
C@              S U B R O U T I N E    H E X D E C                 @
C@                                                                 @
C@    THIS SUBROUTINE IS DESIGNED TO CONVERT INCOMING DATA FROM    @
C@    ITS HEXADECIMAL VALUE INTO ITS EQUIVALENT DECIMAL VALUE.     @
C@                                                                 @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      SUBROUTINE    HEXDEC( HEXDIG, VALUE )
C

      INTEGER      HEXVAL(2), VALUE, I, J
      CHARACTER    HEXS(16)*1, HEXDIG*3
C
      DATA         HEXS /'0','1','2','3','4','5','6','7','8','9',
     *                 'A','B','C','D','E','F'/
C
      DATA         HEXVAL /0,0/
C
C     SEARCH THROUGH HEXADECIMAL ARRAY FOR EQUIVALENT REPRESENTA-
C     TIONS OF DIGITS OR LETTERS.
C
      DO 20 I = 1,2
         DO 10 J = 1,16
            IF (HEXDIG(I:I).EQ.HEXS(J)) HEXVAL(I) = J-1
 10      CONTINUE
 20   CONTINUE
C
      VALUE = HEXVAL(1)*16 + HEXVAL(2)
C
      RETURN
      END


:
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                     @
C@                S U B R O U T I N E    D E C H E X                   @
C@                                                                     @
C@    THIS SUBROUTINE IS DESIGNED TO CONVERT INCOMING DATA FROM        @
C@    ITS DECIMAL VALUE TO ITS EQUIVALENT CHARACTER REPRESENTA-        @
C@    TION.                                                            @
C@                                                                     @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      SUBROUTINE    DECHEX( DECVAL,HEXDIG )
C
      INTEGER       DIGIT1, DIGIT2, DECVAL
      CHARACTER     HEXS(16)*1, HEXDIG*3, HEX1*1, HEX2*1
C

      DATA          HEXS /'0','1','2','3','4','5','6','7','8','9',
     *                    'A','B','C','D','E','F'/
C
      DIGIT1 = DECVAL/16
      DIGIT2 = MOD( DECVAL,16 )
C
      HEX1 = HEXS( DIGIT1 + 1 )
      HEX2 = HEXS( DIGIT2 + 1 )
C
      HEXDIG = HEX1//HEX2
C
      RETURN
      END


:
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                @
C@              S U B R O U T I N E    I D F I X                  @
C@                                                                @
C@    THIS SUBROUTINE IS DESIGNED TO CORRECT ALL DATA TO THE      @
C@    LEFT OF AND INCLUDING THE LOAN-ID ON AN FD-LINE.            @
C@                                                                @
C@    THIS SUBROUTINE IS ONLY INVOKED WHEN WE KNOW THERE EXIST    @
C@    ALL SIX TEMPERATURES.  WHAT WE MAINLY CHECK FOR IN THIS     @
C@    SUBROUTINE IS THE ACCURACY IN SEQUENTIAL TIME.  IF THE      @
C@    TIME ON THE 2ND FD-LINE DOES NOT COMPUTE A DIFFERENCE OF    @
C@    EITHER 2 MINUTES OR 40 MINUTES, THEN WE KNOW A SERIOUS      @
C@    ERROR EXIST AND WE DO NOTHING TO IT.                        @
C@                                                                @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      SUBROUTINE    IDFIX( BUFFER,FIXED )
C
      INTEGER      DAY, HOUR, MIN, FDLEN, DIFF, CNT, I, J
      INTEGER      DEC(3,3), TIME1, TIME2, TIME3, VAL$
C
      CHARACTER    BUFFER(3)*50, MAT(3,12)*3, LOANID*6
      LOGICAL      FIXED
C
C
C     EXTRACT THE CORRECT LOAN-ID FROM THE BUFFER'S 1ST ELEMENT
C
      FIXED = .FALSE.
      LOANID = BUFFER(1)(13:18)
C
C     CONVERT THE BUFFER VECTOR INTO A 3X3 MATRIX FOR SIMPLER
C     ACCESS.
C
      DO 20 I = 1,3
         CNT = 0
         DO 10 J = 1,36,3
            CNT = CNT + 1
            MAT( I,CNT ) = BUFFER(I)(J:J+2)
 10      CONTINUE
 20   CONTINUE
C
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@    SHIFT ALL ELEMENTS WITHIN THE 2ND ROW ONE SPACE TO THE      @
C@    RIGHT.                                                      @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      DO 30 I = 12,6,-1
         MAT( 2,I ) = MAT( 2,I-1 )
 30   CONTINUE
C


:
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@    CONVERT CHARACTER REPRESENTATIONS INTO THEIR DECIMAL          @
C@    EQUIVALENTS.                                                  @
C@                                                                  @
C@    WE ARE CONCERNED WITH THE FOLLOWING:                          @
C@                                                                  @
C@      1   2   3   4   5   6   7   8   9   10  11  12              @
C@     --  --  --  --  --  --  --  --  --  --  --  --               @
C@   1:FD  61  13  02  02  25  3A  3B  3C  3D  3E  3F               @
C@   2:FD  61  13  02  25  4A  4B  4C  4D  4E  4F                   @
C@   3:FD  61  13  04  02  25  5A  5B  5C  5D  5E  5F               @
C@         <>  <>  <>                                               @
C@                                                                  @
C@    WE ARE INTERESTED IN THOSE NUMBERS WHERE '<>' LIES            @
C@    DIRECTLY BENEATH THEM.  THESE NUMBERS ARE USED TO             @
C@    CALCULATE THE MISSING TIME IN THE 2ND FD-LINE.                @
C@                                                                  @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      CNT = 1
C
 40   DO 50 I = 2,4,1
         DEC(CNT,I-1) = VAL$( MAT(CNT,I)(1:1) )*10 +
     *                  VAL$( MAT(CNT,I)(2:2) )
 50   CONTINUE
C
      IF (CNT.LT.3) THEN
         CNT = 3
         GOTO 40
      ENDIF
C
      TIME1 = DEC(1,1)*60*24 + DEC(1,2)*60 + DEC(1,3)
      TIME3 = DEC(3,1)*60*24 + DEC(3,2)*60 + DEC(3,3)
C
      DIFF = TIME3 - TIME1
C
      IF ((DIFF.NE.2) .AND. (DIFF.NE.40)) GOTO 999
C
      FIXED = .TRUE.
C
      IF (DIFF.EQ.2)  TIME2 = TIME1 + 1
      IF (DIFF.EQ.40) TIME2 = TIME1 + 20
C
      DAY = TIME2/1440
      TIME2 = TIME2 - DAY*1440
C
      HOUR = TIME2/60
      TIME2 = TIME2 - HOUR*60
C
      MIN = TIME2
C
C     NOW CONVERT THESE DECIMAL VALUES BACK INTO THERE
C     CHARACTER REPRESENTATION.
C
      CALL STR$( DAY, MAT(2,2) )
      CALL STR$( HOUR,MAT(2,3) )
      CALL STR$( MIN ,MAT(2,4) )
C


 :
```

```
      MAT(2,5)  = LOANID(1:3)
      MAT(2,6)  = LOANID(4:6)
C
      BUFFER(2)  = MAT(2,1)//MAT(2,2)//MAT(2,3)//MAT(2,4)//MAT(2,
     *          5)//MAT(2,6)//MAT(2,7)//MAT(2,8)//MAT(2,9)//MAT(2,
     *         10)//MAT(2,11)//MAT(2,12)
C
 999  RETURN
      END


:
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                     @
C@          I N T E G E R    F U N C T I O N    V A L$  @
C@                                                     @
C@    THIS INTEGER FUNCTION IS DESIGNED TO CONVERT INCOMING @
C@    CHARACTER DATA INTO ITS EQUIVALENT DECIMAL VALUE AND @
C@    RETURN ITS FORTRAN STATEMENT.                    @
C@                                                     @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      INTEGER FUNCTION    VAL$( CHR$ )
C
      INTEGER        I
      CHARACTER      DECMAL(10)*1, CHR$*1
C
      DATA           DECMAL /'0','1','2','3','4','5','6','7','8','9'/
C
      VAL$= 0
C
      DO 10 I = 1,10
         IF (CHR$.EQ.DECMAL(I)) GOTO 20
 10   CONTINUE
C
      VAL$ = -1
      GOTO 30
C
 20   VAL$ = I-1
C
 30   RETURN
      END


:
```

```
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C@                                                                 @
C@              S U B R O U T I N E    S T R$                      @
C@                                                                 @
C@    THIS SUBROUTINE IS DESIGNED TO CONVERT INCOMING DATA FROM    @
C@    ITS DECIMAL VALUE TO ITS EQUIVALENT CHARACTER                @
C@    REPRESENTATION.                                              @
C@                                                                 @
C@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C
      SUBROUTINE    STR$( VAL,STRING )
C
      INTEGER      VAL, LEFT, RIGHT
      CHARACTER    STRING*3, HEXS(10)*1
C
      DATA         HEXS /'0','1','2','3','4','5','6','7','8','9'/
C
      LEFT = VAL/10
      RIGHT = VAL - (LEFT*10)
C
      STRING(1:1) = HEXS( LEFT + 1 )
      STRING(2:2) = HEXS( RIGHT + 1 )
C
      RETURN
      END


:
```

The temperature correcting algorithm can be most easily understood by considering an example. In this example it is simpler to consider only four temperature inputs. A data set with a missing datum might appear as:

5C	41	57	4A
5C	42	4B	*
5C	43	57	4C

First we convert from hexadecimal to decimal:

92	65	87	74
92	66	75	*
92	67	87	76

The test or trial vector for the second row, which is the interpolated mean of the first and third row, can now be calculated.

92	66	87	75

Comparing the test vector against the second row, the sum of the squares of the differences with no shifting of the row is:

$$(92-92)^2 + (66-66)^2 + (75-87)^2 = 144$$

If the last element of the row is shifted one location the sum of the squares of the differences is:

$$(92-92)^2 + (66-66)^2 + (75-75)^2 = 0$$

Shifting the last two elements, the sum of the squares of the differences is:

$$(92-92)^2 + (87-66)^2 + (74-75)^2 = 442$$

Shifting all elements, the sum of the squares of the differences is:

$$(66-92)^2 + (87-66)^2 + (74-75)^2 = 1118$$

Clearly shifting only the last element of the row results in the minimum sum of the squares. Therefore we know that the data array should be:

92	65	87	74
92	66	*	75
92	67	87	76

The trial or test value, 87 is now inserted in the location of the known missing datum.

Finally converting back into hexadecimal the corrected data are:

5C	41	57	4A
5C	42	57	4B
5C	43	57	4C

---

* missing datum.

OCS.S-SPDFIX

```
C
C  OCS.S-SPDFIX IS AN ATTEMPT TO FIRST DETECT AND THEN
C  CORRECT SPEED ERRORS IN OCS ORIGINAL HEX DATA FILES.
C
C
C  HERE IS A SAMPLE OF THE ODD THINGS THAT OCCUR IN
C  ORIGINAL OCS HEX DATA:
C
C  FD LINE
C  FE ON CODE
C  (SPEEDS MAY OR MAY NOT FOLLOW BEFORE NEXT FD)
C  FD
C
C  OR A DIFFERENT SCENARIO:
C
C  FD LINE
C  (SPEEDS MAY OR MAY NOT FOLLOW BEFORE FE OFF CODE)
C  FE OFF CODE
C  FD
C
C  OR TO FURTHER COMPLICATE MATTERS:
C
C  FD LINE
C  FE ON CODE
C  (SPEEDS MAY OR MAY NOT FOLLOW BEFORE FE OFF CODE)
C  FE OFF CODE
C  FE ON CODE
C  (SPEEDS MAY OR MAY NOT FOLLOW BEFORE FE OFF CODE)
C  ETC. UNTIL NEXT FD
C  FD
C  NOTE:  THE OTHER OCS PROGRAMS ONLY PROVIDE FOR THE
C         CAR TO CHANGE ITS MIND (ON-OFF STATUS) SIX
C         TIMES IN ONE MINUTE!  IF THIS SOUNDS OUTRAGEOUS,
C         READ ON...I SET UP ONE PROGRAM TO DEAL WITH
C         THE CAR CHANGING STATUS ONLY FOUR TIMES IN
C         ONE MINUTE, AND THE VERY FIRST TEST RUN CONTAINED
C         AN INSTANCE WITH FIVE CHANGES.  OF COURSE, THE
C         SAME THING HAPPENED WITH THE MODIFIED VERSION
C         TO DEAL WITH FIVE CHANGES.  SO FAR, NO LOAN
C         HAS BEEN DISCOVERED TO HAVE MORE THAN SIX.
C
C  ANOTHER WRINKLE:  SOMETHIMES (LOOSELY TRANSLATED:  OFTEN)
C                    WHEN THE CAR IS TURNED OFF, TWO SPEED
C                    PAIRS FOLLOW THE FE OFF CODE (PROBABLY
C                    DUE TO SOME BUFFER BEING EMPTIED BY
C                    THE RECORDING COMPUTER).  OTHER OCS SOFT-
C                    WARE HANDLES THIS BY IGNORING THE OFFENDING
C                    SPEED.  THIS PROGRAM WILL FOLLOW SUIT, AND
C                    JUST SPIT IT OUT.
C
C  ALSO, BESIDES FE ON AND OFF CODES, THE DATA CONTAINS
C  URBAN<=>RURAL CHANGE CODES.  FOR EXAMPLE, A CAR CAN BE
C  ON RURAL AND CHANGE TO ON URBAN.  WE WANT TO CONTINUE
C  KEEPING TRACK OF THE SPEEDS FOR A MINUTE AFTER A URBAN
C  RURAL CHANGE WHEN IT OCCURS IN THE MIDDLE OF A MINUTE.
C  THESE CHANGES CAN ALSO OCCUR WHEN THE CAR IS OFF, BUT
C  THEY DO NOT CAUSE ANY MAJOR PROBLEMS LIKE WHEN THE CAR
C  IS ON.
C
C
```

```
C
C  I/O UNITS:
C
C  NUMERIC  MNEUMONIC  PURPOSE
C  -------  ---------  -------
C     4     INPUT      ORIGINAL HEX DATA FILE
C     6     MSINK      MESSAGES TO USER
C     7     OUTPUT     CORRECTED HEX DATA FILE
C     8     MESSAG     UNCORRECTED SUSPECTED ERRORS
C
C
C
C  IMPORTANT VARIABLES:
C
C  NAME    TYPE     DESCRIPTION
C  ----    ----     -----------
C  LINE    CHAR*40  DATA LINE JUST READ
C  FDLINE  CHAR*40  COPY OF LAST FD LINE READ
C  SPEEDS  CHAR*40  ARRAY TO STORE A MINUTE'S SPEEDS
C                   USED ONLY FOR PROBLEMS AROUND FC
C  SNOBS   CHAR*2   ARRAY TO STORE SPEEDS WITHOUT BLANKS
C                   USED TO CHECK SPEEDS
C  TOTLIN  INTEGER  COUNTER FOR # OF LINES OF SPEEDS BETWEEN FD'S.
C  TOTSPD  INTEGER  COUNTER FOR # OF SPEED BYTES
C
C
C
C
C
C  DECLARATIONS
C
      INTEGER INPUT,MSINK,OUTPUT,MESSAG,FIXED
      INTEGER TOTLIN,TOTSPD
C
      CHARACTER*40 SPEEDS(11),LINE,FDLINE
      CHARACTER*2 SNOBS(120) /120*'  '/
      CHARACTER*2 FD /'FD'/, FE /'FE'/, FC /'FC'/
      CHARACTER*2 OA /'OA'/, OB /'OB'/, O2 /'02'/, O3 /'03'/
C
      LOGICAL ON /.FALSE./
C
      COMMON /IO/ INPUT,MSINK,OUTPUT,MESSAG,FIXED
C
C  READ A LINE FROM DATA FILE
C
   10 READ(INPUT,50,END=900)LINE
   50 FORMAT(A)
C
C  CHECK LINE FOR FC, FE, OR FD CODES
C
      IF (LINE(1:2) .EQ. FC) THEN
         WRITE(MESSAG,60) 'FILE GOOD.  FC FOUND HERE: ',FDLINE
   60    FORMAT(' ',2A)
C
C        ODD THINGS HAPPEN AROUND THE FC.  JUST DUMP SPEEDS
C        (WITHOUT CHECKING ANYTHING) AND START COLLECTING THEM
C        AGAIN AFTER THE FC.
C
         DO 80 I=1,TOTLIN
            WRITE(OUTPUT,70) SPEEDS(I)
```

:

```
   70      FORMAT(A)
   80   CONTINUE
C
        TOTLIN=0
        TOTSPD=0
C
        WRITE(OUTPUT,70) LINE
C
C     CHECK FOR FE CODE
C
      ELSEIF (LINE(1:2) .EQ. FE) THEN
C
C        DETERMINE WHICH FE CODE IT IS
C
         IF (LINE(4:5) .EQ. 0A .OR. LINE(4:5) .EQ. 02) THEN
C
C           RE-INITIALIZE SPEED COUNTERS WHEN CAR IS TURNED ON
C           AND BLANK OUT SPEEDS WITHOUT BLANKS ARRAY.  ALSO
C           RE-INITIALIZE IF THIS IS AN URBAN<=>RURAL TYPE CHANGE;
C           BUT, FIRST, CHECK PRECEDING SPEEDS.
C
            IF (ON) THEN
               CALL SPDCHK(SNOBS,FDLINE,TOTLIN,TOTSPD)
            ENDIF
C
            ON=.TRUE.
            TOTLIN=0
            TOTSPD=0
C
            DO 90 I=1,120
               SNOBS(I)='   '
   90       CONTINUE
C
            WRITE(OUTPUT,70) LINE
C
C
         ELSEIF (LINE(4:5) .EQ. 0B .OR. LINE(4:5) .EQ. 03) THEN
C
C           CHECK LAST BLOCK OF SPEEDS WHEN CAR IS TURNED OFF.
C           DO NOT CHECK SPEEDS IS URBAN<=>RURAL TYPE CHANGE.
C
            IF (ON) THEN
               ON=.FALSE.
               CALL SPDCHK(SNOBS,FDLINE,TOTLIN,TOTSPD)
            ENDIF
C
            WRITE(OUTPUT,70) LINE
C
C
         ELSE
C
            WRITE(MSINK,100) 'BAD FE CODE FOUND HERE: ',LINE
  100       FORMAT(' ',2A)
C
            GO TO 999
C
         ENDIF
C
C  CHECK FOR FD CODE
C
```

```
      ELSEIF (LINE(1:2) .EQ. FD) THEN
C
C        IF CAR IS ON THEN CHECK THE LAST BLOCK OF SPEEDS
C
         IF (ON) THEN
C
C           CHECK PRECEDING BLOCK OF SPEEDS
C
            CALL SPDCHK(SNOBS,FDLINE,TOTLIN,TOTSPD)
C
C           THEN RE-INITIALIZE SPEED COUNTERS AND BLANK OUT SNOBS
C
            TOTLIN=0
            TOTSPD=0
C
            DO 110 I=1,120
               SNOBS(I)='  '
  110       CONTINUE
C
         ENDIF
C
         WRITE(OUTPUT,70) LINE
C
C        SAVE THIS FD LINE FOR FUTURE USE
C
         FDLINE=LINE
C
C  NOW, FINALLY, WE ARE SURE WE HAVE A LINE OF SPEED BYTES
C  IF THE CAR IS ON, COLLECT SPEEDS FOR CHECKING IN SNOBS.
C  IF THE CAR IS OFF, JUST SPIT OUT THE LINE.  THIS TAKES
C  CARE OF SITUATIONS WHEN FE OFF CODE IS FOLLOWED BY A
C  SPEED PAIR.
C
      ELSE
C
         IF (ON) THEN
C           STORE LINE IN ARRAY SPEEDS
C           HAVE TO CHECK FOR ODD SITUATIONS AROUND FC
C
            IF (TOTLIN .LT. 11) THEN
               TOTLIN=TOTLIN+1
               SPEEDS(TOTLIN) = LINE
            ENDIF
C
C           NOW, SEE HOW MANY SPEEDS ARE IN THIS LINE
C           FIRST STRIP OFF TRAILING BLANKS, LEAVING ONE.
C
            DO 200 I=40,1,-1
               IF (SPEEDS(TOTLIN)(I:I) .NE. ' ') GO TO 210
  200       CONTINUE
C
  210       LEN=I+1
C
C           TAKE OUT BLANKS BETWEEN SPEED BYTES FOR
C           SUBROUTINE SPDCHK.  AT THE SAME TIME,
C           ADD UP THE NUMBER OF SPEED BYTES.
C
            DO 300 I=3,LEN
               IF (SPEEDS(TOTLIN)(I:I) .EQ. ' ') THEN
                  TOTSPD=TOTSPD+1
```

```
                  SNOBS(TOTSPD)=SPEEDS(TOTLIN)(I-2:I-1)
               ENDIF
 300        CONTINUE
C
         ELSE
            WRITE(OUTPUT,70) LINE
         ENDIF
C
C  THAT'S ALL FOR THIS LINE
C
      ENDIF
C
C  GO GET THE NEXT LINE
C
      GO TO 10
C
C
C  END OF DATA SET
C
 900  WRITE(5,910) 'DATA SET DONE'
 910  FORMAT(' ',A)
C
 999  STOP
      END

:
```

```
C
C
C
      SUBROUTINE SPDCHK(SNOBS,FDLINE,TOTLIN,TOTSPD)
C
C
C  SUBROUTINE TO CHECK THE NUMBER OF SPEEDS IN A BLOCK.
C  ERROR MESSAGES PRINTED TO FILE MESSAG AND CORRECTED SPEEDS
C  TO FILE OUTPUT.
C
C  THE THEORY OF CORRECTING SPEED ERRORS FOLLOWS:
C
C    BASICALLY, THE ONLY TIME WE KNOW DEFINITELY THAT
C    THE SPEEDS ARE CORRECT IS WHEN WE HAVE A "LOW HIGH"
C    PAIR.  EXAMPLE:  01 36 01 48 01 69 01 75 01 A1
C    THESE ARE ALL LOW HIGH PAIRS.  HOWEVER, THESE ARE ONLY
C    CONSIDERED GOOD IF THE LOW OCCURS IN AN "ODD" POSITION
C    AND THE HIGH OCCURS IN AN "EVEN" POSITION.
C
C    ODD   EVEN  WHAT WE KNOW
C    ---   ----  ------------
C    LOW   HIGH  GOOD SPEED PAIR
C    LOW   LOW   WE DON'T KNOW ANYTHING--COULD BE GOOD OR BAD
C    HIGH  LOW   BAD SPEED PAIR
C    HIGH  HIGH  BAD SPEED PAIR
C
C    THE DEFINITION OF LOW AND HIGH DOES NOT CONCERN ITSELF
C    WITH THE RELATIONSHIP BETWEEN THE ELEMENTS OF THE PAIR.
C    LOW IS DEFINED AS 00, 01, 02, 03 AND HIGH IS DEFINED AS
C    EVERYTHING ELSE.  HOPEFULLY, THE CAR WILL NOT GO ABOVE
C    03 FF (127.8 MPH)
C
C
C
C
      INTEGER INPUT,MSINK,OUTPUT,MESSAG,FIXED
      INTEGER TOTLIN,TOTSPD
      INTEGER IFIRST,INEXT,LGOOD,FBAD
C
      CHARACTER*40 FDLINE
      CHARACTER*2 SNOBS(120)
C
      LOGICAL LOW,HIGH
C
      COMMON /IO/ INPUT,MSINK,OUTPUT,MESSAG,FIXED
C
C
C
C  CHECK IF THERE ARE 120 SPEED BYTES (60 PAIRS)
C
      IF (TOTSPD .NE. 120) THEN
C
C        IF NOT, CHECK IF THERE ARE AN EVEN OR ODD NUMBER OF BYTES
C
         IF (TOTSPD/2*2 .NE. TOTSPD) THEN
C
C           PRINT ERROR MESSAGE IF ODD NUMBER OF BYTES
C           AND CONTINUE ON TO TRY AND FIX SPEED ERROR
C
            WRITE(MESSAG,100) 'BAD SPEEDS IN THIS BLOCK: ',FDLINE
 100        FORMAT('0',2A)
```

```
C
C           BEGIN LOOKING FOR UN-LOW HIGH PAIRS
C
            LGOOD=0
            FBAD =0
            IFIRST=1
            INEXT =2
  110       IF (INEXT .GT. TOTSPD) GO TO 180
            LOW =.FALSE.
            HIGH=.FALSE.
C
            LOW=(SNOBS(IFIRST).EQ.'00' .OR. SNOBS(IFIRST).EQ.'01' .OR.
     +           SNOBS(IFIRST).EQ.'02' .OR. SNOBS(IFIRST).EQ.'03')
C
            HIGH=(SNOBS(INEXT).NE.'00' .AND. SNOBS(INEXT).NE.'01' .AND.
     +            SNOBS(INEXT).NE.'02' .AND. SNOBS(INEXT).NE.'03')
C
C           IF LOW HIGH PAIR, WE ARE CONFIDENT THAT SPEED PAIR IS
C           GOOD.  SAVE THE INDEX OF THIS GOOD PAIR FOR LATER USE.
C           THEN INCREMENT COUNTERS AND LOOK FOR MORE LOW HIGHS.
C
            IF (LOW .AND. HIGH) THEN
               IF (FBAD .NE. 0) THEN
                  WRITE(MESSAG,150) 'BAD SPEED BETWEEN SPEED PAIR',
     +                             LGOOD,LGOOD+1,' AND ',IFIRST,INEXT,
     +                            'BAD SPEED PAIR IS',FBAD,FBAD+1
  150             FORMAT(' ',2(A,2I4)/A,2I4)
C
C                 FIX THESE SPEEDS
C
                  CALL SPDFIX(SNOBS,LGOOD,IFIRST,TOTSPD)
C
                  GO TO 180
C
C
               ELSE
                  LGOOD= IFIRST
                  IFIRST=IFIRST+2
                  INEXT =INEXT +2
                  GO TO 110
               ENDIF
C
C           NOW, IF A LOW HIGH PAIR DID NOT OCCUR, WE COULD BE IN
C           TROUBLE.  IF THE PAIR IS A LOW LOW, WE DON'T KNOW
C           ANYTHING, SO CONTINUE PAIRING ALONG UNTIL SOMETHING
C           DEFINITELY GOOD OR BAD OCCURS.  INCREMENT IFIRST AND
C           INEXT BY TWO, KEEPING THE PAIRING PATTERN, BUT DO
C           NOT SET VARIABLE LASTGOOD INDEX.
C
            ELSEIF (LOW .AND. .NOT. HIGH) THEN
               IFIRST=IFIRST+2
               INEXT =INEXT+2
               GO TO 110
C
C           IF THE PAIR IS HIGH LOW OR HIGH HIGH WE KNOW SOMETHING
C           HAS GONE WRONG.  HURRAY!  CHANGE THE PAIRING PATTERN
C           BY SHIFTING IT OVER TO THE RIGHT ONE AND PAIR UP ANOTHER
C           SET TO TRY TO FIND A (GOOD) LOW HIGH.  NOW, WE HAVE TWO
C           LOW HIGH PAIRS SANDWICHING THE ERROR.  SET VARIABLE FIRST BAD.
C
            ELSEIF (.NOT. LOW) THEN
```

```
               FBAD  =IFIRST
               IFIRST=INEXT
               INEXT =INEXT+1
               GO TO 110
            ENDIF
C
C           NOW WE HAVE GONE THROUGH ALL THE SPEEDS IN SEARCH
C           OF A BAD PAIR.  IF ONE WAS NOT FOUND, PRINT WARNING.
C
  180       IF (FBAD .EQ. 0) THEN
               WRITE(MESSAG,185) 'SORRY, UNABLE TO DETECT ERROR'
  185          FORMAT(' ',A)
            ENDIF
C
C
C
C
         ELSE
C
C        PRINT WARNING MESSAGE IF EVEN NUMBER OF BYTES.
C        THERE MAY BE TWO ERRORS IN ONE BLOCK.  OR, THE CAR MAY
C        HAVE BEEN TURNED ON OR OFF IN THE MIDDLE OF A MINUTE.
C        DO NOT TRY TO CORRECT THESE ERRORS, BUT PRINT A WARNING.
C
            WRITE(MESSAG,190) 'LESS THAN 60 SPEEDS IN THIS BLOCK: ',
     +                         FDLINE
  190       FORMAT('0',2A)
         ENDIF
      ENDIF
C
C  IN ANY CASE, PRINT OUT SPEEDS
C  NOTE AWFUL METHOD USED TO PRINT THESE OUT:  PROBLEM ENCOUNTERED
C  WITH REPEAT SPECIFICATIONS IN FORMAT STATEMENTS.  (UNFORTUNATELY
C  FOR US, MTS DOESN'T HAVE ALL THE BUGS OUT OF F77 YET)
C
      I=1
  200 IF (I .GT. TOTSPD) GO TO 210
         WRITE(OUTPUT,205)(SNOBS(J),J=I,I+11)
  205    FORMAT(12(A,' '))
         I=I+12
         GO TO 200
  210 CONTINUE
C
C
C
      RETURN
      END


:
```

```
C
C
C
C
C
C
      SUBROUTINE HEXDEC(HEXIN1,HEXIN2,DECOUT)
C
C   THIS SUBROUTINE CONVERTS A HEX SPEED PAIR INTO A DECIMAL NUMBER.
C
C   FOR EXAMPLE, SUPPOSE HEXIN1 = 01 AND HEXIN2 = 3C HEXDEC WOULD
C   RETURN DECOUT = XXX.
C
C   THE METHOD IS NOT TERRIBLY STRAIGHTFORWARD BECAUSE HEXIN1 AND
C   HEXIN2 ARE HEX CHARACTERS AND DECOUT IS AN INTEGER.  ALSO, THERE
C   ARE PROBLEMS IF THE LEFT-MOST MEMBER OF HEXIN2 IS 8 OR GREATER.
C   THE FAMOUS "B-PROBLEM" ENCOUNTERED IN OCS.S-PREPAR.  THERE IS
C   THE ADDITIONAL B-PROBLEM IN THE RIGHT-MOST MEMBER OF HEXIN1
C   HERE BECAUSE OF THE SHIFTING.  IN "NORMAL" SITUATIONS,
C   HEXIN1 IS ALWAYS BETWEEN 00 AND 03; HOWEVER, SINCE WE
C   ARE SHIFTING THE PAIRING SCHEME, IT IS LIKELY THAT HEXIN1
C   MAY BE "BIG" I.E. NOT BETWEEN 00 AND 03.  THE LONG-HAND WAY OF
C   CONVERTING CHOPS OFF THE LEFT-MOST BINARY DIGIT OF BOTH B
C   PROBLEM HEX DIGITS.  SO A CORRECTION MUST BE MADE.
C
C   TO CONVERT HEX TO DECIMAL THE LONG WAY:
C
C       01     3C     TAKE THE ORIGINAL PAIR
C     0000  0001  0011  1100     CONVERT EACH DIGIT TO BINARY
C            XXX   XXX  XXDD     SEPARATE OUT DECIMAL PART
C                 00101111.00    STICK IT TOGETHER
C                 76543210       CONVERT BACK TO DECIMAL: 2**X
C               32+8+4+2+1.00    ADD UP COMPONENTS
C                       59.00    DONE!
C
C   TO CONVERT HEX TO DECIMAL USING J.P.'S METHOD:
C
C       (128*1 + 3*16 + C) / 4   USE LONG EQUATION
C       (128   + 96   +12) / 4
C                         59.00  DONE!
C
C   THIS SUBROUTINE IS BUILT ON THE J.P. CONCEPT. HOWEVER,
C   IT WILL NOT CONCERN ITSELF WITH THE DIVIDE BY FOUR PART.
C   ALL SPEEDS WILL BE DEALT WITH IN THEIR WHOLE NUMBER
C   COUNTERPARTS.  THIS IS TOUGH ENOUGH WITHOUT HASSLING WITH
C   THE DECIMAL VALUES.  (SUGGESTED BY GLENN THOMPSON)
C
C
C   DECLARATIONS
C
      CHARACTER*2 HEXIN1,HEXIN2
      CHARACTER*1 HEXMAP(16)
C
      INTEGER VAL1(2),VAL2(2)
      INTEGER DECOUT
C
      DATA HEXMAP /'0','1','2','3','4','5','6','7',
     +             '8','9','A','B','C','D','E','F'/
C
C   BEGIN WITH THE FIRST HEX NUMBER
C
```

```
    DO 200 I=1,2
C
      DO 100 J=1,16
        IF (HEXIN1(I:I) .EQ. HEXMAP(J)) GO TO 150
 100  CONTINUE
C
      VAL1(I) = -1
      GO TO 200
C
 150  VAL1(I)=J-1
C
 200 CONTINUE
C
C REPEAT WITH THE SECOND HEX NUMBER
C
    DO 400 I=1,2
C
      DO 300 J=1,16
        IF (HEXIN2(I:I) .EQ. HEXMAP(J)) GO TO 350
 300  CONTINUE
C
      VAL2(I) = -1
      GO TO 400
C
 350  VAL2(I)=J-1
C
 400 CONTINUE
C
C FIX THE RIGHT-MOST MEMBER OF THE FIRST SPEED
C AND THE LEFT-MOST MEMBER OF THE SECOND SPEED.
C
    IF (VAL1(2) .GE. 8) THEN
       VAL1(2)=VAL1(2)-8
    ENDIF
C
    IF (VAL2(1) .GE. 8) THEN
       VAL2(1)=VAL2(1)-8
    ENDIF
C
C NOW DO THE MULTIPLYING
C
    DECOUT= 128*VAL1(2) + 16*VAL2(1) + VAL2(2)
C
C
    RETURN
    END
```

```
C
C
C
C
C
C
C
      SUBROUTINE DECHEX(DECIN,HEX1,HEX2)
C
C   THIS SUBROUTINE CONVERTS A DECIMAL SPEED INTO A HEX SPEED PAIR.
C   IT IS THE INVERSE OF SUBROUTINE HEXDEC.  SEE THAT SUBROUTINE
C   FOR A DETAILED EXPLANATION OF THE HEX SPEED CONVERSION.
C
C
C   DECLARATIONS
C
      CHARACTER*2 HEX1,HEX2
      CHARACTER*1 HEXMAP(16)
C
      INTEGER VAL1(2),VAL2(2)
      INTEGER DECIN
C
      DATA HEXMAP /'0','1','2','3','4','5','6','7',
     +             '8','9','A','B','C','D','E','F'/
C
C   TURN THE DECIMAL NUMBER INTO ITS HEX EQUIVALENT
C
      VAL1(1)=0
      VAL1(2)=DECIN/128
      TEMP   =DECIN-(VAL1(2)*128)
      VAL2(1)=TEMP/16
      VAL2(2)=TEMP-(VAL2(1)*16)
C
C   NOW CONVERT EACH NUMBER INTO A CHARACTER
C
      HEX1(1:1)=HEXMAP(VAL1(1)+1)
      HEX1(2:2)=HEXMAP(VAL1(2)+1)
C
      HEX2(1:1)=HEXMAP(VAL2(1)+1)
      HEX2(2:2)=HEXMAP(VAL2(2)+1)
C
C
      RETURN
      END


:
```

```
C
C
C
C
C
C
C
      SUBROUTINE SPDFIX(SNOBS,IGOOD1,IGOOD2,TOTSPD)
C
C  SUBROUTINE TO FIND INCORRECT SPEED BETWEEN TWO GOOD SPEEDS.
C  NOT AN ALL-TOGETHER EASY TASK, SINCE WE HAVE NO IDEA HOW MANY
C  SPEEDS ARE IN BETWEEN THESE TWO GOOD SPEEDS.
C  EXAMPLE:
C  --------
C    L H H L H                1 POSSIBLE PLACE FOR ERROR
C    L H L L H L H            2 POSSIBLE PLACES FOR ERROR
C    L H L L L L H L H        3 POSSIBLE PLACES FOR ERROR
C    ETC.
C
C
C
C  THE GENERAL IDEA OF SUBROUTINE SPDFIX IS TO TRY ALL POSSIBLE
C  COMBINATIONS OF SPEED PAIRS, EACH TIME CHOOSING A DIFFERENT HALF
C  PAIR TO SKIP.  THE SPECIFIC APPROACH WILL MINIMIZE THE SUM OF
C  THE SQUARES OF THE CHANGES IN THE SPEED.  THIS PROGRAM
C  WILL KEEP A RUNNING SUM OF THE SQUARED CHANGES AND A
C  "RUNNING MINIMUM"  SUM.  AFTER TRYING ALL THE POSSIBLE
C  PAIRING COMBINATIONS, WE WILL KNOW WHICH SPEED IS MISSING
C  AND CAN REPLACE IT WITH THE AVERAGE OF THE TWO GOOD SPEEDS
C  ON EITHER SIDE.
C
C
C
C  HERE IS AN EXAMPLE:
C
C  SUPPOSE OUR DATA LINE SEGMENT LINE LOOKED LIKE THIS:
C    01 5B 01 5C 5A 01 5B
C     L  H  L  H  H  L  H
C
C  NOW IT IS OBVIOUS TO US THAT THE LINE SHOULD BE:
C    01 5B 01 5C 01 5A 01 5B
C     L  H  L  H  L  H  L  H
C
C  BUT, LET'S PRETEND WE ARE A (DUMB) COMPUTER:
C
C  THE POSSIBLE PAIRINGS ARE:
C  1) 01 5B             2) 01 5B
C     -----                01 5C
C     5C 5A                -----
C     01 5B                01 5B
C
C  THESE TRANSLATE INTO:
C  1) 219               2) 219
C     730                  220
C     219                  219
C
C  NOW THE SUM OF THE SQUARED DIFFERENCES ARE:
C  SUM1  = 511**2 + 511**2
C        = HUGE NUMBER
C
C  SUM2 = 1**2 + 1**2
```

```
C       = 2
C
C  SO THE WINNER IS SUM2
C
C  NOW PUT THE AVERAGE OF THE TWO SURROUNDING 'GOOD' PAIRS
C  INTO THE SPOT WHERE ERROR WAS:
C    01 5B 01 5B 01 5B 01 5B
C
C  NOTE:   WHEN WE DO THIS AVERAGING, IT IS VERY LIKELY THAT
C          THE GOOD PART OF THE BAD SPEED (?) WILL BE LOST.
C          LIKE IN THIS EXAMPLE, 5A WAS GOOD BUT WHEN WE
C          AVERAGED 5C AND 5B WE GOT 5B.  ALL IS NOT LOST,
C          WE HAVE DECIDED TO BE HAPPY WITH THIS AVERAGED
C          NUMBER.
C
C
C
C  DECLARATIONS
C
      INTEGER INPUT,MSINK,OUTPUT,MESSAG,FIXED
      INTEGER TOTSPD
      INTEGER DIFF,SHIFTS,AVE,AVE1,AVE2,SUM
      INTEGER DEC1,DEC2
      INTEGER WHICHS
C
      CHARACTER*2 SNOBS(120),TEMP(120),FIX1,FIX2
C
      COMMON /IO/ INPUT,MSINK,OUTPUT,MESSAG,FIXED
C
C  DEBUG
C
C     WRITE(FIXED,5)TOTSPD
C   5 FORMAT(' ',I4)
C     I=1
C   6 IF ( I .GT. TOTSPD) GO TO 8
C        WRITE(FIXED,10)(SNOBS(J),J=I,I+11)
C  10    FORMAT(12(A,' '))
C        I=I+12
C        GO TO 6
C   8 CONTINUE
C
C  BEGIN BY CHECKING THAT THERE ARE INDEED TWO GOOD SPEED PAIRS
C  SENT HERE.
C
      IF (IGOOD1 .EQ. 0) THEN
         WRITE(MESSAG,20)'SORRY, NO FIRST LOW-HIGH PAIR.  CANNOT FIX'
   20    FORMAT(' ',A)
         RETURN
      ENDIF
C
C  NEXT FIND OUT HOW MANY SPEEDS ARE IN BETWEEN TWO
C  GOOD SPEED PAIRS.  THIS MUST BE AN ODD NUMBER.
C
      DIFF=IGOOD2-IGOOD1
      IF (DIFF/2*2 .EQ. DIFF) THEN
         WRITE(MESSAG,50)'SORRY, EVEN NUMBER OF SPEEDS.  CANNOT FIX.'
   50    FORMAT(' ',A)
         RETURN
      ENDIF
C
C  NOW DECIDE HOW MANY DIFFERENT PAIRINGS THERE ARE TO TRY MISSING
```

:

```
C  SPEED IN EACH POSSIBLE PLACE.
C
      SHIFTS=DIFF/2
C
C  NOW TRY THE MISSING SPEED IN EACH PLACE
C  ASSUME EACH IS IN THE 2*I+1 PLACE. I.E. IF SHIFTS=3, THEN TRY
C  THE 3RD, 5TH, AND 7TH SPEEDS.
C
      DO 300 I=1,SHIFTS
         SUM=0
C
C        GET THE DECIMAL EQUIVALENT OF THE FIRST GOOD PAIR
C
         J=IGOOD1
         CALL HEXDEC(SNOBS(J),SNOBS(J+1),DEC1)
C
C        GET THE DECIMAL EQUIVALENT OF THE NEXT PAIR
C
  100    J=J+2
         IF (J .GT. IGOOD2) GO TO 200
C
C           CHECK IF THIS PAIR SHOULD BE SKIPPED
C
            IF (J .EQ. 2*I+IGOOD1) J=J+1
C
C           NOW GET THE DECIMAL EQUIVALENT OF THE NEXT PAIR
C
            CALL HEXDEC(SNOBS(J),SNOBS(J+1),DEC2)
C
C           TAKE THE AVERAGE OF THE TWO SPEEDS
C
            AVE=(DEC1+DEC2)/2
C
C           KEEP A RUNNING SUM OF THE SQUARES OF THE DIFFERENCES
C
            SUM=SUM+((DEC1-AVE)**2 + (DEC2-AVE)**2)
C
C           GET READY FOR THE NEXT PAIR
C
            DEC1=DEC2
         GO TO 100
C
C        NOW KEEP TRACK OF THE SMALLEST SUM
C
 200     IF (I .EQ. 1) THEN
            MIN=SUM
            WHICHS=I
         ELSEIF (SUM .LT. MIN) THEN
            MIN=SUM
            WHICHS=I
         ENDIF
C
C
  300 CONTINUE
C
C
C  NOW WE KNOW WHICH SCENARIO GAVE US THE SMALLEST SUM OF THE
C  SQUARED DIFFERENCES.  ANOTHER PROBLEM ARISES:  WE HAVE TO
C  FIND THE TWO NEAREST GOOD PAIRS SURROUNDING THE ERROR AND
C  CALCULATE THEIR AVERAGE.  THE AVERAGE IS EASY, THE FINDING
C  IS NOT.
```

```
C
      CALL HEXDEC(SNOBS(WHICHS*2+IGOOD1-2),
     +            SNOBS(WHICHS*2+IGOOD1-1),AVE1)
      CALL HEXDEC(SNOBS(WHICHS*2+IGOOD1+1),
     +            SNOBS(WHICHS*2+IGOOD1+2),AVE2)
C
      AVE=(AVE1+AVE2)/2
C
      CALL DECHEX(AVE,FIX1,FIX2)
C
C  NOW INSERT THE CORRECT HEX PAIR INTO THE SPEED ARRAY.
C  PROBLEM:  THE PAIR GOES WHERE A HALF PAIR WAS BEFORE,
C            SO WILL HAVE TO MOVE THE REST OF THE SNOBS
C            ARRAY DOWN ONE.
C
C
C  DEBUG
C
C     WRITE(FIXED,5)TOTSPD
C     WRITE(FIXED,11)MIN,WHICHS,FIX1,FIX2
C  11 FORMAT(' ',2I10,2A)
      SNOBS(WHICHS*2+IGOOD1)=FIX1
C
      DO 400 I=WHICHS*2+IGOOD1+2,TOTSPD+1
         TEMP(I)=SNOBS(I-1)
  400 CONTINUE
C
      DO 450 I=WHICHS*2+IGOOD1+2,TOTSPD+1
         SNOBS(I)=TEMP(I)
  450 CONTINUE
C
      SNOBS(WHICHS*2+IGOOD1+1)=FIX2
C
      TOTSPD=TOTSPD+1
C
C
C
C  DEBUG
C
C     WRITE(FIXED,5)TOTSPD
C     I=1
C 500 IF (I .GT. TOTSPD) GO TO 510
C        WRITE(FIXED,10)(SNOBS(J),J=I,I+11)
C        I=I+12
C        GO TO 500
C 510 CONTINUE
      RETURN
      END


:
```

```
      BLOCK DATA
C
      COMMON /IO/ INPUT,MSINK,OUTPUT,MESSAG,FIXED
C
      INTEGER INPUT,MSINK,OUTPUT,MESSAG,FIXED
C
      DATA IN[illegible] /4/, MSINK /6/, OUTPUT /7/, MESSAG /8/,
     +     FI[illegible] /9/
C
      END
```

OCS.S-PREPAR

```
C  SNH7:OCS.S-PREPAR HOLDS THE NEWEST VERSION OF THE OCS
C  TRANSCRIPTION PROGRAM.  IT CONVERTS HEX DATA INTO
C  DECIMAL VALUES AND WRITES THEM TO THREE FILES:
C    CHART:    PSEUDO-STRIP CHART USED FOR VISUAL
C              EDITING OF SPEED DATA.
C    RESULT:   BLOCK DATA LAYOUT WITH 6 ROWS OF 10
C              SPEEDS FOLLOWING THE ID RECORD.
C    POST:     BLOCK DATA LAYOUT WITH FLAGS TO HELP THE
C              ANALYSIS PROGRAM READ IT.  THIS FILE
C              ONLY CONTAINS DATA BEGINNING THE MINUTE
C              AFTER THE CALIBRATION MODE.
C
C  THIS PROGRAM IS A COMPOSITE OF:
C    1. J.P.CHENG'S ORIGINAL STRIP CHART PROGRAM
C    2. R.L.PARSON'S MODIFICATIONS TO DEAL WITH 5 TEMPERATURES
C    3. M.K.MELICK'S MODIFICATIONS FOR DIFFERENT DATA LAYOUTS
C
C
C
C  THIS PROGRAM ALSO CONTAINS A SECTION TO CORRECT TEMPERATURE
C  DATA:
C    CORRECT = M * SENSOR + B
C      WHERE M = 0.82473
C            B = 0.41840 WHEN SENSOR <= 90
C      AND
C            M = 0.51254
C            B = 28.847  WHEN SENSOR > 90
C
C  THESE COEFFICIENTS WERE DETERMINED FROM MIDAS ANALYSIS OF
C  TEMPERATURE CALIBRATION DATA.
C
C  I/O UNIT ASSIGNMENTS:
C
C    MNEMONIC  UNIT #  FILE ASSIGNMENT  PURPOSE
C    --------  ------  ---------------  -------
C     HEX        4     <HEX DATA FILE>  INPUT HEXIDECIMAL ENGINE DATA
C     ANSWER     5        *MSOURCE*     USER RESPONSES TO PROMPTS
C     ASK        6         *MSINK*      PROMPTS TO USER
C     RESULT     7      <OUTPUT FILE>   CONVERSION RESULTS
C     POST       8      <OUTPUT FILE>   CONVERSION RESULTS PLUS FLAGS
C     CHART      9      <OUTPUT FILE>   STRIP CHART OF SPEEDS
C
C  FLAG ASSIGNMENTS:
C
C   MNEMONIC       HEX    VALUE      EXPLANATION
C   --------       ---    -----      -----------
C
C    FLAGS(1)      FE 0A   1          CAR ON--URBAN
C                  FE 0B   2          CAR OFF---URBAN
C                  FE 02   3          CAR ON--RURAL
C                  FE 03   4          CAR OFF--RURAL
C                  FC      5          CALIBRATION MODE
C                  FF      6          PROBLEM:  FF CODE
C   FLAGS(2)               1-60       NUMBER OF SPEEDS FOLLOWING
C   FLAGS(3,5,7,9.11)      1-4        SWITCH CHANGE CODE
C   FLAGS(4,6,8,10,12)     1-60       NUMBER OF SPEEDS FOLLOWING
C
C  THE ORIGINAL PROGRAM WAS WRITTEN APR 04, 1979 BY J. P. CHENG.  THE
C  LAST VERSION BEFORE THIS ONE WAS COMPLETED MAY 24, 1982 BY SDSB.
C
```

:

```
C  MODIFICATIONS MADE OCT 22, 1982 BY R. PARSONS, CSC, UNDER DELIVERY
C  ORDER 6363-201, SUBTASK 025 (OPERATIONAL CHARACTERISTICS STUDY).
C
C  FURTHER MODIFICATIONS MADE MARCH 8, 1983 BY M. K. MELICK, CSC,
C  UNDER DELIVERY ORDER 6363-201, SUBTASK 025, OPERATIONAL
C  CHARACTERISTICS STUDY.
C
C  TEMPERATURE CORRECTION CODE ADDED APRIL 19,1983 BY M. K. MELICK.
C  BUG IN POSTFL WRITE-OUT FIXED APRIL 26, 1983, MKM.
C
C  01-20-84:  MARY KAY MELICK.  CHANGES MADE TO FACILITATE *BATCH*
C             RUNS OF PREPAR.  SEE LINE 144.
C
C
      INTEGER ASK, ANSWER, HEX, RESULT, POST, CHART
      INTEGER FCSW
      LOGICAL*1 CH(600), SIXTSW
      LOGICAL EQUC
      COMMON /IO/ ASK, ANSWER, HEX, RESULT, POST, CHART
      COMMON /BEGIN/ FCSW
      COMMON /LOGIC/ SIXTSW
C
C  INITIATE RUN.
C
      CALL INIT
C
C  I1:  FIRST CHARACTER'S INDEX RIGHT AFTER "FD" FIELD
C  I2:  LAST  CHARACTER'S INDEX JUST BEFORE "FD" FIELD
C  II:  LAST  CHARACTER'S INDEX WHICH IS NOT A BLANK
C  L1:  FIRST CHARACTER'S INDEX OF A DATA LINE TO BE READ
C  L2:  LAST  CHARACTER'S INDEX OF A DATA LINE TO BE READ
C  IP:  INDEX POINTER FOR STARTING TO SEARCH A FIELD
C
C  LOCATE THE FIRST 'FD' FIELD IN THE DATA FILE.
C  DATA HAS BEEN RE-FORMATTED BY APPLE PROGRAM
C  SO THAT ALL FD, FE, FC CODES BEGIN IN COLUMN 1.
C  ALSO, ONLY ONE FD, FE, OR FC PER LINE.
C
      IP = 1
      L1 = 1
      L2 = 40
   10 READ (HEX,30,END=210) (CH(L),L=1,40)
   30 FORMAT (20(2A1,1X))
      CALL FINDST(CH, 40, 'FD', 2, IP, IFNSH, &10)
C
C  CUT OFF BLANK FIELDS AT RIGHT END OF DATA LINE.
C
      DO 50 K = 1, 20
        II = 40 - 2 * (K - 1)
        IF (EQUC(' ',CH(II))) GO TO 50
        GO TO 70
   50 CONTINUE
C
   70 I1 = IFNSH + 2
      IP = IFNSH + 2
      L2 = II
C
C  THIS SECTION COMMENTED OUT SINCE APPLE RE-FORMATTED
C  DATA DOES NOT HAVE TWO FD'S ON ONE LINE.
C  LOCATE THE SECOND 'FD' FIELD IN THIS DATA LINE.
```

```
C
CMK   CALL FINDST(CH, L2, 'FD', 2, IP, IFNSH, &170)
CMK   GO TO 190
C
C  LOCATE THE NEXT 'FD' FIELD.
C  RESET CHARACTER INDICES FIRST
C
   80 L1 = II + 1
      L2 = II + 40
      IP = L1
   90 READ (HEX,30,END=210) (CH(L),L=L1,L2)
      CALL FINDST(CH, L2, 'FD', 2, IP, IFNSH, &110)
C
C  CUT OFF BLANK FIELDS.
C
  110 DO 130 K = 1, 20
        II = L2 - 2 * (K - 1)
        IF (EQUC(' ',CH(II))) GO TO 130
        GO TO 150
  130 CONTINUE
C
C  IF THE NEXT 'FD' WAS FOUND, PROCESS CHARACTERS BETWEEN THE
C  'FD' FIELDS.   IF NEXT 'FD' WAS NOT FOUND (IFNSH = 0),
C  RESET L1, L2 AND IP (INDEX POINTER), AND CONTINUE SEARCH.
C
  150 IF (IFNSH .NE. 0) GO TO 190
      GO TO 80
C
C  WORK ON THE CHARACTERS BETWEEN THE TWO 'FD' FIELDS.
C
  190 I2 = IFNSH - 1
      CALL OCSOUT(CH, I1, I2)
C
C  BACKUP THE HEX FILE LINEPOINTER ONE LINE TO ALLOW REREAD OF
C  CURRENT LINE AND SET INDICES TO SEARCH FOR NEXT 'FD' FIELD.
C
      BACKSPACE HEX
      I1 = (I2 - L1) + 4
      L1 = 1
      L2 = 40
      IP = I1
      GO TO 90
C
  210 CALL CMD('$SOURCE OCS.C-TSAVE ',20)
      STOP
      END


:
```

```
C
C
C
C
C
      SUBROUTINE INIT
C
C   INIT PROMPTS FOR THE RUN TITLE, WRITES THE RESULT, CHART AND
C   POST TABLE HEADERS, PROMPTS FOR THE NUMBER OF TEMPERATURES AND
C   SETS THE LOGIC SWITCH FOR THE NUMBER READ.
C
      INTEGER ASK, ANSWER, HEX, RESULT, POST, CHART
      INTEGER NTEMPS
      REAL*4 COMM
      LOGICAL*1 SIXTSW, FMT
      LOGICAL*1 AST,BLNK,PT(81)
      DIMENSION COMM(25), FMT(1)
      DATA FMT /'*'/
      COMMON /IO/ ASK, ANSWER, HEX, RESULT, POST, CHART
      COMMON /LOGIC/ SIXTSW
      COMMON /GRAPH/ AST,BLNK,PT
C
C   INPUT COMMENT STATEMENT FOR OUTPUT HEADING.
C
      WRITE (ASK,10)
      READ (ANSWER,30) COMM
   10 FORMAT (' ', 'YOU ARE IN THE OCS PROGRAM, ENTER INPUT COMMENTS',
     +        ' (MAX OF 100 CHARACTERS)', /' ', 10('----:----I'))
   30 FORMAT (25A4)
C
C   WRITE OUTPUT HEADING.
C
      WRITE (CHART,50) COMM
      WRITE (RESULT,50) COMM
   50 FORMAT ('1', 25A4/' ', 'DD HH MM   ID  T1  T2  T3  T4  T5  T6 FF R
     1S ')
C
C   NUMBER OF TEMPERATURES MUST BE 5 OR 6.
C
   70 WRITE (ASK,75)
   75 FORMAT ('0', 'ENTER NUMBER OF TEMPERATURES (5 OR 6):')
      READ (ANSWER,FMT) NTEMPS
      IF (NTEMPS .EQ. 6) GO TO 80
      IF (NTEMPS .NE. 5) GO TO 70
      SIXTSW = .FALSE.
C
C   WRITE NUMBER OF TEMPERATURES TO POST
C
   80 WRITE(POST,85)NTEMPS
   85 FORMAT(' ',I1)
C
C   INTIALIZE ARRAY PT FOR GRAPHICS
C
      DO 90 I=1,81
         PT(I)=BLNK
   90 CONTINUE
C
      RETURN
      END
```

:

```
C
C
C
C
C
      SUBROUTINE OCSOUT(CH, I1, I2)
C
C  THE SUBROUTINE "OCSOUT" SEPARATES, CONVERTS, AND OUTPUTS
C  TIME, VEHICLE ID, TEMPERATURE, SWITCH, AND SPEED VALUES
C  FROM THE HEX DATA FILE.          (BY J. P. CHENG 04/06/79)
C
C  MODIFICATIONS TO WRITE TO FILE POST ASSUME THAT THE FIRST FE CODE
C  FOLLOWS THE FIRST FD.  IF THIS IS NOT THE CASE, IT WILL BE NECESSARY
C  TO CHANGE THE HEX INPUT FILE SO THAT THE FIRST FD PRECEDES THE FE
C  CODE.  THIS WILL NOT CHANGE ANY OF THE STATISTICS SINCE THE POST
C  FILE ONLY CONTAINS DATA STARTING THE MINUTE AFTER THE CALIBRATION
C  MODE.  ASSUME FURTHER THAT THE CALIBRATION MODE FOLLOWS THE FIRST FD.
C
C  IF, IN ANY MINUTE, THERE HAPPENS TO BE MORE
C  THAN 60 SPEEDS, THE POST FILE WILL CONTAIN ONLY THE FIRST
C  60--THE REST BEING IGNORED.
C
      INTEGER ASK, ANSWER, HEX, RESULT, POST, CHART
      INTEGER FCSW
      INTEGER NTEMPS, XX, FLAGS(12)
      INTEGER FLGCT,SAVEFL
      LOGICAL*1 CH(600), IN(14), BLANK,SIXTSW
      LOGICAL*1 AST,BLNK,PT(81)
      DIMENSION ITEMP(6), BUFFER(60), POSTBF(60)
      DATA BLANK /' '/
      DATA XX /'  XX'/
      DATA SAVEFL /1/
      COMMON /IO/ ASK, ANSWER, HEX, RESULT, POST, CHART
      COMMON /BEGIN/ FCSW
      COMMON /LOGIC/ SIXTSW
      COMMON /GRAPH/ AST,BLNK,PT
C
C  ACCORDING TO REQUESTOR, NO SPLIT HEX PAIRS WILL OCCUR IN INPUT DATA:
C  ...00 77 00 70 E0 ... INSTEAD OF ... 00 77 00 7E ...
C  WILL NOT HAPPEN.  THEREFORE THE NUMBER OF CHARACTERS TO BE PROCESSED IS
C  ALWAYS AN EVEN NUMBER.  IN ADDITION, THE NUMBER OF SPEEDS IS ALWAYS A
C  MULTIPLE OF 4, SINCE EACH SPEED IS WRITTEN INTO THE INPUT DATA FILE
C  AS TWO HEX PAIRS.
C
C  INITIALIZE NUMBER OF TEMPERATURES (NTEMPS) AS PER SIXTHSW
C
      NTEMPS = 5
      IF (SIXTSW) NTEMPS = 6
C
C  INITIALIZE FLAGS AND FLGCT
C
      FLGCT=1
      DO 20 I=1,12
        FLAGS(I)=0
   20 CONTINUE
      FLAGS(1)=SAVEFL
C
C  OBTAIN DD-HH-MM, ID1 & ID2 CHARACTERS.
C
C  ASSUME I2 > I1 (SEQUENCE "FD FD" NEVER OCCURS).  THUS, INT ALWAYS 0.
C
```

```
      INT = I2 - I1 + 1
      DO 10 ID = 1, 10
        IN(ID) = BLANK
   10 CONTINUE
C
C  INT >= 10 => STORE (NO CONVERSION REQUIRED) THE FIRST 10 CHARACTERS.
C  IN IN(1) THRU IN(10).  THESE CORRESPOND TO DD HH MM VEID" IN OUTPUT.
C
C  INT < 10 => BAD DATA - DUMP WHAT'S THERE (UNCONVERTED) INTO IN(1)
C  THROUGH IN(INT).
C
      IDTEST = 10
      IF (INT .LT. 10) IDTEST = INT
      DO 30 ID = 1, IDTEST
        IN(ID) = CH(I1 + ID - 1)
   30 CONTINUE
C
C  INITIALIZE TEMPERATURES TO ZERO.
C
      DO 50 IT = 1, 6
        ITEMP(IT) = 0
   50 CONTINUE
C
C  INT < 12 => PROCESSING DONE => TEMPERATURES ALL 0, NO CODES, AND N
C  SPEEDS.
C
      IF (INT .LT. 12) GO TO 370
C
C  FOR SIX TEMPERATURES:
C    INT >= 22 => CONVERT THE NEXT 6 HEX PAIRS TO THE CORRESPONDING 6
C    TEMPERATURES, IN DEGREES CENTIGRADE, AND STORE IN ITEMP(1) THRU
C    ITEMP(6).
C    INT < 22 => MISSING DATA, CONVERT THE (INT-10)/2 PAIRS AND USE THE
C    DEFAULT ZEROES FOR THE MISSING PAIRS.
C
C
C
C  FOR FIVE TEMPERATURES:
C    INT >= 20 => CONVERT THE NEXT 5 HEX PAIRS TO THE CORRESPONDING 5
C    TEMPERATURES. IN DEGREES CENTIGRADE. AND STORE IN ITEMP(1) THRU
C    ITEMP(5). THE OUTPUT FIELD FOR THE SIXTH TEMPERATURE WILL CONTAIN "XX"
C    TO INDICATE ONLY FIVE TEMPERATURES.
C
C   INT < 20 => MISSING DATA, CONVERT THE (INT-10)/2 PAIRS AND USE THE
C   DEFAULT ZEROES FOR THE MISSING PAIRS.
C
      K1 = I1 + 8
C
C  K1 IS THE INDEX OF THE 3RD DIGIT OF THE VEHICLE ID.
C
      ITTEST = NTEMPS
      IF (INT .LT. ((NTEMPS*2) + 10)) ITTEST = (INT - 10) / 2
      DO 70 IT = 1, ITTEST
        K1 = K1 + 2
C
C  K1, K1+1 ARE INDICES OF TWO HEX DIGITS IN "CH" REPRESENTING A
C  TEMPERATURE.
C
C  TEMPERATURE CORRECTION HERE.
C
        CALL CONTWO(CH, K1, K1 + 1, IT1, IT2)
```

```
        ITEMP(IT) = (16*IT1) + IT2 - 57
        RTEMP=FLOAT(ITEMP(IT))
        IF (RTEMP .LE. 90.) RTEMP=(0.82473*RTEMP)+0.4184
        IF (RTEMP .GT. 90.) RTEMP=(0.51254*RTEMP)+28.847
        ITEMP(IT)=IFIX(RTEMP)
   70 CONTINUE
      IF (SIXTSW) WRITE (RESULT,230) (IN(L),L=1,10), ITEMP
      IF (SIXTSW) WRITE (CHART,230) (IN(L),L=1,10), ITEMP
C
      IF ( .NOT. SIXTSW) WRITE (RESULT,330) (IN(L),L=1,10), (ITEMP(L),
     +      L=1,5), XX
      IF ( .NOT. SIXTSW) WRITE (CHART,330) (IN(L),L=1,10), (ITEMP(L),
     +      L=1,5), XX
C
      N = 0
      NTOTAL=N
C
C  FOR SIX TEMPERATURES:
C    INT < 24 => PROCESSING DONE.  INT > 24 => CHECK REMAINING (INT-22)
C    HEX PAIR(S) FOR THE FF, FC, FE, AND SPEED CASES AND PROCESS ACCORDINGLY.
C
C
C  FOR FIVE TEMPERATURES:
C    INT < 22 => PROCESSING DONE.  INT > 22 => CHECK REMAINING (INT-20)
C    HEX PAIRS FOR THE FF, FC, FE, AND SPEED CASES AND PROCESS
C    ACCORDINGLY.
C
      IF (INT .LT. ((NTEMPS*2) + 12)) GO TO 350
   90 IGATE = 0
  110 K1 = K1 + 2
      K2 = K1 + 1
C
C  K1, K2 ARE INDICES OF NEXT PAIR OF HEX DIGITS IN "CH".
C
      IF (K2 + 2 .GT. I2) GO TO 350
      IF (IGATE .NE. 0) GO TO 130
C
      IN(11) = CH(K1)
      IN(12) = CH(K2)
      IN(13) = CH(K1 + 2)
      IN(14) = CH(K2 + 2)
C
C  IF NO 'FF,' 'FC,' OR 'FE' ARE FOUND, THEN THE NEXT
C  HEX DIGITS REPRESENT ONE SPEED DATUM POINT.
C
  130 CALL FINDST(CH, K2, 'FF', 2, K1, IFNSH, &150)
      IF (N .NE. 0) WRITE (RESULT,250) (BUFFER(I),I=1,N)
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=N
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=6
      N = 0
C
      WRITE (RESULT,290)
      WRITE (CHART, 290)
C
      GO TO 90
C
  150 CALL FINDST(CH, K2, 'FC', 2, K1, IFNSH, &170)
      IF (N .NE. 0) WRITE (RESULT,250) (BUFFER(I),I=1,N)
      FLGCT=FLGCT+1
```

```
      FCSW = FCSW+1
      FLAGS(FLGCT)=N
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=5
      N = 0
C
      WRITE (RESULT,310)
      WRITE (CHART, 310)
C
      GO TO 90
C
  170 CALL FINDST(CH, K2, 'FE', 2, K1, IFNSH, &190)
      IH = IFNSH
      IN(11) = CH(IH)
      IN(12) = CH(IH + 1)
      IN(13) = CH(IH + 2)
      IN(14) = CH(IH + 3)
C
      IF (N .NE. 0) WRITE (RESULT,250) (BUFFER(I),I=1,N)
      WRITE (CHART,332) (IN(L),L=11,14)
C
C  DETERMINE FE CODE
C
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=N
      CALL FINDST(CH,K2+2,'0A',2,K1+2,IFND,&174)
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=1
      GO TO 185
C
  174 CALL FINDST(CH,K2+2,'0B',2,K1+2,IFND,&176)
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=2
      GO TO 185
C
  176 CALL FINDST(CH,K2+2,'02',2,K1+2,IFND,&180)
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=3
      GO TO 185
C
  180 CALL FINDST(CH,K2+2,'0C',2,K1+2,IFND,&186)
      FLGCT=FLGCT+1
      FLAGS(FLGCT)=4
  185 N = 0
      WRITE (RESULT,270) (IN(L).L=11,14)
      IGATE = 1
      GO TO 210
  186 WRITE(RESULT,280)
C
C  SEND FIRST PAIR OF SPEED BYTE TO BE CONVERTED
C
  190 CALL CONTWO(CH. K1, K2, IT1, IT2)
      SPEED = IT1 * 16 + IT2
C
      IF (IT1 .GT. 0) GO TO 110
C
C  SEND SECOND PAIR OF SPEED BYTE TO BE CONVERTED.  BUT FIRST,
C  IT MUST BE CHECKED FOR THE '8' PROBLEM    --
C  FIRST PLACE OF SECOND PAIR GETS CUT OFF WHEN CONVERTING MANUALLY.
C  EXAMPLE:  00 B4 => 00000000 10110100 => 00001101.00 => 13.0.
C  CONTWO GIVES (0*128 + 11*16 +4)/4 = 45.  BAD NEWS.
```

```
C  SOLUTION:  CHANGE B4 TO THE EQUIVALENT 34 => 00110100 => 13.0.
C
      CALL CONFIX(CH, K1 + 2, K1 + 3, IT1, IT2)
      SPEED = (SPEED*128 + IT1*16 + IT2) / 4
C
C  WRITE SPEED TO STRIP CHART
C
      IFIXED=(SPEED+1.5)
      IF (IFIXED .GT. 81) IFIXED=81
      PT(IFIXED)=AST
      WRITE(CHART,331)SPEED,PT
C
      PT(IFIXED)=BLNK
C
C  REPEAT DATE AND TIME AFTER 20TH AND 40TH SPEED SO THAT
C  THERE IS ALWAYS AN IDENTIFIER SHOWING ON THE SCREEN.
C
      IF (N .EQ. 19 .OR. N .EQ. 39)
     +    WRITE(CHART,230)(IN(L),L=1,10)
C
C  WRITE SPEED TO BUFFER FOR WRITING TO RESULT AND POST LATER
C
      N=N+1
      BUFFER(N)=SPEED
      IF (NTOTAL+1 .GT. 60) GO TO 210
      NTOTAL=NTOTAL+1
      POSTBF(NTOTAL)=SPEED
C
  210 K1 = K1 + 2
C
      GO TO 110
C
C  FORMAT STATEMENTS
C
  230 FORMAT ('-', 3(2A1,1X), 4A1, 6I4)
  235 FORMAT(' ',2A1,1X,4A1,2X,4A1,6I4,12I4)
  240 FORMAT(' ',2A1,1X,4A1,2X,4A1,5I4,A4,12I4)
  245 FORMAT(' ',20F5.1)
  250 FORMAT (' ', 10F6.1)
  270 FORMAT (' ', 37X, 2(1X,2A1))
  280 FORMAT(' ', '*****       BAD FE CODE        *****')
  290 FORMAT (' ', '*****  "FF" FIELD SHOWS UP   *****')
  310 FORMAT (' ', '*****      CALIB MODE HERE    *****')
  330 FORMAT ('-', 3(2A1,1X), 4A1, 5I4, A4)
  331 FORMAT (' ',F6.1,81A1)
  332 FORMAT (38X,2(1X,2A1))
C
C
C  WRITE FINAL SPEED COUNTS TO FLAGS, THEN SAVE LAST STATUS OF
C  CAR ON/OFF URBAN/RURAL STATUS FOR NEXT MINUTE'S IDENTIFICATION
C
  350 FLGCT=FLGCT+1
      FLAGS(FLGCT)=N
C
      DO 360 I=1,11,2
        IF(FLAGS(I) .LE. 4 .AND. FLAGS(I) .NE. 0) SAVEFL=FLAGS(I)
  360 CONTINUE
C
C  WRITE RESULT AND POST
C
      IF (N .NE. 0) WRITE (RESULT,250) (BUFFER(I),I=1,N)
```

```
C     USER DESIRES HOT STARTS ONLY, AND DATA IS COLD
      IF (HOTCOL .EQ. HOT  .AND. START .NE. 1) GO TO 500

      USER DESIRES COLD START ONLY, AND DATA IS HOT
      IF (HOTCOL .EQ. COLD .AND. START .NE. 2) GO TO 500
C
C  OK, NOW SAFE TO ADD ON SECONDS IN SPEED AND ACCEL BANDS
C
      DO 450 I=1,8
         ISPEED(I)=ISPEED(I)+TSPD(I)
         CSPEED(I)=CSPEED(I)+TSPD(I)
  450 CONTINUE
C
      DO 460 I=1,23
         IACCEL(I)=IACCEL(I)+TACC(I)
         CACCEL(I)=CACCEL(I)+TACC(I)
  460 CONTINUE
C
      DO 480 I=1,8
         DO 470 J=1,23
            IMATRX(J,I)=IMATRX(J,I)+TMTX(J,I)
            CMATRX(J,I)=CMATRX(J,I)+TMTX(J,I)
  470    CONTINUE
  480 CONTINUE
C
      ISUM=ISUM+SEC(1)
      CSUM=CSUM+SEC(1)
C
C  END OF READING/SUMMING
C
  500 CONTINUE
C
C  GO GET MORE DATA
C
      GO TO 300
C
C  FINALLY, DONE READING ALL DATA
C
  600 CALL FINCAL(OLOAN,ISPEED,IACCEL,IMATRX,ISUM)
      CALL FINCAL(LSUM ,CSPEED,CACCEL,CMATRX,CSUM)
      GO TO 999
C
C  READ ERROR
C
  700 WRITE(MSINK,710)LOAN.DATE,TRIPNO(1).TIME(1)
  710 FORMAT('0','OH NO!  READ ERROR ON LOAN ',I5,
     +          ' ',' HEX DAY ',I3,
     +          ' ',' TIME ',I5,
     +          ' ',' TRIP NUMBER ',I3)
      GO TO 999
C
C  EARLY EOF
C
  800 WRITE(MSINK,810)LOAN.DATE(1),TIME(1),TRIPNO(1)
  810 FORMAT('0','OH NO!  UNEXPECTED EOF ON LOAN ',I5,
     +     / ' ',' HEX DAY ',I3,
     +     / ' ',' TIME ',I5,
     +     / ' ',' TRIP NUMBER ',I3)
C
  999 STOP
      END
```

```
C
C
C


      SUBROUTINE INIT(SHORT,LONG,BOX,SUM)
C
C  THIS SUBROUTINE ZEROES OUT ALL RUNNING SUMS
C
      INTEGER SHORT(8),LONG(23),BOX(23,8),SUM
C
C
C
      DO 10 I=1,8
         SHORT(I)=0
   10 CONTINUE
C
      DO 20 I=1,23
         LONG(I)=0
   20 CONTINUE
C
      DO 40 I=1,23
         DO 30 J=1,8
            BOX(I,J)=0
   30    CONTINUE
   40 CONTINUE
C
      SUM=0
C

      RETURN
      END
```

```
C
C
C
C


      SUBROUTINE LINIT(OLOAN,IRTN)
C
C   THIS SUBROUTINE INITIALIZES THE LOAN NUMBER
C
      INTEGER OLOAN
C
      INTEGER TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
      COMMON /IO/ TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
      IRTN=0
C
      READ(TRPIN,100,END=800,ERR=700) OLOAN
  100 FORMAT(88X,I5)
      BACKSPACE TRPIN
      RETURN
C
C
  700 IRTN=10
      WRITE(MSINK,710)
  710 FORMAT('0','READ ERROR TRYING TO INITIALIZE LOAN.',
     +       / ' ','CHECK THAT THE TRIP FILE IS ATTACHED TO',
     +       / ' ','THE CORRECT I/O UNIT NUMBER')
      RETURN
C

  800 IRTN=10
      WRITE(MSINK,810)
  810 FORMAT('0','UNEXPECTED EOF TRYING TO INITIALIZE LOAN.',
     +       / ' ','CHECK THAT THE TRIP FILE IS ATTACHED TO',
     +       / ' ','THE CORRECT I/O UNIT NUMBER')
      RETURN
      END
```

```
C
C
C
      SUBROUTINE PROMPT
C
C  THIS ROUTINE PROMPTS USER FOR THE RUN OPTIONS
C
      INTEGER*2 DEFALT,PARTIC,URBRUR,HOTCOL
      INTEGER*2 YES,NO,DAY,TRIP,URBAN,RURAL,ALL,HOT,COLD
C
      INTEGER TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
      COMMON /IO/ TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
      COMMON /OPTION/ DEFALT,PARTIC,URBRUR,HOTCOL
      COMMON /WORDS/ YES,NO,DAY,TRIP,URBAN,RURAL,ALL,HOT,COLD
C
C
C
   10 WRITE(MSINK,100)
  100 FORMAT('0','YOU ARE IN THE OCS.S-SPEEDS PROGRAM.',
     +      / ' ','THE FOLLOWING ARE DEFAULT VALUES:',
     +      / '0','1.  COMPLETE PARTICIPANT DAYS ONLY',
     +      / ' ','2.  URBAN AND RURAL TRIPS',
     +      / ' ','3.  HOT AND COLD STARTS',
     +      / '0',
     +      / '&','DO YOU WISH TO TAKE THE DEFAULT? (YE/NO): ')
C

      READ(MSOURC,110) DEFALT
  110 FORMAT(A2)
      IF (DEFALT .EQ. YES) GO TO 300
      IF (DEFALT .NE. NO) GO TO 10
C
C
  120 WRITE(MSINK,130)
  130 FORMAT('0','YOU MUST NOW ANSWER THE FOLLOWING QUESTIONS:',
     +      / '0','1.  COMPLETE PARTICIPANT DAYS (LESS DATA)',
     +      / '&','    OR PARTICIPANT TRIPS (MORE DATA)? (DA/TR): ')
C
      READ(MSOURC,110) PARTIC
      IF (PARTIC .NE. DAY .AND. PARTIC .NE. TRIP) GO TO 120
C
  140 WRITE(MSINK,150)
  150 FORMAT('0',
     +      / '&','2.  URBAN, RURAL, OR ALL TRIPS? (UR/RU/AL): ')
C
      READ(MSOURC,110) URBRUR
      IF (URBRUR .NE. URBAN .AND. URBRUR .NE. RURAL .AND.
     +    URBRUR .NE. ALL) GO TO 140
C
  160 WRITE(MSINK,170)
  170 FORMAT('0',
     +      / '&','3.  HOT, COLD, OR ALL STARTS? (HO/CO/AL): ')
C
      READ(MSOURC,110) HOTCOL
      IF (HOTCOL .NE. HOT .AND. HOTCOL .NE. COLD .AND.
     +    HOTCOL .NE. ALL) GO TO 160
C
      IF (PARTIC .EQ. DAY .AND. URBRUR .EQ. ALL .AND.
```

```
[illegible] HOTCOL .EQ. ALL) WRITE (MSINK,180)
 180 FORMAT('0','YOU HAVE JUST SELECTED ALL DEFAULT VALUES.',
    +     / ' ','NEXT TIME JUST ANSWER "YES" TO THE FIRST QUESTION!')
     RETURN

C  ASSIGN DEFAULT VALUES
C
 300 PARTIC=DAY
     URBRUR=ALL
     HOTCOL=ALL
     RETURN
     END


:
```

```
C
C
C

      SUBROUTINE CHECK(DATE,TRIPNO,DAYTR,TIME,SEC,IRTN)
C
C  THIS SUBROUTINE MAKES SURE THAT THE INFORMATION FROM THE TRIP
C  FILE MATCHES THE INFORMATION IN THE SPEEDS FILE.
C
      INTEGER DATE(2),TRIPNO(2),DAYTR(2),TIME(2),SEC(2)
      INTEGER IRTN
C
      INTEGER TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
      COMMON /IO/ TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
C
C
      IRTN=0
C
C
C
      IF (DATE(1)  .EQ. DATE(2)  .AND. TRIPNO(1) .EQ.  TRIPNO(2) .AND.
     +    DAYTR(1) .EQ. DAYTR(2) .AND. TIME(1)   .EQ.  TIME(2)   .AND.
     +    SEC(1)   .EQ. SEC(2))  RETURN
C
C  IF HERE, DATA IN TRIP AND SPEED FILES IS IN WRONG ORDER
C
      IRTN=10
:
      WRITE(MSINK,910)DATE(1),DATE(2),TRIPNO(1),TRIPNO(2),DAYTR(1),DAYTR(2),
     +               TIME(1),TIME(2),SEC(1),SEC(2)
 910  FORMAT('0','OH NO!  THIS PROGRAM HAS BOMBED BECAUSE THE',
     +      / ' ','DATA IN THE TRIP AND SPEED FILES DOES NOT MATCH.'
     +      / ' ','CHECK THE ORDER OF EACH FILE.'
     +      / ' ','HERE ARE SOME CLUES:',
     +      / '0','                  TRIP FILE VALUE    SPEED FILE VALUE',
     +      / ' ','DAY NUMBER              ',  I6,'            ',   I6,
     +      / ' ','TRIP NUMBER             ',  I6,'            ',   I6,
     +      / ' ','DAYS TRIP NUMBER        ',  I6,'            ',   I6,
     +      / ' ','TIME AT START           '.  I6,'            ',   I6,
     +      / ' '.'LEN IN SEC              ',  I6,'            ',   I6)
C
C
C
      RETURN
      END

:
```

```
C
C
C


      SUBROUTINE FINCAL(NUM,SPEED,ACCEL,MATRX,SUM)
C
C  SUBROUTINE TO CALCULATE AND PRINT OUT THE FINAL RESULS
C
      INTEGER NUM,SPEED(8),ACCEL(23),MATRX(23,8),SUM
C
      INTEGER TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
      REAL SPDFR(8)
      REAL ACCFR(23)
      REAL MATFR(23,8)
C
      COMMON /IO/ TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
C  DEBUG HERE
C
      WRITE(DEBUG,1)NUM,(SPEED(I),I=1,8),SUM
    1 FORMAT(I5,8I7,I10)
C
C  CHECK HERE FOR DIVIDE BY ZERO
C
      IF (SUM .EQ. 0) GO TO 800
C
C  OK, SAFE TO DIVIDE

      DO 100 I=1,8
         SPDFR(I)=FLOAT(SPEED(I))/SUM
  100 CONTINUE
C
      DO 200 I=1,23
         ACCFR(I)=FLOAT(ACCEL(I))/SUM
  200 CONTINUE
C
      DO 400 I=1,23
         DO 300 J=1,8
            MATFR(I.J)=FLOAT(MATRX(I,J))/SUM
  300    CONTINUE
  400 CONTINUE
C
C  NOW PRINT OUT THESE FRACTIONS
C
      WRITE(SPDOUT,450)NUM,(SPDFR(I),I=1,8)
  450 FORMAT(I5,8F10.5)
C
      WRITE(ACCOUT,550)NUM,(ACCFR(I),I=1,23)
  550 FORMAT(I5,23F10.5)
C
      DO 700 I=1.23
         WRITE(MTXOUT,650)NUM,(MATFR(I,J),J=1,8)
  650    FORMAT(I5,8F10.5)
  700 CONTINUE


      RETURN
C
C
```

```
C DEAL WITH DIVIDE BY ZERO
C
  800 WRITE(MSINK,810)NUM
  810 FORMAT('0','NO TRIPS OCCURRED WITH THESE CHARACTERISTICS',
     +      /' ','IN LOAN NUMBER ',I5,
     +      /' ','BEWARE:  OUTPUT FILES MAY HAVE "HOLES"')
      RETURN
      END

:
```

```
C
C
C
      BLOCK DATA
C
C
      INTEGER TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
C
      INTEGER*2 DEFALT,PARTIC,URBRUR,HOTCOL
      INTEGER*2 YES,NO,DAY,TRIP,URBAN,RURAL,ALL,HOT,COLD
C
      COMMON /IO/ TRPIN,SPDIN,MSOURC,MSINK,DEBUG,SPDOUT,ACCOUT,MTXOUT
      COMMON /OPTION/ DEFALT,PARTIC,URBRUR,HOTCOL
      COMMON /WORDS/  YES,NO,DAY,TRIP,URBAN,RURAL,ALL,HOT,COLD
C
      DATA TRPIN /2/, SPDIN  /3/, MSOURC /5/, MSINK /6/,
     +     DEBUG /7/, SPDOUT /8/, ACCOUT /9/, MTXOUT /10/
C
      DATA YES /'YE'/, NO /'NO'/, DAY /'DA'/, TRIP /'TR'/,
     +     URBAN /'UR'/, RURAL /'RU'/, ALL /'AL'/,
     +     HOT /'HO'/, COLD /'CO'/
C
C
C
      END

:
```

```
C
      IF (FCSW .LT. 1) GO TO 370
      IF (FCSW .EQ. 1) GO TO 365
C
      IF (SIXTSW) WRITE(POST,235)(IN(I),I=1,10),ITEMP,FLAGS
      IF (.NOT. SIXTSW)
     +WRITE(POST,240)(IN(I),I=1,10),(ITEMP(I),I=1,5),XX,FLAGS
      IF(NTOTAL .EQ. 0) GO TO 370
      L1=1
      L2=0
      DO 262 K=2,12,2
         L2=L2+FLAGS(K)
         IF (FLAGS(K) .NE. 0) WRITE(POST,245)(POSTBF(I),I=L1,L2)
         L1=L2+1
  262 CONTINUE
      GO TO 370
C
  365 FCSW=FCSW+1
C
  370 RETURN
      END
```

```
C
C
C
C
C
      SUBROUTINE CONTWO(CH, C1, C2, N1, N2)
C
C  CONVERT TWO HEX DIGITS TO DECIMAL.
C
      LOGICAL*1 ARR(16), CH(600)
      INTEGER C1, C2, CHFND
C
      DATA ARR /'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A',
     +     'B', 'C', 'D', 'E', 'F'/
C
      N1 = -10000
      N2 = -10000
C
      CALL FINDC(ARR, 16, CH(C1), 1, 1, N1, CHFND, &10)
      N1 = N1 - 1
C
      CALL FINDC(ARR, 16, CH(C2), 1, 1, N2, CHFND, &10)
      N2 = N2 - 1
C
   10 RETURN
      END


:
```

```
C
C
C
C
C
      SUBROUTINE CONFIX(CH, CH1, CH2, DEC1, DEC2)
C
C   SOLUTION TO THE 'B' PROBLEM IN THE FIRST CHARACTER OF SECOND SPEED
C   PAIR. FORMULA IN MAIN DOES NOT WORK CORRECTLY IF FIRST CHARACTER IS IN
C   RANGE 8 TO F (HEX).
C
      LOGICAL*1 ALL(16), PROBLM(8), CH(600)
      INTEGER CH1, CH2, CHFND, DEC1, DEC2
      DATA ALL /'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A',
     +      'B', 'C', 'D', 'E', 'F'/
      DATA PROBLM /'8', '9', 'A', 'B', 'C', 'D', 'E', 'F'/
C
C
      DEC1 = -10000
      DEC2 = -10000
C
C
      CALL FINDC(PROBLM, 8, CH(CH1), 1, 1, DEC1, CHFND, &10)
      DEC1 = DEC1 - 1
      GO TO 30
C
   10 CALL FINDC(ALL, 16, CH(CH1), 1, 1, DEC1, CHFND, &30)
      DEC1 = DEC1 - 1
C
   30 CALL FINDC(ALL, 16, CH(CH2), 1, 1, DEC2, CHFND, &50)
      DEC2 = DEC2 - 1
C
   50 RETURN
      END
```

:

```
C
C
C
C
C
      BLOCK DATA
C
      COMMON /IO/ ASK, ANSWER, HEX, RESULT, POST, CHART
      COMMON /BEGIN/ FCSW
      COMMON /LOGIC/ SIXTSW
      COMMON /GRAPH/ AST, BLNK, PT
C
      INTEGER ASK, ANSWER, HEX, RESULT, POST, CHART
      INTEGER FCSW
      LOGICAL*1 SIXTSW
      LOGICAL*1 AST, BLNK, PT(81)
C
      DATA ASK /6/, ANSWER /5/, HEX /4/, RESULT /7/ , POST /8/,
     +     CHART /9/
      DATA FCSW /0/
      DATA SIXTSW / .TRUE. /
      DATA AST /'*'/, BLNK /' '/
C
      END

:
```

```
C
C
C  OCS.S-SCAN IS A DATA-VERIFICATION PROGRAM FOR USE ON
C  XXX.FLAGS BEFORE RUNNING THE STATS PROGRAM.
C
C  THIS PROGRAM READS THE FLAGS FILE IN A MANNER SIMILAR
C  TO THE STATS PROGRAM.  IF A READ-ERROR OCCURS, THE
C  PROGRAM WILL WRITE A WARNING MESSAGE TO THE ERROR FILE
C  AND MAKE AN ATTEMPT TO RE-READ THE LINE WITH A DIFFERENT
C  FORMAT.  THERE ARE ONLY TWO DATA FORMATS ASSOCIATED WITH
C  THE FLAGS DATA.
C
C  THE PROGRAM WILL CHECK THE FOLLOWING DATA:
C
C  DAY     SHOULD BE SAME AS PREVIOUS DAY EXCEPT AT MIDNIGHT
C  HOUR    SHOULD BE SAME AS PREVIOUS HOUR EXCEPT ON THE HOUR
C  MINUTE  SHOULD BE PREVIOUS+1 EXCEPT WHEN CAR OFF MORE THAN
C          EIGHT HOURS
C  IDCODE  SHOULD BE SAME AS PREVIOUS IDCODE THROUGHOUT ENTIRE FILE
C
C  TO CHECK FOR MISSING SPEEDS AND TEMPERATURES, THE PROGRAM
C  WILL CHECK THE CHANGE IN SPEEDS AND TEMPERATURES:
C
C  SPEED  CHANGE SHOULD BE LESS THAN 7.0 MPH/SEC
C  TEMP   CHANGE SHOULD BE LESS THAN 10.0 DEGREES/MIN
C
C
C  I/O ASSIGNMENTS
C
C  MNEMONIC      UNIT #      FILE ASSIGNMENT        PURPOSE
C  --------      ------      ---------------        -------
C  INPUT           4         <   INPUT FILE  >      FLAGS FILE FROM PREPAR
C  MESSAG          7         <  OUTPUT FILE  >      ERROR/WARNING MESSAGES
C
C
C
C  SEE PROGRAM OCS.S-STATS FOR DETAILED EXPLANATION OF THE
C  THEORY OF THE FLAGS.  THIS PROGRAM WILL USE MOST ASSUMPTIONS
C  MADE IN OCS.S-STATS AND THE SAME GENERAL STRUCTURE.  SOME
C  OF THE EXTRA PROVISIONS MADE IN STATS ARE NOT HERE (EXAMPLE:
C  POSSIBILITY OF 5 OR 6 TEMPERATURES IS NOT HERE- THIS WILL
C  WORK ONLY ON FILES WITH 6 TEMPERATURES.)
C
C
C  MAIN PROGRAM STRUCTURE:
C
C    READ NTEMPS (PROMPTL/FORGET NTEMPS)
C    READ FIRST MINUTE'S ID RECORD
C      INITIALIZE ODAY,OHOUR.OMIN,OID (O=OLD)
C      DETERMINE STATUS OF CAR
C      CHECK FOR STATUS CHANGE INSIDE FIRST MINUTE
C      IF CAR IS ON, CALL SCHECK TO CHECK SPEEDS
C
C    READ NEXT MINUTE'S ID RECORD
C      CHECK DAY,HOUR,MIN.ID.TEMPS AGAINS OLD VERSIONS
C      CHECK FOR STATUS CHANGE BETWEEN MINUTES
C      IF CAR IS ON, CALL CARON TO CHECK SPEEDS
C      CHECK FOR STATUS CHANGE INSIDE THIS MINUTE
C    GET ANOTHER ID RECORD
C
```

```
[redacted]
C
      SUBROUTINE CAROFF(DAY,TIME)
      ----------

C  03-07-84:  MARY KAY MELICK.  MODIFIED SUBROUTINE CAROFF TO FIX
C             BUG WHEN TRIPS CONTINUE DURING MIDNIGHT.  DISCOVERED
C             WITH OHIO DATA #312.  NOTE:  THIS ROUTINE WAS ENTIRELY
C             RESTRUCTURED.
C
C
C
C  SINCE THE RECORDING DEVICE ONLY RECORDS EVERY 20 MINUTES AFTER
C  THE CAR HAS BEEN OFF FOR 9 HOURS, OFFCT(MINS) MUST BE CALCULATED
C  INSTEAD OF SIMPLY INCREMENTED.
C  ALSO, MUST CHECK FOR "ON THE HOUR" AND MIDNIGHT DISCONTINUITIES.
C
      INTEGER DAY,TIME
      INTEGER OFFCT,DIFF,OLDTIM
      INTEGER POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      INTEGER SAVDAY,DAYTRP,COMPL,TYPECT(3),STRTCT(2)
C
      REAL DAYDIS
C
      LOGICAL*1 HOT
      LOGICAL*1 MIDNIT
C
C
      COMMON /IO/ POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ SAVDAY,DAYTRP,COMPL,TYPECT,STRTCT
      COMMON /RDAYT/ DAYDIS
      COMMON /ONTYPE/ HOT
      COMMON /LMIDNT/ MIDNIT
C
C
C
      WRITE(ONOFLG,10) TIME
   10 FORMAT(' ','CAR OFF ',I6)
C
C  SPECIAL HANDLING FOR MIDNIGHT DURING A CAROFF PERIOD .OR.
C  THE FIRST CAROFF MINUTE AFTER MIDNIGHT OCCURRED DURING
C  THE LAST TRIP.
C
   20 IF (TIME .NE. 0 .AND. (.NOT. MIDNIT)) GO TO 30
C
C     WRITE OUT DAY TOTALS AT MIDNIGHT
C
      DAYDIS=DAYDIS/1609.34
C
      WRITE(DAYLG,22)COMPL,SAVDAY,DAYTRP,TYPECT,STRTCT,DAYDIS
   22 FORMAT(8I6,F10.3)
C
      MIDNIT=.FALSE.
C
C     ZERO OUT DAY TOTALS
C
      DAYTRP=0
C
      DO 24 I=1,3
         TYPECT(I)=0
```

```
   r , 40 , 50

   24 CONTINUE
C
      DO 26 I=1,2
         STRTCT(I)=0
   26 CONTINUE
C
      DAYDIS=0.0
      COMPL=1
      SAVDAY=DAY
C
C     CHECK THAT THIS IS NOT THE FIRST MINUTE OF A CAROFF PERIOD
C
      IF (OFFCT .EQ. 0) GO TO 50
      OFFCT=OFFCT+2360-OLDTIM
      GO TO 40
C
C  TIME NOT MIDNIGHT.  CHECK FOR HOUR CHANGEOVER AND 20 MINUTE
C  PROBLEMS BEFORE INCREMENTING OFFCT.
C  BUT FIRST, CHECK THAT THIS IS NOT THE FIRST MINUTE OF A CAROFF PERIOD
C
   30 IF (OFFCT .EQ. 0) GO TO 50
      DIFF=TIME-OLDTIM
      IF (DIFF .GT. 20) DIFF=DIFF-40
      OFFCT=OFFCT+DIFF
C
   40 IF (OFFCT .GE. 480) HOT=.FALSE.
      OLDTIM=TIME
C
      RETURN
C
C  SPECIAL HANDLING FOR FIRST MINUTE DURING CAROFF PERIOD
C
   50 OFFCT=OFFCT+1
      OLDTIM=TIME
      RETURN
      END
```

```
C
C
C
      SUBROUTINE ONOFF(DAY,TIME,OLDST,J,ONCT,TRIPCT)
C     ----------
C
      INTEGER DAY,TIME,OLDST,J,ONCT,TRIPCT
      INTEGER FLAGS(12)
      INTEGER POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      INTEGER OFFCT
      LOGICAL*1 HOT
      DIMENSION FORGET(60)
      COMMON /IO/ POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      COMMON /STATUS/ FLAGS
      COMMON /OFFMIN/ OFFCT
      COMMON /ONTYPE/ HOT
C
      IF (FLAGS(J) .GT. OLDST) GO TO 50
      IF (FLAGS(J) .LT. OLDST)
     +CALL CARON(DAY,TIME,FLAGS(J+1),ONCT,TRIPCT,OLDST)
C
      RETURN
C
C  COMPUTE LAST CARON STATS
C
   50 IF(FLAGS(J+1) .EQ. 0) GO TO 80
C
      JJ=FLAGS(J+1)
      READ(POST,70)(FORGET(I),I=1,JJ)
   70 FORMAT(F6.1,19F5.1)
C
   80 CALL STATON(OLDST)
C
C  RESET ON AND OFF COUNTERS
C
      ONCT=0
      OFFCT=0
      HOT=.TRUE.
C
      CALL CAROFF(DAY,TIME)
C
      RETURN
      END


:
```

```
C
C
C
      SUBROUTINE URBRUR(DAY,TIME,OLDST,K,ONCT,TRIPCT)
      ----------
C
      INTEGER DAY,TIME,OLDST,ONCT,TRIPCT
      INTEGER POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      INTEGER FLAGS(12)
      LOGICAL*1 COMBIN
      COMMON /IO/ POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      COMMON /STATUS/ FLAGS
      COMMON /COMBO/ COMBIN
C
      COMBIN=.TRUE.
C
      IF (FLAGS(K) .GT. OLDST) GO TO 50
      WRITE(ONOFLG,20)
   20 FORMAT(' ','RURAL TO URBAN CHANGE')
      GO TO 70
C
   50 WRITE(ONOFLG,60)
   60 FORMAT(' ','URBAN TO RURAL CHANGE')
C
   70 IF (FLAGS(K) .EQ. 1 .OR. FLAGS(K) .EQ. 3)
     +       CALL CARON(DAY,TIME,FLAGS(K+1),ONCT,TRIPCT,OLDST)
      IF (FLAGS(K) .EQ. 2 .OR. FLAGS(K) .EQ. 4)
     +                         CALL CAROFF(DAY,TIME)
C
      OLDST=FLAGS(K)
      RETURN
      END


:
```

[redacted]

```
C
      BLOCK DATA

      COMMON /I[redacted],DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      COMMON /S[redacted]/ FLAGS
      COMMON /ID[redacted]/ SAVDAY,DAYTRP,COMPL,TYPECT,STRTCT
      COMMON /RDAYT/ DAYDIS
      COMMON /LMIDNT/ MIDNIT
C
      INTEGER POST,DAYLG,TRIPLG,STOPLG,SPEDLG,TEMPLG,ONOFLG
      INTEGER FLAGS(12)
      INTEGER SAVDAY,DAYTRP,COMPL,TYPECT(3),STRTCT(2)
      REAL DAYDIS
      LOGICAL*1 MIDNIT
C
      DATA POST /2/, DAYLG /3/, TRIPLG /4/, STOPLG /7/
      DATA SPEDLG /8/, TEMPLG /9/, ONOFLG /10/
      DATA FLAGS /12*0/
      DATA SAVDAY /0/, DAYTRP /0/, COMPL /0/,
     +     TYPECT /3*0/, STRTCT /2*0/
      DATA DAYDIS /0.0/
      DATA MIDNIT /.FALSE./
C
      END


:
```

Appendix B

Data Reduction Programs

B 2

OCS-STATS.FOR

```
C  OCS.S-STATS IS THE 2 PROCESSOR FOR OCSDATA AFTER IT HAS BEEN
C  THROUGH THE TRANSCRIPTION PROGRAM.  THE TRANSCRIPTION
C  PROGRAM OUTPUT USED BY THIS PROGRAM IS ESSENTIALLY THE SAME
C  AS THE PAGEPRINTER OUTPUT SAVED AS FINALIZED; HOWEVER, THERE IS
C  AN ADDITIONAL FIELD APPENDED TO THE MINUTE IDENTIFICATION
C  LINE--THE FLAGS FIELD.    ALSO, THE SPEEDS ARE IN BLOCKS OF 3 BY 20.
C  IF THERE ARE TWO SEPARATE CARON PERIODS DURING A MINUTE, THE SPEEDS
C  BLOCKS BEGIN AGAIN IN POSITION 1, NEXT LINE.
C
C
C  STRATEGY OF FLAGS:
C  FLAGS(1)-FLAGS(12) CONTAIN INFORMATION TO READ SPEED DATA
C  (IF PRESENT) OR GO DIRECTLY TO NEXT MINUTE IF NO SPEEDS.
C  FLAGS(1),(3),(5),(7),(9),(11) CONTAIN INFORMATION ON DRIVING STATUS
C  I.E. CAR ON/OFF; URBAN/RURAL. FLAGS(2),(4),(6),(81),(10),(12) CONTAIN
C  THE NUMBER OF SPEEDS TO BE READ (IF THE CAR IS ON).
C
C
C  SBM NOTE:  This program could have been improved if the "Flags"
C                 used a  2 X 6 matrix rather than a 1 X 12 matrix
C                 with odd and even values.  This would have improved
C                 program efficiency because iteration could have been
C                 implemented easier.  This is one feature that makes
C                 this program such a tacky one to follow.  This would
C                 also make it easier for all programmers and others
C                 that may look at this code.  The purpose of this
C                 message is for remembering this when the next OCS
C                 project is undertaken using higher-tech equipment
C                 and hopefully a more efficient program as well.
C
C
C  I/O UNIT ASSIGNMENTS:
C                                                                FILE
C  UNIT #      FILE ASSIGNMENT       PURPOSE                     NAME
C  ------      ---------------       -------                     ----
C
C     2       < INPUT FILE  >     DATA FILE WITH FLAGS           FLAGS
C     3       < OUTPUT FILE >     DAY STATISTICS                -DAY
C     4       < OUTPUT FILE >     TRIP STATISTICS               -TRIP
C     6       < OUTPUT FILE >     SPEED/ACCEL BANDS             -CMP
C     7       < OUTPUT FILE >     STOP STATISTICS               -STOP
C     8       < OUTPUT FILE >     SPEED/ACCEL BANDS             -BAND
C     9       < OUTPUT FILE >     TEMPERATURES FOR PLOTTING     -TEMP
C     10      < OUTPUT FILE >     ERROR CHECKING FILE           -DIAG
C
C
C  SPECIFICALLY:
C  NUMERIC           MEANING
C  -------           -------
C     1          CAR ON--URBAN
C     2          CAR OFF--URBAN
C     3          CAR ON--RURAL
C     4          CAR OFF--RURAL
C
C  SOME TYPICAL EXAMPLES:
C     1 60   0   0   0   0   0   0   0   0   0   0
C     1 28   2   0   1  17   0   0   0   0   0   0
C     2  0   0   0   0   0   0   0   0   0   0   0
```

```
C     1 51  2  1  0  0  0  0  0  0  0  0
C     3 12  1 48  0  0  0  0  0  0  0  0
C
C   SBM NOTE:  The above set of numbers is a set of sample
C                 input lines, each containing 12 flags.  The
C                 even ones tell how many speeds to read in
C                 as input on the next lines.  It is important
C                 to know how many speeds will be read in so the
C                 data will be read  correctly in a looped read
C                 statement.  The odd numbered flags tell the
C                 car status as defined above.  Each line represents
C                 one minute.  If the status does not change within the
C                 minute, the unused flags are set to 0.  If the status
C                 does change within the minute, the next pair of flags
C                 tell the status.  If the vehicle is off, no speeds
C                 will be read in.  The OFFTIME is then determined by
C                 setting the total number of points in the line to 60
C                 and subtracting the number of speeds read.  This is
C                 fine when no more than 1 'off engine' status, 2 or 4,
C                 is read on a given line.  If there is greater than 1
C                 off status on a given line, the first is set to 1 since
C                 it had to use up at least one second and the second one
C                 is given what's remaining of the 60 seconds on the
C                 line.  Also, note in the above sample flag lines, on
C                 line 4, there is a number other than 0 associated with
C                 the even numbered flag when the motor is off.  This
C                 number has no real meaning and occurred as a result of
C                 a hardware glitch in the testing.  Perhaps as the speed
C                 was about to be read, the status simultaneously
C                 changed.  These are rare occurrences.  Such speed data
C                 were kept in the input file, but are not used in the
C                 calculations.
C
C  NOTE:  IF CAR IS TURNED OFF ANYTIME DURING A MINUTE,
C            THE SPEEDS WILL NOT TOTAL 60.
C
C  TABLE OF POSSIBLE STATUS CHANGES (- MEANS IMPOSSIBLE COMBINATION)
C
C               1           2           3           4
C      --------------------------------------------
C      |                                              |
C      |                  URBAN        ON              |
C   1  |        X        ON=>OFF    URB=>RUR       -   |
C      |                                              |
C      |      URB                               OFF    |
C   2  |    OFF=>ON      X           -       URB=>RUR  |
C      |                                              |
C      |      ON                                RUR    |
C   3  |    RUR=>URB     -            X       ON=>OFF  |
C      |                                              |
C      |                 OFF         RUR               |
C   4  |        -      RUR=>URB    OFF=>ON      X      |
C      |                                              |
C       --------------------------------------------
C
C  THE WAY THAT THE FLAGS ARE SET UP MAKES DETERMINATION OF THE TYPE
C  OF STATUS CHANGE EASY.  IF THE DIFFERENCE BETWEEN THE CURRENT
C  STATUS AND THE NEW ONE IS ONE (1), THE STATUS CHANGE IS AN ON-
```

```
C  OFF TYPE.  CONVERSELY, IF THE DIFFERENCE IS TWO (2), THE STATUS
C  CHANGE IS AN URBAN-RURAL TYPE.
C
C  MAIN PROGRAM LOGIC:
C    READ FIRST MINUTE'S ID RECORD
C      DETERMINE STATUS
C      CALL CARON/CAROFF ACCORDINGLY
C
C    READ SECOND MINUTE'S ID RECORD
C      LOOP1:
C        IF STATUS CHANGE BETWEEN MINUTE
C        THEN CALL ONOFF/URBRUR ACCORDINGLY
C        ELSE CALL CARON/CAROFF
C
C        IF STATUS CHANGE INSIDE MINUTE
C        THEN CALL ONOFF/URBRUR ACCORDINGLY
C        READ NEXT MINUTE'S ID RECORD
C      END LOOP1
C
C
C  REVISIONS FOLLOW:
C
C  11-29-83:  MARY KAY MELICK.  MODIFIED SUBROUTINE CARON TO FIX
C             BUG WHEN CAR IS TURNED ON DURING A MINUTE AND NO SPEEDS
C             FOLLOW UNTIL THE NEXT MINUTE--ALIAS N=0.  DISCOVERED WITH
C             OHIO DATA #206.
C
C  01-04-84:  MARY KAY MELICK.
C              BUG FOUND:  IF CAR IS OFF FOR AN ENTIRE DAY, NO LINE FOR
C                          THAT DAY APPEARED IN DAYLOG.  DISCOVERED WITH
C                          OHIO DATA #220.
C              REASON:     NO WRITE STATEMENTS IN SUBROUTINE CAROFF AT
C                          THE CHECK FOR MIDNIGHT.
C             SOLUTION:   PUT WRITE STATEMENTS INTO SUBROUTINE CAROFF.
C                         ALSO ADDED TWO NEW COMMON AREAS TO HANDLE
C                         GETTING ALL NEEDED VARIABLES AROUND.
C                           /IDAYT/  INTEGER DAY TOTALS
C                           /RDAYT/  REAL DAY TOTALS
C
C  01-09-84:  MARY KAY MELICK.
C              PROBLEM WITH VARIABLE STPERM (STOPS PER MILE). WHEN
C              DISTANCE IS VERY CLOSE TO ZERO, STOPS/MILE IS
C              RIDICULOUSLY LARGE (OBVIOUSLY).  NEW VARIABLE AVEDBS
C              (AVERAGE DISTANCEC BETWEEN STOPS) REPLACED STPERM AND
C              IS DEFINED AS FOLLOWS:
C               IF STOPCT > 1 THEN AVEDBS=DISTNC/(STOPCT-1)
C               IF STOPCT = 1 THEN AVEDBS=DISTNC/STOPCT
C             DISCOVERED WITH OHIO DATA #220.
C
C  03-07-84:  MARY KAY MELICK.
C             BUG FOUND:  IF TRIP BEGINS BEFORE MIDNIGHT AND ENDS AFTER
C                         MIDNIGHT, ALL DAY TOTALS ARE ZERO FOR FIRST
C                         DAY AND EVERYTHING ADDED ONTO THE NEXT DAY.
C                         DISCOVERED WITH OHIO DATA #312.
C             REASON:     PROGRAM COULD ONLY DEAL PROPERLY WITH MID-
C                         NIGHT IF CAR IS TURNED OFF.
C             SOLUTION:   ADDED LOGICAL VARIABLE MIDNIT IN COMMON AREA
C                         /LMIDNT/ TO SUBROUTINES CARON AND CAROFF.
```

```
C                         WHEN MIDNIGHT OCCURS DURING A TRIP, THE
C                         VARIABLE MIDNIT IS SET TO .TRUE. IN SUBROUTINE
C                         CARON. THE NEXT TIME THE CAR IS TURNED OFF
C                         (END OF THIS TRIP) THE DAY TOTALS ARE WRITTEN
C                         OUT AND ALL VARIABLES ARE REINITIALIZED FOR
C                         THE NEXT (NOW CURRENT) DAY.
C
C  11-01-84:  STEVEN B. MICHLIN
C             BUG FOUND:  SUBROUTINE SPDACC LOGIC IS OBSCURE AND
C                         LIMITED TO SMALL BANDS.  THE ENTIRE SPDACC
C                         SUBROUTINE WILL BE COMPLETELY REWRITTEN ANYWAY
C                         TO PUT SPEED BANDS IN ONE MPH INTERVALS RATHER
C                         THAN TEN.  ALSO, THE ACCELERATION BANDS WILL
C                         BE INCREMENTED BY .25 RATHER THAN .5.  AS IT
C                         TURNS OUT, EVEN THOUGH THE LOGIC OF THE FIRST
C                         VERSION OF SPDACC APPEARS NONETHELESS CORRECT,
C                         THERE WERE SOME POINTS THAT DUE TO ROUNDING
C                         ERROR TEND TO BE OFF BY ONE IN THE
C                         ACCELERATION BANDS.  THIS WAS DUE TO ROUNDING
C                         ERROR.  WHEN REAL NUMBER WERE ADDED TO RESULT
C                         IN AN INTEGER, ROUNDING ERROR OCCURRED.
C                         THIS OCCURRANCE WAS A SURPRISE, BUT WAS TESTED
C                         AND IT WAS FOUND THAT ADDING .01 TO THE 51 IN
C                         THE EQUATION SOLVED THE PROBLEM, EVEN THOUGH
C                         THE NEW SPDACC ROUTINE DOESN'T USE THAT
C                         OBSCURE, MORE EXPENSIVE METHOD.
C             REASON:     TO GIVE GLEN ANOTHER PERSPECTIVE FOR HIS
C                         REPORT.
C             SOLUTION:   REWRITE SPDACC SUBROUTINE WITH NEW ALGORITHM
C                         AND CHANGE SPEED BAND INCREMENTS TO 1SEC AND
C                         ACCELERATION BAND INCREMENTS TO .25 M/H-S.
C             STATUS:     DONE.    11-06-84
C
C
C  11-01-84:  STEVEN B. MICHLIN
C             BUG FOUND:  IF THE CAR IS TURNED OFF AND BACK ON WITHIN
C                         A GIVEN MINUTES TIME, SOME TIME MAY BE
C                         ATTRIBUTED TO THE PREVIOUS MINUTE AND SOME
C                         TO THE NEXT MINUTE.  WHAT TO DO SUCH WITH DATA
C                         MUST BE DEALT WITH, IN ANY CASE.  IS IT A
C                         STALL, A STOP, DATA ERROR, OR WHAT.
C             REASON:     THE CAROFF-ON IS HANDLED ON A CLOCK-MINUTE
C                         BY MINUTE BASIS.  IF THE STOP IS LESS THAN A
C                         MINUTE, THIS DATA IS CONSIDERED AS PART OF
C                         THE SAME TRIP. ALSO, IF THIS OCCURRED AT
C                         STARTUP, THE REAL STARTUP WOULD HAVE BEEN
C                         CONSIDERED AS A HOT START.
C             SOLUTION:   CHECK FOR STALLS  WHEN THE CAR IS TURNED OFF
C                         AND THEN ON WITHIN A MINUTE.  WRITE THEM IN
C                         DAYLOG AND CONSIDER THE DISTANCE, ETC AS
C                         PART OF THE SAME TRIP.  THIS DIFFERENTIATES
C                         IT FROM A STOP. THIS REQUIRES KNOWING WHAT
C                         IS HAPPENING ON THE NEXT DATA LINE.  THIS
C                         THEREFORE REQUIRED THE MAJOR WORK OF
C                         READING 2 LINES AT A TIME TO ALWAYS HAVE THE
C                         VALUES ON THE NEXT LINE AVAILABLE TO CHECK FOR
C                         STALL.  IF A LINE ENDS IN 15 SECONDS OF ENGINE
C                         OFF, YOU DON'T YET KNOW IF IT IS STALL OR STOP
```

```
C                        UNTIL THE FIRST 45 SECONDS OF THE NEXT LINE
C                        HAS BEEN LOOKED AT.  IF THE NEXT 45 SECONDS
C                        ARE ENGINE OFF, THIS IS A STOP BY OUR
C                        DEFINITION, OTHERWISE IT IS STALL.
C             STATUS:
C
C
C      ..........DEFINITIONS OF VARIABLES..........
C
C        DAY............I*4...Julian Day number (1-99).
C        FLAGS(12)......I*4...Odd...1,3,5,7,9,11;driving status
C                                    information; on/off; urban/rural.
C                             Even..2,4,6,8,10,12;number of speeds to
C                                    read if the car is on.
C        HOT............L*1...True when hot start, otherwise false.
C                               initially true.
C        I..............I*4...Counter.
C        ID.............I*4...Vehicle ID number.
C        NTEMPS.........I*4...
C        OFFCT..........I*4...Time (min) car was off before trip.
C        OLDST..........I*4...After 20 minutes off, counts once each
C                               20 minutes (counter).
C        ONCT...........I*4...Length of trip (sec) =TOTSS+GOSC (counter)
C        2...........I*4...Unit 2; Input file; Data file with flags.
C        SAVDAY.........I*4...Specific day. See variation with DAYLOG
C                               value.
C        SIXTSW.........L*1...
C        TEMP(6)........I*4...The six measured temperature data.
C        9.........I*4...Unit 9; Output file; Temperatures for plot
C        TIME...........I*4...24 hour clock time at end of stop.
C        TRIPCT.........I*4...Sequential trip number.
C
C     CALL FTNCMD('ASSIGN 2=SHYX:454.FLAGS;')
C     CALL FTNCMD('ASSIGN 2=454.FLAGS;')
C     CALL FTNCMD('ASSIGN 3=-454.DAY;')
C     CALL CMDNOE('$SET TIME=10', 12)
C     CALL FTNCMD('ASSIGN 4=-454.TRIP;')
C     CALL FTNCMD('ASSIGN 6=-454.CMP;')
C     CALL FTNCMD('ASSIGN 7=-454.STOP;')
C     CALL FTNCMD('ASSIGN 8=-454.SPEED;')
C     CALL FTNCMD('ASSIGN 9=-454.TEMP;')
C     CALL FTNCMD('ASSIGN 10=-454.DIAG;')
C     CALL EMPTYF('3  ')
C     CALL EMPTYF('4  ')
C     CALL EMPTYF('6  ')
C     CALL EMPTYF('8  ')
C     CALL EMPTYF('9  ')
C     CALL EMPTYF('10 ')
C
C     CALL CMDNOE('$EMPTY -454? OK ', 16)
      INTEGER ITIME(6)
      CALL RUNTME( ITIME)
      CALL GUINFO   ( 'SIGNONID',  ICCID  )
      WRITE(3,19)  ICCID,  (ITIME(I),  I  =  1,   6)
      WRITE(4,19)  ICCID,  (ITIME(I),  I  =  1,   6)
      WRITE(6,19)  ICCID,  (ITIME(I),  I  =  1,   6)
      WRITE(7,19)  ICCID,  (ITIME(I),  I  =  1,   6)
      WRITE(8,19)  ICCID,  (ITIME(I),  I  =  1,   6)
```

```
      WRITE(9,19)  ICCID,  (ITIME(I),  I  =  1,   6)
      WRITE(10,19) ICCID,  (ITIME(I),  I  =  1,   6)
 19   FORMAT(10X, A4, 2X, 6A4 //  )
C
      WRITE (3,140)
      WRITE (4,160)
      WRITE (7,180)
      WRITE (9,200)
      WRITE (6,220)
C
C
      LOGICAL*1 SIXTSW, STALFL, GO, HOT
C
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, OLDST, ONCT,
     1        TRIPCT, OFFCT, SAVDAY, DAYTRP, COMPL, TYPECT(3),
     2        STRTCT(2), SPBAND, PRBAND, ACBAND
C
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1       NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2       NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
     3       TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4       ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5       NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ COMPL, DAYTRP, SAVDAY, STRTCT, TYPECT
      COMMON /ONTYPE/ HOT
C
      SIXTSW = .TRUE.
      DATA ONCT /0/, TRIPCT /0/
C
C  ASSUME FIRST TRIP IS A HOT START.  THIS IS LOGICAL, SINCE
C  TECHNICIANS USUALLY HAVE PREPARED THE CAR FOR A NEW RUN
C  --FOR EXAMPLE, DYNO RUNS USED FOR QUALITY CONTROL.
C  ALSO INITIALIZE OFFCT (MINS).
C
      IFSTAL = 0
      NSTRIP = 0
      NSTCNT = 0
      NSTOT = 0
      NSTSEC = 0
      STALFL = .FALSE.
      HOT = .TRUE.
      OFFCT = 0
C
C     .....WRITE TEMPERATURES TO TEMPLOG, AND INITIALIZE SAVDAY
C
C     .....DO FIRST READ.....
C
      READ (2,240) NTEMPS
      IF (NTEMPS .EQ. 5) SIXTSW = .FALSE.
      INIT = 1
      CALL CHECK(TIME,OLDST)
      CALL CHECK(TIME,OLDST)
C     IF (SIXTSW) READ (2,160) DAY, TIME, ID, TEMP, FLAGS
```

```
C     IF ( .NOT. SIXTSW) READ (2,180) DAY, TIME, ID, (TEMP(I),I=1,5),
C    1                                  XX,  FLAGS
      IF (SIXTSW) WRITE (9,300) DAY, TIME, ID, TEMP
      SAVDAY = DAY
C
C        ....DETERMINE STATUS OF FIRST RECORD
C
C        .....THE LINE BELOW IS DONE TO INITIALIZE OLDST AND TO CAUSE
C               ADDRESS 20 TO CAUSE A GO TOO 40.  A GO TO 40 WOULD HAVE
C               BEEN SUFFICIENT.
C
      OLDST = FLAGS(1)
C
C        ....CHECK FOR STATUS CHANGE BETWEEN MINUTES
C
   20 IF (IABS(FLAGS(1) - OLDST) .EQ. 0) GO TO 40
C
      IF (IABS(FLAGS(1) - OLDST) .EQ. 1) CALL ONOFF(TIME, OLDST,  1,
     1     ONCT, TRIPCT)
C
      IF (IABS(FLAGS(1) - OLDST) .EQ. 2) CALL URBRUR(TIME, OLDST, 1,
     1     ONCT, TRIPCT)
C
      GO TO 60
C
C  DO CARON OR CAROFF CALCULATIONS
C
   40 IF (FLAGS(1) .EQ. 1 .OR. FLAGS(1) .EQ. 3) CALL CARON(TIME,
     1     FLAGS(2), ONCT, TRIPCT, OLDST, 1)
C
      IF (FLAGS(1) .EQ. 2 .OR. FLAGS(1) .EQ. 4) CALL CAROFF(TIME)
C
C CHECK FOR STATUS CHANGES INSIDE MINUTE
C
   60 II = 3
   80 IF (FLAGS(II) .EQ. 0) GO TO 120
      IF(NSTAT(1,INCSPD-1).NE. 2.AND.II .LE. NFLAG)  OLDST = FLAGS(II-2)
C
      IF (IABS(FLAGS(II) - OLDST) .EQ. 1) CALL ONOFF(TIME, OLDST, II,
     1     ONCT, TRIPCT)
C
      IF (IABS(FLAGS(II) - OLDST) .EQ. 2) CALL URBRUR(TIME, OLDST, II,
     1     ONCT, TRIPCT)
      IF ( .NOT. (IABS(FLAGS(II) - OLDST) .EQ. 0)) GO TO 100
C
C     .....The status has not changed, indicating
C
C
C
C     OLDST    =    IFSTAL
  100 IF(NSTAT(1,INCSPD-1).NE. 2.AND.II .LE. NFLAG)  OLDST = FLAGS(II)
C 100 IF(ONCT .NE. 0 .AND.(FLAGS(II).EQ.2.OR.FLAGS(II).EQ. 4))
C    1                                                  OLDST =  FLAGS(II)
      IFSTAL  =  0
      II = II + 2
C     IF (II .GT. NFLAG)  GO TO 20
C
      IF (II .LE. 11) GO TO 80
```

```
  120 IF(NSTAT(1,INCSPD - 1) .NE. 2) OLDST = FLAGS(II-2)
C     IF(IFSTAL .NE. 0) OLDST = IFSTAL
C 120 OLDST  =  FLAGS(II-2)
C 120 CONTINUE
C
C  READ NEXT RECORD
C
      CALL CHECK(TIME,OLDST)
C     IF (SIXTSW) READ (2,160,END=120) DAY, TIME, ID, TEMP, FLAGS
C
C     .....ERROR IN OLD PROGRAM FOR < 6 SPEEDS..  SHOULDN'T
C            USE "I", WHICH IS FLAG COUNTER.....
C
C*ERROR
C     IF ( .NOT. SIXTSW) READ (2,180,END=120) DAY, TIME, ID,
C    1(TEMP(I),I=1,5), XX, FLAGS
C*ERROR
C
      IF (SIXTSW) WRITE (9,300) DAY, TIME, ID, TEMP
C
      GO TO 20
C
C
C 120 CALL FINISH
C     STOP
  140 FORMAT ('   COMPL       DAYTRP  TYPECT(2)  STRTCT(1)  DAYDIS '/
     1       '        SAVDAY      TYPECT(1)  TYPECT(3)  STRTCT(2)',
     2       '      LOAN#' )
  160 FORMAT ('  SAVDAY      DAYTRP   OLDST ', '  OFFCT            DISTN
     1C           AVEDBS       TOTSS        NSTRIP      LOAN#'/
     2       '        TRIPCT    SAVTIM  START ',
     3       '      AVESPD             STOPCT           ONCT       GOSC',
     4       '        NSTCNT '/)
  180 FORMAT ('  SAVDAY       DAYTRP      STOPCT ',
     1       '      STOPDI             LOAN#'/
     2       '        TRIPCT       TIME        STOPSC ',
     3       '            GODI         '/)
  200 FORMAT ('DAY       ID        TEMP(2)   TEMP(4)   TEMP(6)'/
     1       '    TIME   TEMP(1)   TEMP(3)   TEMP(5) '/)
  220 FORMAT (' SAVDAY   DAYTRP    ONCT '/'       TRIPCT    SAVTIM ',
     1       '     LOAN# '  )
  240 FORMAT (I2)
  260 FORMAT (I3, I5, I6, 6F4.0, 12I4)
  280 FORMAT (I3, I5, I6, 5F4.0, A4, 12I4)
  300 FORMAT (I2, I5, I6, 6F4.0)
      END
      SUBROUTINE CARON(TIME, N, ONCT, TRIPCT, OLDST, III)
C     ----------
C
C             SBM......CARON STANDS FOR CAR ON.
C  11-29-83:  MARY KAY MELICK.  MODIFIED SUBROUTINE CARON TO FIX
C             BUG WHEN CAR IS TURNED ON DURING A MINUTE AND NO SPEEDS
C             FOLLOW UNTIL THE NEXT MINUTE--ALIAS N=0.  DISCOVERED WITH
C             OHIO DATA #206.
C
C  03-07-84:  MARY KAY MELICK.  MODIFIED SUBROUTINE CARON TO FIX
C             BUG WHEN TRIPS CONTINUE DURING MIDNIGHT.  DISCOVERED
C             WITH OHIO DATA #312.
```

```
C
C   SUBROUTINE CARON LOGIC:
C   READ N SPEEDS
C   IF NEW TRIP
C   THEN INITIALIZE COUNTERS
C
C   DO I=1,N
C     IF STOP
C     THEN INCREMENT STOP COUNTERS:  TOTAL STOPS
C                                    SECONDS AT THIS STOP
C     ELSE INCREMENT GO COUNTERS:  TOTAL GO SECONDS
C
C     CALL SPDACC TO DO SPEED AND ACCEL BAND FREQUENCIES
C
C     INCREMENT COUNTERS:  TOTAL SECONDS
C                          DISTANCE TRAVELED
C   END
C
C   COMPUTE STATS:
C     AVERAGE SPEED(MPH)=DISTANCE(METERS)/TOTAL SECONDS
C                            *2.237(MILE-SEC/METER-HOUR)
C     DISTANCE(MILES)=DISTANCE(METERS)/1609.34(METERS/MILE)
C     AVEDBS=DISTANCE/(TOTAL STOPS - 1)
C     WRITE TO OUTPUT FILE TRIPLOG AND SPEEDLOG
C   END
C
C
C   THE STOP IS DEFINED AS FOLLOWS:
C     ANYTIME THE SPEED DROPS FROM GREATER THAN 10 MPH TO LESS THAN
C     4 MPH AND THEN RETURNS TO GREATER THAN 10 MPH.   HOWEVER, WHEN
C     THE CAR IS PARKED, THE FIRST ACCELERATION AND THE LAST
C     DECELERATION WILL BE CONSIDERED "STOPS."  THE TIME SPENT AT A
C     STOP INCLUDES ONLY THOSE SECONDS WHEN THE SPEED IS BELOW 4 MPH.
C
C   THE DISTANCE BETWEEN STOPS IS DEFINED AS FOLLOWS:
C     THE DISTANCE TRAVELED WHEN THE CAR IS GOING 4 MPH AND GREATER.
C     KEEP TWO COUNTERS:
C       STOPDISTANCE (SPEED < 4)
C       GODISTANCE (SPEED >= 4)
C     THESE COUNTERS ARE RESET TO 0 AT THE END OF EACH STOP.
C
C   STOP SECTION LOGIC (MAIN LOOP):
C     DO I=1,N
C       IF GO=TRUE
C       THEN IF SPEED < 10
C            THEN IF SPEED < 4
C                 THEN GO=FALSE
C                      STOPSEC=STOPSEC+1
C                      STOPDISTANCE=STOPDISTANCE+METERSPEED
C                 ELSE GOSEC=GOSEC+1
C                 GODISTANCE=GODISTANCE+METERSPEED
C            ELSE GOSEC=GOSEC+1
C            GODISTANCE=GODISTANCE+METERSPEED
C
C       ELSE (GO=FALSE)
C            IF SPEED < 4
C            THEN STOPSEC=STOPSEC+1
C                 STOPDISTANCE=STOPDISTANCE+METERSPEED
```

```
C           ELSE IF SPEED >10
C                THEN STOPCT=STOPCT+1
C                     GO=TRUE
C                     TOTAL STOPSEC=TOTALSTOPSEC+STOPSEC
C                     WRITE OUTPUT TO FILE STOPLOG
C                     GOSEC=GOSEC+1
C                     GODISTANCE=GODISTANCE+METERSPEED
C                     STOPSEC=0
C                ELSE GOSEC=GOSEC+1
C                     GODISTANCE=GODISTANCE+METERSPEED
C    END
C
C  OUTPUT FILES CONTENTS
C
C
C  DAYLOG:
C    COMPL:   0=PARTIAL DAY (*F AND *L DAYS)
C             1=COMPLETE DAY
C    SAVDAY:  DAY
C    DAYTRP:  TOTAL NUMBER OF TRIPS PER DAY # SAVDAY
C    URBCT:   NUMBER OF URBAN TRIPS
C    COMBCT:  NUMBER OF COMBINATION TRIPS
C    RURCT:   NUMBER OF RURAL TRIPS
C    HOTCT:   NUMBER OF HOT STARTS
C    COLDCT:  NUMBER OF COLD STARTS
C    DAYDIS:  TOTAL DISTANCE (MILES) TRAVELED DURING DAY # SAVDAY
C
C  TRIPLOG:
C    SAVDAY:  DAY AT BEGINNING OF TRIP
C    TRIPCT:  SEQUENTIAL NUMBER OF TRIP
C    DAYTRP:  SEQUENTIAL NUMBER OF TRIP DURING DAY # SAVDAY
C    SAVTIM:  CLOCK (24-HOUR) TIME AT BEGINNING OF TRIP
C    TYPE:    1=URBAN
C             2=COMBINATION
C             3=RURAL
C    START:   1=HOT START
C             2=COLD START
C    OFFCT:   TIME(MINS) CAR WAS OFF BEFORE TRIP
C    AVESPD:  AVERAGE SPEED DURING TRIP
C    DISTNC:  DISTANCE TRAVELED DURING TRIP
C    STOPCT:  TOTAL NUMBER OF STOPS PER TRIP
C    AVEDBS:  AVERAGE DISTANCE BETWEEN STOPS DURING TRIP
C    ONCT:    LENGTH (SEC) OF TRIP = TOTSS+GOSC
C    TOTSS:   TOTAL STOP SECONDS OF TRIP
C    GOSC:    TOTAL GO SECONDS OF TRIP
C
C  STOPLOG:
C    SAVDAY:  DAY AT BEGINNING OF TRIP # TRIPCT
C    TRIPCT:  SEQUENTIAL TRIP NUMBER ASSOCIATED WITH STOP
C    DAYTRP:  SEQUENTIAL NUMBER OF TRIP DURING DAY # SAVDAY
C    TIME:    TIME AT END OF STOP
C    STOPCT:  SEQUENTIAL NUMBER OF STOP DURING TRIP # TRIPCT
C    STOPSC:  LENGTH (SEC) OF STOP
C    STOPDI:  DISTANCE TRAVELED DURING "STOP"
C    GODI:    DISTANCE TRAVELED WHEN "GOING"
C
C  SPEEDLOG:
C    SAVDAY:  DAY AT BEGINNING OF TRIP # TRIPCT
```

```
C     TRIPCT:  SEQUENTIAL TRIP NUMBER
C     DAYTRP:  SEQUENTIAL NUMBER OF TRIP DURING DAY # SAVDAY
C     SAVTIM:  TIME AT BEGINNING OF TRIP
C     ONCT:    LENGTH (SEC) OF TRIP
C     SPBAND:  1X81 MATRIX OF TIME SPENT IN EACH SPEED BAND
C     PRBAND:  1X43 MATRIX OF TIME SPENT IN EACH ACCELERATION BAND
C     ACBAND:  81X43 MATRIX THAT COMBINES SPBAND AND PRBAND
C
C      ..........DEFINITIONS OF VARIABLES..........
C
C         ACBAND(43,81)...I*4...Acceleration band: see table in subroutine
C                                   SPDACC.
C         AVEDBS.........R*4...Average distance between stops during trip.
C                                   AVEDBS  =  DISTANCE / (total stops - 1)
C         AVESPD.........R*4...Average speed during trip (MPH).
C         COMBIN.........L*1...
C         COMPL..........I*4...COMPL=0 Partial day; COMPL=1 complete day.
C         DAY............I*4...Julian day number (1-99).
C         DAYDIS.........R*4...
C         3....................Unit 3; Output file; Day statistics.
C         DAYTRP...  .....I*4...Trip number or trip number total. Varies
C                                   with DAYLOG.
C         DISTNC.........R*4...Distance travelled during trip (miles).
C         GODI...........R*4...Distance travelled when going.
C         GOSC...........I*4...Total go seconds of trip.
C         I..............L*1...Iteration counter.
C         MIDNIT.........L*1...Set to TRUE when trip occurs during
C                                   midnite.
C         MILSPD(60).....R*4...Speed (MPH). See subroutine SPDACC.
C         GO.........L*1...True when motor is on; False when off.
C         MTRSPD(60).....R*4...Speed (Meters/sec). "METERSPEED"
C         N..............I*4...The number of seconds to be processed
C                                   in the particular loop.
C         OLDST..........I*4...Old flag status. Used to compare with new.
C         ONCT...........I*4...Length of trip (sec) = TOTSS  +  GOSC.
C         10...................Unit 10; Output file; Error checking file.
C         2....................Unit 2; Input file; Data file with flags.
C         PRBAND(43).....I*4...Matrix of time spent in each acceleration
C                                   band.
C         SAVDAY.........I*4...Specific day.  See variation with DAYLOG
C                                   value.
C         SAVTIM.........I*4...24 hour clock time at beginning of trip.
C         SPBAND(81).....I*4...Speed band.  See subroutine SPDACC.
C         8....................Unit 8; Output file; Speed/Accel Bands.
C         START..........I*4...1=hot start.  2=cold start.
C         STOPCT.........I*4...See DAYLOG variation table. The number of
C                                   stops.
C         STOPDI.........R*4...Distance travelled during stop.
C                                   NOTE:THE UNITS OF THIS VARIABLE ARE
C                                            CHANGED DURING PROGRAM EXECUTION.
C         7....................Unit 7; Output file; Stop statistics.
C         STOPSC.........I*4...Length of stop (sec).
C         STRTCT(2)......I*4...
C         TIME...........I*4...24 hour clock time at end of stop.
C         TOTSS..........I*4...Total stop seconds of trip.
C         TRIPCT.........I*4...Sequential trip number.
C         4....................Unit 4; Output file; Trip statistics.
C         TWO............I*4...Contains the Integer "2";Hardly used.
```

```
C                              Will change to "2" where required
C                              and delete later.
C       TYPECT(3)......I*4...
C
      INTEGER STOPCT, SSTEMP, STOPSC, TOTSS, GOSC, SAVTIM, TWO, START
C
      REAL MILSPD(60), MTRSPD(60)
C
      LOGICAL*1 SIXTSW, STALFL, GO, HOT, COMBIN, MIDNIT
C
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, OLDST, ONCT,
     1        TRIPCT, OFFCT, SAVDAY, DAYTRP, COMPL, TYPECT(3),
     2        STRTCT(2), SPBAND, PRBAND, ACBAND
C
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1       NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2       NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
     3       TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4       ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5       NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ COMPL, DAYTRP, SAVDAY, STRTCT, TYPECT
      COMMON /ONTYPE/ HOT
CC
      COMMON /RDAYT/ DAYDIS
      COMMON /COMBO/ COMBIN
      COMMON /LMIDNT/ MIDNIT
C
      DATA TWO /2/
C
      STALFL = .FALSE.
      WRITE (10,340) DAY, TIME, ID
C
C  SET THE MIDNIGHT FLAG IF MIDNIGHT OCCURS DURING A TRIP
C  ONLY IF 'DURING' TRIP, NOT IF TRIP 'BEGINS' AT MIDNIGHT!
C
      IF (TIME .EQ. 0 .AND. ONCT .NE. 0) MIDNIT = .TRUE.
C
C  READ SPEEDS
C
      IF (N .EQ. 0) GO TO 220
C     READ (2,380) (MILSPD(I),I=1,N)
C
C     .....RETRIEVE NEXT SPEEDSET FROM 'SPD' ARRAY.....
C
      IF (FLAGS(INCSPD*2 - 1) .EQ. 2 .OR. FLAGS(INCSPD*2 - 1) .EQ. 4)
     1INCSPD = INCSPD + 1
C
C     .....WATCH THIS (ABOVE) FOR POTENTIAL BUG.....
C
      DO 20 I = 1, N
   20 MILSPD(I) = SPD(INCSPD,I)
C
      INCSPD = INCSPD + 1
```

```
C
C  CHECK IF THIS IS A NEW TRIP
C
      IF (ONCT .NE. 0) GO TO 80
C
C  INITIALIZE COUNTERS FOR EACH CARON PERIOD
C
      NSTSEC    =   NSTSEC    +     NSTCNT
      NSTOT     =   NSTOT     +     NSTRIP
      NSTCNT    =   0
      NSTRIP    =   0
      GOSC      =   0
      STOPSC    =   0
      STOPCT    =   0
      AVEDBS    =   0
      TOTSS     =   0
      GO     =    .FALSE.
      COMBIN    =    .FALSE.
C
C  THE FOLLOWING (JUNK) COMMENTED OUT DURING THE MIDNIGHT FIX:
C
C*MKM IF (TIME .NE. 0) GO TO 31
C*MKM
C*MKM DAYDIS=DAYDIS/1609.34
C*MKM WRITE(3,25)COMPL,SAVDAY,DAYTRP,TYPECT,STRTCT,DAYDIS
C*M25 FORMAT(8I6,F10.3)
C*MKM
C*MKM DAYTRP=0
C*MKM DO 26 I=1,3
C*MKM    TYPECT(I)=0
C*M26 CONTINUE
C*MKM DO 27 I=1,2
C*MKM    STRTCT(I)=0
C*M27 CONTINUE
C*MKM
C*MKM DAYDIS=0.0
C*MKM COMPL=1
C*MKM
C*M31 TRIPCT=TRIPCT+1
C*MKM DAYTRP=DAYTRP+1
C
C  CONTINUE WITH INITIALIZING FOR A CARON PERIOD
C
      AVESPD = 0.0
      NSTOT  = 0
      NSTSEC = 0
      NSTRIP = 0
      NSTCNT = 0.0
      DISTNC = 0.0
      GODI = 0.0
      STOPDI = 0.0
      SAVDAY = DAY
      SAVTIM = TIME
      TRIPCT = TRIPCT + 1
      DAYTRP = DAYTRP + 1
C
C     .....Zero all "band" arrays.....
C
```

```
      DO 60 I = 1, 81
        SPBAND(I) = 0
C
        DO 40 J = 1, 43
          PRBAND(J) = 0
   40   ACBAND(J,I) = 0
C
   60 CONTINUE
C
C  TAKE CARE OF ODD SITUATIONS THAT OCCUR IMMEDIATELY AFTER THE
C  CALIBRATION MODE.
C
      IF (MILSPD(1) .LE. 10.) GO TO 80
      GO = .TRUE.
C
C  MAIN CARON SECTION
C
C
C  CONVERT SPEED FROM MILES/HOUR TO METERS/SECOND TO AID IN
C  COMPUTING DISTANCE--(*0.447).  LATER, CONVERT DISTANCE
C  BACK TO MILES--(/1609.34).
C
C
C
   80 DO 200 I = 1, N
C
C         STOP SECTION
        MTRSPD(I) = MILSPD(I) * 0.447
        IF ( .NOT. GO) GO TO 120
C
        IF (MILSPD(I) .GE. 10. .OR. MILSPD(I) .GE. 4.) GO TO 100
C
C       .....MILSPD(I) IS <  4  .....
C
        GO = .FALSE.
        STOPSC = STOPSC + 1
C
C       .....Temporary write for error checking;used unit 3 for
C              convenience.
C
C       WRITE(3,1000) I, STOPSC, TIME, INCSPD, MILSPD(I),SPD(INCSPD-1,I)
C1000   FORMAT(' I=',I2,' STOPSC=',I4,' TIME=',I3,' INCSPD=',I2,
C    1         ' MILSPD(I)=',F5.1,' SPD=',F5.1)
C
C       .....THE BELOW IS TRUE SINCE WE ARE USING THE SPEED (MET/SEC)
C              TO DISTANCE (METER) OVER A 1 SECOND INTERVAL.....
C
        STOPDI = STOPDI + MTRSPD(I)
        GO TO 180
C
C       .....MILSPD(I) > 4.  .....
C
  100   GOSC = GOSC + 1
        GODI = GODI + MTRSPD(I)
        GO TO 180
C
C       .....STOP CONDITIONS.....
C
```

```
  120   IF (MILSPD(I) .GE. 10.) GO TO 140
        STOPSC = STOPSC + 1
        STOPDI = STOPDI + MTRSPD(I)
C
C       .....Temporary write for error checking;used unit 3 for
C              convenience.
C
C       WRITE(3,1000) I, STOPSC, TIME, INCSPD, MILSPD(I),SPD(INCSPD-1,I)
        GO TO 180
C
CC140   GOSC = GOSC + 1
CC      GODI = GODI + MTRSPD(I)
C
CC      GO TO 160
C
  140   STOPCT = STOPCT + 1
CC160   STOPCT = STOPCT + 1
        TOTSS = TOTSS + STOPSC
        STOPDI = STOPDI / 1609.34
        GODI = GODI / 1609.34
C
C       .....WRITE "STOPLOG" OUTPUT ON UNIT 7.....
C
        WRITE (7,380) SAVDAY, TRIPCT, DAYTRP, TIME, STOPCT, STOPSC,
     1  STOPDI, GODI, ID
        GO = .TRUE.
        STOPSC =  0
        STOPDI =  0.0
        GODI   =  MTRSPD(I) / 1609.34
        GOSC   =  GOSC  +  1
C
C        SPEED BAND SECTION
  180   CALL SPDACC(ONCT, MILSPD(I), SPBAND, PRBAND, ACBAND, ISBSAV,
     1       SPDSAV)
C
C        INCREMENT TRIP COUNTERS SECTION
        ONCT = ONCT + 1
        DISTNC = DISTNC + MTRSPD(I)
        DAYDIS = DAYDIS + MTRSPD(I)
  200 CONTINUE
C
  220 RETURN
C
C
C  COMPUTE STATS FOR A CARON PERIOD
C
C  THE STOP SECTION DOES NOT CATCH STOPS AT ENDS OF TRIPS SINCE
C  SPEED NEVER GOES ABOVE 10 MPH.  FIND THOSE STOPS HERE AND WRITE
C  RESULTS TO STOPLOG AND CORRECT TOTALSTOPSEC FOR TRIPLOG.
C
C     .....CALLED FROM ONOFF.....
C
      ENTRY STATON(OLDST)
      IF (ONCT .GT. 0) AVESPD = DISTNC * 2.237 / ONCT
      DISTNC = DISTNC / 1609.34
      IF (STOPSC .EQ. 0) GO TO 240
      STOPCT = STOPCT + 1
      TOTSS = TOTSS + STOPSC
```

```
      STOPDI = STOPDI / 1609.34
      GODI = GODI / 1609.34
      NSTOT = NSTOT + NSTRIP
      NSTSEC = NSTSEC + NSTCNT
C
C       .....WRITE "STOPLOG" OUTPUT ON UNIT 7.....
C
      WRITE (7,380) SAVDAY, TRIPCT, DAYTRP, TIME, STOPCT, STOPSC,
     1STOPDI, GODI, ID
C
      STOPSC = 0
      STOPDI = 0.0
      GODI = 0.0
      NSTRIP = 0
      NSTCNT = 0
C
  240 IF (STOPCT .GT. 1) AVEDBS = DISTNC / (STOPCT - 1)
      IF (STOPCT .EQ. 1) AVEDBS = DISTNC
      IF (HOT) GO TO 260
      START = 2
      STRTCT(START) = STRTCT(START) + 1
      GO TO 280
C
  260 START = 1
      STRTCT(START) = STRTCT(START) + 1
C
  280 IF (COMBIN) GO TO 300
      TYPECT(OLDST) = TYPECT(OLDST) + 1
C
C       .....WRITE "TRIPLOG" OUTPUT ON UNIT 4.....
C
      WRITE (4,400) SAVDAY, TRIPCT, DAYTRP, SAVTIM, OLDST, START, OFFCT,
     1AVESPD, DISTNC, STOPCT, AVEDBS, ONCT, TOTSS, GOSC, NSTOT, NSTSEC,
     2ID
      GO TO 320
C
  300 WRITE (4,400) SAVDAY, TRIPCT, DAYTRP, SAVTIM, TWO, START, OFFCT,
     1AVESPD, DISTNC, STOPCT, AVEDBS, ONCT, TOTSS, GOSC, NSTOT, NSTSEC,
     2ID
      TYPECT(TWO) = TYPECT(TWO) + 1
C
C
C
C
C
  320 CONTINUE
C
C       .....WRITE "SPEEDLOG" OUTPUT ON UNIT 6 (COMPRESSED).....
C
      WRITE (6,420) SAVDAY, TRIPCT, DAYTRP, SAVTIM, ONCT, ID
      CALL CMPRES(SPBAND, PRBAND, ACBAND)
C
C       .....WRITE "SPEEDLOG" OUTPUT ON UNIT 8.....
C
      WRITE (8,420) SAVDAY, TRIPCT, DAYTRP, SAVTIM, ONCT, ID
      WRITE (8,440) (SPBAND(I),I=1,81)
      WRITE (8,460) (PRBAND(I),I=1,43)
      WRITE (8,480) ((ACBAND(I,J),I=1,43),J=1,81)
```

```
      RETURN
C
C  WRITE LAST (INCOMPLETE) DAY'S STATISTICS TO DAYLOG
C
      ENTRY FINISH
      COMPL = 0
      DAYDIS = DAYDIS / 1609.34
C
C       .....WRITE "DAYLOG" OUTPUT ON UNIT 3.....
C
      WRITE (3,500) COMPL, SAVDAY, DAYTRP, TYPECT, STRTCT, DAYDIS, ID
      RETURN
C
  340 FORMAT (' ', 'CAR ON  ',3(I6,5X) )
  360 FORMAT (F6.1, 19F5.1)
  380 FORMAT (6I6, 2F10.3, I6)
  400 FORMAT (4I6, 2I2, I6, 2F10.3, I6, F10.3, 6I6)
  420 FORMAT (6I6)
  440 FORMAT (' ', 'SPEED  ', 81I6)
  460 FORMAT (' ', 'ACCEL  ', 43I5)
  480 FORMAT (' ', 'COMBIN ', 43I5)
  500 FORMAT (8I6, F10.3, I6)
      END
      SUBROUTINE CAROFF(TIME)
C     ----------
C
C      SBM....CAROFF STANDS FOR CAR OFF.
C
C  03-07-84:  MARY KAY MELICK.  MODIFIED SUBROUTINE CAROFF TO FIX
C             BUG WHEN TRIPS CONTINUE DURING MIDNIGHT.  DISCOVERED
C             WITH OHIO DATA #312.  NOTE:  THIS ROUTINE WAS ENTIRELY
C             RESTRUCTURED.
C
C  For the first 8 hours of "caroff", data is recorded once per
C  minute.  Then after 20 minutes, data is recorded every 20 minutes.
C
C  SINCE THE RECORDING DEVICE ONLY RECORDS EVERY 20 MINUTES AFTER
C  THE CAR HAS BEEN OFF FOR 8 HOURS, OFFCT(MINS) MUST BE CALCULATED
C  INSTEAD OF SIMPLY INCREMENTED.
C  ALSO, MUST CHECK FOR "ON THE HOUR" AND MIDNIGHT DISCONTINUITIES.
C
C     ..........DEFINITIONS OF VARIABLES..........
C
C       COMPL..........I*4...COMPL=0 Partial day; COMPL=1 Complete day.
C       DAY............I*4...Julian day number (1-99).
C       DAYDIS.........R*4...
C       3....................Unit 3; Output file; Day statistics.
C       DAYTRP.........I*4...Trip number or trip number total. Varies
C                              with DAYLOG.
C       DIFF...........I*4...TIME - OLDTIM. If > 20, DIFF = DIFF - 40
C       MIDNIT.........L*1...Set to TRUE when trip occurs during
C                              midnight.
C       OFFCT..........I*4...The amount of time the car is off.
C       OLDTIM.........I*4...The old time.  This is to be compared with
C                              the new.
C       10...................Unit 10; Output file; Error checking file.
C       SAVDAY.........I*4...Specific day. See variationwith DAYLOG
C                              value.
```

```
C        STRTCT(2).......I*4...
C        TIME............I*4...24 hour clock time at end of stop.
C        TYPECT(3).......I*4...
C
C
      INTEGER DIFF, OLDTIM

C
C
      LOGICAL*1 SIXTSW, STALFL, GO, HOT, MIDNIT
C
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, OLDST, ONCT,
     1        TRIPCT, OFFCT, SAVDAY, DAYTRP, COMPL, TYPECT(3),
     2        STRTCT(2), SPBAND, PRBAND, ACBAND
C
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1       NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2       NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
     3       TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4       ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5       NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ COMPL, DAYTRP, SAVDAY, STRTCT, TYPECT
      COMMON /ONTYPE/ HOT
C
C
      COMMON /RDAYT/ DAYDIS
      COMMON /LMIDNT/ MIDNIT
C
C
C
      STALFL  =  .FALSE.
      IF ( .NOT. STALFL) WRITE (10,140) DAY, TIME, ID
C
C  SPECIAL HANDLING FOR MIDNIGHT DURING A CAROFF PERIOD .OR.
C  THE FIRST CAROFF MINUTE AFTER MIDNIGHT OCCURRED DURING
C  THE LAST TRIP.
C
   20 IF (TIME .NE. 0 .AND. ( .NOT. MIDNIT)) GO TO 80
C
C     WRITE OUT DAY TOTALS AT MIDNIGHT
C
      DAYDIS = DAYDIS / 1609.34
C
      WRITE (3,160) COMPL, SAVDAY, DAYTRP, TYPECT, STRTCT, DAYDIS, ID
C
      MIDNIT = .FALSE.
C
C     ZERO OUT DAY TOTALS
C
      DAYTRP = 0
C
      DO 40 I = 1, 3
   40 TYPECT(I) = 0
```

```
C
      DO 60 I = 1, 2
   60 STRTCT(I) = 0
C
      DAYDIS = 0.0
      COMPL = 1
      SAVDAY = DAY
C
C     CHECK THAT THIS IS NOT THE FIRST MINUTE OF A CAROFF PERIOD
C
      IF (OFFCT .EQ. 0) GO TO 120
      OFFCT = OFFCT + 2360 - OLDTIM
      GO TO 100
C
C TIME NOT MIDNIGHT.  CHECK FOR HOUR CHANGEOVER AND 20 MINUTE
C PROBLEMS BEFORE INCREMENTING OFFCT.
C BUT FIRST, CHECK THAT THIS IS NOT THE FIRST MINUTE OF A CAROFF PERIOD
C
   80 IF (OFFCT .EQ. 0) GO TO 120
      DIFF = TIME - OLDTIM
      IF (DIFF .GT. 20) DIFF = DIFF - 40
      OFFCT = OFFCT + DIFF
C
  100 IF (OFFCT .GE. 480) HOT = .FALSE.
      OLDTIM = TIME
      INCSPD = INCSPD    +      1
C
      RETURN
C
C SPECIAL HANDLING FOR FIRST MINUTE DURING CAROFF PERIOD
C
  120 OFFCT = OFFCT + 1
      OLDTIM = TIME
      INCSPD = INCSPD  +  1
      RETURN
C
  140 FORMAT (' ', 'CAR OFF ', I6, 2(5X, I6) )
  160 FORMAT (8I6, F10.3, I6)
      END
      SUBROUTINE ONOFF(TIME, OLDST, J, ONCT, TRIPCT)
C     ----------
C
C     SBM......ONOFF STANDS FOR 'ON OFF'.
C
C     ..........DEFINITIONS OF VARIABLES.........
C
C       DAY............I*4...Julian day number (1-99).
C       FLAGS(12)......I*4...Odd...1,3,5,7,9,11;driving status
C                                    information; on/off; urban/rural.
C                            Even..2,4,6,8,10,12;number of speeds to
C                                    read if the car is on.
C       FORGET(60).....R*4...Dummy variable read in.  Number of fields
C                              varies.
C       HOT............L*1...
C       JJ.............I*4...Used as FLAGS(J+1) substitution once.
C                              Could be changed.
C       OFFCT..........I*4...
C       OLDST..........I*4...Old onoff status to be compared with new.
```

```
C       ONCT...........I*4...Length of trip (sec) = TOTSS  +  GOSC.
C       2....................Unit 2; Input file; Data file with flags.
C       TIME...........I*4...24 hour clock time at begining of trip.
C       TRIPCT.........I*4...Sequential trip number.
C
      DIMENSION FORGET(60)
C
C
      LOGICAL*1 SIXTSW, STALFL, GO, HOT
C
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, OLDST, ONCT,
     1        TRIPCT, OFFCT, SAVDAY, DAYTRP, COMPL, TYPECT(3),
     2        STRTCT(2), SPBAND, PRBAND, ACBAND, STOPSC
C
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1       NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2       NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
     3       TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4       ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5       NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ COMPL, DAYTRP, SAVDAY, STRTCT, TYPECT
      COMMON /ONTYPE/ HOT
C
C
      IF (FLAGS(J) .GT. OLDST) GO TO 20
      IF (FLAGS(J) .LT. OLDST) CALL CARON(TIME, FLAGS(J + 1), ONCT,
     1     TRIPCT, OLDST, J)
C
      RETURN
C
C  COMPUTE LAST CARON STATS
C
   20 STALFL  =  .FALSE.
      IF (NSTAT(1,INCSPD) .EQ. 2) STALFL = .TRUE.
      IF ( .NOT. STALFL) GO TO 40
C
C       .....WRITE "DIAGNOSTICSLOG" OUTPUT ON UNIT 10.....
C
C     WRITE(10,2300) INCSPD,TIME,NFLAG,FLAGS(NFLAG),FLAGS(NFLAG-1),
C    1   NSTAT(1,INCSPD),NSTAT(2,INCSPD)
C2300 FORMAT(' INCSPD=',I12,' TIME=',I5,' NFLAG=',I3,' FLAGS(NFLAG)=',
C    1        I3,/' FLAGS(NFLAG-1)=',I3,' NSTAT(1,INCSPD=',I10,
C    2         ' NSTAT(2,INCSPD=', I10/)
C
      WRITE (10,80) TIME, NSTAT(2,INCSPD)
      IF(STOPSC .LT. 0)  STOPSC =  0
      STOPSC  =  STOPSC  +  NSTAT(2,INCSPD)
      GO  =  .FALSE.
      ONCT = ONCT + NSTAT(2,INCSPD)
      IF(NFLAG   .NE. J+1)     OLDST    =  FLAGS(J)
      IFSTAL  =  OLDST
C
      NSTRIP = NSTRIP + 1
```

```
      NSTCNT = NSTCNT + NSTAT(2,INCSPD)
      INCSPD = INCSPD + 1
      GO = .FALSE.
      RETURN
   40 IF (FLAGS(J + 1) .EQ. 0) GO TO 60
C
C
C     JJ = FLAGS(J + 1)
C     READ (2,60) (FORGET(I),I=1,JJ)
C
C
   60 CALL STATON(OLDST)
C
C     IF( NSTAT(1,J/2) .EQ. 2)  GO TO 50
C
C
C  RESET ON AND OFF COUNTERS
C
      ONCT = 0
      OFFCT = 0
      HOT = .TRUE.
C
      CALL CAROFF(TIME)
C  50 CALL CAROFF( TIME)
C
      RETURN
   80 FORMAT (' CAR STALL AT', I6, ' FOR ', I4, ' SECONDS')

  100 FORMAT (F6.1, 19F5.1)
      END
      SUBROUTINE URBRUR(TIME, OLDST, K, ONCT, TRIPCT)
C     ----------
C
C     SBM......URBRUR STANDS FOR URBAN/RURAL MODES.
C     ..........DEFINITIONS OF VARIABLES..........
C
C       COMBIN.........L*1...
C       DAY............I*4...Julian day number (1-99).
C       FLAGS(12)......I*4...Odd...1,3,5,7,9,11;driving status
C                                     information; on/off; urban/rural.
C                             Even..2,4,6,8,10,12;number of speeds to
C                                     read if the car is on.
C       OLDST..........I*4...Old on/off status to be compared to new.
C       ONCT...........I*4...Length of trip (sec) = TOTSS  +  GOSC
C       10...................Unit 10; Output file; Error checking file.
C       TIME...........I*4...24 hour clock time at end of stop.
C       TRIPCT.........I*4...Sequential trip number.
C
C
      LOGICAL*1 SIXTSW, STALFL, GO, HOT, COMBIN
C
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, OLDST, ONCT,
     1        TRIPCT, OFFCT, SAVDAY, DAYTRP, COMPL, TYPECT(3),
     2        STRTCT(2), SPBAND, PRBAND, ACBAND
C
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1       NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2       NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
```

```
     3        TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4        ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5        NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ COMPL, DAYTRP, SAVDAY, STRTCT, TYPECT
      COMMON /ONTYPE/ HOT
C
C
      COMMON /COMBO/ COMBIN
C
      COMBIN = .TRUE.
C
      IF (FLAGS(K) .GT. OLDST) GO TO 20
C
C       .....WRITE "DIAGNOSTICSLOG" OUTPUT ON UNIT 10.....
C
      WRITE (10,60)
      GO TO 40
C
   20 WRITE (10,80)
C
   40 IF (FLAGS(K) .EQ. 1 .OR. FLAGS(K) .EQ. 3) CALL CARON(TIME,
     1     FLAGS(K + 1), ONCT, TRIPCT, OLDST, K)
      IF (FLAGS(K) .EQ. 2 .OR. FLAGS(K) .EQ. 4) CALL CAROFF(TIME)
C
      OLDST = FLAGS(K)
      RETURN
   60 FORMAT (' ', 'RURAL TO URBAN CHANGE')
   80 FORMAT (' ', 'URBAN TO RURAL CHANGE')
      END
      BLOCK DATA
C
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1        NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2        NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
     3        TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4        ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5        NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
      COMMON /OFFMIN/ OFFCT
      COMMON /IDAYT/ COMPL, DAYTRP, SAVDAY, STRTCT, TYPECT
      COMMON /ONTYPE/ HOT
C
      COMMON /RDAYT/ DAYDIS
      COMMON /LMIDNT/ MIDNIT
C
C
      LOGICAL*1 SIXTSW, STALFL, GO, HOT, MIDNIT
C
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, OLDST, ONCT,
     1         TRIPCT, OFFCT, SAVDAY, DAYTRP, COMPL, TYPECT(3),
```

```
     2        STRTCT(2), SPBAND, PRBAND, ACBAND
C
C
      DATA FLAGS /12*0/, SAVDAY /0/, DAYTRP /0/, COMPL /0/, TYPECT /3*0/
     1     , STRTCT /2*0/, DAYDIS /0.0/, MIDNIT / .FALSE. /
C
      END
      SUBROUTINE SPDACC(ONCT, RSPEED, SPBAND, PRBAND, ACBAND, ISBSAV,
     1           SPDSAV)
C
C     THIS SUBROUTINE WAS TOTALLY REWRITTEN BY STEVEN B. MICHLIN
C     USING A CHEAPER, MORE EFFICIENT AND MORE ACCURATE ALGORITHM
C     THAN THAT PREVIOUSLY USED.
C
C     SBM.......SPDACC STANDS FOR 'SPEED-ACCELERATION'.
C
C  THE SPEED BANDS ARE SET UP AS FOLLOWS:
C
C    SPEED(RSPEED)      SPEED BAND(SPBAND)
C     0.0    -  1.0             1
C     1.01   -  2.0             2
C     2.01   -  3.0             3
C      .        .               .
C      .        .               .
C      .        .               .
C    79.01   - 80.0             80
C
C    SPEED BANDS ARE DETERMINED BY ROUNDING UP THE SPEED.
C
C
C
C
C  THE ACCELERATION BANDS ARE SET UP AS FOLLOWS:
C
C    ACCEL(M/H-S)          ACCEL BAND(PRBAND)
C              - -5.01           1
C    -5.00    - -4.76            2
C    -4.75    - -4.51            3
C    -4.50    - -4.26            4
C    -4.25    - -4.01            5
C    -4.00    - -3.76            6
C      .         .               .
C      .         .               .
C      .         .               .
C    -1.00    - -0.76            18
C    -0.75    - -0.51            19
C    -0.50    - -0.26            20
C    -0.25    - -0.01            21
C    -0.0099 -  0.0099           22
C     0.01    -  0.25            23
C     0.26    -  0.50            24
C     0.51    -  0.75            25
C     0.76    -  1.00            26
C      .         .               .
C      .         .               .
C      .         .               .
C     4.01    -  4.25            39
C     4.26    -  4.50            40
```

```
C     4.51   -  4.75                41
C     4.76   -  5.00                42
C     5.10   -                      43
C
C
C
C     ......ACBAND SUBSCRIPT MAP (BELOW)..........
C
C         PICK SPEED BASED ON SPEED 1.
C         SPEED BASED ON PREVIOUS SPEED BAND.
C
C      SPEED BAND>>      1       2       3       4
C      ACCEL BAND
C          1          (1,1)   (1,2)   (1,3)   (1,4)
C          2          (2,1)   (2,2)   (2,3)   (2,4)
C          3          (3,1)   (3,2)   (3,3)   (3,4)
C          4          (4,1)   (4,2)   (4,3)   (4,4)
C
C
C
C     ..........DEFINITIONS OF VARIABLES..........
C
C       ACBAND(43,81)..I*4...Acceleration-Speed band: this array
C                              contains the number of occurrances of
C                              a specific acceleration at a specific
C                              speed.  See subscript map above.
C       ACCEL..........R*4...Acceleration (M/H-S). RSPEED - SPDSAV
C       IACBND.........I*4...Numerical value of acceleration band
C                              number.
C       ISPEED.........I*4...Numerical value of speed band.
C       ISBSAV.........I*4...Previous value of ISPEED.
C       RSPEED.........R*4...The actual speed.
C       SPDSAV.........R*4...The previous speed value.
C       ONCT...........I*4...Length of trip (sec) = TOTSS  +  GOSC
C       PRBAND(43).....I*4...Matrix of time spent in each acceleration
C                              band.
C       SPBAND(81).....I*4...Speed band: This vector counts the number
C                              of times in a specific speed band.
C
C
      INTEGER ONCT, SPBAND(81), PRBAND(43), ACBAND(43,81)
C
C     .....SPEED BAND SECTION
C
      IF (RSPEED .LE. 80.0) GO TO 20
C
C     .....THERE ARE 81 SPEED BANDS.  YOU HAVE EXCEEDED THE 80th.
C
      ISPEED = 81
      GO TO 40
C
C     .....ROUND UP THE VALUE OF THE SPEED TO GET THE SPEED BAND.
C
   20 ISPEED = RSPEED + .9999
C
C     .....THE BELOW STEP IS ESSENTIAL TO COVER THE CASE WHEN RSPEED=0
C            OTHERWISE ISPEED=0 WHICH IMPOSSIBLE.
C
```

```
      IF (RSPEED .LE. 1.0) ISPEED = 1
C
C     .....INCREMENT THE NUMBER OF OCCURRANCES AT THAT SPEED BAND.
C
   40 SPBAND(ISPEED) = SPBAND(ISPEED) + 1
C
C     .......ACCELLERATION BAND SECTION
C
      IF (ONCT .NE. 0) GO TO 60
C
C     .....SAVE THE PREVIOUS VALUE OF THE SPEED, SPEED BAND....
C
      SPDSAV = RSPEED
      ISBSAV = ISPEED
      RETURN
C
C     .....THE BELOW STATEMENT IS VALID BECAUSE 1 SEC INCREMENT.....
C
   60 ACCEL = RSPEED - SPDSAV
C     IF (ACCEL .GE. - 5. .AND. ACCEL .LE. 5.) GO TO 100
      IF (ACCEL .LT. - 5.) IACBND = 1
      IF (ACCEL .LT. - 5.) GO TO 80
      IF (ACCEL .GT. 5.) IACBND = 43
      IF (ACCEL .GT. 5.) GO TO 80
C
C     .....THE EQUATION BELOW FINDS THE IACBND.  ALTHOUGH IT IS
C             UNOBVIOUS, IT WORKS, AND IS QUITE EFFICIENT COMPARED TO
C             THE PREVIOUS WAY THIS WAS CALCULATED.
C
      IACBND = ACCEL * 4. + 22.00
      IF (ACCEL .GE. .01) IACBND = ACCEL * 4. + 22.9
      IF (ABS(ACCEL) .LT. .01) IACBND = 22
   80 ACBAND(IACBND,ISBSAV) = ACBAND(IACBND,ISBSAV) + 1
      PRBAND(IACBND) = PRBAND(IACBND) + 1
C
C     .....SAVE THE PREVIOUS VALUE OF THE SPEED, SPEED BAND....
C
      SPDSAV = RSPEED
      ISBSAV = ISPEED
      RETURN
      END
      SUBROUTINE CMPRES(SPBAND, PRBAND, ACBAND)
C
C     SBM
C
C     .....IN THIS SUBROUTINE, CMPRES, THE BANDS ARE COMPRESSED FROM
C             43 X 81 TO 23 X 8.  THE ACTUAL CALCULATIONS, WHICH FIND
C             SPEED BANDS IN THE INCREMENTS SEEN IN SUBROUTINE SPDACC,
C             ARE DONE THAT WAY TO MAKE IT EASY TO MANIPULATE THE
C             MATRIX TO GET OTHER BAND SIZES FOR THE FUTURE WITHOUT
C             HAVING TO RERUN THIS PROGRAM IN MODIFIED FORM.  THE
C             PROGRAMMER CAN SIMPLY READ THE TAPES AND HAVE A SHORT
C             PROGRAM THAT READS THE BIG MATRIX AND COMPRESSES IT TO
C             A SMALL ONE.  IN OUR CASE, WE'LL USE AN OUTPUT MATRIX
C             OF THE SIZE DEPICTED BELOW.
C
C  THE SPEED BANDS ARE SET UP AS FOLLOWS:
C
```

```
C    SPEED(MILSPD)      ARRAY 'ISPD'       SPEED BAND(SPBAND)
C     0.0 -  10.0          0 - 100                1
C    10.1 -  20.0        101 - 200                2
C    20.1 -  30.0        201 - 300                3
C    30.1 -  40.0        301 - 400                4
C    40.1 -  50.0        401 - 500                5
C    50.1 -  60.0        501 - 600                6
C    60.1 -  70.0        601 - 700                7
C    70.1 -              701 -                    8
C
C  THE ACCELERATION BANDS ARE SET UP AS FOLLOWS:
C
C    ACCEL(M/H-S)      ARRAY 'IACCEL'        ACCEL BAND(ACBAND)
C         - -5.1             - -51                1
C    -5.0 - -4.6         -50 - -46                2
C    -4.5 - -4.1         -45 - -41                3
C    -4.0 - -3.6         -40 - -36                4
C    -3.5 - -3.1         -35 - -31                5
C    -3.0 - -2.6         -30 - -26                6
C    -2.5 - -2.1         -25 - -21                7
C    -2.0 - -1.6         -20 - -16                8
C    -1.5 - -1.1         -15 - -11                9
C    -1.0 - -0.6         -10 -  -6               10
C    -0.5 - -0.1          -5 -  -1               11
C     0.0 -  0.0           0 -   0               12
C     0.1 -  0.5           1 -   5               13
C     0.6 -  1.0           6 -  10               14
C     1.1 -  1.5          11 -  15               15
C     1.6 -  2.0          16 -  20               16
C     2.1 -  2.5          21 -  25               17
C     2.6 -  3.0          26 -  30               18
C     3.1 -  3.5          31 -  35               19
C     3.6 -  4.0          36 -  40               20
C     4.1 -  4.5          41 -  45               21
C     4.6 -  5.0          46 -  50               22
C     5.1 -               51 -                   23
C
C
      INTEGER SPBAND(81), PRBAND(43), ACBAND(43,81), SPBTWO(8),
     1        PRBTWO(23), ACBTWO(23,8)
C
      DO 40 I = 1, 8
        SPBTWO(I) = 0
C
        DO 20 J = 1, 23
          PRBTWO(J) = 0
   20   ACBTWO(J,I) = 0
C
   40 CONTINUE
C
      DO 80 I = 1, 81
C
        II = I / 10. + .999
C
C     .....NOW CHECK FOR POINTS ABOVE 70 MPH
C
        IF (II .LE. 7) SPBTWO(II) = SPBTWO(II) + SPBAND(I)
        IF (II .GT. 7) SPBTWO(8) = SPBTWO(8) + SPBAND(I)
```

```
        IF (II .LE. 7) ACBTWO(12,II) = ACBTWO(12,II) + ACBAND(22,I)
        IF (II .GT. 7) ACBTWO(12,8) = ACBTWO(12,8) + ACBAND(22,I)
C
        DO 60 K = 1, 22
          K1 = K / 2 + 1
          KK = K + 21
          RK = KK
          KK1 = RK / 2 + 1.6
C
C     .....NOW CHECK FOR POINTS ABOVE 70 MPH
C
          IF (II .GT. 7 .AND. K .NE. 22) ACBTWO(K1,8) = ACBTWO(K1,8) +
     1    ACBAND(K,I)
          IF (II .LE. 7 .AND. K .NE. 22) ACBTWO(K1,II) = ACBTWO(K1,II) +
     1     ACBAND(K,I)
C         IF(K .EQ. 22)    GO TO 100
C
C     .....II CAN NOT EXCEED THE VALUE 7.....
C
          IF (II .LE. 7 .AND. K .NE. 1) ACBTWO(KK1,II) = ACBTWO(KK1,II)
     1    + ACBAND(KK,I)
          IF (II .GT. 7 .AND. K .NE. 1) ACBTWO(KK1,8) = ACBTWO(KK1,8) +
     1    ACBAND(KK,I)
   60   CONTINUE
C
C
C     .....NOW ADD POINTS FROM -.1 TO +.1 MPH/S ACCELERATION.....
C
   80 CONTINUE
C
      DO 100 K = 1, 22
        K1 = K / 2 + 1
        KK = K + 21
        RK = KK
        KK1 = RK / 2. + 1.6
C
C         IF(K .EQ. 22)  GO TO 300
        IF (K .NE. 1.) PRBTWO(KK1) = PRBTWO(KK1) + PRBAND(KK)
  100 IF (K .NE. 22) PRBTWO(K1) = PRBTWO(K1) + PRBAND(K)
C
      PRBTWO(12) = PRBAND(22)
C
C     .....NOW WRITE BANDS IN OUTPUT FILE ON UNIT 6.....
C
C
C       .....WRITE "SPEEDLOG" OUTPUT ON UNIT 6 (COMPRESSED)....
C
      WRITE (6,120) SPBTWO
      WRITE (6,140) PRBTWO
      WRITE (6,160) ((ACBTWO(I,J),J=1,8),I=1,23)
      RETURN
C
  120 FORMAT (' ', 'SPEED   ', 8I6)
  140 FORMAT (' ', 'ACCEL   ', 23I4)
  160 FORMAT (' ', 'COMBIN ', 8I6)

      END
      SUBROUTINE CHECK(TIME,OLDST)
```

```
C
C     .....WRITTEN BY STEVEN B. MICHLIN.....
C
C     .....THIS SUBROUTINE IS USED TO CHECK FOR STALLS.   ON
C            11-14-84, THIS SUBROUTINE WAS ADDED BECAUSE DATA
C            WAS READ ONE LINE AT A TIME.  IN ADDING A COUNTER
C            TO COUNT STALL TIME, WHICH IS 60 OR LESS
C            CONSECUTIVE SECONDS OF ENGINE OFF TIME, IT WAS
C            REALIZED THAT TO ACCURATELY DETERMINE STALL TIME,
C            TWO SIXTY SECOND LINES WITH FLAGS MUST BE CHECKED
C            SINCE A 60 SECOND STALL COULD OCCUR WITHIN TWO LINES.
C
C     .....THIS SUBROUTINE READS TWO CONSECUTIVE LINES,  CHECKS
C            FOR THE STALL CONDITION, AND APPROPRIATELY CHANGES THE
C            FLAG VALUES TO BE ABLE TO USE THE PREVIOUS VERSION OF
C            THE PROGRAM WITH AS LITTLE CHANGE AS POSSIBLE.  IT IS
C            THEREFORE, *ASSUMED* THAT THE PREVIOUS PROGRAM WORKED
C            FINE.
C
C
C.....NAME...MODE...TAG
C
C.....DAY.........I*4.....The 'current' DAY value.
C.....3..................
C.....DAYNXT......I*4.....The 'next' DAY value.
C.....FLAGNX(12)..R*4.....The 'next' FLAGS value.
C.....FLAGS(12)...I*4.....The 'current' FLAGS value.
C.....I...........I*4.....Iteration counter.
C.....ID..........I*4.....Vehicle ID number.
C.....IDNEXT......I*4.....Next vehicle ID number.
C.....J...........I*4.....Iteration counter.
C.....NDIFF.......I*4.....The difference between the number of speed
C                          values from FLAGS and 60.  This difference
C                          is the number of seconds of engine off or
C                          stall in that minute time interval.
C.....NDUMMY......I*4.....Dummy variable used in conversion from
C                          Fortran-77 to Fortran-4.  Arrays in Fortran-4
C                          can't be used as maximum Do-Loop limit.
C.....NFLAG.......I*4.....The current number of FLAGS that have
C                          nonzero values.
C.....NFLGNX......I*4.....The next value of NFLAG.
C.....NN..........I*4.....The total number of consecutive seconds of
C                          stop time.  If NN <= 60, it is the total
C                          number of consecutive seconds of stall
C                          time by definition in this program.
C.....NOFFSC......I*4.....The off time in a given trip (seconds).
C.....NSEC........I*4.....The total number of speeds read in after a
C                          line with FLAGS values.  If NSEC < 60
C                          then there is some stop or stall time.
C.....NSECNX......I*4.....The next value of NSEC.
C.....NSTCNT......I*4.....The number of stalls in a given trip.
C.....NSTFLG(6)...I*4.....Current stall flag value NSTFLG(I); 0=nostall;
C                           1=stall.
C.....NSTFNX(6)...I*4.....Next stall flag value NSTFLG(I); 0=nostall;
C                           1=stall.
C.....NSTOP.......I*4.....Number of stop seconds.
C.....NSTRIP......I*4.....The stall-time in a trip (seconds).
C.....NTRIP.......I*4.....The number of trips on the current line.
```

```
C.....NTRPNX......I*4.....The number of trips on the next line.
C.....10.................
C.....2...................Unit for read.
C.....SIXTSW......L*1.....Used to specify whether or not 5 or 6
C                          temperatures are to be input.
C.....SPD(6,60)...R*4.....Array of speed values associated with each
C                          even numbered flag.
C.....SPDNXT(6,60)R*4.....The next SPD value.
C.....8.................
C.....7.................
C.....TEMP(6).....R*4.....Temperature array to be read in.
C.....9.................
C.....TEMPNX(6)...R*4.....The next temperature array to be read in.
C.....TIME........I*4.....Time on 24 hour clock.
C.....TIMENX......I*4.....The next value of time on 24 hour clock read
C                          in.
C.....4.................
C.....XX..........I*4.....Dummy variable to read in where the 6th TEMP
C                          value would have been if the data was
C                          flawless.
C
C
      LOGICAL*1 SIXTSW, STALFL, GO
      INTEGER XX, DAY, DAYNXT, TIME, TIMENX, FLAGS, FLAGNX, SPBAND,
     1         PRBAND, ACBAND, OLDST
      COMMON /CHCK/ DAYNXT, FLAGNX(12), IDNEXT, NDIFF, NFLGNX, NOFF,
     1       NOFFCN, NOFFSC, NSEC, NSECNX, NSTCNT, NSTCNX, NSTFLG(6),
     2       NSTFNX(6), NSTRIP, SPD(6,60), SPDNXT(6,60), TEMPNX(6),
     3       TIMENX, INIT, SPBAND(81), PRBAND(43), ACBAND(43,81),
     4       ISBSAV, SPDSAV, GODI, STOPSC, STOPDI, NSTAT(2,6),
     5       NSTNX(2,6), NSTOT, NSTSEC, IFSTAL
      COMMON /STATUS/ FLAGS(12)
      COMMON /STUFF/ DAY, ID, NFLAG, TEMP(6), INCSPD
      COMMON /LOGICL/ SIXTSW, STALFL, GO
C
C     .....THE 'CURRENT' VALUE NOW EQUALS THE 'NEXT' VALUE OF
C           THE READ-IN VARIABLES AFTER WHICH A NEW VALUE WILL BE
C           READ IN.
C
C     NOFFSC  =  NOFFCN
C     NOFF    =  0
C
      NFLAG = NFLGNX
      TIME = TIMENX
      IF(IDNEXT .NE. 0)          ID  =  IDNEXT
      DAY = DAYNXT
      NSEC = NSECNX
      NSECNX = 0
C
      DO 60 I = 1, 6
        NSTFLG(I) = NSTFNX(I)
        FLAGS(I) = FLAGNX(I)
        FLAGS(I + 6) = FLAGNX(I + 6)
        TEMP(I) = TEMPNX(I)
        NSTFNX(I) = 0
C
        DO 20 J = 1, 2
   20   NSTAT(J,I) = NSTNX(J,I)
```

```
C
C         .....The below puts a 5 to check for no value given.
C
C         NSTNX(1,I)      =   4
        NSTNX(1,I) = 0
C
C         .....The below step is there to automatically pad all
C                stops and stalls with 1's for the unknown
C                where there are more than 1 stall or stop on the
C                same 60 second input line.  They total 60 - NSEC. .
C                These are also included in NSEC.*****
C
        NSTNX(2,I) = 0
C
        NDUMMY = FLAGNX(I*2)
C
        DO 40 J = 1, NDUMMY
   40   SPD(I,J) = SPDNXT(I,J)
C
   60 CONTINUE
C
C
C
C     .....USED IN TEMPERATURE AND FLAG READS.....
C
      IF (SIXTSW) READ (2,460,END=420) DAYNXT, TIMENX, IDNEXT, TEMPNX,
     1FLAGNX
      IF ( .NOT. SIXTSW) READ (2,480,END=420) DAYNXT, TIMENX, IDNEXT,
     1(TEMPNX(I),I=1,5), XX, FLAGNX
C
      NFLGNX = 0
      NSECNX = 0
C
      DO 100 I = 1, 6
C
        IF (FLAGNX(I*2 - 1) .NE. 0) NFLGNX = 2 * I
C
C       .....READ SPEEDS.....
C
        NN = FLAGNX(2*I)
        IF (FLAGNX(I*2) .NE. 0) READ (2,500) (SPDNXT(I,J),J=1,NN)
        IF (FLAGNX(I*2 - 1) .EQ. 0) GO TO 100
C
        IF ( .NOT. (FLAGNX(I*2 - 1) .EQ. 1 .OR. FLAGNX(I*2 - 1) .EQ. 3))
     1     GO TO 80
        NSECNX = NSECNX + FLAGNX(I*2)
        NSTNX(1,I) = 1
        NSTNX(2,I) = FLAGNX(I*2)
        GO TO 100
C
   80   IF ( .NOT. (FLAGNX(I*2 - 1) .EQ. 2 .OR. FLAGNX(I*2 - 1) .EQ. 4))
     1     GO TO 100
C
        IF (NFLGNX .EQ. 2 .AND. I .NE. 1) GO TO 100
C
C      .....IT'S PROBABLY A STALL.....
C
C       NSTNX(1,I)    =   2
```

```
C       NSTNX(2,I)     =    1
C
C       .....PAD "1's".....
C         ***REMOVE THIS SECTION LATER***
C
        IF ( .NOT. (NFLGNX .GT. 4 .AND. (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1)
     1      .EQ. 4) .OR. (NFLGNX .GT. 6 .AND. (FLAGNX(2) .EQ. 2 .OR.
     2     FLAGNX(2) .EQ. 4)))) GO TO 90
        NSECNX = NSECNX + 1
        NSTNX(1,I) = 2
C
C      .....ONE SECOND FOR THE PADDED MULTIPLE 'ENGINE OFF' ON SAME LINE
C
        NSTNX(2,I) = 1
C
C
C       .....SPEED VALUES WERE INSERTED IN INPUT FILES IN AN UNUSUAL
C              WAY TO SAVE FILE SPACE.  THIS WAS *NOT* DONE BY ME..SBM
C              FOR NEXT OCS PROGRAM, USE INTEGERS TO SAVE SPACE, FOR
C              EXAMPLE, SPEED MULTIPILED BY 100.
C
C                 XX.0=======>XX.00
C                 XX.3=======>XX.25
C                 XX.5=======>XX.50
C                 XX.8=======>XX.75
C                 XX.0=======>XX.00
C
C       .....THE BELOW  DO-LOOP CORRECTS FOR THE ABOVE SPECIAL CASE...
C
C
  90    DO  50  J  =  1,  NN
          ISPEED =  SPDNXT(I,J)
          DEL    =  SPDNXT(I,J)    -  ISPEED
          IF(ABS(DEL  -  .3)  .LT.  .01)  SPDNXT(I,J)  =  ISPEED + .25
  50      IF(ABS(DEL  -  .8)  .LT.  .01)  SPDNXT(I,J)  =  ISPEED + .75
C
C
C
  100 CONTINUE
      NPAD     =    0
      NPDNUM   =    0
      IF ( .NOT. (NFLGNX .EQ. 2 .AND. (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1)
     1   .EQ. 4))) GO TO 120
C
C      .....IT'S DEFINITELY A 60 SECOND 'ENGINE OFF'
C
      NSECNX = 60
      NSTNX(1,1) = 0
C
C      .....60 SECONDS FOR AN 'ENGINE OFF'
C
      NSTNX(2,1) = 60
C
  120 IF ( .NOT. (NFLGNX .GT. 4 .AND. (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1)
     1   .EQ. 4))) GO TO 140
      NSTNX(1,1) = 0
      NSTNX(2,1) = 60 - FLAGNX(2)
      NSTNX(1,2) = 1
```

```
      NSTNX(2,2) = FLAGNX(4)
      NSTNX(2,2) = FLAGNX(2)
C
C  610  IF(.NOT.(                      (FLAGNX(1).EQ.2.OR.FLAGNX(1).EQ.4)))
  140 IF ( .NOT. (NFLGNX .GE. 4 .AND. (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1)
     1   .EQ. 4))) GO TO 180
C
      NPAD   =     1
      NPDNUM =    NPDNUM   +   1
C
C         .....EVERY OTHER ONE WILL STALL OUT UNTIL LAST SO PUT 2 IN
C                THE NSTNX(1,I*2-1) FOR ONES THAT FLAGNX(1)=2 OR 4.....
C
C         .....PAD "1's" IN FOR THE STALLS.....
C
      NDUM   = 0
      NDUMMY = FLOAT(NFLGNX) / 4.5    +    1.
C
      DO 160 I = 1, NDUMMY
        IF(FLAGNX(I*4-3) .NE. 2 .AND. FLAGNX(I*4-3) .NE. 4) GO TO 170
        NSTNX(2,I*2-1)  =  1
        NSTNX(1,I*2-1)  =  2
        NDUM   =   NDUM   +  NSTNX(2,I*2-1)
  150   IF(FLAGNX(I*4-1) .NE. 1 .AND. FLAGNX(I*4-1) .NE. 3) GO TO 160
        NSTNX(2,I*2  )  =  FLAGNX(I*4)
        NSTNX(1,I*2  )  =  1
      NDUM   =  NDUM   +   NSTNX(2,I*2  )
  160 CONTINUE
      GO TO 171
  170 NDUMMY  =  NDUMMY  -  1
  171 CONTINUE
C
C         .....UNPAD THE LAST VALUE.....
C
      IF(FLAGNX(NDUMMY*4-3) .NE.2 .AND.FLAGNX(NDUMMY*4-3).NE. 4)GOTO 161
      NSTNX(2,NDUMMY*2-1)  =  61  - NDUM
      NSTNX(1,NDUMMY*2-1)  =  2
      NSECNX =  60
 161  CONTINUE
C
C
      IF( NFLAG .EQ. 2 .OR. FLAGS(NFLAG-1) .EQ. 1 .OR.
     1   FLAGS(NFLAG-1).EQ. 3 )   GO TO 180
      NDIFF  =  60  -  NSEC
      NN     =  NDIFF  +  1
      IF( NN .LT. 60 ) GO TO 320
      GO TO 400
C
C
C         .....EVERY OTHER ONE WILL STALL OUT UNTIL LAST SO PUT 2 IN
C                THE NSTNX(1,I*2) FOR ONES THAT FLAGNX(1)=1 OR 3.....
C
  180 IF ( .NOT. (NFLGNX .GT. 2 .AND. (FLAGNX(1) .EQ. 1 .OR. FLAGNX(1)
     1   .EQ. 3))) GO TO 200
      NSTNX(1,1) = 1
      NSTNX(2,1) = FLAGNX(2)
      NSTNX(1,2) = 0
      NSTNX(2,2) = 60 - FLAGNX(2)
```

```
      IF (NFLGNX .GT. 4) NSTNX(2,2) = 60 - FLAGNX(2) - FLAGNX(6)
      IF (NFLGNX .GT. 4) NSTNX(1,3) = 1
      IF (NFLGNX .GT. 4) NSTNX(2,3) = FLAGNX(6)
C 630    IF(.NOT.(                          (FLAGNX(1).EQ.1.OR.FLAGNX(1).EQ.3)))
  200 IF ( .NOT. (NFLGNX .GE. 6 .AND. (FLAGNX(1) .EQ. 1 .OR. FLAGNX(1)
     1    .EQ. 3))) GO TO 240
C
C         .....PAD "1's" IN FOR THE STALLS.....
C
      NPAD   = 1
C
      NDUM   = 0
      NDUMMY = NFLGNX / 4.  + 1.
C
C
      DO 220 I = 1, NDUMMY
        IF(FLAGNX(I*4-3) .NE. 1 .AND. FLAGNX(I*4-3) .NE. 3) GO TO 210
        NSTNX(2,I*2-1)  =  FLAGNX(I*4-2)
        NSTNX(1,I*2-1)  =  1
        NDUM  =  NDUM  +  NSTNX(2,I*2 - 1)
  210   IF(FLAGNX(I*4-1) .NE. 2 .AND. FLAGNX(I*4-1) .NE. 4) GO TO 225
        NSTNX(2,I*2  )  =  1
        NSTNX(1,I*2  )  =  2
        NDUM  =  NDUM  +  NSTNX(2,I*2  )
  220 CONTINUE
      GO TO 226
  225 NDUMMY  =   NDUMMY  -  1
  226 CONTINUE
C
C         .....UNPAD THE LAST VALUE.....
C
      IF(FLAGNX(NDUMMY*4-1) .NE. 2.AND.FLAGNX(NDUMMY*4-1).NE. 4)GOTO 221
      NSTNX(2,NDUMMY*2  )  =  61  - NDUM
      NSTNX(1,NDUMMY*2  )  =  2
      NSECNX =  60
 221  CONTINUE
C
C
      IF( NFLAG .EQ. 2 .OR. FLAGS(NFLAG-1) .EQ. 1 .OR.
     1    FLAGS(NFLAG-1).EQ. 3 )   GO TO 240
      NDIFF  =  60  -  NSEC
      NN     =  NDIFF  +  1
      IF( NN .LT. 60 ) GO TO 320
      GO TO 400
C
  240 CONTINUE
C
C
C     INIT  =  1
C
      IF ( .NOT. (NFLGNX .EQ. 2 .AND. (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1)
     1   .EQ. 4))) GO TO 280
C
      NSTNX(1,1) = 0
      NSTNX(2,1) = 60
      IF (FLAGS(NFLAG - 1) .EQ. 2 .OR. FLAGS(NFLAG - 1) .EQ. 4)
     1   GO TO 260
      NSTAT(1,NFLAG/2) = 0
```

```
      NSTAT(2,NFLAG/2) = 60 - NSEC
  260 GO TO 400
C
  280 IF (INIT .EQ. 1) GO TO 400
C
C
      IF (NFLAG .EQ. 2 .AND. NSEC .LT. 60) GO TO 400
C
C     .....DUE TO A DATA COLLECTION EQUIPMENT BUG, THERE WERE SOME
C            ENGINE OFF PERIODS  WHEN A SPEED WAS READ, BUT NO MORE THAN
C            ONE SPEED PER RUN.  ALL SUCH CASES WILL BE DEALT WITH AS IF
C            ZERO SPEEDS WERE READ. SINCE THE SPEED READ WAS CAUSED BY A
C            DATA COLLECTION EQUIPMENT BUG AND SHOULD NOT HAVE BEEN
C            THERE.
C
C     .....NOW CHECK IF ANY ENGINE OFF PERIODS OF LESS THAN 60 SECONDS
C            OCCUR.  SUCH 'STALLS' COULD OCCUR FROM EITHER ONE OR TWO
C            LINES OF FLAG DATA, SO TWO LINES WERE READ IN.
C
C     IF(NFLAG .EQ. 2 .AND. (FLAGS(1) .EQ. 2 .OR. FLAGS(1) .EQ. 4) )
C    1                                         NSTAT(2,1)    =    60
C
      IF (NSEC .NE. 60 .AND. NSTAT(2,1) .NE. 60) GO TO 300
      GO TO 400
C
  300 IF (NSEC .GT. 60) PRINT 440
      IF (NSEC .GT. 60) STOP
CC    IF((NFLAG .EQ. 4) .AND. (FLAGS(1) .EQ. 2 .OR. FLAGS(1) .EQ.4))
CC   1                                     GO TO 160
CC    IF((NFLGNX .GT. 2) .AND. (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1) .EQ.4))
CC   1                                     GO TO 160
C
C     .....CALCULATE THE NUMBER OF MOTOR OFF SECONDS, NDIFF, ON CURRENT
C            LINE IN DATA FILE.
C
      NDIFF = 60 - NSEC
      IF (NFLAG .EQ. 2 .AND. (FLAGNX(1) .EQ. 1 .OR. FLAGNX(1) .EQ. 3))
     1   GO TO 400
      IF (NFLAG .EQ. 2) NDIFF = 60
      IF ((FLAGNX(1) .EQ. 1) .OR. (FLAGNX(1) .EQ. 3)) NSTFLG(NFLAG/2) =
     11
      IF ((FLAGNX(1) .EQ. 1) .OR. (FLAGNX(1) .EQ. 3)) GO TO 400
      IF ((FLAGNX(2) + NDIFF) .GT. 60) GO TO 400
      IF ((FLAGS(NFLAG - 1) .EQ. 1) .OR. (FLAGS(NFLAG - 1) .EQ. 3))
     1   GO TO 400
C
C     IF( ( NDIFF  + 60 - NSECNX) .GT. 60) GO TO 160
C
C     NSEC   =   60
C     .....FOR NDIFF  LESS THAN 60
C
C     .....SUM UP THE NUMBER OF PROSPECTIVE STOP SECONDS AND THE NUMBER
C            OF SECONDS OF THE FLAG(2) 'NEXT'.  IF <= 60, THIS IS A
C            STALL, OTHERWISE IT IS A STOP.
C
      NN = NDIFF + 1
      IF (FLAGNX(1) .EQ. 2 .OR. FLAGNX(1) .EQ. 4) NN = NDIFF + 60 -
     1NSECNX
```

```
      IF (((FLAGNX(1) .EQ. 2 .OR. FLAGNX(1) .EQ. 4)) .AND. NFLGNX .EQ.
     12) NN = NDIFF + 60
C
C     .....IF NN  >  60, IT IS AN OFF, NOT A STALL.....
      IF (NFLAG .EQ. 2 .AND. (FLAGNX(1) .EQ. 1 .OR. FLAGNX(1) .EQ. 3))
     1   GO TO 400
C
C
C     .....BEWARE OF THE CASE AFTER A STALL AND COMBINING THE
C              STALL SECONDS, THAT NSEC CAN BE LESS THAN 60.....
C
      IF (NN .LE. 60 .AND. NFLAG .NE. 2) GO TO 320
C
      NSTAT(1,NFLAG/2) = 0
      NSTAT(2,NFLAG/2) = NDIFF
      NSTNX(1,1) = 0
C
C     .....THE BELOW IS TRUE ONLY WHEN FLAGNX(NFLGNX-1) .NE. 2 OR 4
C
      IF (FLAGNX(NFLGNX - 1) .NE. 2 .AND. FLAGNX(NFLGNX - 1) .NE. 4)
     1NSTNX(2,1) = 60 - NSECNX
C
      GO TO 400
C
C     .....COMBINE THE STALL SECONDS WHICH CONTINUES FROM THE PREVIOUS
C              LINE....
C
  320 IF (NFLGNX .EQ. 2 .AND. FLAGNX(2) .EQ. 0) GO TO 400
C
C     .....ITS A STALL, SO NSTAT(1,I) = 2; 1=ON; 0=OFF; 5=INITIALIZED
C                        FIX LATER
      NSTAT(1,NFLAG/2) = 2
C
      NDUMMY = FLAGNX(2)
C
      DO 340 I = 1, NDUMMY
  340 SPD(NFLAG/2,FLAGS(NFLAG) + I) = SPDNXT(1,I)
C     FLAGS(NFLAG) = FLAGS(NFLAG) + FLAGNX(2)
      NSTAT(1,NFLAG/2) = 2
      NSTAT(2,NFLAG/2) = NN
C
      NFLGNX = NFLGNX - 2
C
      NDUM    =  NFLGNX / 2
C
      DO 380 I = 1, NDUM
C
CC        IF (I .GT. 6) GO TO 380
          IF (I .LT. 7) NSTNX(1,I) = NSTNX(1,I + 1)
          IF (I .LT. 7) NSTNX(2,I) = NSTNX(2,I + 1)
C
          NDUMMY = FLAGNX(I*2+2)
          IF(NDUMMY .EQ. 0)  GO TO 370
C
          DO 360 J = 1, NDUMMY
          TEMPP        = SPDNXT(I + 1,J)
C         PRINT 2000, TIME, I, J, TEMPP
C2000     FORMAT(' TIME=',I4, ' I=',I3, ' J=',I3, ' TEMPP=', F5.1)
```

```
  360   SPDNXT(I,J) = TEMPP
C
  370  FLAGNX(2*I)  =  FLAGNX(2*I+2)
  380 FLAGNX(2*I-1) = FLAGNX(2*I + 1)
C
      FLAGNX(NFLGNX + 1) = 0
      FLAGNX(NFLGNX + 2) = 0
      NSTNX(1,NFLGNX/2 + 1) = 5
C     NSTNX(1,NFLGNX/2+2)     =     5
      NSTNX(2,NFLGNX/2 + 1) = 0
C     NSTNX(2,NFLGNX/2+2)     =     0
C
C
  400 INIT = 0
      INCSPD = 1
      RETURN
  420 IF (OLDST .NE. 2 .AND. OLDST .NE. 4)  CALL STATON(OLDST)
      CALL FINISH
      STOP
  440 FORMAT ('  MORE THAN 60 SPEEDS IN A MINUTE-IMPOSSIBLE!!!')
  460 FORMAT (I3, I5, I6, 6F4.0, 12I4)
  480 FORMAT (I3, I5, I6, 5F4.0, A4, 12I4)
  500 FORMAT (F6.1, 19F5.1)
      END
      SUBROUTINE RUNTME(ITIME)
      DIMENSION ITIME(6)
      CALL TIME(22,0, ITIME)
      RETURN
      END
```

OCS-CMP.FOR

```
LEVEL 1.3.1 (FEB 1984)           VS FORTRAN          DATE: MAR 16, 1987    TIME: 11:23:07                PAGE:

REQUESTED OPTIONS (EXECUTE): TEST

OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST   TRMFLG   SRCFLG   SYM
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)     CHARLEN(500)     SDUMP

                *....*...1.........2.........3.........4.........5.........6.........7.*.......8.........9.........*

                C
                C  PROGRAM OCS_CMP.FOR
                C
                C    THIS PROGRAM IS USED TO COMBINE OPERATIONAL CHARACTERISTIC STUDY
                C    DATA (GENERATED BY THE PROGRAM OCS_STAT.FOR) INTO TRIPS ACCORDING
                C    TO A USER SPECIFIED BETWEEN TRIP TIME LIMIT.
                C
                C    PROGRAMMER: STEVE MICHELIN
                C
                C    3/16/87 - MODIFIED PROGRAM TO INCLUDE OFFCT IN TOTAL SECONDS AND
                C              STOP SECONDS FOR COMBINED TRIPS. ALSO PUT IN OPTION
                C              FOR HEADERS IN OUTPUT OR NO. ALSO CLEANED UP CODE AND
                C              PUT IN COMMENTS. JDC.
                C
ISN        1          INTEGER*4 SVDAY1(125),  TRPCT1(125),   DVTRP1(125),   SVTIM1(125),
                     1          TYPE1(125),   START1(125),   OFFCT1(125),
                     2          STPCT1(125),  TOTSC1(125),    STPSC1(125),   GOSC1(125),
                     3          LOAN1(125),   ECHO11(125),   WHO1(125),     GOOD1(125),
                     4          DATE1(125),   ECHO31(125),
                     5          SVDAY2(125),  TRPCT2(125),   DVTRP2(125),   SVTIM2(125),
                     6          TYPE2(125),   START2(125),   OFFCT2(125),
                     7          STPCT2(125),  TOTSC2(125),    STPSC2(125),   GOSC2(125),
                     8          LOAN2(125),   ECHO12(125),   WHO2(125),     GOOD2(125),
                     9          DATE2(125),   ECHO32(125),   ITIME(6),
                     *          ERR1(125),    ERR2(125),     ERR3(125),     ERR4(125),
                     1          NEWTRP(125)
ISN        2          INTEGER*4 OFFSEC/0/, TOTSEC/0/, TOTSTP/0/
                C
ISN        3          REAL*8    AVSPD1(125),  AVDBS1(125),   AVSPD2(125),   AVDBS2(125),
                     1            DSTNC1(125),   DSTNC2(125)
ISN        4          CHARACTER  DATFLE*30, OUFLE*7, CONSOL*10
ISN        5          CHARACTER HEADER, YES/'Y'/, NO/'N'/
                C
                C  DEFINE INPUT FILE AS SHYY:OCS.D-TRIPS. THIS SHOULD BE DATA
                C  GENERATED FROM THE PROGRAM OCS_STAT.FOR.
                C  DEFINE OUTPUT FILE AS -TRIP, CONSOLE AS *MSOURCE*.
                C
ISN        6          DATFLE  =  'SHYY:OCS.D-TRIPS'
ISN        7          OUFLE   =  '-TRIP'
ISN        8          CONSOL  =  '*MSOURCE*'
ISN        9          OPEN (0, FILE=CONSOL      )
ISN       10          OPEN (5, FILE=DATFLE      )
ISN       11          OPEN (6, FILE=OUFLE       )
                C
ISN       12          CALL FTNCMD('SET ATTENTION=ON;')
                C
                C  GET USER SPECIFIED BETWEEN TRIP TIME LIMIT.
                C
ISN       13          WRITE(0,*) 'ENTER A BETWEEN TRIP TIME LIMIT:
ISN       14          READ (0,*) ITOLER
ISN       15          WRITE(0,*) 'TOLERANCE IS ', ITOLER
```

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

                C
                C  CHECK IF USER WANTS HEADERS IN OUTPUT FILE -TRIP.
                C
ISN      16           WRITE(0,*) 'DO YOU WANT HEADERS IN OUTPUT FILE -TRIP? (Y/N):
ISN      17           READ (0,505) HEADER
ISN      18       505 FORMAT(A1)
                C
                C  READ IN ORIGINAL OCS TRIP DATA.
                C
ISN      19           NPAGE    =   1
ISN      20           NBACK    =   0
ISN      21           NEND     =   0
ISN      22       10  CONTINUE
ISN      23           IF(NBACK .EQ. 1) BACKSPACE 5
ISN      25           NBACK        =  1
                C
ISN      26           DO 200   I  =  1,    1000
ISN      27           READ(5,100,END=250)
                     1            SVDAY1(I), TRPCT1(I), DVTRP1(I), SVTIM1(I), TYPE1(I),
                     2            START1(I), OFFCT1(I), AVSPD1(I), DSTNC1(I), STPCT1(I),
                     3            AVDBS1(I), TOTSC1(I),  STPSC1(I), GOSC1(I), LOAN1(I),
                     4            ECHO11(I), WHO1 (I),  ECHO31(I), DATE1(I), GOOD1(I)
ISN      28       100 FORMAT(4I6, 2I2, I6, 2F10.3, I6, F10.3, 3I6, I5, 3I2, I7, I2)
                C
ISN      29           DVTRP2(I)  =  0
ISN      30           TRPCT2(I)  =  0
ISN      31           ICOUNT  =  I  -  1
ISN      32           IF( I .EQ. 1 )  GO TO 200
ISN      33           IF( LOAN1(I) .NE. LOAN1(I-1) ) GO TO 300
ISN      34       200 CONTINUE
                C
ISN      35       250 CONTINUE
ISN      36           NEND  =   1
ISN      37       300 CONTINUE
                C
                C---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
                C
                C  COMBINE TRIPS IF THE OFFCT FALLS WITHIN THE USER SPECIFIED LIMIT.
                C
ISN      38           TRPCT2(1)  =  0
ISN      39           J  =  0
ISN      40           DO  500  I  =  1, ICOUNT
ISN      41           IF(OFFCT1(I) .GT. ITOLER .OR. I .EQ. 1)  THEN
                C
                C  TRIP NOT WITHIN LIMIT.
                C
ISN      42             J            =   J   +   1
ISN      43             ERR1(J) =   0
ISN      44             ERR2(J) =   0
ISN      45             ERR3(J) =   0
ISN      46             ERR4(J) =   0
ISN      47             NEWTRP(J)   =   0
ISN      48             SVDAY2(J)   =   SVDAY1(I)
ISN      49             IF(I .EQ. 1) THEN
ISN      50               TRPCT2(J)   =   1
ISN      51             ELSE
```

```
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8

ISN      52               TRPCT2(J)   =   TRPCT2(J-1)    +  1
ISN      53             ENDIF
ISN      54             IF(SVTIM1(I) .LT. SVTIM1(I-1) .OR. I .EQ. 1 .OR.
              1                    OFFCT1(I).GE.1440) THEN
ISN      55               DYTRP2(J)   =   1
ISN      56             ELSE
ISN      57               DYTRP2(J)   =   DYTRP2(J-1)    +  1
ISN      58             ENDIF
                C
ISN      59             SVTIM2(J)   =   SVTIM1(I)
ISN      60             TYPE2 (J)   =   TYPE1 (I)
ISN      61             START2(J)   =   START1(I)
ISN      62             OFFCT2(J)   =   OFFCT1(I)
ISN      63             AVSPD2(J)   =   AVSPD1(I)
ISN      64             DSTNC2(J)   =   DSTNC1(I)
ISN      65             STPCT2(J)   =   STPCT1(I)
ISN      66             AVDBS2(J)   =   AVDBS1(I)
ISN      67             TOTSC2 (J)   =   TOTSC1 (I)
ISN      68             STPSC2(J)   =   STPSC1(I)
ISN      69             GOSC2 (J)   =   GOSC1 (I)
ISN      70             LOAN2 (J)   =   LOAN1 (I)
ISN      71             ECHO12(J)   =   ECHO11(I)
ISN      72             WHO2  (J)   =   WHO1  (I)
ISN      73             ECHO32(J)   =   ECHO31(I)
ISN      74             DATE2 (J)   =   DATE1 (I)
ISN      75             GOOD2 (J)   =   GOOD1 (I)
                C
                C TRIP WITHIN LIMIT.
                C
ISN      76           ELSE
ISN      77             OFFSEC = OFFCT1(I) * 60
ISN      78             TOTSEC = TOTSC1(I) + OFFSEC
ISN      79             TOTSTP = STPSC1(I) + OFFSEC
                C
ISN      80             NEWTRP(J)  =  NEWTRP(J)  +  1
ISN      81             IF(TYPE2(J) .NE. TYPE1(I) )      TYPE2(J)  =  2
                C
ISN      83             DSTNC2(J)   =   DSTNC2(J)  + DSTNC1(I)
ISN      84             STPCT2(J)   =   STPCT2(J)  +  STPCT1(I)
ISN      85             TOTSC2 (J)   =   TOTSC2(J)  +  TOTSEC
                C
ISN      86             AVDBS2(J)   =   DSTNC2(J)  /  (STPCT2(J) )
ISN      87             AVSPD2(J) = (DSTNC2(J) * 3600) / TOTSC2(J)
                C
ISN      88             STPSC2(J)   =   STPSC2(J) +  TOTSTP
ISN      89             GOSC2 (J)   =   GOSC2(J)  +  GOSC1 (I)
                C
ISN      90             IF(ECHO12(J) .NE.  ECHO11(I) )  ERR1(J)   =  1
ISN      92             IF(WHO2  (J) .NE.  WHO1  (I) )  ERR2(J)   =  2
ISN      94             IF(ECHO32(J) .NE.  ECHO31(I) )  ERR3(J)   =  3
ISN      96             IF(GOOD2 (J) .NE.  GOOD1 (I) )  ERR4(J)   =  4
                C
ISN      98           ENDIF
                C
                C
ISN      99      500  CONTINUE
```

```
              *....*...1.........2.........3.........4.........5.........6.........7.*.......8

              C
              C
              C---+----1----+----2----+----3----+----4----+----5----+----6----+----7--
              C
              C  WRITE COMBINED TRIP DATA TO TEMPORARY FILE -TRIP.
              C
ISN     100         DO 700  I  =  1,  J
ISN     101         IF(HEADER.NE.YES)GO TO 580
ISN     102         TESTI       =  FLOAT(I) / 50.
ISN     103         IF( I .EQ. 1 .OR. TESTI .EQ. 1.0 .OR. TESTI .EQ. 2.0) THEN
ISN     104         CALL GUINFO(         'SIGNONID',                   ICCID)
ISN     105         CALL CUINFO(         '$=ON     ',                   1     )
ISN     106         CALL TIME(           22,   0,                       ITIME)
              C
ISN     107         WRITE(6,510) (ICCID, (    ITIME(K), K = 1,     6),
                   1                ITOLER, J, LOAN2(I), NPAGE)
ISN     108    510  FORMAT('1', 10X, A4, 2X, 6A4, 10X, 'OFFCOUNT LIMIT=',I3,9X,
                   1I3,2X,'TOTAL TRIPS        LOAN#=', I4, 10X, 'PAGE ', I3)
ISN     109         WRITE(6,550)
ISN     110    550  FORMAT (/,
                   1        '  SAVDAY       DAYTRP   OLDST ', '  OFFCT              DIS',
                   2'TNC          AVEDBS        STPSC       LOAN#        DATE ',
                   3'   ERRORS',/,         '          TRIPCT     SAVTIM  START ',
                   4        '       AVESPD              STOPCT           TOTSC        GOSC'
                   5        '       FLAGS      TRPFLG     #COMB ',/)
              C
ISN     111         NPAGE   =   NPAGE   +   1
ISN     112         ENDIF
              C
ISN     113    580  CONTINUE
ISN     114         WRITE(6,600)
                   2            SVDAY2(I), TRPCT2(I), DYTRP2(I), SVTIM2(I), TYPE2(I),
                   2            START2(I), OFFCT2(I), AVSPD2(I), DSTNC2(I), STPCT2(I),
                   3            AVDBS2(I), TOTSC2(I),  STPSC2(I), GOSC2(I), LOAN2(I),
                   4            ECHO12(I), WHO2 (I),  ECHO32(I), DATE2(I), GOOD2(I),
                   5            ERR1(I),   ERR2(I),   ERR3(I),   ERR4(I),  NEWTRP(I)
ISN     115    600  FORMAT(4I6, 2I2, I6, 2F10.3, I6, F10.3, 3I6, I5, 3I2, I7, 6I2)
              C
ISN     116    700  CONTINUE
ISN     117         IF(NEND .EQ. 0)   GO TO 10
              C
              C  TERMINATE.
              C
ISN     118         STOP
ISN     119         END

*STATISTICS*   SOURCE STATEMENTS = 113, PROGRAM SIZE = 35232 BYTES, PROGRAM NAME = MAIN    PAGE:    1.

*STATISTICS*    NO DIAGNOSTICS GENERATED.

**MAIN** END OF COMPILATION 1 ******
```

```
SUMMARY OF MESSAGES AND STATISTICS FOR ALL COMPILATIONS

*STATISTICS*   SOURCE STATEMENTS = 113, PROGRAM SIZE = 35232 BYTES, PROGRAM NAME = MAIN    PAGE:

*STATISTICS*    NO DIAGNOSTICS GENERATED.

**MAIN** END OF COMPILATION 1 ******

******* SUMMARY STATISTICS ******* 0 DIAGNOSTICS GENERATED. HIGHEST SEVERITY CODE IS 0.
```

Appendix C

Histograms: Urban Only Trips

Trips Separated By An Engine Off Period Exceeding 10 Minutes

Histogram Key:

LEFT-END = Lower limit of interval
TOT % = Percent of total trips found in interval
COUNT = Number of trips in interval

AVERAGE TRIP SPEED (0-64 mph)

each X = 2 trips

```
LEFT-END     TOT%   COUNT

  0.          4.4     44 +XXXXXXXXXXXXXXXXXXXXXX
  2.0000      2.2     22 +XXXXXXXXXXX
  4.0000      3.7     37 +XXXXXXXXXXXXXXXXXXX
  6.0000      4.2     42 +XXXXXXXXXXXXXXXXXXXXX
  8.0000      5.4     54 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
  10.000      6.0     60 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  12.000      9.3     93 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  14.000     10.4    104 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  16.000      9.7     97 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  18.000      7.9     79 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  20.000     10.3    103 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  22.000      7.7     77 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  24.000      5.6     56 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
  26.000      3.5     35 +XXXXXXXXXXXXXXXXXX
  28.000      2.5     25 +XXXXXXXXXXXXX
  30.000      1.9     19 +XXXXXXXXXX
  32.000      1.5     15 +XXXXXXXX
  34.000      1.4     14 +XXXXXXX
  36.000       .5      5 +XXX
  38.000       .3      3 +XX
  40.000       .3      3 +XX
  42.000       .3      3 +XX
  44.000       .3      3 +XX
  46.000       .2      2 +X
  48.000       .1      1 +X
  50.000       .2      2 +X
  52.000     0.        0 +
  54.000     0.        0 +
  56.000     0.        0 +
  58.000     0.        0 +
  60.000       .1      1 +X
  62.000       .1      1 +X

  TOTAL     100.0   1000  (INTERVAL WIDTH= 2.0000)
```

NUMBER OF STOPS DURING TRIP (0-35) each X = 2 trips

```
 LEFT-END     TOT%   COUNT

  0.           0.       0 +
  1.0000       3.5     35 +XXXXXXXXXXXXXXXXXX
  2.0000       5.1     51 +XXXXXXXXXXXXXXXXXXXXXXXXXX
  3.0000       7.1     71 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4.0000       9.5     95 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  5.0000       9.0     90 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  6.0000      10.4    104 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  7.0000       9.1     91 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  8.0000       8.3     83 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  9.0000       6.1     61 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  10.000       5.1     51 +XXXXXXXXXXXXXXXXXXXXXXXXXX
  11.000       4.2     42 +XXXXXXXXXXXXXXXXXXXXX
  12.000       3.0     30 +XXXXXXXXXXXXXXX
  13.000       3.6     36 +XXXXXXXXXXXXXXXXXX
  14.000       2.1     21 +XXXXXXXXXXX
  15.000       2.3     23 +XXXXXXXXXXXX
  16.000       1.8     18 +XXXXXXXXX
  17.000       1.6     16 +XXXXXXXX
  18.000       1.7     17 +XXXXXXXXX
  19.000       1.2     12 +XXXXXX
  20.000       1.1     11 +XXXXXX
  21.000        .5      5 +XXX
  22.000        .9      9 +XXXXX
  23.000        .3      3 +XX
  24.000        .3      3 +XX
  25.000        .6      6 +XXX
  26.000        .2      2 +X
  27.000        .4      4 +XX
  28.000        .2      2 +X
  29.000       0.       0 +
  30.000       0.       0 +
  31.000       0.       0 +
  32.000        .1      1 +X
  33.000       0.       0 +
  34.000        .3      3 +XX

                .4      4 > 35.000
  TOTAL      100.0   1000  (INTERVAL WIDTH= 1.0000)
```

TRIP DISTANCE (0-25 miles) **each X = 2 trips**

```
LEFT-END     TOT%  COUNT

 0.          10.5   105 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000       8.5    85 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000       8.1    81 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000       7.6    76 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       5.2    52 +XXXXXXXXXXXXXXXXXXXXXXXXXX
 2.5000       5.3    53 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.0000       6.0    60 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.5000       4.0    40 +XXXXXXXXXXXXXXXXXXXX
 4.0000       6.5    65 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.5000       4.2    42 +XXXXXXXXXXXXXXXXXXXXX
 5.0000       2.8    28 +XXXXXXXXXXXXXX
 5.5000       4.5    45 +XXXXXXXXXXXXXXXXXXXXXXX
 6.0000       3.6    36 +XXXXXXXXXXXXXXXXXX
 6.5000       2.3    23 +XXXXXXXXXXXX
 7.0000       2.0    20 +XXXXXXXXXX
 7.5000       1.5    15 +XXXXXXXX
 8.0000       1.1    11 +XXXXXX
 8.5000       1.5    15 +XXXXXXXX
 9.0000       1.3    13 +XXXXXXX
 9.5000       1.1    11 +XXXXXX
 10.000       1.9    19 +XXXXXXXXXX
 10.500        .9     9 +XXXXX
 11.000       1.3    13 +XXXXXXX
 11.500       1.3    13 +XXXXXXX
 12.000       1.2    12 +XXXXXX
 12.500        .9     9 +XXXXX
 13.000        .8     8 +XXXX
 13.500        .3     3 +XX
 14.000        .3     3 +XX
 14.500        .4     4 +XX
 15.000        .5     5 +XXX
 15.500        .2     2 +X
 16.000        .3     3 +XX
 16.500        .2     2 +X
 17.000        .2     2 +X
 17.500        .2     2 +X
 18.000       0.      0 +
 18.500        .2     2 +X
 19.000        .4     4 +XX
 19.500        .1     1 +X
 20.000        .1     1 +X
 20.500        .1     1 +X
 21.000        .1     1 +X
 21.500       0.      0 +
 22.000        .1     1 +X
 22.500       0.      0 +
 23.000       0.      0 +
 23.500       0.      0 +
 24.000       0.      0 +
 24.500       0.      0 +

               .4     4 > 25.000
 TOTAL      100.0  1000  (INTERVAL WIDTH= .50000)
```

TRIP DISTANCE (0-2 miles)

each X = 1 trip

```
LEFT-END     TOT%  COUNT

 0.           4.6    46 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .10000       1.4    14 +XXXXXXXXXXXXXX
 .20000       1.9    19 +XXXXXXXXXXXXXXXXXXX
 .30000       2.1    21 +XXXXXXXXXXXXXXXXXXXXX
 .40000        .5     5 +XXXXX
 .50000       1.4    14 +XXXXXXXXXXXXXX
 .60000       2.2    22 +XXXXXXXXXXXXXXXXXXXXXX
 .70000       1.7    17 +XXXXXXXXXXXXXXXXX
 .80000       1.4    14 +XXXXXXXXXXXXXX
 .90000       1.8    18 +XXXXXXXXXXXXXXXXXX
 1.0000       1.8    18 +XXXXXXXXXXXXXXXXXX
 1.1000       1.5    15 +XXXXXXXXXXXXXXX
 1.2000       1.2    12 +XXXXXXXXXXXX
 1.3000       1.8    18 +XXXXXXXXXXXXXXXXXX
 1.4000       1.8    18 +XXXXXXXXXXXXXXXXXX
 1.5000       1.7    17 +XXXXXXXXXXXXXXXXX
 1.6000       1.2    12 +XXXXXXXXXXXX
 1.7000       2.0    20 +XXXXXXXXXXXXXXXXXXXX
 1.8000       1.0    10 +XXXXXXXXXX
 1.9000       1.8    18 +XXXXXXXXXXXXXXXXXX

             65.2   652 > 2.0000
 TOTAL      100.0  1000  (INTERVAL WIDTH= .10000)
```

AVERAGE DISTANCE BETWEEN STOPS (0-5 miles) each X = 2 trips

```
LEFT-END     TOT%   COUNT

 0.           7.0     70 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .10000       9.3     93 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .20000      11.9    119 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .30000      12.9    129 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .40000      11.7    117 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000      10.5    105 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .60000       8.2     82 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .70000       5.3     53 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 .80000       5.3     53 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 .90000       3.7     37 +XXXXXXXXXXXXXXXXXXX
 1.0000       2.0     20 +XXXXXXXXXX
 1.1000       1.6     16 +XXXXXXXX
 1.2000       1.4     14 +XXXXXXX
 1.3000        .7      7 +XXXX
 1.4000       1.7     17 +XXXXXXXXX
 1.5000       1.3     13 +XXXXXXX
 1.6000        .4      4 +XX
 1.7000        .5      5 +XXX
 1.8000        .2      2 +X
 1.9000        .2      2 +X
 2.0000        .7      7 +XXXX
 2.1000        .5      5 +XXX
 2.2000        .2      2 +X
 2.3000        .2      2 +X
 2.4000        .2      2 +X
 2.5000        .1      1 +X
 2.6000        .2      2 +X
 2.7000       0.       0 +
 2.8000        .2      2 +X
 2.9000        .1      1 +X
 3.0000        .2      2 +X
 3.1000        .1      1 +X
 3.2000       0.       0 +
 3.3000       0.       0 +
 3.4000       0.       0 +
 3.5000       0.       0 +
 3.6000       0.       0 +
 3.7000       0.       0 +
 3.8000        .1      1 +X
 3.9000       0.       0 +
 4.0000        .3      3 +XX
 4.1000       0.       0 +
 4.2000        .1      1 +X
 4.3000       0.       0 +
 4.4000       0.       0 +
 4.5000       0.       0 +
 4.6000       0.       0 +
 4.7000       0.       0 +
 4.8000        .1      1 +X
 4.9000       0.       0 +

               .9      9 > 5.0000
 TOTAL      100.0   1000  (INTERVAL WIDTH= .10000)
```

TIME SINCE LAST TRIP (0-24 hours)

each X = 3 trips

```
LEFT-END     TOT%   COUNT

 0.          25.0    250 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000      15.7    157 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000       8.6     86 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000       7.2     72 +XXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       3.6     36 +XXXXXXXXXXXX
 2.5000       3.4     34 +XXXXXXXXXXXX
 3.0000       3.1     31 +XXXXXXXXXXX
 3.5000       2.2     22 +XXXXXXXX
 4.0000       2.1     21 +XXXXXXX
 4.5000       2.7     27 +XXXXXXXXX
 5.0000       1.1     11 +XXXX
 5.5000       1.0     10 +XXXX
 6.0000        .5      5 +XX
 6.5000        .5      5 +XX
 7.0000        .2      2 +X
 7.5000        .4      4 +XX
 8.0000       1.6     16 +XXXXXX
 8.5000       2.1     21 +XXXXXXX
 9.0000       1.1     11 +XXXX
 9.5000       1.6     16 +XXXXXX
 10.000       1.7     17 +XXXXXX
 10.500       1.1     11 +XXXX
 11.000       1.1     11 +XXXX
 11.500       1.3     13 +XXXXX
 12.000        .9      9 +XXX
 12.500        .9      9 +XXX
 13.000        .6      6 +XX
 13.500        .8      8 +XXX
 14.000       1.3     13 +XXXXX
 14.500        .4      4 +XX
 15.000        .2      2 +X
 15.500        .2      2 +X
 16.000        .4      4 +XX
 16.500        .3      3 +X
 17.000        .2      2 +X
 17.500        .5      5 +XX
 18.000        .1      1 +X
 18.500        .1      1 +X
 19.000        .4      4 +XX
 19.500        .4      4 +XX
 20.000        .1      1 +X
 20.500        .1      1 +X
 21.000        .1      1 +X
 21.500        .4      4 +XX
 22.000        .1      1 +X
 22.500        .3      3 +X
 23.000        .4      4 +XX
 23.500        .1      1 +X

              1.8     18 > 24.000
 TOTAL      100.0   1000  (INTERVAL WIDTH= .50000)
```

TIME SINCE LAST TRIP (10-60 minutes)   each X = 1 trip

```
LEFT-END    TOT%  COUNT

 10.000     0.       0 +
 11.000     2.1     21 +XXXXXXXXXXXXXXXXXXXXX
 12.000     2.0     20 +XXXXXXXXXXXXXXXXXXXX
 13.000     2.1     21 +XXXXXXXXXXXXXXXXXXXXX
 14.000     1.9     19 +XXXXXXXXXXXXXXXXXXX
 15.000     1.8     18 +XXXXXXXXXXXXXXXXXX
 16.000     1.6     16 +XXXXXXXXXXXXXXXX
 17.000      .6      6 +XXXXXX
 18.000     2.2     22 +XXXXXXXXXXXXXXXXXXXXXX
 19.000     1.5     15 +XXXXXXXXXXXXXXX
 20.000     1.0     10 +XXXXXXXXXX
 21.000     1.3     13 +XXXXXXXXXXXXX
 22.000      .9      9 +XXXXXXXXX
 23.000      .5      5 +XXXXX
 24.000      .9      9 +XXXXXXXXX
 25.000      .6      6 +XXXXXX
 26.000     1.5     15 +XXXXXXXXXXXXXXX
 27.000     1.0     10 +XXXXXXXXXX
 28.000     1.2     12 +XXXXXXXXXXXX
 29.000      .3      3 +XXX
 30.000      .7      7 +XXXXXXX
 31.000     1.0     10 +XXXXXXXXXX
 32.000     1.3     13 +XXXXXXXXXXXXX
 33.000      .8      8 +XXXXXXXX
 34.000      .7      7 +XXXXXXX
 35.000      .6      6 +XXXXXX
 36.000      .6      6 +XXXXXX
 37.000     1.0     10 +XXXXXXXXXX
 38.000      .7      7 +XXXXXXX
 39.000      .8      8 +XXXXXXXX
 40.000     1.0     10 +XXXXXXXXXX
 41.000      .4      4 +XXXX
 42.000      .5      5 +XXXXX
 43.000      .3      3 +XXX
 44.000      .3      3 +XXX
 45.000      .4      4 +XXXX
 46.000      .4      4 +XXXX
 47.000      .3      3 +XXX
 48.000      .4      4 +XXXX
 49.000      .4      4 +XXXX
 50.000    0.        0 +
 51.000      .2      2 +XX
 52.000      .3      3 +XXX
 53.000      .5      5 +XXXXX
 54.000      .4      4 +XXXX
 55.000      .4      4 +XXXX
 56.000      .6      6 +XXXXXX
 57.000      .1      1 +X
 58.000      .2      2 +XX
 59.000      .7      7 +XXXXXXX

           59.0    590 > 60.000
 TOTAL    100.0   1000  (INTERVAL WIDTH= 1.0000)
```

TOTAL TRIP TIME (0-90 minutes)

each X = 2 trips

```
LEFT-END      TOT%   COUNT

 0.            6.5     65 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000        6.7     67 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.0000       11.0    110 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 6.0000        9.1     91 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.0000        8.1     81 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000        7.1     71 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000        8.4     84 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 14.000        7.5     75 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 16.000        5.7     57 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18.000        5.9     59 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 20.000        3.9     39 +XXXXXXXXXXXXXXXXXXXX
 22.000        2.4     24 +XXXXXXXXXXXX
 24.000        2.7     27 +XXXXXXXXXXXXXX
 26.000        2.3     23 +XXXXXXXXXXXX
 28.000        1.6     16 +XXXXXXXX
 30.000        2.0     20 +XXXXXXXXXX
 32.000        1.1     11 +XXXXXX
 34.000         .6      6 +XXX
 36.000        1.2     12 +XXXXXX
 38.000        1.2     12 +XXXXXX
 40.000         .7      7 +XXXX
 42.000        1.0     10 +XXXXX
 44.000         .7      7 +XXXX
 46.000         .5      5 +XXX
 48.000         .4      4 +XX
 50.000        0.       0 +
 52.000         .2      2 +X
 54.000         .3      3 +XX
 56.000         .2      2 +X
 58.000         .1      1 +X
 60.000         .2      2 +X
 62.000        0.       0 +
 64.000         .1      1 +X
 66.000         .1      1 +X
 68.000        0.       0 +
 70.000         .1      1 +X
 72.000        0.       0 +
 74.000         .1      1 +X
 76.000         .1      1 +X
 78.000        0.       0 +
 80.000        0.       0 +
 82.000        0.       0 +
 84.000        0.       0 +
 86.000        0.       0 +
 88.000        0.       0 +

                .2      2 > 90.000
 TOTAL       100.0   1000  (INTERVAL WIDTH= 2.0000)
```

TOTAL TIME MOVING (0-25 minutes)   each X = 1 trip

```
LEFT-END     TOT%  COUNT

 0.          4.5    45 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000      1.5    15 +XXXXXXXXXXXXXXX
 1.0000      2.9    29 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000      2.2    22 +XXXXXXXXXXXXXXXXXXXXXX
 2.0000      3.4    34 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.5000      3.9    39 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.0000      2.9    29 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.5000      4.3    43 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.0000      3.6    36 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.5000      3.7    37 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 5.0000      2.8    28 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 5.5000      2.4    24 +XXXXXXXXXXXXXXXXXXXXXXXX
 6.0000      2.5    25 +XXXXXXXXXXXXXXXXXXXXXXXXX
 6.5000      3.3    33 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 7.0000      2.8    28 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 7.5000      2.8    28 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.0000      2.7    27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.5000      3.2    32 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 9.0000      2.8    28 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 9.5000      3.4    34 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000      1.7    17 +XXXXXXXXXXXXXXXXX
 10.500      2.2    22 +XXXXXXXXXXXXXXXXXXXXXX
 11.000      2.5    25 +XXXXXXXXXXXXXXXXXXXXXXXXX
 11.500      2.7    27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000      2.4    24 +XXXXXXXXXXXXXXXXXXXXXXXX
 12.500      2.1    21 +XXXXXXXXXXXXXXXXXXXXX
 13.000      1.8    18 +XXXXXXXXXXXXXXXXXX
 13.500      1.5    15 +XXXXXXXXXXXXXXX
 14.000      1.0    10 +XXXXXXXXXX
 14.500      1.8    18 +XXXXXXXXXXXXXXXXXX
 15.000      1.0    10 +XXXXXXXXXX
 15.500      1.1    11 +XXXXXXXXXXX
 16.000       .8     8 +XXXXXXXX
 16.500       .7     7 +XXXXXXX
 17.000       .7     7 +XXXXXXX
 17.500      1.4    14 +XXXXXXXXXXXXXX
 18.000       .3     3 +XXX
 18.500       .5     5 +XXXXX
 19.000       .6     6 +XXXXXX
 19.500       .7     7 +XXXXXXX
 20.000       .8     8 +XXXXXXXX
 20.500       .9     9 +XXXXXXXXX
 21.000       .4     4 +XXXX
 21.500       .8     8 +XXXXXXXX
 22.000      1.2    12 +XXXXXXXXXXXX
 22.500       .5     5 +XXXXX
 23.000      1.0    10 +XXXXXXXXXX
 23.500       .6     6 +XXXXXX
 24.000       .3     3 +XXX
 24.500       .5     5 +XXXXX

             3.9    39 > 25.000
 TOTAL     100.0  1000  (INTERVAL WIDTH= .50000)
```

TOTAL TIME STOPPED (0-20 minutes)

each X = 2 trips

```
LEFT-END     TOT%   COUNT

 0.           5.7     57 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000      10.1    101 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000      11.9    119 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000       8.4     84 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       9.8     98 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.5000       6.3     63 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.0000       5.0     50 +XXXXXXXXXXXXXXXXXXXXXXXXX
 3.5000       4.1     41 +XXXXXXXXXXXXXXXXXXXXX
 4.0000       3.4     34 +XXXXXXXXXXXXXXXXX
 4.5000       3.6     36 +XXXXXXXXXXXXXXXXXX
 5.0000       1.7     17 +XXXXXXXXX
 5.5000       2.0     20 +XXXXXXXXXX
 6.0000       2.3     23 +XXXXXXXXXXXX
 6.5000       1.7     17 +XXXXXXXXX
 7.0000       1.7     17 +XXXXXXXXX
 7.5000       1.5     15 +XXXXXXXX
 8.0000        .7      7 +XXXX
 8.5000        .9      9 +XXXXX
 9.0000       1.5     15 +XXXXXXXX
 9.5000       1.1     11 +XXXXXX
 10.000       1.2     12 +XXXXXX
 10.500        .7      7 +XXXX
 11.000        .4      4 +XX
 11.500       1.6     16 +XXXXXXXX
 12.000        .5      5 +XXX
 12.500        .9      9 +XXXXX
 13.000       1.1     11 +XXXXXX
 13.500        .3      3 +XX
 14.000        .8      8 +XXXX
 14.500        .7      7 +XXXX
 15.000        .5      5 +XXX
 15.500        .2      2 +X
 16.000        .5      5 +XXX
 16.500        .3      3 +XX
 17.000        .4      4 +XX
 17.500        .3      3 +XX
 18.000        .5      5 +XXX
 18.500        .5      5 +XXX
 19.000        .4      4 +XX
 19.500        .6      6 +XXX

              4.2     42 > 20.000
 TOTAL      100.0   1000  (INTERVAL WIDTH= .50000)
```

NUMBER OF TRIPS MADE PER DAY (0-17) each x = 1 day

URBAN DAYS

```
MIDPOINT   HIST%  COUNT

   0.0       9.0     17 +XXXXXXXXXXXXXXXXX
   1.0       6.4     12 +XXXXXXXXXXXX
   2.0      11.2     21 +XXXXXXXXXXXXXXXXXXXXX
   3.0      14.4     27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.0      10.1     19 +XXXXXXXXXXXXXXXXXXX
   5.0      14.4     27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
   6.0       8.0     15 +XXXXXXXXXXXXXXX
   7.0       8.0     15 +XXXXXXXXXXXXXXX
   8.0       8.0     15 +XXXXXXXXXXXXXXX
   9.0       4.3      8 +XXXXXXXX
  10.0       2.7      5 +XXXXX
  11.0       1.1      2 +XX
  12.0       0.0      0 +
  13.0       0.5      1 +X
  14.0       0.0      0 +
  15.0       0.5      1 +X
  16.0       1.1      2 +XX
  17.0       0.5      1 +X

 TOTAL              188  (INTERVAL WIDTH= 1.0000)
```

Appendix D

Histograms: Urban And/Or Rural Trips

Trips Separated By An Engine Off Period Exceeding 10 Minutes

Histogram Key:

LEFT-END = Lower limit of interval
TOT % = Percent of total trips found in interval
COUNT = Number of trips in interval

AVERAGE TRIP SPEED (0-64 mph) **each X = 2 trips**

```
LEFT-END     TOT%  COUNT

 0.           4.1    52 +XXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       2.1    26 +XXXXXXXXXXXXX
 4.0000       3.7    46 +XXXXXXXXXXXXXXXXXXXXXXX
 6.0000       4.1    51 +XXXXXXXXXXXXXXXXXXXXXXXXXX
 8.0000       5.0    63 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000       5.9    74 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000       8.4   106 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 14.000       9.8   123 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 16.000       9.2   115 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18.000       7.3    91 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 20.000       9.6   121 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 22.000       7.1    89 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 24.000       5.7    72 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 26.000       3.6    45 +XXXXXXXXXXXXXXXXXXXXXXX
 28.000       2.9    36 +XXXXXXXXXXXXXXXXXX
 30.000       2.1    26 +XXXXXXXXXXXXX
 32.000       1.9    24 +XXXXXXXXXXXX
 34.000       1.8    23 +XXXXXXXXXXXX
 36.000        .8    10 +XXXXX
 38.000       1.0    12 +XXXXXX
 40.000        .9    11 +XXXXXX
 42.000        .7     9 +XXXXX
 44.000        .7     9 +XXXXX
 46.000        .3     4 +XX
 48.000        .3     4 +XX
 50.000        .5     6 +XXX
 52.000        .1     1 +X
 54.000        .1     1 +X
 56.000        .2     2 +X
 58.000       0.      0 +
 60.000        .2     2 +X
 62.000        .1     1 +X

 TOTAL      100.0  1255  (INTERVAL WIDTH= 2.0000)
```

NUMBER OF STOPS DURING TRIP (0-35) **each X = 2 trips**

```
LEFT-END      TOT%  COUNT

 0.            0.       0 +
 1.0000        3.5     44 +XXXXXXXXXXXXXXXXXXXXXX
 2.0000        5.2     65 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.0000        7.5     94 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.0000       10.3    129 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 5.0000        8.7    109 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 6.0000       10.3    129 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 7.0000        8.3    104 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.0000        8.1    102 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 9.0000        6.1     77 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000        5.3     66 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 11.000        4.1     52 +XXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000        3.1     39 +XXXXXXXXXXXXXXXXXXXX
 13.000        3.3     42 +XXXXXXXXXXXXXXXXXXXXX
 14.000        2.2     27 +XXXXXXXXXXXXXX
 15.000        2.2     27 +XXXXXXXXXXXXXX
 16.000        1.8     22 +XXXXXXXXXXX
 17.000        1.5     19 +XXXXXXXXXX
 18.000        1.5     19 +XXXXXXXXXX
 19.000        1.4     17 +XXXXXXXXX
 20.000        1.2     15 +XXXXXXXX
 21.000         .6      8 +XXXX
 22.000        1.0     12 +XXXXXX
 23.000         .3      4 +XX
 24.000         .4      5 +XXX
 25.000         .5      6 +XXX
 26.000         .2      2 +X
 27.000         .4      5 +XXX
 28.000         .2      2 +X
 29.000         .1      1 +X
 30.000         .1      1 +X
 31.000        0.       0 +
 32.000         .2      2 +X
 33.000        0.       0 +
 34.000         .2      3 +XX

                .5      6 > 35.000
 TOTAL       100.0   1255  (INTERVAL WIDTH= 1.0000)
```

TRIP DISTANCE (0-25 miles)

each X = 2 trips

```
 LEFT-END     TOT%   COUNT

  0.          10.1    127 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  .50000       8.3    104 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  1.0000       8.0    101 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  1.5000       6.8     85 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2.0000       4.9     62 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  2.5000       5.0     63 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3.0000       5.4     68 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  3.5000       3.8     48 +XXXXXXXXXXXXXXXXXXXXXXXX
  4.0000       6.2     78 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4.5000       4.0     50 +XXXXXXXXXXXXXXXXXXXXXXXXX
  5.0000       2.6     33 +XXXXXXXXXXXXXXXXX
  5.5000       3.8     48 +XXXXXXXXXXXXXXXXXXXXXXXX
  6.0000       3.3     42 +XXXXXXXXXXXXXXXXXXXXX
  6.5000       2.2     28 +XXXXXXXXXXXXXX
  7.0000       1.9     24 +XXXXXXXXXXXX
  7.5000       1.5     19 +XXXXXXXXXX
  8.0000       1.4     17 +XXXXXXXXX
  8.5000       1.5     19 +XXXXXXXXXX
  9.0000       1.7     21 +XXXXXXXXXXX
  9.5000       1.0     13 +XXXXXXX
  10.000       1.7     21 +XXXXXXXXXXX
  10.500        .8     10 +XXXXX
  11.000       1.3     16 +XXXXXXXX
  11.500       1.1     14 +XXXXXXX
  12.000       1.0     12 +XXXXXX
  12.500        .8     10 +XXXXX
  13.000        .7      9 +XXXXX
  13.500        .4      5 +XXX
  14.000        .3      4 +XX
  14.500        .5      6 +XXX
  15.000        .5      6 +XXX
  15.500        .4      5 +XXX
  16.000        .4      5 +XXX
  16.500        .2      3 +XX
  17.000        .3      4 +XX
  17.500        .4      5 +XXX
  18.000        .2      2 +X
  18.500        .2      3 +XX
  19.000        .6      7 +XXXX
  19.500        .3      4 +XX
  20.000        .1      1 +X
  20.500        .2      3 +XX
  21.000        .1      1 +X
  21.500      0.        0 +
  22.000        .2      2 +X
  22.500        .1      1 +X
  23.000        .3      4 +XX
  23.500        .2      2 +X
  24.000        .1      1 +X
  24.500        .1      1 +X

               3.0     38 > 25.000
  TOTAL      100.0   1255  (INTERVAL WIDTH= .50000)
```

TRIP DISTANCE (0-2 miles) **each X = 1 trip**

```
 LEFT-END     TOT%  COUNT

  0.           4.6    58 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  .10000       1.4    17 +XXXXXXXXXXXXXXXXX
  .20000       1.8    23 +XXXXXXXXXXXXXXXXXXXXXXX
  .30000       1.8    22 +XXXXXXXXXXXXXXXXXXXXXX
  .40000        .6     7 +XXXXXXX
  .50000       1.5    19 +XXXXXXXXXXXXXXXXXXX
  .60000       2.1    26 +XXXXXXXXXXXXXXXXXXXXXXXXXX
  .70000       1.4    17 +XXXXXXXXXXXXXXXXX
  .80000       1.4    17 +XXXXXXXXXXXXXXXXX
  .90000       2.0    25 +XXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000        1.9    24 +XXXXXXXXXXXXXXXXXXXXXXXX
 1.1000        1.6    20 +XXXXXXXXXXXXXXXXXXXX
 1.2000        1.4    18 +XXXXXXXXXXXXXXXXXX
 1.3000        1.5    19 +XXXXXXXXXXXXXXXXXXX
 1.4000        1.6    20 +XXXXXXXXXXXXXXXXXXXX
 1.5000        1.5    19 +XXXXXXXXXXXXXXXXXXX
 1.6000        1.1    14 +XXXXXXXXXXXXXX
 1.7000        1.6    20 +XXXXXXXXXXXXXXXXXXXX
 1.8000        1.0    13 +XXXXXXXXXXXXX
 1.9000        1.6    20 +XXXXXXXXXXXXXXXXXXXX

              66.7   837 > 2.0000
 TOTAL       100.0  1255  (INTERVAL WIDTH= .10000)
```

AVERAGE DISTANCE BETWEEN STOPS (0-5 miles) each X = 2 trips

```
LEFT-END     TOT%  COUNT

 0.           6.9    86 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .10000       8.8   110 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .20000      11.5   144 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .30000      11.7   147 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .40000      11.0   138 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000       9.7   122 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .60000       7.6    96 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .70000       4.6    58 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .80000       4.7    59 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .90000       3.7    46 +XXXXXXXXXXXXXXXXXXXXXXX
 1.0000       2.2    28 +XXXXXXXXXXXXXX
 1.1000       1.8    22 +XXXXXXXXXXX
 1.2000       1.3    16 +XXXXXXXX
 1.3000       1.0    13 +XXXXXXX
 1.4000       1.8    22 +XXXXXXXXXXX
 1.5000       1.1    14 +XXXXXXX
 1.6000        .6     8 +XXXX
 1.7000        .4     5 +XXX
 1.8000        .5     6 +XXX
 1.9000        .4     5 +XXX
 2.0000        .7     9 +XXXXX
 2.1000        .6     7 +XXXX
 2.2000        .3     4 +XX
 2.3000        .3     4 +XX
 2.4000        .2     2 +X
 2.5000        .4     5 +XXX
 2.6000        .4     5 +XXX
 2.7000        .4     5 +XXX
 2.8000        .3     4 +XX
 2.9000        .7     9 +XXXXX
 3.0000        .2     2 +X
 3.1000        .1     1 +X
 3.2000        .2     3 +XX
 3.3000        .2     3 +XX
 3.4000      0.       0 +
 3.5000      0.       0 +
 3.6000        .1     1 +X
 3.7000      0.       0 +
 3.8000        .1     1 +X
 3.9000        .2     2 +X
 4.0000        .3     4 +XX
 4.1000      0.       0 +
 4.2000        .1     1 +X
 4.3000        .1     1 +X
 4.4000      0.       0 +
 4.5000      0.       0 +
 4.6000        .1     1 +X
 4.7000      0.       0 +
 4.8000        .2     2 +X
 4.9000        .2     2 +X

              2.5    32 > 5.0000
 TOTAL      100.0  1255  (INTERVAL WIDTH= .10000)
```

TIME SINCE LAST TRIP (0-24 hours)

each X = 4 trips

```
LEFT-END    TOT%  COUNT

 0.         24.7   310 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000     15.9   199 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000      9.0   113 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000      6.9    86 +XXXXXXXXXXXXXXXXXXXXXX
 2.0000      3.7    47 +XXXXXXXXXXXX
 2.5000      3.7    46 +XXXXXXXXXXXX
 3.0000      2.9    37 +XXXXXXXXXX
 3.5000      2.5    31 +XXXXXXXX
 4.0000      2.5    31 +XXXXXXXX
 4.5000      2.4    30 +XXXXXXXX
 5.0000      1.0    13 +XXXX
 5.5000       .9    11 +XXX
 6.0000       .5     6 +XX
 6.5000       .4     5 +XX
 7.0000       .2     3 +X
 7.5000       .4     5 +XX
 8.0000      1.6    20 +XXXXX
 8.5000      2.3    29 +XXXXXXXX
 9.0000      1.0    12 +XXX
 9.5000      1.5    19 +XXXXX
 10.000      1.4    18 +XXXXX
 10.500      1.0    12 +XXX
 11.000      1.4    18 +XXXXX
 11.500      1.2    15 +XXXX
 12.000       .9    11 +XXX
 12.500       .8    10 +XXX
 13.000       .7     9 +XXX
 13.500       .6     8 +XX
 14.000      1.4    17 +XXXXX
 14.500       .4     5 +XX
 15.000       .5     6 +XX
 15.500       .4     5 +XX
 16.000       .4     5 +XX
 16.500       .3     4 +X
 17.000       .3     4 +X
 17.500       .5     6 +XX
 18.000       .1     1 +X
 18.500       .1     1 +X
 19.000       .4     5 +XX
 19.500       .3     4 +X
 20.000       .2     2 +X
 20.500       .2     2 +X
 21.000       .1     1 +X
 21.500       .3     4 +X
 22.000       .2     2 +X
 22.500       .2     3 +X
 23.000       .3     4 +X
 23.500       .2     2 +X

             1.4    18 > 24.000
 TOTAL     100.0  1255  (INTERVAL WIDTH= .50000)
```

TIME SINCE LAST TRIP (10-60 minutes)   each X = 1 trip

```
LEFT-END     TOT%  COUNT

 10.000       0.      0 +
 11.000       2.4    30 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000       1.8    23 +XXXXXXXXXXXXXXXXXXXXXXX
 13.000       1.9    24 +XXXXXXXXXXXXXXXXXXXXXXXX
 14.000       1.8    23 +XXXXXXXXXXXXXXXXXXXXXXX
 15.000       1.9    24 +XXXXXXXXXXXXXXXXXXXXXXXX
 16.000       1.5    19 +XXXXXXXXXXXXXXXXXXX
 17.000       1.0    12 +XXXXXXXXXXXX
 18.000       1.8    23 +XXXXXXXXXXXXXXXXXXXXXXX
 19.000       1.4    18 +XXXXXXXXXXXXXXXXXX
 20.000       1.0    12 +XXXXXXXXXXXX
 21.000       1.4    17 +XXXXXXXXXXXXXXXXX
 22.000        .9    11 +XXXXXXXXXXX
 23.000        .6     7 +XXXXXXX
 24.000       1.2    15 +XXXXXXXXXXXXXXX
 25.000        .6     7 +XXXXXXX
 26.000       1.4    17 +XXXXXXXXXXXXXXXXX
 27.000       1.0    13 +XXXXXXXXXXXXX
 28.000       1.0    12 +XXXXXXXXXXXX
 29.000        .2     3 +XXX
 30.000        .6     8 +XXXXXXXX
 31.000        .8    10 +XXXXXXXXXX
 32.000       1.2    15 +XXXXXXXXXXXXXXX
 33.000       1.0    12 +XXXXXXXXXXXX
 34.000        .6     8 +XXXXXXXX
 35.000        .6     7 +XXXXXXX
 36.000        .6     7 +XXXXXXX
 37.000       1.0    13 +XXXXXXXXXXXXX
 38.000        .9    11 +XXXXXXXXXXX
 39.000        .8    10 +XXXXXXXXXX
 40.000       1.0    13 +XXXXXXXXXXXXX
 41.000        .4     5 +XXXXX
 42.000        .5     6 +XXXXXX
 43.000        .5     6 +XXXXXX
 44.000        .2     3 +XXX
 45.000        .4     5 +XXXXX
 46.000        .5     6 +XXXXXX
 47.000        .2     3 +XXX
 48.000        .6     7 +XXXXXXX
 49.000        .3     4 +XXXX
 50.000        .2     3 +XXX
 51.000        .2     2 +XX
 52.000        .2     3 +XXX
 53.000        .4     5 +XXXXX
 54.000        .6     7 +XXXXXXX
 55.000        .3     4 +XXXX
 56.000        .6     7 +XXXXXXX
 57.000        .2     2 +XX
 58.000        .2     2 +XX
 59.000        .9    11 +XXXXXXXXXXX

             59.0   740 > 60.000
 TOTAL      100.0  1255  (INTERVAL WIDTH= 1.0000)
```

TOTAL TRIP TIME (0-90 minutes)   each X = 2 trips

```
LEFT-END     TOT%   COUNT

 0.           6.5     82 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       6.9     87 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.0000      10.4    131 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 6.0000       8.3    104 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.0000       7.8     98 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000       6.9     86 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000       7.5     94 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 14.000       6.8     85 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 16.000       5.6     70 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18.000       5.7     71 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 20.000       4.0     50 +XXXXXXXXXXXXXXXXXXXXXXXXX
 22.000       2.3     29 +XXXXXXXXXXXXXXX
 24.000       2.9     37 +XXXXXXXXXXXXXXXXXXX
 26.000       2.2     27 +XXXXXXXXXXXXXX
 28.000       1.7     21 +XXXXXXXXXXX
 30.000       1.8     23 +XXXXXXXXXXXX
 32.000       1.2     15 +XXXXXXXX
 34.000       1.0     13 +XXXXXXX
 36.000       1.5     19 +XXXXXXXXXX
 38.000       1.4     17 +XXXXXXXXX
 40.000        .8     10 +XXXXX
 42.000       1.2     15 +XXXXXXXX
 44.000       1.0     13 +XXXXXXX
 46.000        .5      6 +XXX
 48.000        .4      5 +XXX
 50.000        .2      3 +XX
 52.000        .3      4 +XX
 54.000        .4      5 +XXX
 56.000        .2      2 +X
 58.000        .2      2 +X
 60.000        .3      4 +XX
 62.000        .1      1 +X
 64.000        .2      2 +X
 66.000        .1      1 +X
 68.000        .2      2 +X
 70.000        .2      3 +XX
 72.000       0.       0 +
 74.000        .1      1 +X
 76.000        .1      1 +X
 78.000        .1      1 +X
 80.000       0.       0 +
 82.000       0.       0 +
 84.000        .1      1 +X
 86.000       0.       0 +
 88.000        .1      1 +X

              1.0     13 > 90.000
 TOTAL      100.0   1255  (INTERVAL WIDTH= 2.0000)
```

TOTAL TIME MOVING (0-25 minutes)   each X = 1 trip

```
LEFT-END     TOT%  COUNT

 0.           4.5    56 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000       1.5    19 +XXXXXXXXXXXXXXXXXXX
 1.0000       2.7    34 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000       2.2    28 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       3.2    40 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.5000       4.0    50 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.0000       3.1    39 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.5000       3.7    46 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.0000       3.5    44 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4.5000       3.7    46 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 5.0000       2.6    33 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 5.5000       2.2    27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 6.0000       2.2    27 +XXXXXXXXXXXXXXXXXXXXXXXXXXX
 6.5000       3.0    38 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 7.0000       2.9    36 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 7.5000       2.5    32 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.0000       2.5    32 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8.5000       3.3    41 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 9.0000       2.4    30 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 9.5000       3.1    39 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 10.000       1.5    19 +XXXXXXXXXXXXXXXXXXX
 10.500       2.3    29 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 11.000       2.3    29 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 11.500       2.5    31 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 12.000       2.1    26 +XXXXXXXXXXXXXXXXXXXXXXXXXX
 12.500       1.7    21 +XXXXXXXXXXXXXXXXXXXXX
 13.000       1.5    19 +XXXXXXXXXXXXXXXXXXX
 13.500       1.5    19 +XXXXXXXXXXXXXXXXXXX
 14.000       1.0    12 +XXXXXXXXXXXX
 14.500       1.8    22 +XXXXXXXXXXXXXXXXXXXXXX
 15.000       1.4    17 +XXXXXXXXXXXXXXXXX
 15.500       1.0    12 +XXXXXXXXXXXX
 16.000       1.0    13 +XXXXXXXXXXXXX
 16.500        .9    11 +XXXXXXXXXXX
 17.000        .6     7 +XXXXXXX
 17.500       1.4    18 +XXXXXXXXXXXXXXXXXX
 18.000        .4     5 +XXXXX
 18.500        .6     8 +XXXXXXXX
 19.000        .6     8 +XXXXXXXX
 19.500        .6     7 +XXXXXXX
 20.000        .8    10 +XXXXXXXXXX
 20.500        .9    11 +XXXXXXXXXXX
 21.000        .5     6 +XXXXXX
 21.500        .7     9 +XXXXXXXXX
 22.000       1.1    14 +XXXXXXXXXXXXXX
 22.500        .4     5 +XXXXX
 23.000       1.0    13 +XXXXXXXXXXXXX
 23.500        .6     7 +XXXXXXX
 24.000        .4     5 +XXXXX
 24.500        .5     6 +XXXXXX

              7.9    99 > 25.000
 TOTAL      100.0  1255  (INTERVAL WIDTH= .50000)
```

TOTAL TIME STOPPED (0-20 minutes)   each X = 2 trips

```
LEFT-END     TOT%   COUNT

 0.           6.3     79 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 .50000      10.3    129 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.0000      11.3    142 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 1.5000       8.0    101 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.0000       9.3    117 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2.5000       6.3     79 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.0000       5.2     65 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3.5000       3.8     48 +XXXXXXXXXXXXXXXXXXXXXXXX
 4.0000       3.7     46 +XXXXXXXXXXXXXXXXXXXXXXX
 4.5000       3.5     44 +XXXXXXXXXXXXXXXXXXXXXX
 5.0000       1.8     22 +XXXXXXXXXXX
 5.5000       2.1     26 +XXXXXXXXXXXXX
 6.0000       2.2     27 +XXXXXXXXXXXXXX
 6.5000       1.7     21 +XXXXXXXXXXX
 7.0000       1.5     19 +XXXXXXXXXX
 7.5000       1.5     19 +XXXXXXXXXX
 8.0000       1.0     13 +XXXXXXX
 8.5000        .9     11 +XXXXXX
 9.0000       1.5     19 +XXXXXXXXXX
 9.5000       1.0     13 +XXXXXXX
 10.000       1.2     15 +XXXXXXXX
 10.500       1.0     12 +XXXXXX
 11.000        .6      8 +XXXX
 11.500       1.4     18 +XXXXXXXXX
 12.000        .6      7 +XXXX
 12.500       1.0     12 +XXXXXX
 13.000       1.0     13 +XXXXXXX
 13.500        .2      3 +XX
 14.000        .9     11 +XXXXXX
 14.500        .6      7 +XXXX
 15.000        .5      6 +XXX
 15.500        .4      5 +XXX
 16.000        .5      6 +XXX
 16.500        .2      3 +XX
 17.000        .5      6 +XXX
 17.500        .3      4 +XX
 18.000        .5      6 +XXX
 18.500        .5      6 +XXX
 19.000        .4      5 +XXX
 19.500        .5      6 +XXX

              4.5     56 > 20.000
 TOTAL      100.0   1255  (INTERVAL WIDTH= .50000)
```

NUMBER OF TRIPS MADE PER DAY (0-17) each x = 1 day

```
MIDPOINT   HIST%  COUNT

   0.0       6.8    17 +XXXXXXXXXXXXXXXXX
   1.0       4.8    12 +XXXXXXXXXXXX
   2.0      10.4    26 +XXXXXXXXXXXXXXXXXXXXXXXXXX
   3.0      12.4    31 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.0      11.2    28 +XXXXXXXXXXXXXXXXXXXXXXXXXXXX
   5.0      13.9    35 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   6.0      11.6    29 +XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   7.0      10.0    25 +XXXXXXXXXXXXXXXXXXXXXXXXX
   8.0       8.4    21 +XXXXXXXXXXXXXXXXXXXXX
   9.0       4.4    11 +XXXXXXXXXXX
  10.0       2.8     7 +XXXXXXX
  11.0       0.8     2 +XX
  12.0       0.4     1 +X
  13.0       0.8     2 +XX
  14.0       0.0     0 +
  15.0       0.4     1 +X
  16.0       0.8     2 +XX
  17.0       0.4     1 +X

 TOTAL             251  (INTERVAL WIDTH= 1.0000)
```

Appendix E

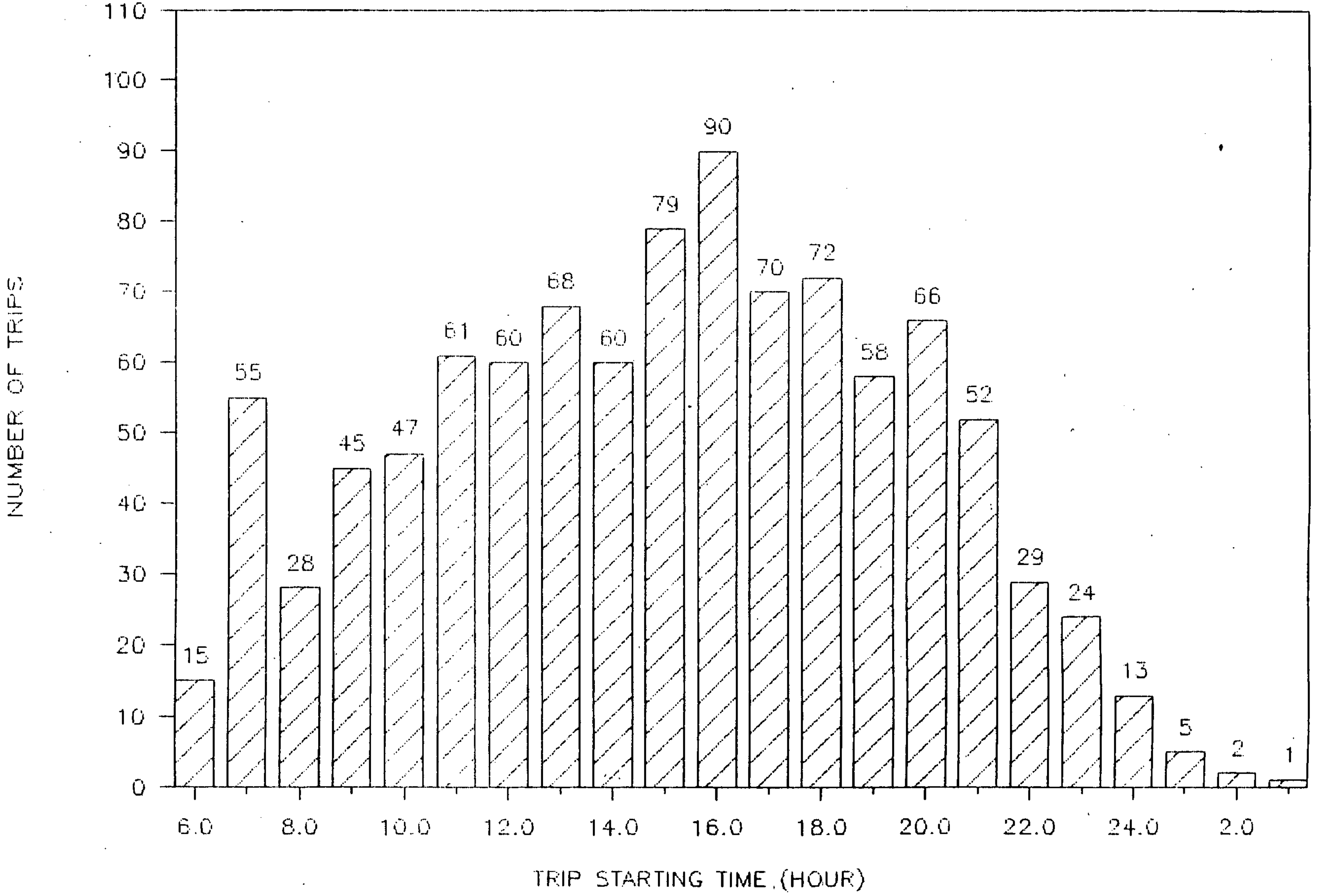
Graphs

Trip Parameter vs. Trip Starting Time
Urban Only Trips

Trips Separated By An Engine Off
Period Exceeding 10 Minutes

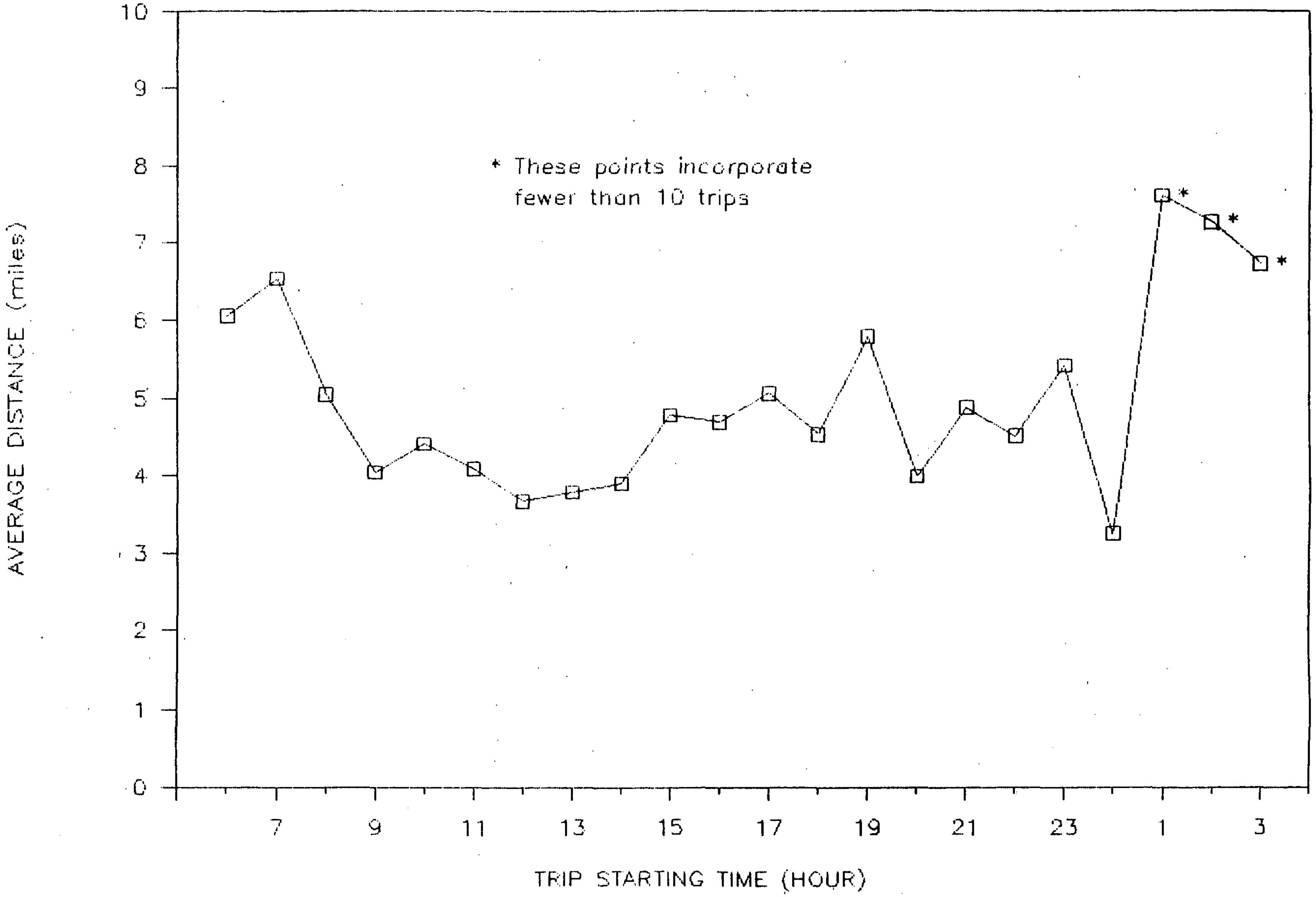
NUMBER OF TRIPS vs TIME OF DAY

(Urban Only)



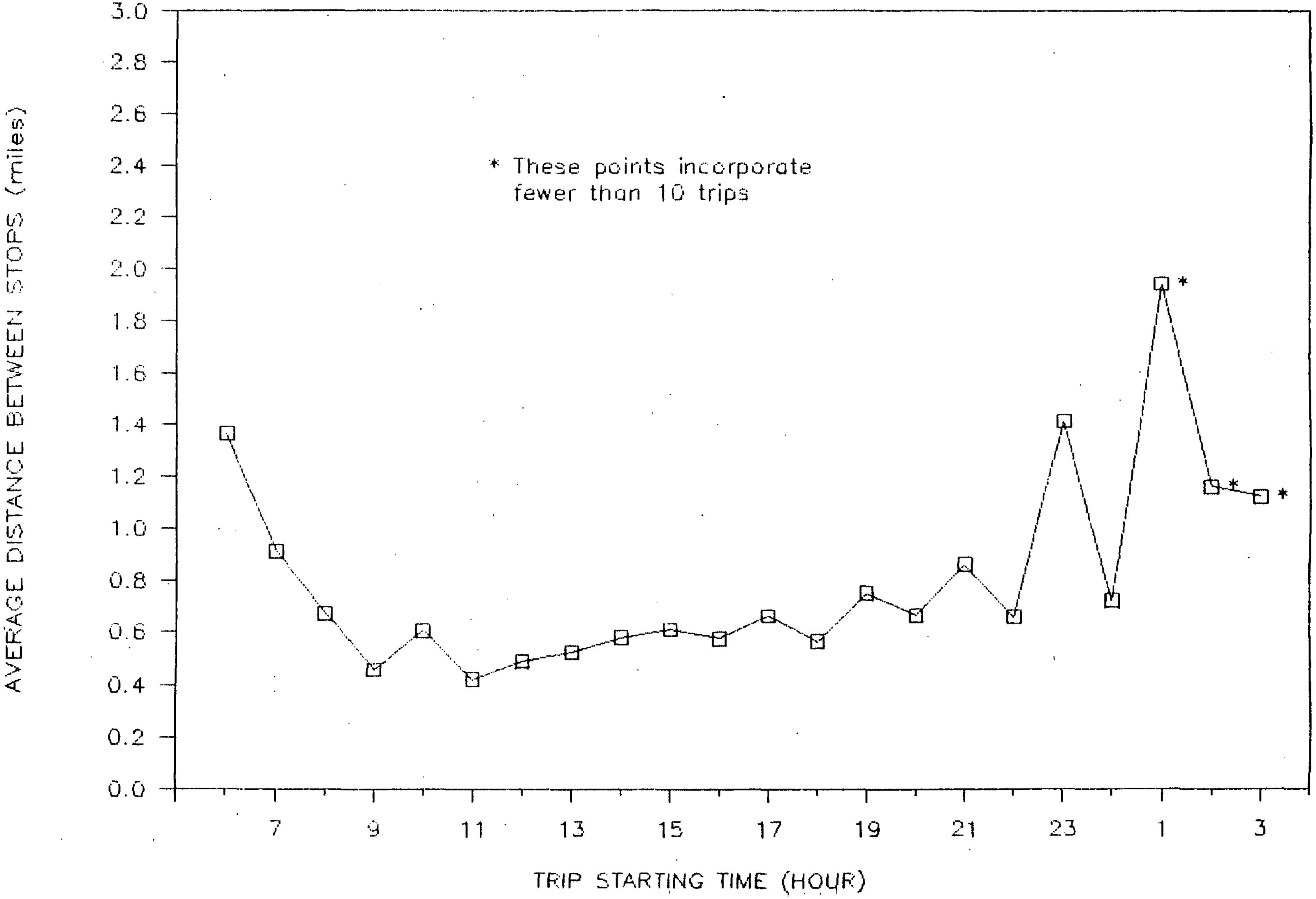
AVERAGE DISTANCE TRAVELLED

(Urban Only)



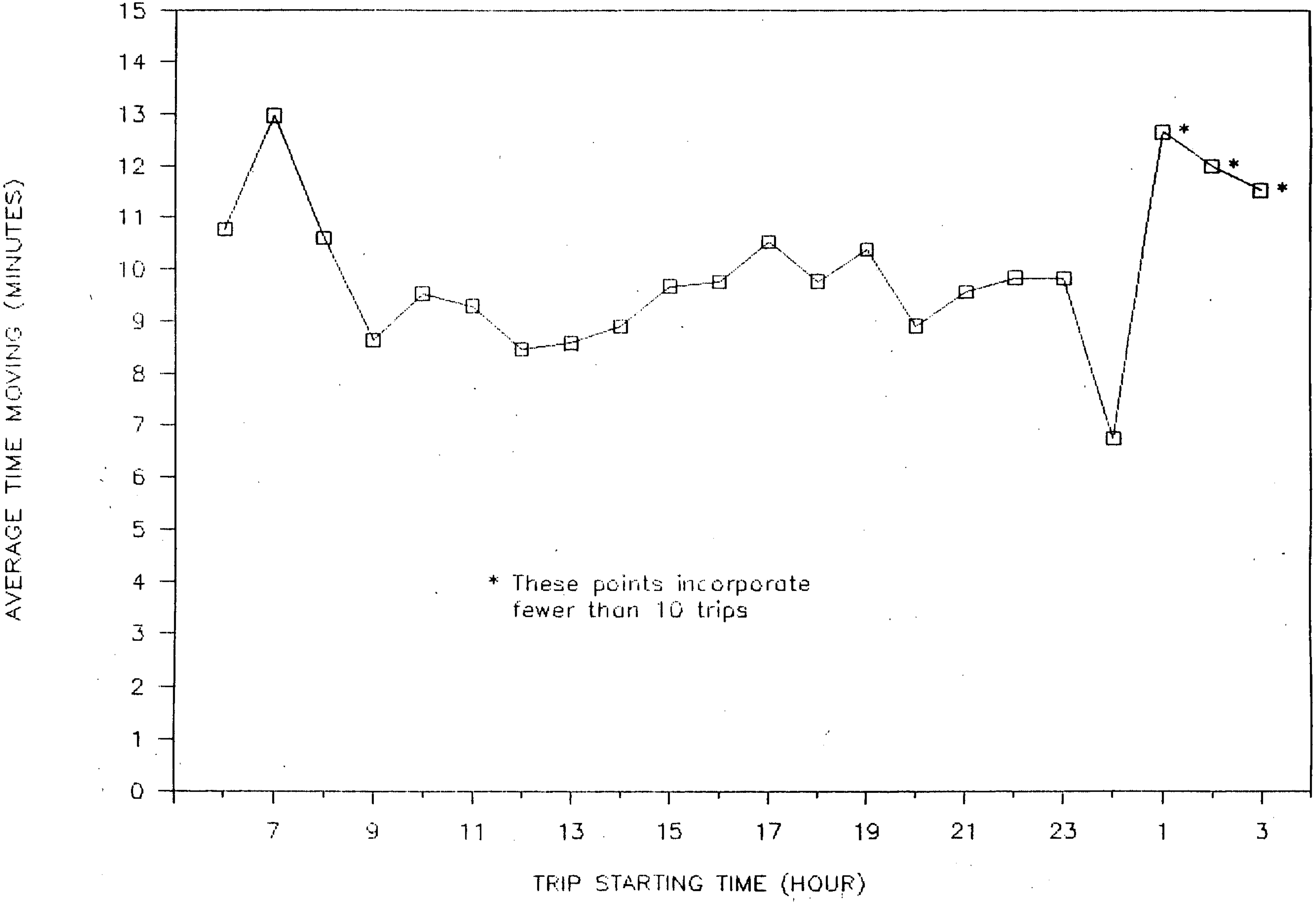
DISTANCE BETWEEN STOPS

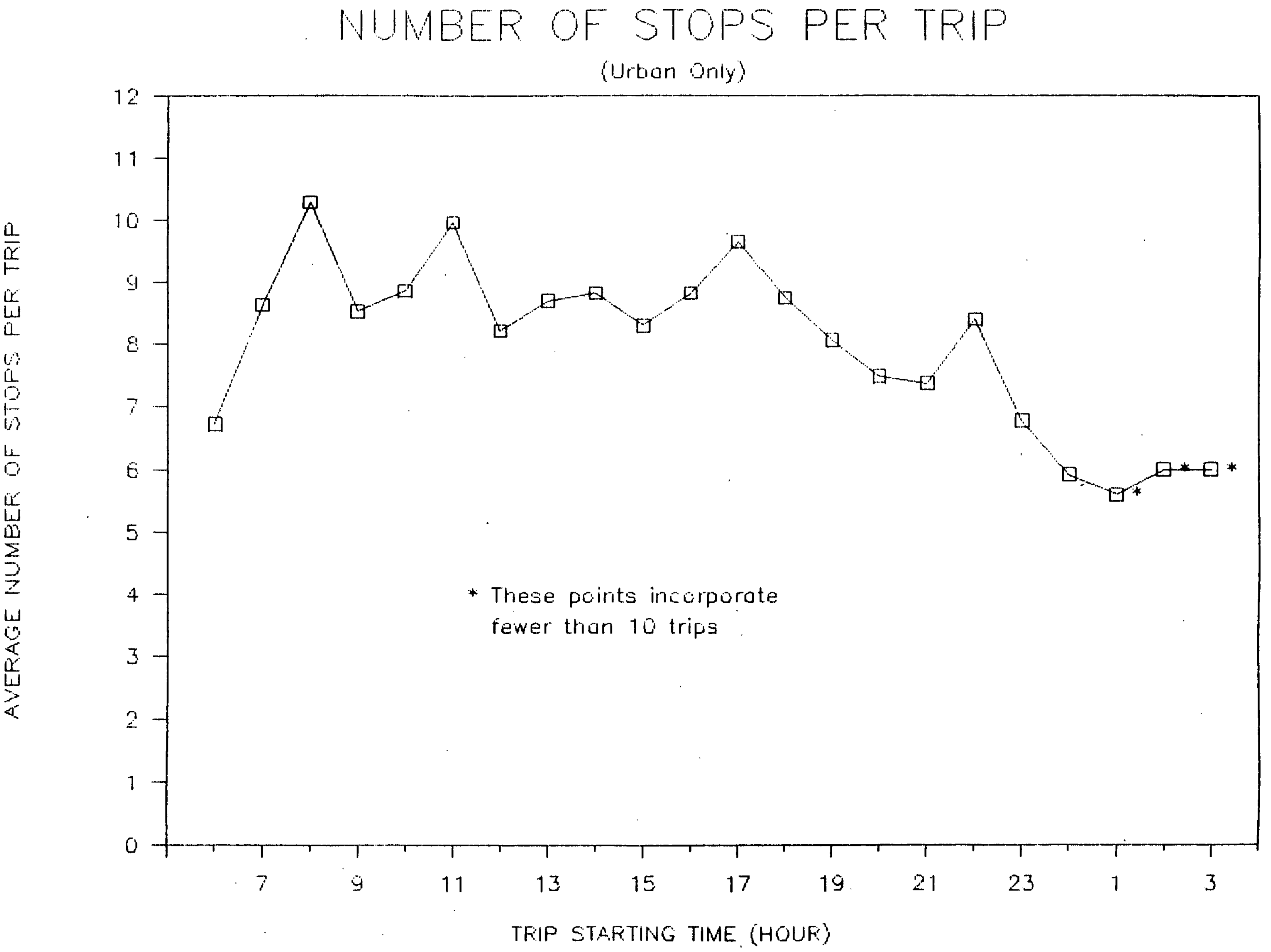
(Urban Only)



TIME MOVING PER TRIP

(Urban Only)





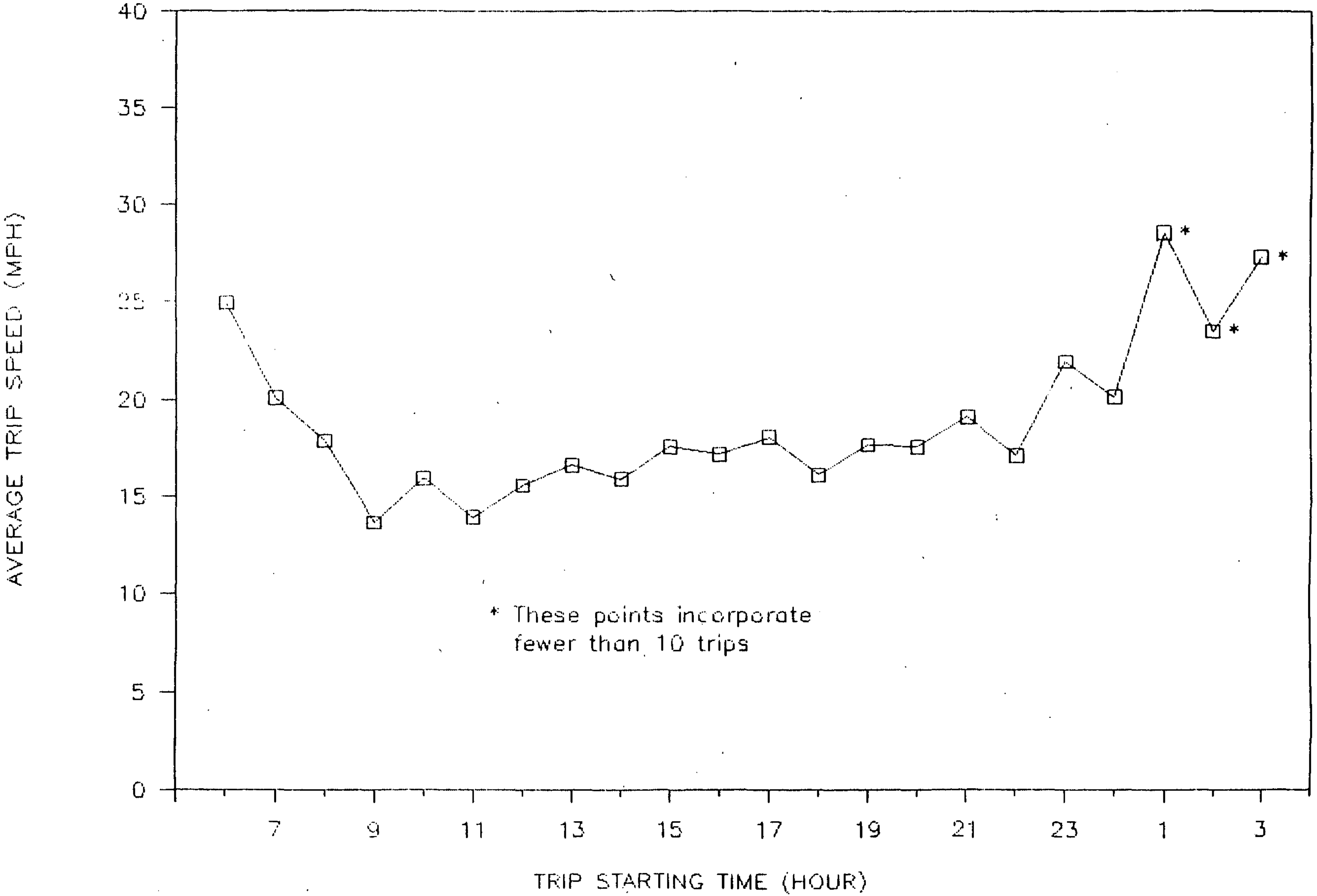
NUMBER OF STOPS PER TRIP

(Urban Only)

* These points incorporate fewer than 10 trips

AVERAGE TRIP SPEED

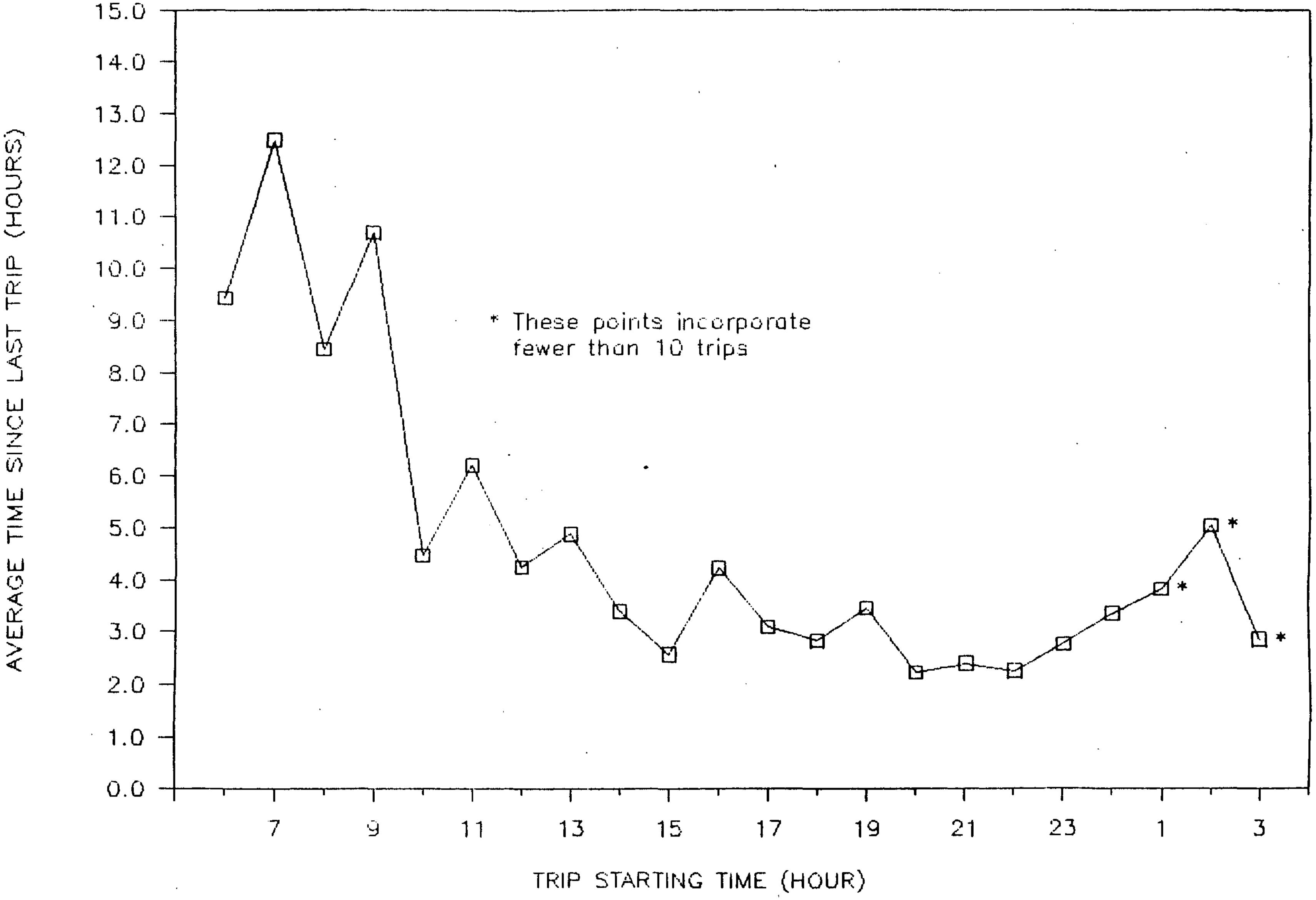
(Urban Only)

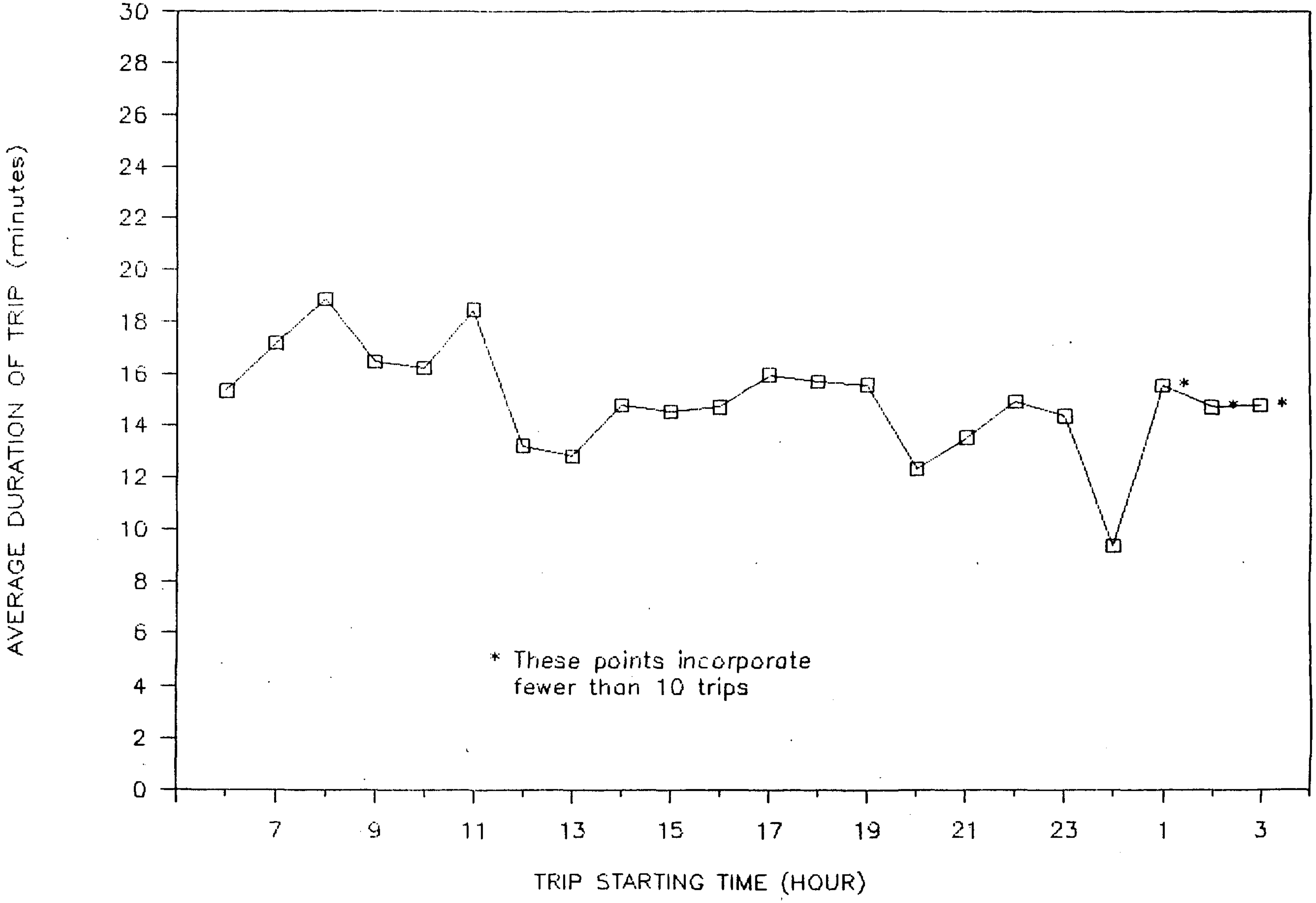


* These points incorporate fewer than 10 trips

TIME SINCE LAST TRIP

(Urban Only)





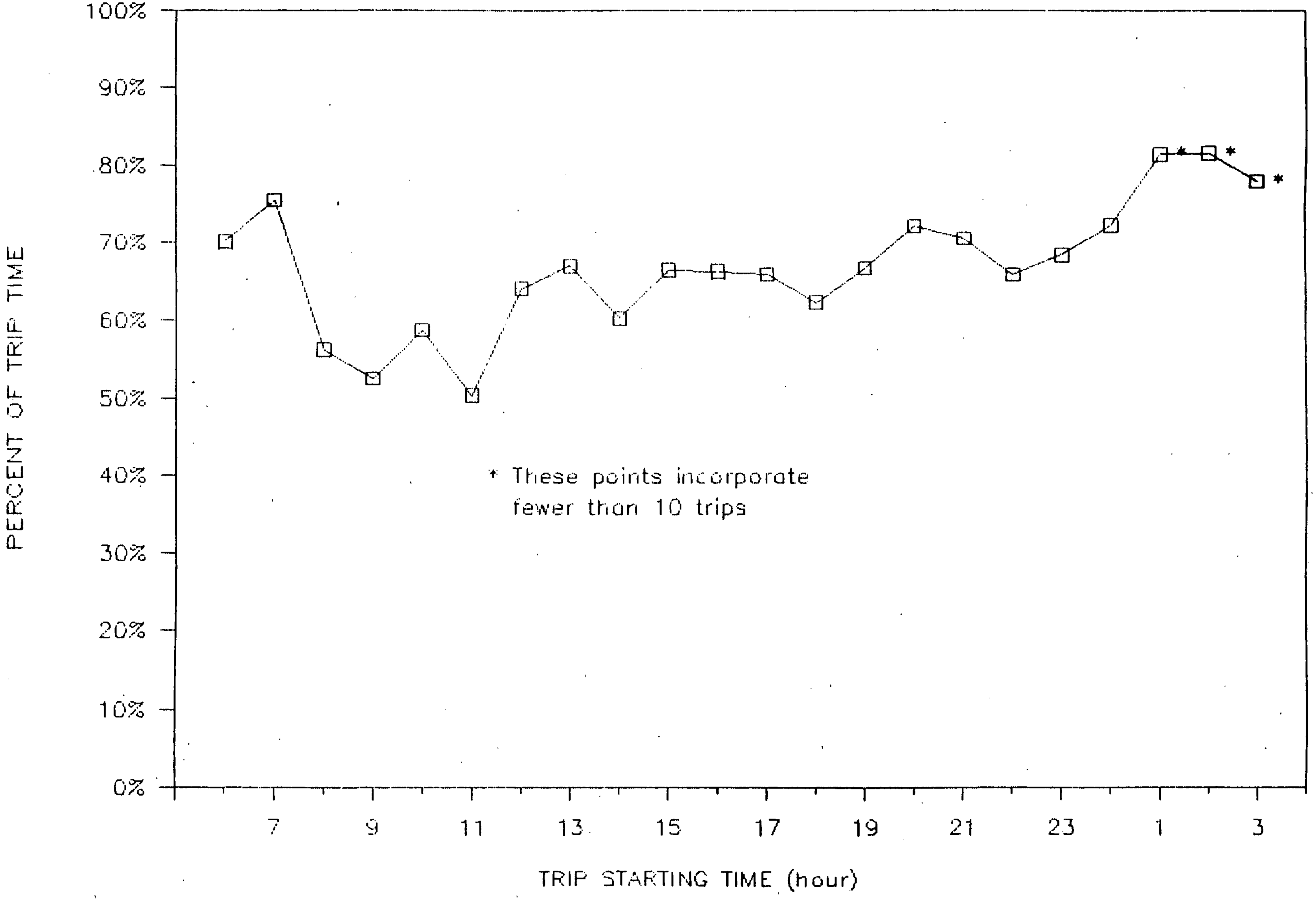
TRIP DURATION

(Urban Only)

* These points incorporate fewer than 10 trips

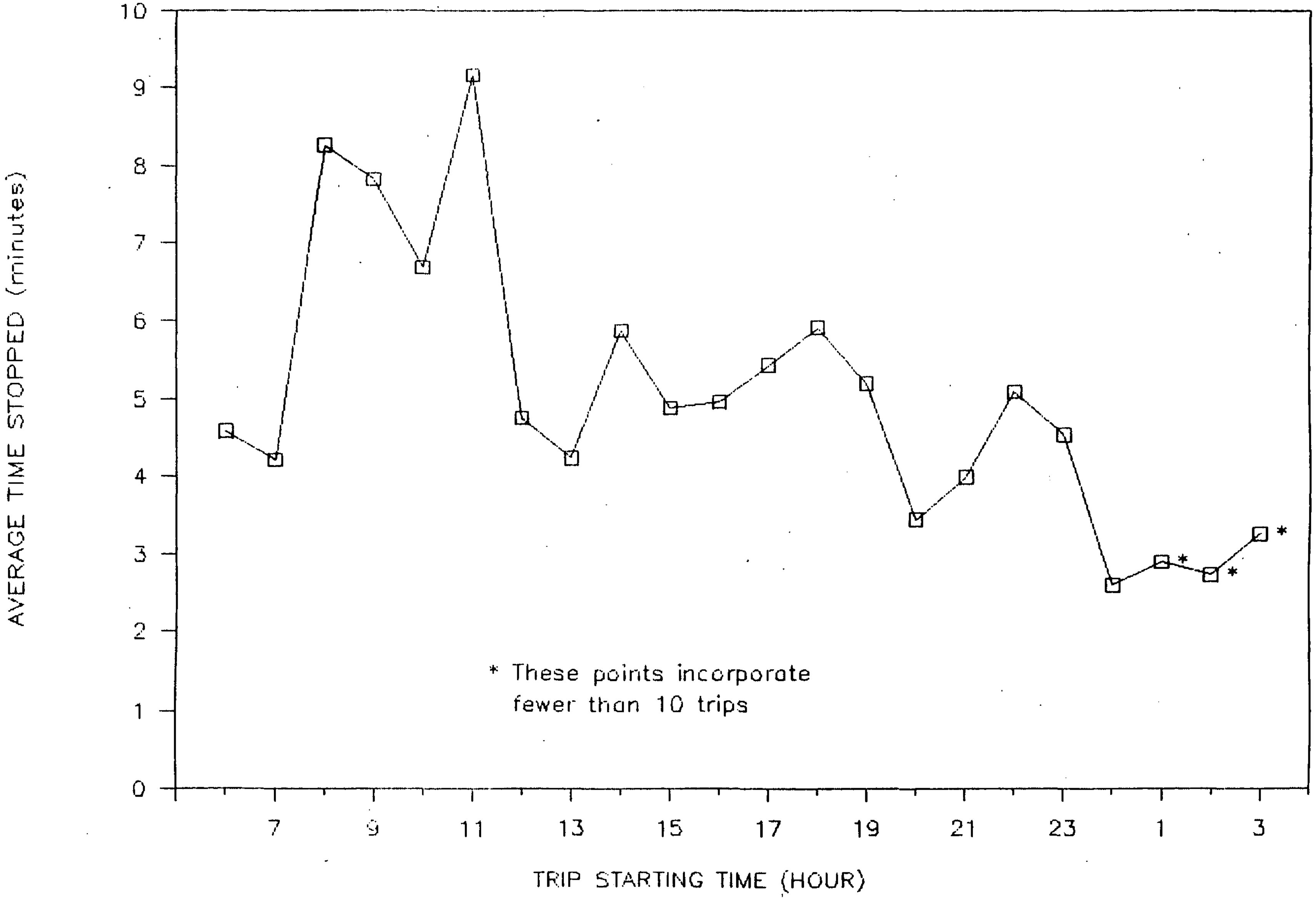
FRACTION OF TIME MOVING

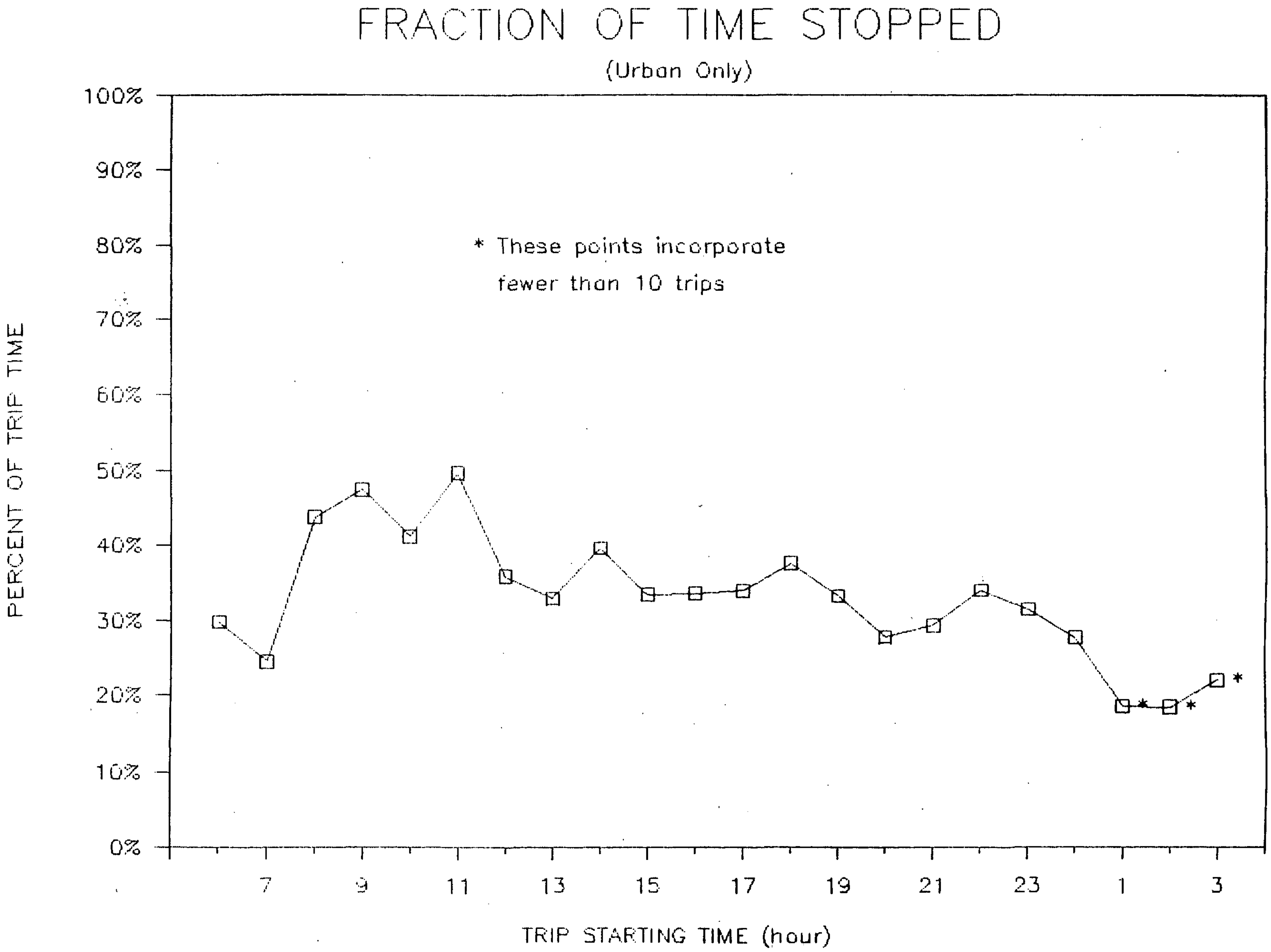
(Urban Only)



TIME STOPPED PER TRIP

(Urban Only)




FRACTION OF TIME STOPPED
(Urban Only)
* These points incorporate fewer than 10 trips
PERCENT OF TRIP TIME
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
7
9
11
13
15
17
19
21
23
1
3
TRIP STARTING TIME (hour)
*
*
*

URBAN TRIPS ONLY - AVERAGE VALUES

Minimum of 10 minutes engine off time between trips

TRIP STARTING TIME	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0
VEHICLES	15	55	28	45	47	61	60	68	60	79	90
AVE SPEED (mph)	25.0	20.1	17.9	13.6	15.9	13.9	15.6	16.6	15.9	17.6	17.2
DISTANCE (miles)	6.06	6.54	5.05	4.05	4.42	4.10	3.68	3.80	3.91	4.78	4.70
# OF STOPS	6.7	8.6	10.3	8.5	8.9	10.0	8.2	8.7	8.8	8.3	8.8
AVE DISTANCE btwn STOPS (miles)	1.4	0.91	0.67	0.46	0.61	0.42	0.49	0.53	0.58	0.61	0.58
ENGINE OFF (hours)	9.4	12.5	8.4	10.7	4.5	6.2	4.3	4.9	3.4	2.6	4.2
TRIP TIME (min)	15.4	17.2	18.9	16.5	16.2	18.5	13.2	12.8	14.8	14.6	14.7
STOP TIME (min)	4.6	4.2	8.2	7.8	6.7	9.2	4.8	4.2	5.9	4.9	5.0
MOVING (min)	10.8	13.0	10.6	8.6	9.5	9.3	8.5	8.6	8.9	9.7	9.8

TRIP STARTING TIME	17.0	18.0	19.0	20.0	21.0	22.0	23.0	24.0	1.0	2.0	3.0
VEHICLES	70	72	58	66	52	29	24	13	5	2	1
AVE SPEED (mph)	18.1	16.1	17.7	17.6	19.1	17.2	21.9	20.1	28.6	23.5	27.3
DISTANCE (miles)	5.07	4.54	5.80	4.00	4.88	4.52	5.43	3.26	7.61	7.28	6.73
# OF STOPS	9.7	8.8	8.1	7.5	7.4	8.4	6.8	5.9	5.6	6.0	6.0
AVE DISTANCE btwn STOPS (miles)	0.66	0.57	0.75	0.67	0.86	0.66	1.4	0.72	1.9	1.2	1.1
ENGINE OFF (hours)	3.1	2.8	3.5	2.2	2.4	2.3	2.8	3.4	3.8	5.0	2.9
TRIP TIME (min)	16.0	15.7	15.6	12.4	13.6	14.9	14.4	9.4	15.6	14.7	14.8
STOP TIME (min)	5.4	5.9	5.2	3.4	4.0	5.1	4.5	2.6	2.9	2.7	3.3
MOVING (min)	10.5	9.8	10.4	8.9	9.6	9.8	9.8	6.8	12.7	12.0	11.5

Appendix F

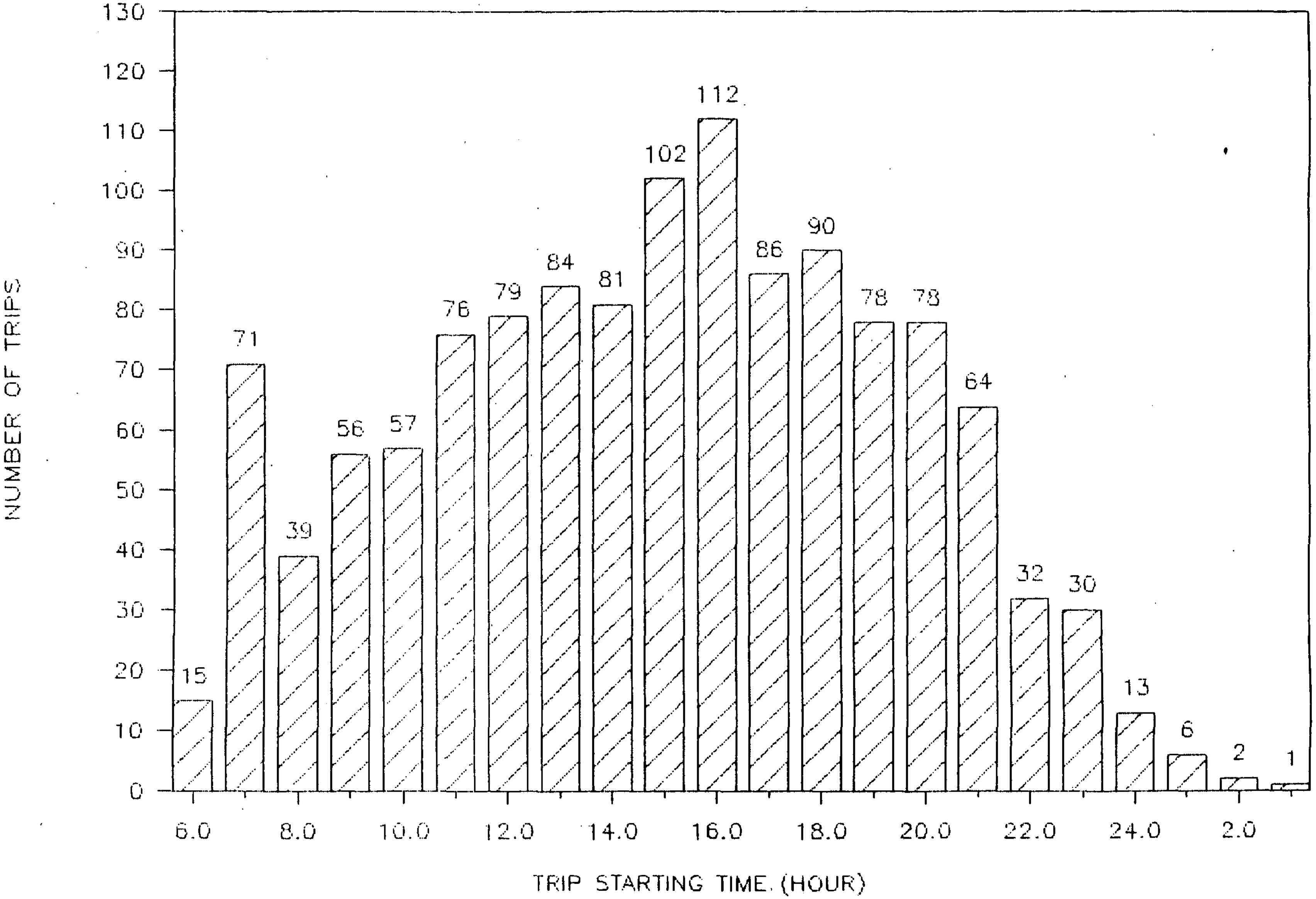
Graphs

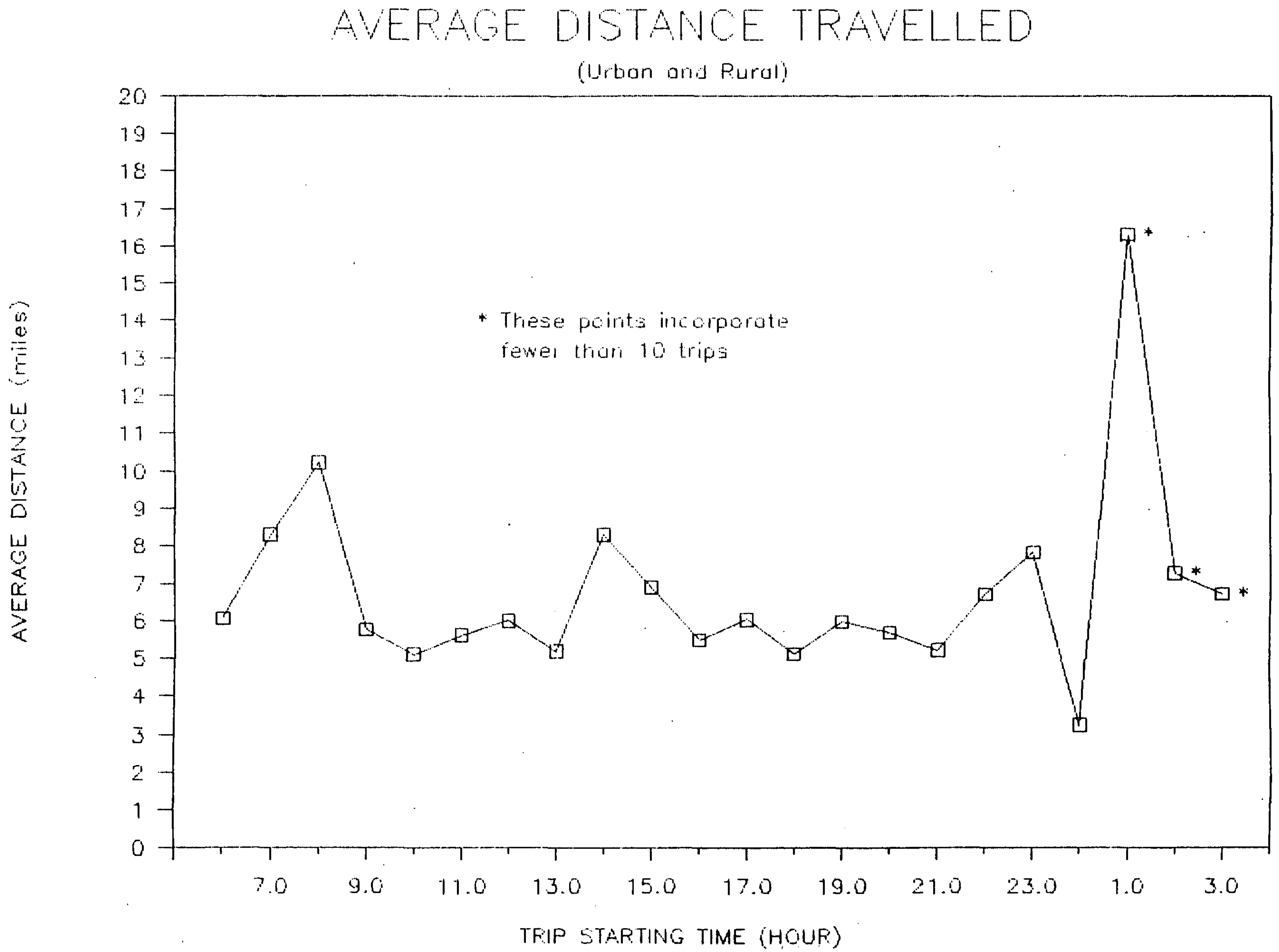
Trip Parameter vs. Trip Starting Time
Urban And/Or Rural Trips

Trips Separated By An Engine Off Period Exceeding 10 Minutes

NUMBER OF TRIPS vs TIME OF DAY

(Urban and Rural)

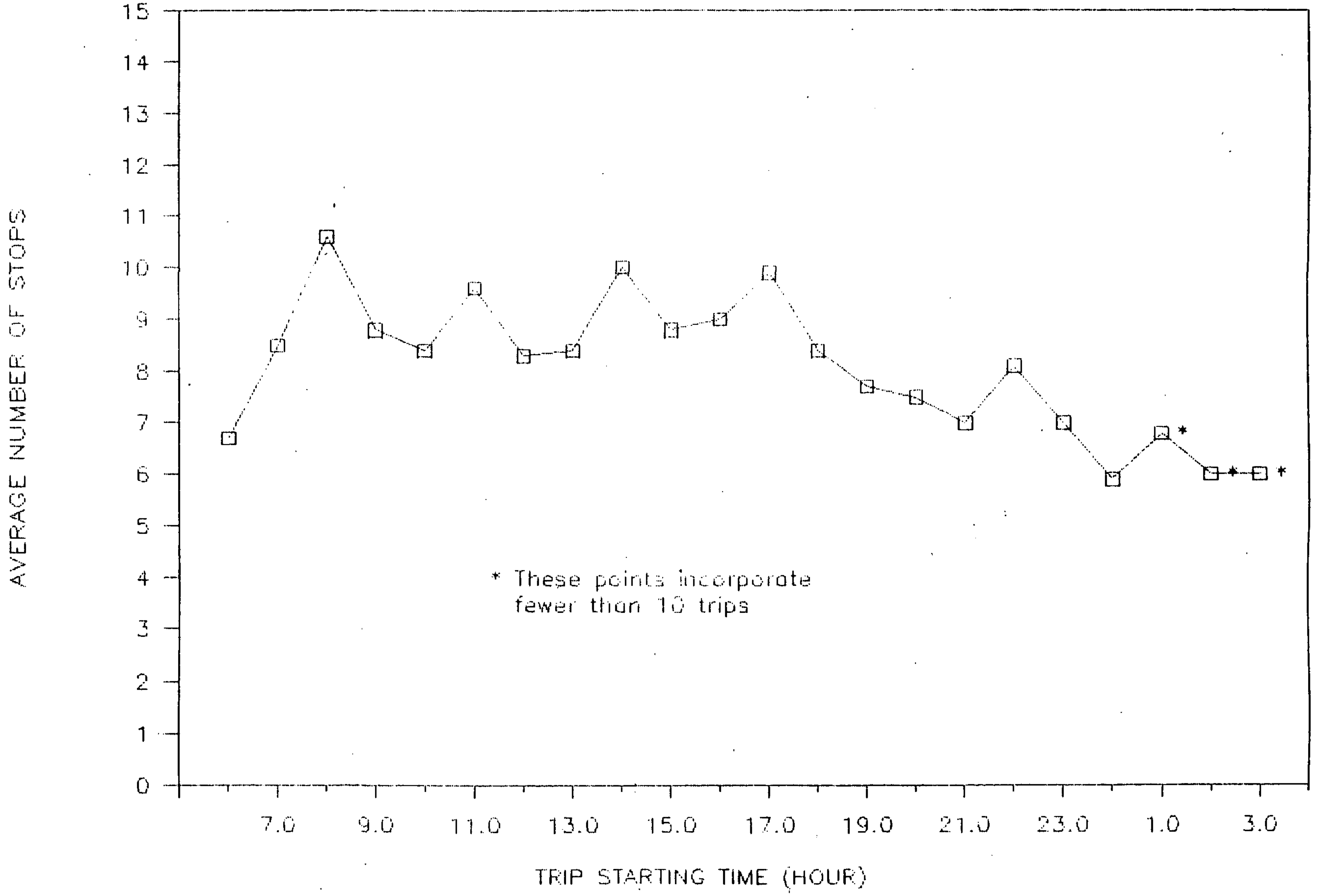



AVERAGE DISTANCE TRAVELLED
(Urban and Rural)
AVERAGE DISTANCE (miles)
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
* These points incorporate fewer than 10 trips
*
*
*
7.0
9.0
11.0
13.0
15.0
17.0
19.0
21.0
23.0
1.0
3.0
TRIP STARTING TIME (HOUR)

F 3

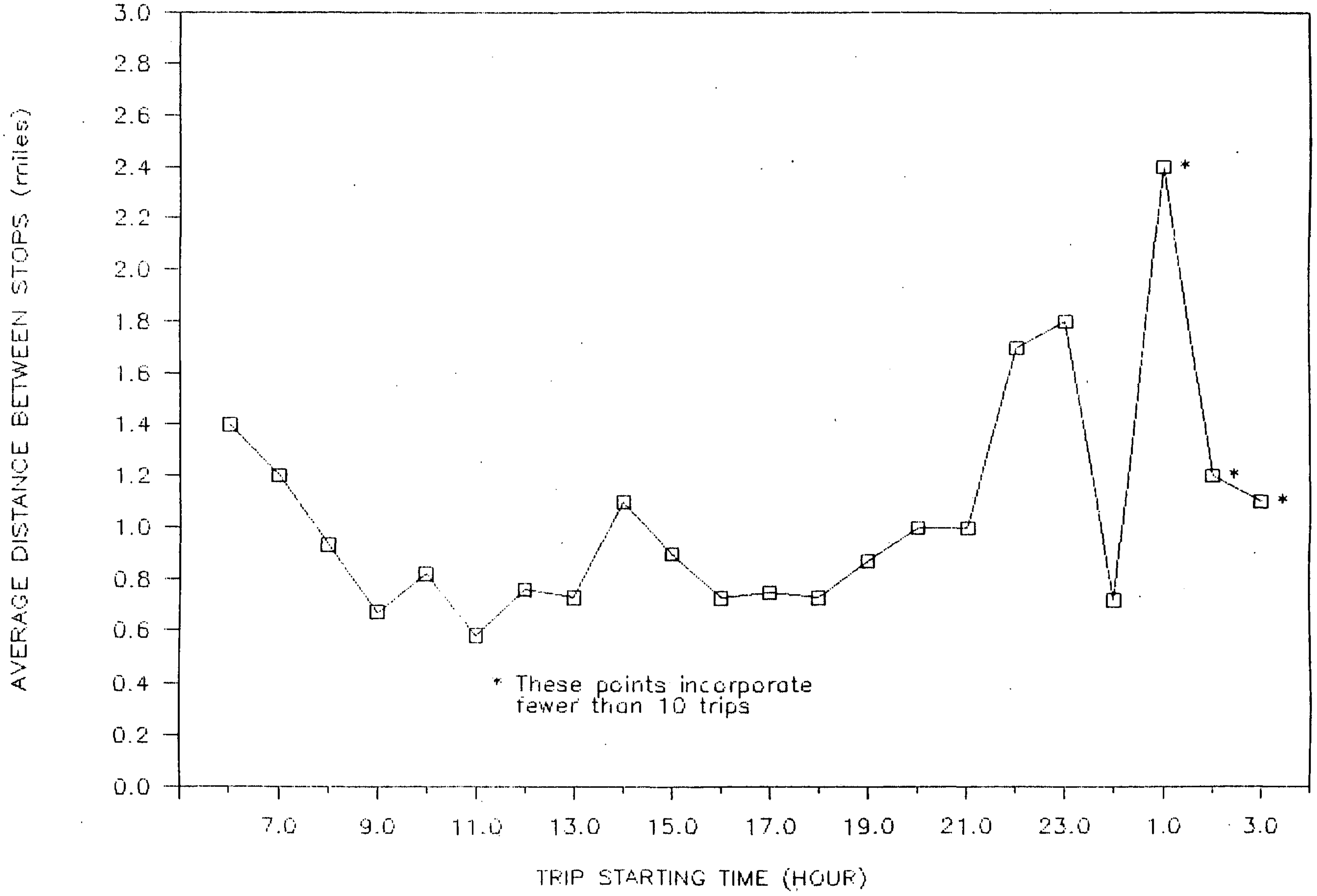
NUMBER OF STOPS PER TRIP

(Urban and Rural)



* These points incorporate fewer than 10 trips

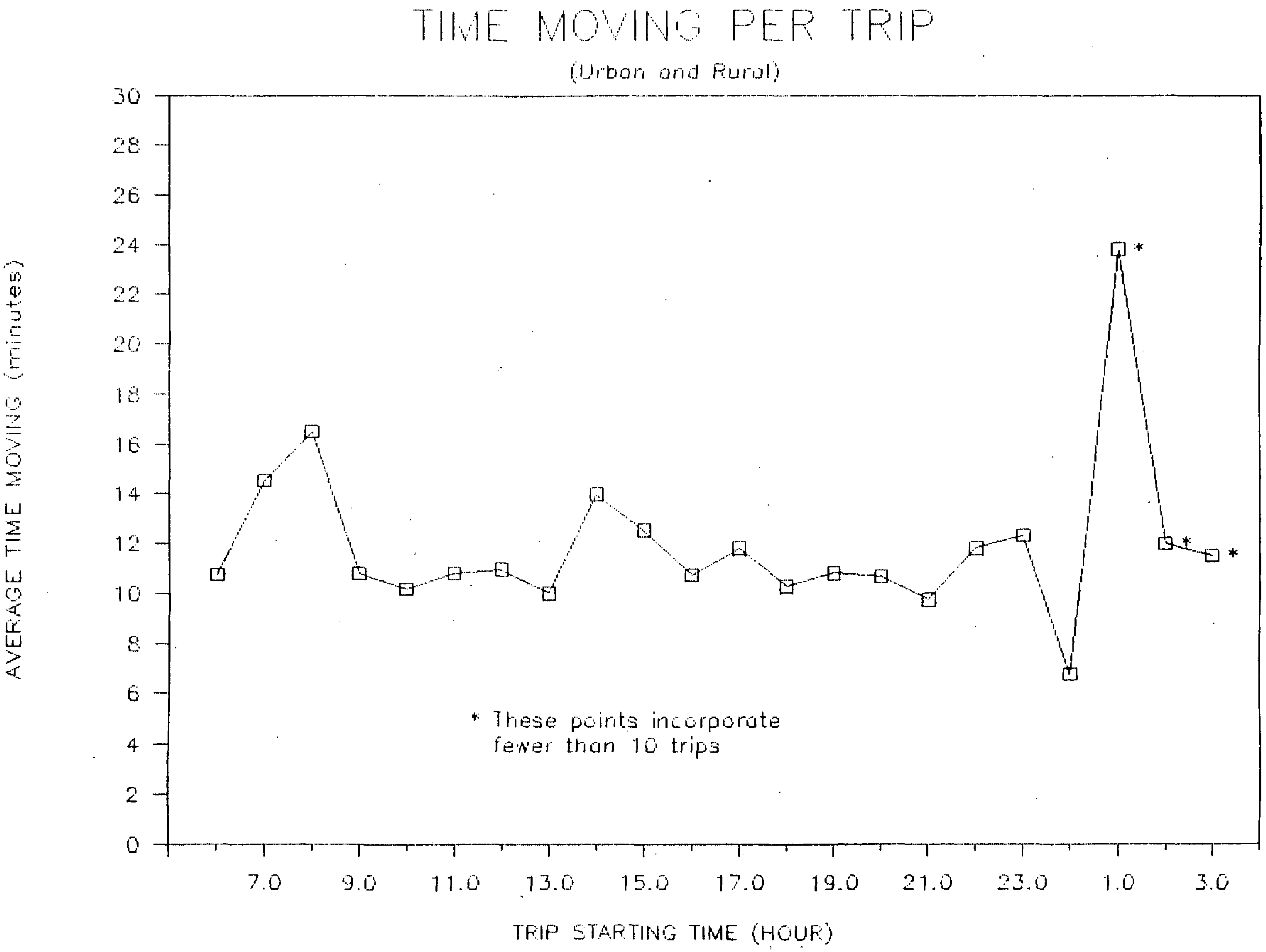
F 4



DISTANCE BETWEEN STOPS

(Urban and Rural)

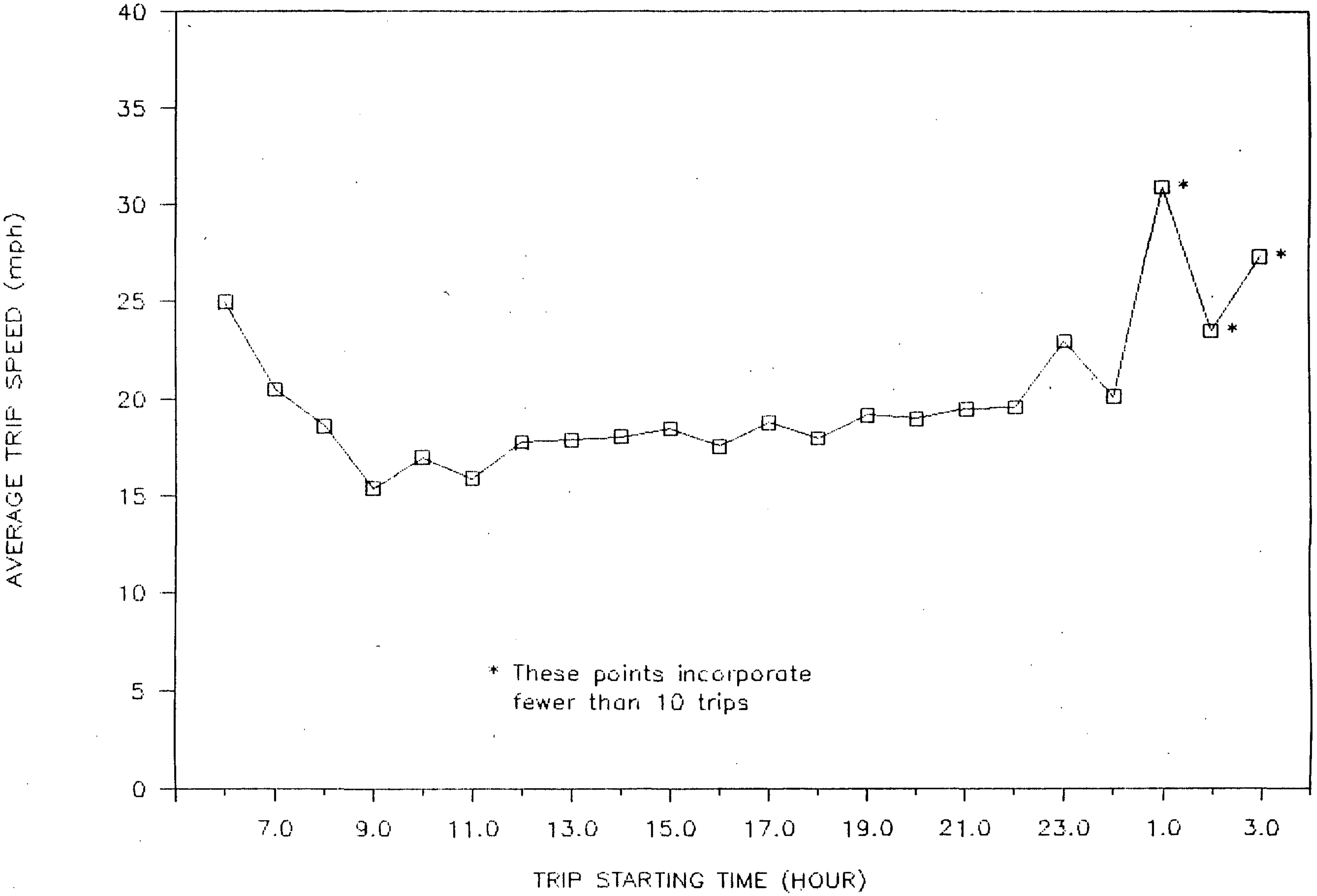
F 5


TIME MOVING PER TRIP
(Urban and Rural)
AVERAGE TIME MOVING (minutes)
TRIP STARTING TIME (HOUR)
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
7.0
9.0
11.0
13.0
15.0
17.0
19.0
21.0
23.0
1.0
3.0
*
* These points incorporate fewer than 10 trips

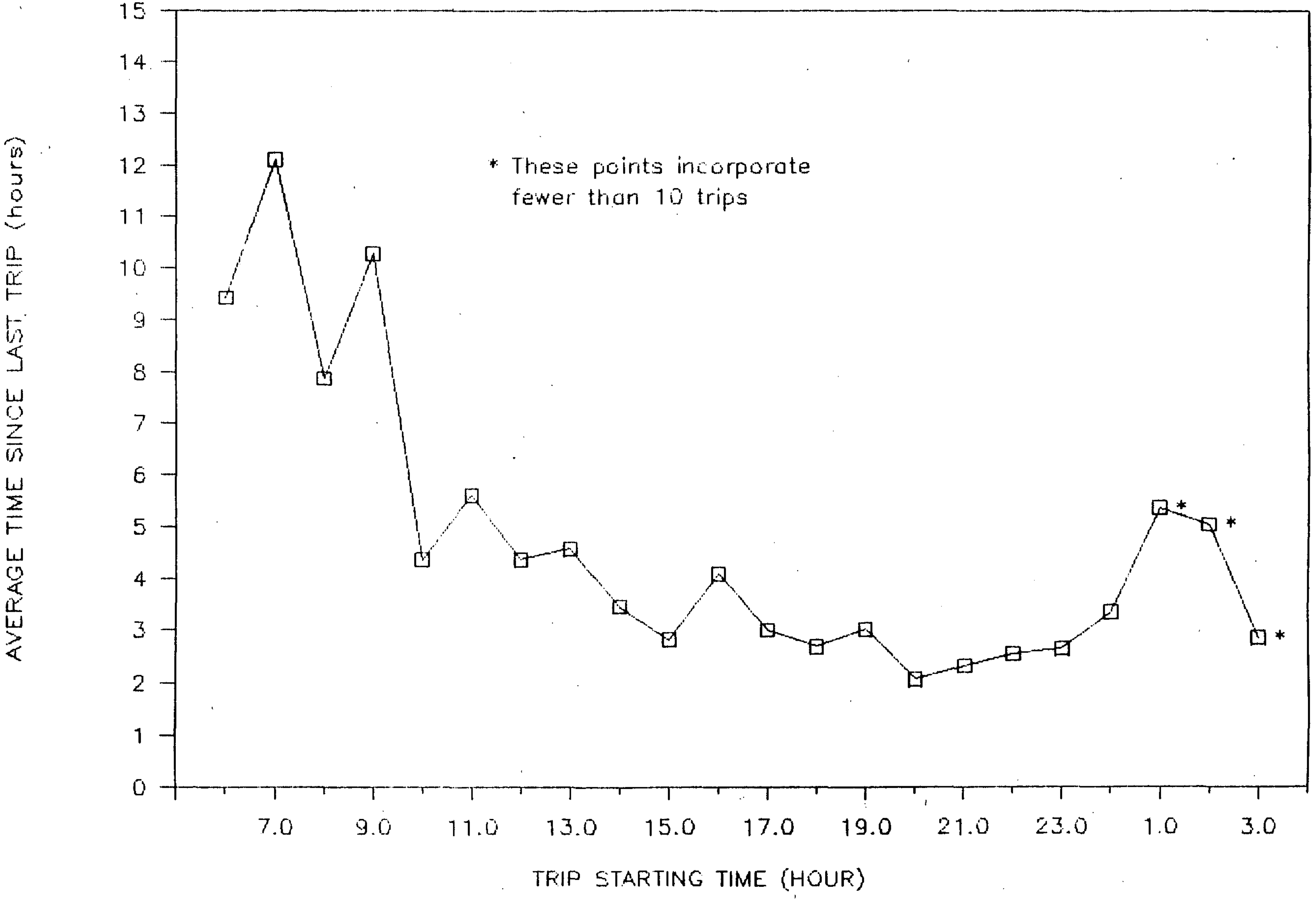
F 6

AVERAGE TRIP SPEED

(Urban and Rural)



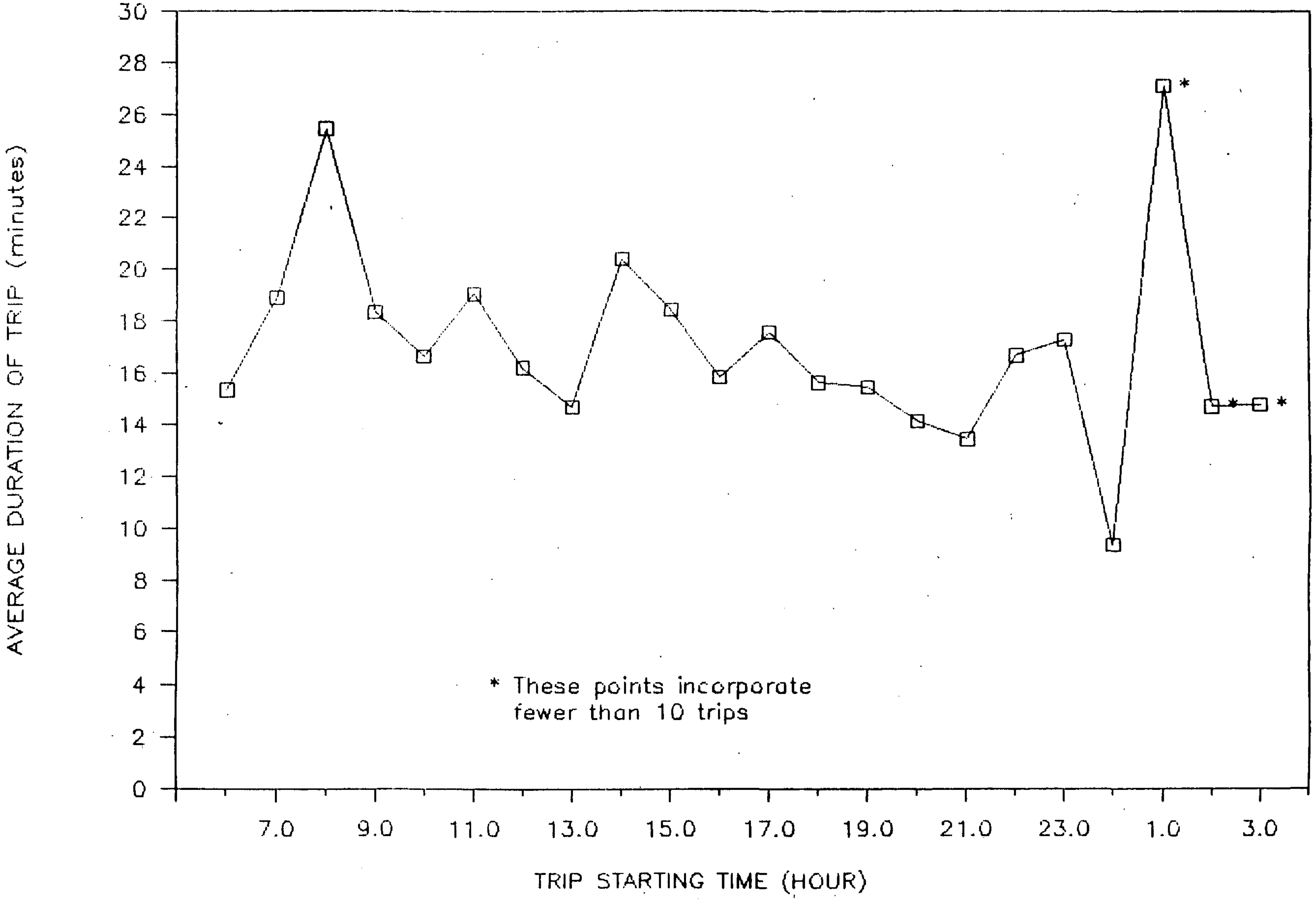
* These points incorporate fewer than 10 trips



TIME SINCE LAST TRIP

(Urban and Rural)

F 8

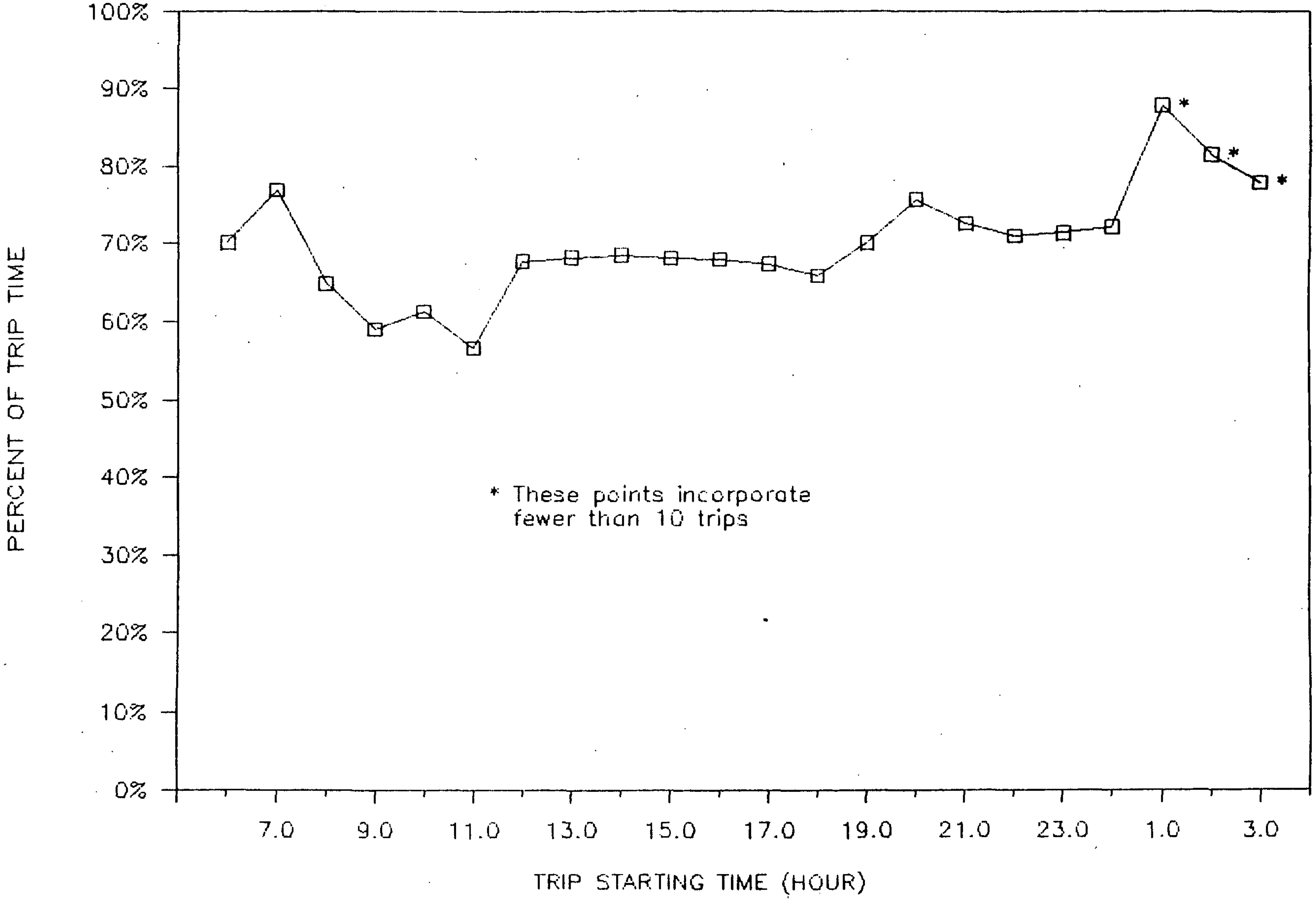


TRIP DURATION

(Urban and Rural)

* These points incorporate
fewer than 10 trips

F 9



FRACTION OF TRIP TIME MOVING

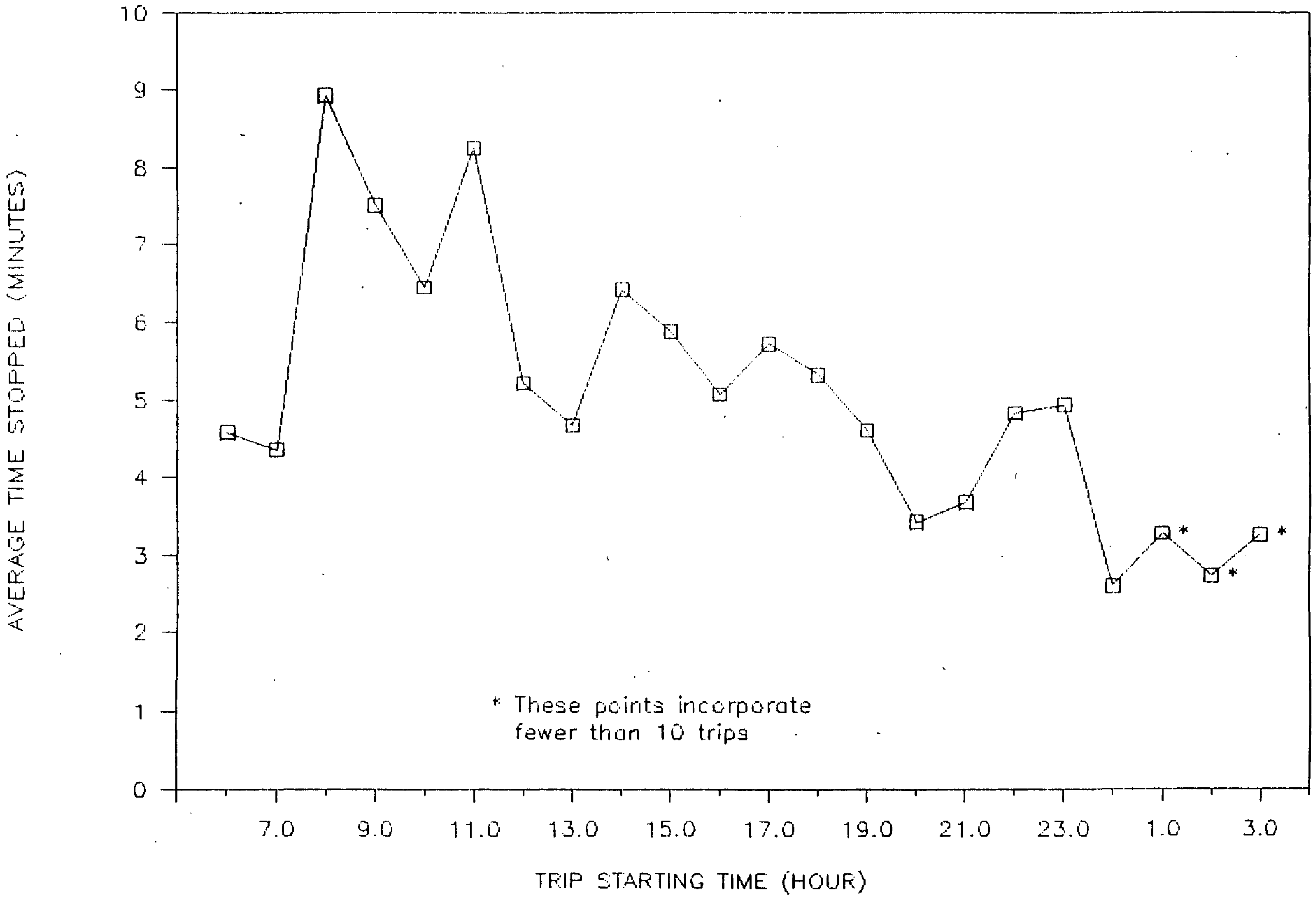
(Urban and Rural)

* These points incorporate fewer than 10 trips

F 10

TIME STOPPED PER TRIP

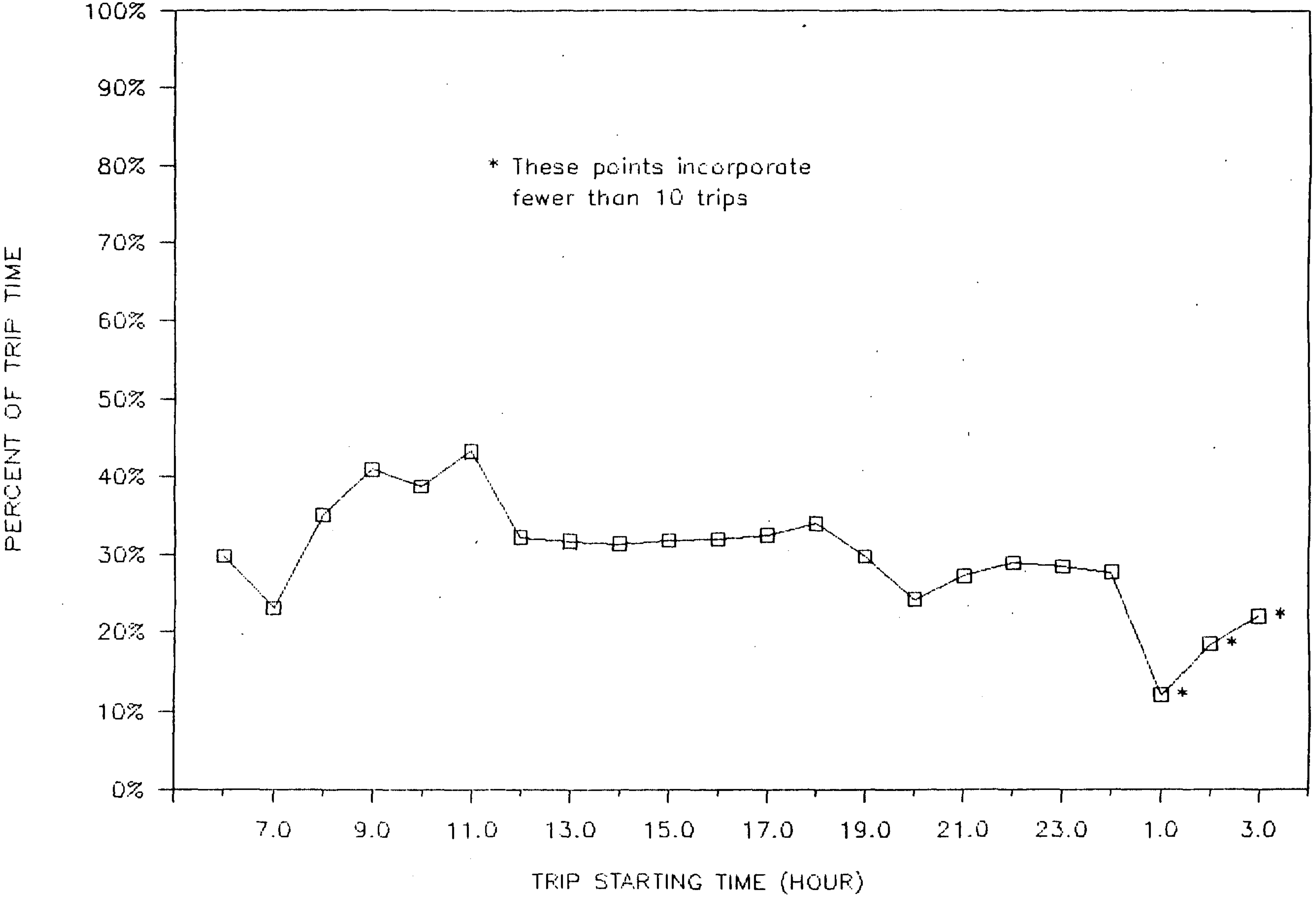
(Urban and Rural)



F 11

FRACTION OF TRIP TIME STOPPED

(Urban and Rural)



F 12

URBAN AND RURAL TRIPS - AVERAGE VALUES
Minimum of 10 minutes engine off time between trips

TRIP START TIME	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0
VEHICLES	15	71	39	56	57	76	79	84	81	102	112
AVE SPEED (mph)	25.0	20.5	18.6	15.4	17.0	15.9	17.8	17.9	18.1	18.5	17.6
DISTANCE (miles)	6.06	8.30	10.24	5.77	5.11	5.62	6.01	5.20	8.31	6.91	5.52
# OF STOPS	6.7	8.5	10.6	8.8	8.4	9.6	8.3	8.4	10.0	8.8	9.0
AVE DISTANCE btwn STOPS (miles)	1.4	1.2	0.93	0.67	0.82	0.58	0.76	0.73	1.1	0.90	0.73
ENGINE OFF (hours)	9.4	12.1	7.9	10.3	4.4	5.6	4.4	4.6	3.5	2.8	4.1
TRIP TIME (min)	15.4	18.9	25.4	18.3	16.6	19.1	16.2	14.7	20.4	18.4	15.8
STOP TIME (min)	4.6	4.4	8.9	7.5	6.4	8.2	5.2	4.7	6.4	5.9	5.1
MOVING TIME (min)	10.8	14.5	16.5	10.8	10.2	10.8	11.0	10.0	14.0	12.6	10.8

TRIP START TIME	17.0	18.0	19.0	20.0	21.0	22.0	23.0	24.0	1.0	2.0	3.0
VEHICLES	86	90	78	78	64	32	30	13	6	2	1
AVE SPEED (mph)	18.8	18.0	19.2	19.0	19.5	19.6	23.0	20.1	30.9	23.5	27.3
DISTANCE (miles)	6.05	5.13	5.99	5.71	5.24	6.73	7.84	3.26	16.31	7.28	6.73
# OF STOPS	9.9	8.4	7.7	7.5	7.0	8.1	7.0	5.9	6.8	6.0	6.0
AVE DISTANCE btwn STOPS (miles)	0.75	0.73	0.87	1.0	1.0	1.7	1.8	0.72	2.4	1.2	1.1
ENGINE OFF (hours)	3.0	2.7	3.0	2.1	2.3	2.6	2.6	3.4	5.4	5.1	2.9
TRIP TIME (min)	17.6	15.6	15.4	14.1	13.5	16.7	17.3	9.4	27.1	14.7	14.8
STOP TIME (min)	5.7	5.3	4.6	3.4	3.7	4.8	4.9	2.6	3.3	2.7	3.3
MOVING TIME (min)	11.8	10.3	10.8	10.7	9.8	11.8	12.4	6.8	23.8	12.0	11.5

Appendix G

Participant Survey

- Sample Questionnaire
- Summary of Answers

DATE: 7/6/83 ID #: 324

DAY: 87 HOUR: 13 MIN: 33

Attachment C

OCS Driver/Vehicle Questionnaire

At Time of OCS Installation/Initialization of Data Tape

OCS Test Car: Make CHEVY Model CHEVETTE Year [illegible]

V.I.N. 1G1AB68[illegible]

OCS Instrument Package Number: 1 2 [✓ 3] (circle one)

Date: 7-6-83 Time: 14:15 AM/[✓ PM]

Fuel tank level (to nearest 1/8): 7/8 Amount of fuel required to fill: 1.9

Odometer Reading: 6856.2

Tape initialized by: LARRY BELL

Installed by: [illegible]

Speed sensor calibrated and quality control procedure run by: LARRY BELL, on 7-6-83

Quality control procedure was (check one):

X the dynamometer procedure.

___ an EPA approved alternate.

At the Time of Participant Intreview

Questions for the participant. Should be filled in before he drives the test car.

1. Vehicle model and year (participant's car): 79 CHEVETTE

   Manufacturer's Vehicle Identification Number (VIN): ________

2. Date purchased 4 (Month) 79 (Year)

ID # 324

'79 Chevette

3. How many miles/year is your car driven?

A.___ 0 - 5,000
B.___ [✓ 5,000 - 10,000]
C.___ 10,000 - 15,000
D.___ 15,000 - 20,000
E.___ 20,000 - 30,000
F.___ Over 30,000

4. Where is this driving done?

A. City expressways:

1.___ 75% (almost all)
2.___ [✓ 51 - 75% (most)]
3.___ 21 - 50% (some)
4.___ 0 - 20% (little or none)

B. Major city streets:

1.___ 75% (almost all)
2.___ [✓ 51 - 75% (most)]
3.___ 21 - 50% (some)
4.___ 0 - 20% (little or none)

C. City side streets:

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4.___ [✓ 0 - 20% (little or none)]

D. Rural expressways:

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4.___ [✓ 0 - 20% (little or none)]

E. Other rural roads:

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4.___ [✓ 0 - 20% (little or none)]

5. How is this driving done?

A. To and from work:

1.___ [✓ 75% (almost all)]
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4.___ 0 - 20% (little or none)

B. Shopping and errands:

1.___ 75% (almost all)
2. [✓ 51 - 75% (most)]
3.___ 21 - 50% (some)
4.___ 0 - 20% (little or none)

C. Business (not to and from work):

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4. [✓ 0 - 20% (little or none)]

D. Other (social, vacations, etc.), please state: ______________________________

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4. [✓ 0 - 20% (little or none)]

6. How did you get here today?

A. [✓ City streets]
B. [✓ Expressway]
C.___ Rural roads
D.___ Combination of above

7. Approximately how many miles did you travel to get here? 50

8. How is your car used?

A. Driver only:

1. [✓ 75% (almost all)]
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4.___ 0 - 20% (little or none)

B. Driver and one passenger:

1.___ 75% (almost all)
2. [✓ 51 - 75% (most)]
3.___ 21 - 50% (some)
4.___ 0 - 20% (little or none)

C. Driver and 2 or more passengers:

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4. [✓ 0 - 20% (little or none)]

D. Driver only with heavy cargo:

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4. [✓ 0 - 20% (little or none)]

E. Driver, passengers, and cargo:

1.___ 75% (almost all)
2.___ 51 - 75% (most)
3.___ 21 - 50% (some)
4. [✓ 0 - 20% (little or none)]

9. During a typical week how many different people drive your car?

A. [✓ Usually only one]
B.___ Usually only two
C.___ Three or more people may drive the car during a typical week.

10. On a typical day, how many trips are made with your car? (One trip is defined as starting the engine, driving some distance, and stopping the engine.) 4

11. Is your car operated regularly on unpaved roads, in competition, or hauling loads heavier than for which it was designed?

A.___ Yes
B. [✓ No]
C.___ Don't know

12. If you drive your car to and from work, approximately how many miles is that trip, one way? 7 (Enter N/A if not applicable. If you enter N/A here also enter N/A on the next four questions.)

13. Would you, in your judgement, consider that

A. [✓ an urban trip]
B.___ a rural trip
C.___ a combination of both urban and rural driving
D.___ N/A

14. At approximately what time to do you leave for work?
2:45 AM/[✓ PM]

15. What would you say is your average speed during this trip?
55 mph

16. Approximately how long does this trip take you, on the average? ____ hours 12 minutes

17. Do you keep record of the fuel economy of your car?

A.___ Yes
B. [✓ No]

We are interested in the fuel economy people actually get with their cars. If the answer above was yes, answer questions 18 - 20, otherwise enter N/A.

18. How many miles per gallon do you get with your car?

A._____ mpg in the city
B._____ mpg on the highway
C._____ mpg combined city and highway
D. [✓ N/A]

19. Are you concerned with the fuel economy of your car?

A.___ Yes
B. [✓ No]

20. What type of tires do you use on your car?

A.___ Bias-ply
B.___ Bias-belt
C. [✓ Radial]
   1. [✓ Steel-belted]
   2.___ Fiberglass-belted
   3.___ Other

(If participant does not know this, contractor should look at the tires and find out.)

The test car was turned over to participant for the beginning of the test period on:

7-6-83 Date

16:20 Time AM/[✓ PM]

At Time of OCS Data Tape Removal

Date: 7-12-83 Time: ____________ AM/[✓ PM]

Fuel tank level (to nearest 1/8): ________ Amount of fuel required to fill: ________

Instruments rmvd by: ________
Tape removed by: ________

Summary From Participant Questionnaire*

1. How many miles/year is your car driven?

Ranges	Responses
0- 5,000	2
5,000-10,000	7
10,000-15,000	23
15,000-20,000	9
20,000-30,000	3
Over 30,000	2
Total Response =	46

2. Where is this driving done?**

Fraction of Time	City Express-ways	Major City Streets	City Side Streets	Rural Express-ways	Other Rural Roads
76-100% (almost all)	7	3	2	1	0
51-75% (most)	9	17	6	5	1
21-50% (some)	19	17	12	18	7
0-20% (little)	11	9	25	22	38
Total Responses	46	46	45	46	46

3. How is this driving done?**

Fraction of time	To and from Work	Shopping & Errands	Business	Other (Vacations, etc.)
76-100% (almost all)	11	4	1	0
51-75% (most)	12	8	2	3
21-50% (some)	12	22	14	15
0-20% (little)	11	12	29	27
Total Responses	46	46	46	45

---

* Forty-six of the forty-seven participants used in this analysis responded.

** The combined fractions of time indicated by some participants exceeded 100%.

4. How is your car used?*

Fraction of Time	Driver Only	Driver & 1 Psgr.	Driver & 2+ Psgr	Driver & Heavy Cargo	Driver, Psgrs, Cargo
76-100% (almost all)	22	3	2	0	0
51-75% (most)	8	7	3	0	0
21-50% (some)	7	15	6	4	4
0-20% (little)	9	20	35	42	42
Total Responses	46	45	46	46	46

5. During a typical week, how many different people drive the car?

Options	Responses
Usually only one	31
Usually only two	14
Three or more	1
Total Response =	46

6. On a typical day, how many trips are made with your car? (One trip is defined as starting the engine, driving some distance, and stopping the engine.)

Answers varied from 1 to 15.

7. Is your car operated regularly on unpaved roads, in competition, or hauling loads heavier than for which it was designed?

Yes	1
No	45
Total Response =	46

8. If you drive your car to and from work, approximately how many miles is that trip, one way?

Answers varied from 1/4 to 30 miles.
Fifteen participants answered with "N/A".
Forty-six participants responded.

---

* The combined fractions of time indicated by some participants exceeded 100%.

9. Would you, in your judgement, consider that:

Options	Responses
An urban trip	26
A rural trip	1
A combination	4
N/A	15
Total Response =	46

10. At approximately what time do you leave for work?

Answers varied from 6 a.m. to 9 p.m. Nineteen of the participants indicated a time. Sixteen of these left for work between 6 a.m. and 9 a.m., as follows:

6:00 - 6:59	5
7:00 - 7:59	8
8:00 - 9:00	3

11. What would you say is your average speed during this trip?

Answers varied from 20 to 65 mph.

12. Approximately how long does this take you?

Answers varied from 2 minutes to 1-1/2 hours.

13. Do you keep a record of the fuel economy of your car?

Yes	13
No	33
Total Responses =	46

14. How many miles per gallon do you actually get with your car?

City	13-27 mpg
Highway	16-35 mpg
Combined	14-39* mpg

---

* The participant responding with 39 mpg did not answer the city and highway parts of the question.

15. Are you concerned with the fuel economy of your car?

Yes	28
No	16
Total Responses =	44

16. What type of tires do you use on your car?

Bias-ply	1
Bias-belt	0
Radial	39
Total Responses =	40

If radial, what type?

Steel-belted	32
Fiberglass-belted	2
Other	1
Total Responses =	35

Appendix H

Temperature Analysis

This appendix contains an expanded discussion of the temperature analysis described in the main text. Some parts were included in the Summary of Temperature Analysis (Section IX of main text), but are repeated here for continuity. The discussion is divided into four sections: typical trip temperatures versus time, average soak period, individual temperature profiles, and soak periods.

A. Typical Trip Temperatures vs. Time

The temperatures obtained for the typical trip are shown in Figure H-1. At the start of the trip all of the temperatures are clustered at 12°C. As the trip progresses, the water pump exhibits a fairly steep increase in temperature, reaching a level more than 30°C above "ambient" or pre-trip levels within fifteen minutes, and remaining fairly steady for the remainder of the trip. The oil pan temperature also rises about 30°C, but does so more gradually, taking over twenty minutes to do so before leveling off. The remaining temperatures: the fuel tank, air cleaner, and underhood, show gradual increase of 4°C, 6°C and 15°C, respectively, beginning around four minutes after the trip begins, and never appearing to level off.

B. Average Soak Period

Significant changes occur in the temperature profiles when the vehicle completes its trip. The first portion of the soak is characterized by the vehicle's response to having its cooling system shut off, and having to dissipate the engine heat in another manner. The second portion is characterized by the vehicle components cooling down to near ambient levels.

Figure H-2 shows the temperature profiles for the first forty-five minutes after engine shutoff, plotted at one minute intervals. The water pump temperature jumps 20°C to a peak level of nearly 70°C in the first ten minutes of the soak before beginning to cool off toward the ambient temperature.

The oil pan temperature rises slightly, approximately 5°C to a peak level of 50°C, before beginning to fall off after only five minutes of soak. The fuel tank temperature begins dropping as soon as engine shutoff occurs to around 12°C after ten minutes, after which it exhibits a profile that appears to track the ambient temperatures. (As noted earlier, this temperature is measured on the underside of the fuel tank, and

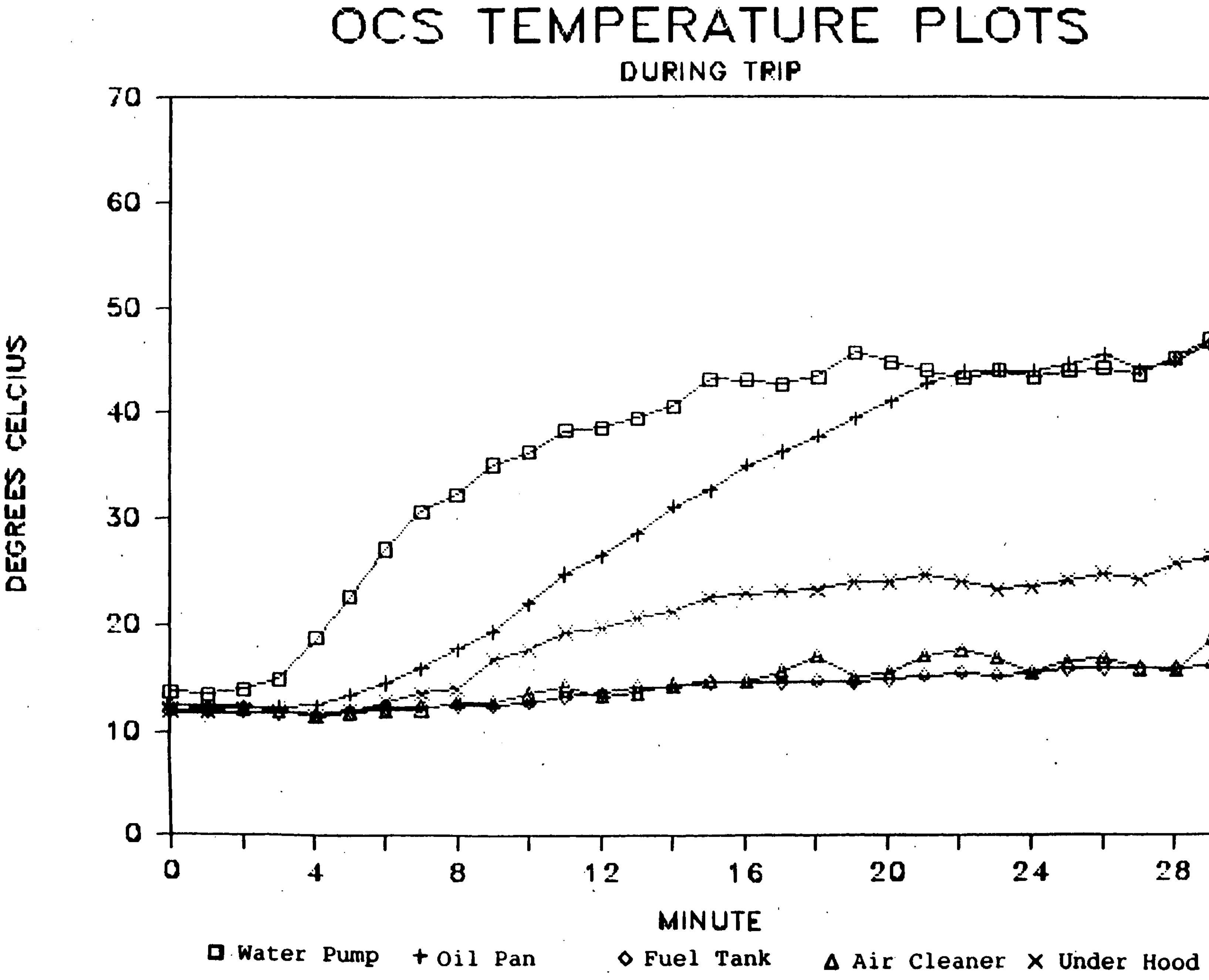

OCS TEMPERATURE PLOTS
DURING TRIP
DEGREES CELCIUS
0
10
20
30
40
50
60
70
MINUTE
0
4
8
12
16
20
24
28
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-1

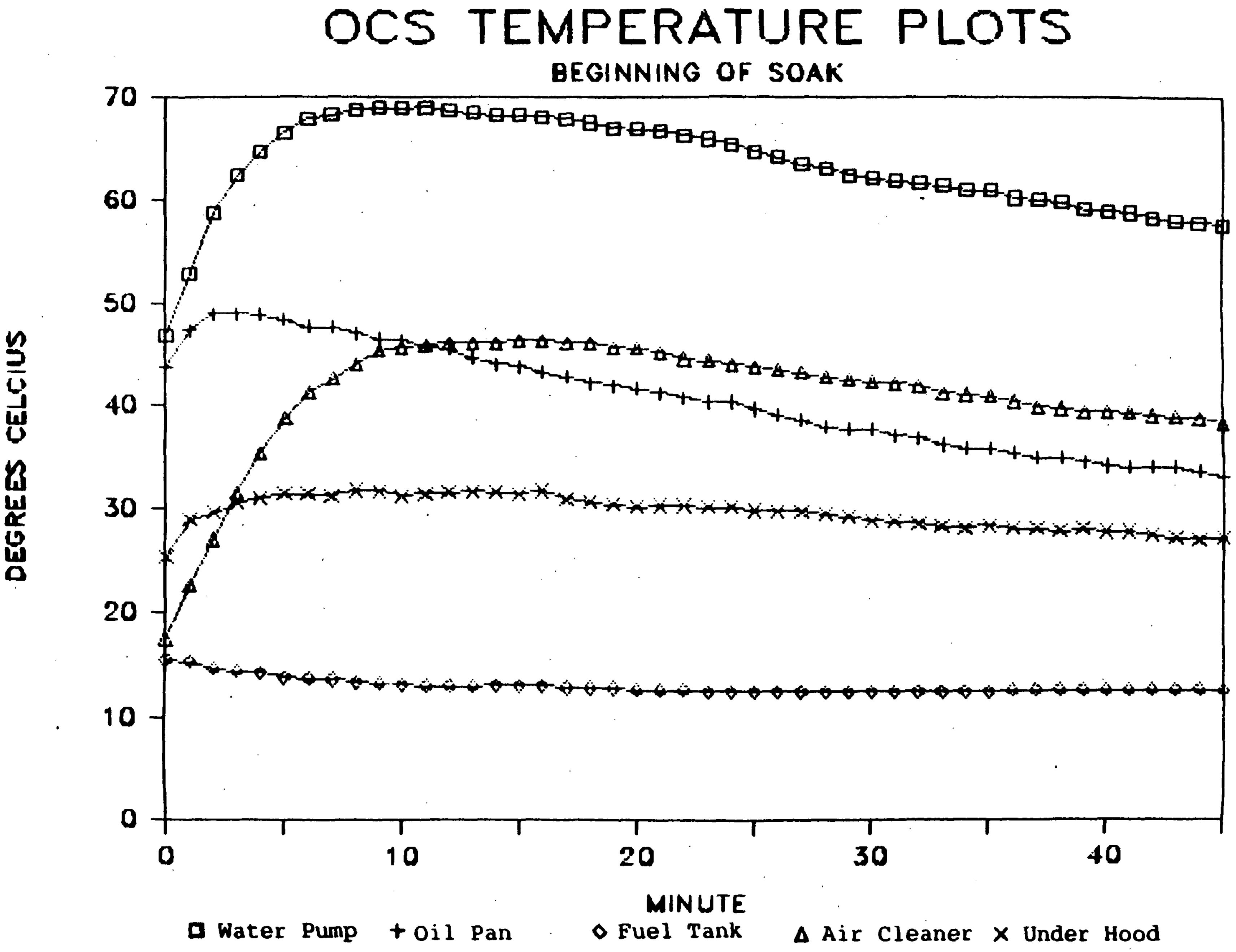

OCS TEMPERATURE PLOTS
BEGINNING OF SOAK
DEGREES CELCIUS
0
10
20
30
40
50
60
70
MINUTE
0
10
20
30
40
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-2

does not appear to represent the temperature of the fuel in the tank.) The temperature at the air cleaner rises markedly, almost 30°C, in this portion of the soak. After ten minutes, it eclipses the oil pan temperature, reaching a peak of just under 50°C after fifteen minutes before it begins to cool off. The underhood temperatures rises slightly, about 5°C, to a peak level of just over 30°C before beginning to fall off.

The complete soak, with the temperatures plotted every minute for the first forty-five minutes and every twelve minutes thereafter, is shown in Figure H-3. The water pump, oil pan, air cleaner, and underhood temperatures all exhibit decay down to a presumed ambient temperature. The fuel tank temperature rises through the soak period, apparently tracking a rise in the ambient temperature. This rise should be expected, as all but one of these ten trips occurred in the morning with the soak occurring during the midday hours. After around five and one-half to six hours soak, the measured temperatures have converged to the same level. After this point, the temperatures move along together, tracking the ambient level.

Beginning at twenty minutes after the soak begins, exponential decay curves were fit to the water pump, oil pan, air cleaner, and underhood temperatures. These were first adjusted by subtracting the fuel tank temperature in order to model decay to ambient temperature. These models fit the data extremely well, with R^2 values over .99 in each case. The resulting decay constants, when time is measured in minutes, are .007, .011, .008, and .007 (1/minutes) for the water pump, oil pan, air cleaner, and underhood temperatures, respectively. To put these values in perspective, Table H-1 shows the time required for the difference between the peak soak temperatures and ambient temperatures to be reduced to 50 percent, 20 percent, 10 percent, 5 percent, and 1 percent of their original values for the decay constants of .007 and .011 (1/minutes). These may be useful in determining when a vehicle is experiencing a "hot start" versus a "cold-start".

C. Individual Temperature Profiles

Time-temperature profiles for individual trips were also developed in order to make comparisons between trips, and to see how well the "average trip" reflects individual trips. Three trips were chosen for this portion of the analysis: a trip representing a summer day (loan #234), a trip representing a winter night (loan #468, first trip), and a trip representing a winter day (loan #468, second trip). Since the last two of these involve the same vehicle some consideration may also be given to the differences between trips for an individual vehicle.

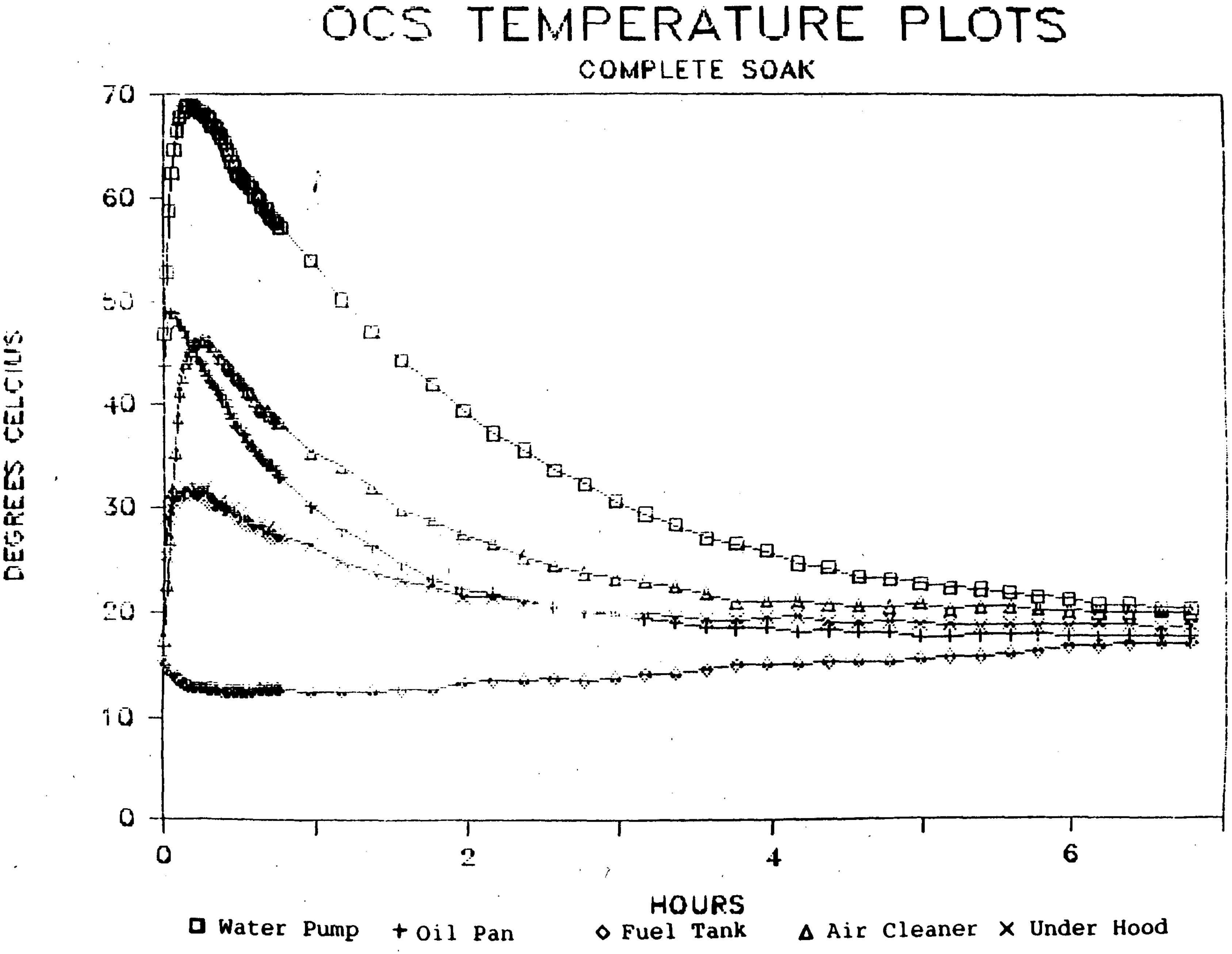

OCS TEMPERATURE PLOTS
COMPLETE SOAK
DEGREES CELCIUS
70
60
50
40
30
20
10
0
0
2
4
6
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-3

Table H-1

Hot Soak Times

Decay Constant (1/minutes)	Time Required to Reduce Temperature Difference to Indicated Percentage of Original (hrs:min)				
	50%	20%	10%	5%	1%
.007	1:39	3:50	5:29	7:08	10:58
.011	1:03	2:26	3:29	4:32	6:59

Figures H-4 through H-6 show the temperature profiles developed from these three trips. The rise in water pump temperature is significantly steeper and higher for the summer trip, although in all three trips the pattern closely reflects that of the average trip. The rise in oil pan temperature is similar for all three trips. The summer day and winter night trips, however, are not long enough for this temperature to rise up to the level of the water pump as it does in the average trip. The fuel tank and air cleaner temperature profiles are similar for all these trips. The underhood temperature profile differs somewhat from trip to trip. For the two day trips this temperature appears closely linked to the oil pan temperature. The winter day trip, which lasts longer, shows the underhood temperature leveling off after eighteen minutes, whereas the oil pan temperature continues to rise. This is not seen in the summer day trip due to its shorter length. The underhood temperature profile in the winter night trip is more reflective of the average trip, exhibiting a rise above the fuel tank and air cleaner temperatures, but well below the oil pan and water pump temperatures.

D. Soak Periods

Figures H-7 through H-9 show the time-temperature profiles for the first forty-five minutes of the soak periods. The different trips exhibit profiles quite similar to the average trip in: 1) the slight rise in the oil pan temperature, 2) the steep rise in the air cleaner temperature, and 3) the relative inactivity of the fuel tank temperature. There is a striking difference in the behavior of the water pump temperatures. The two winter trip profiles show water pump temperature rises of over 30°C to peak levels above 60°C. Meanwhile the summer trip profile shows only a slight increase of around 5°C to a peak level of 75°C. This may reflect some maximum attainable temperature. The rises associated with the underhood temperature also differ from trip to trip. This may be a carryover effect due to the length of the trips, however as the two short trips show more of a rise in the underhood temperature than do either the longer winter day trip or the average trip.

Figures H-10 through H-12 show the time-temperature profiles for the complete soak periods. The pattern of decay to ambient seen in the average trip is seen again here for all three trips. Note that for the winter night trip, the fuel tank is cooling off during the soak overnight.

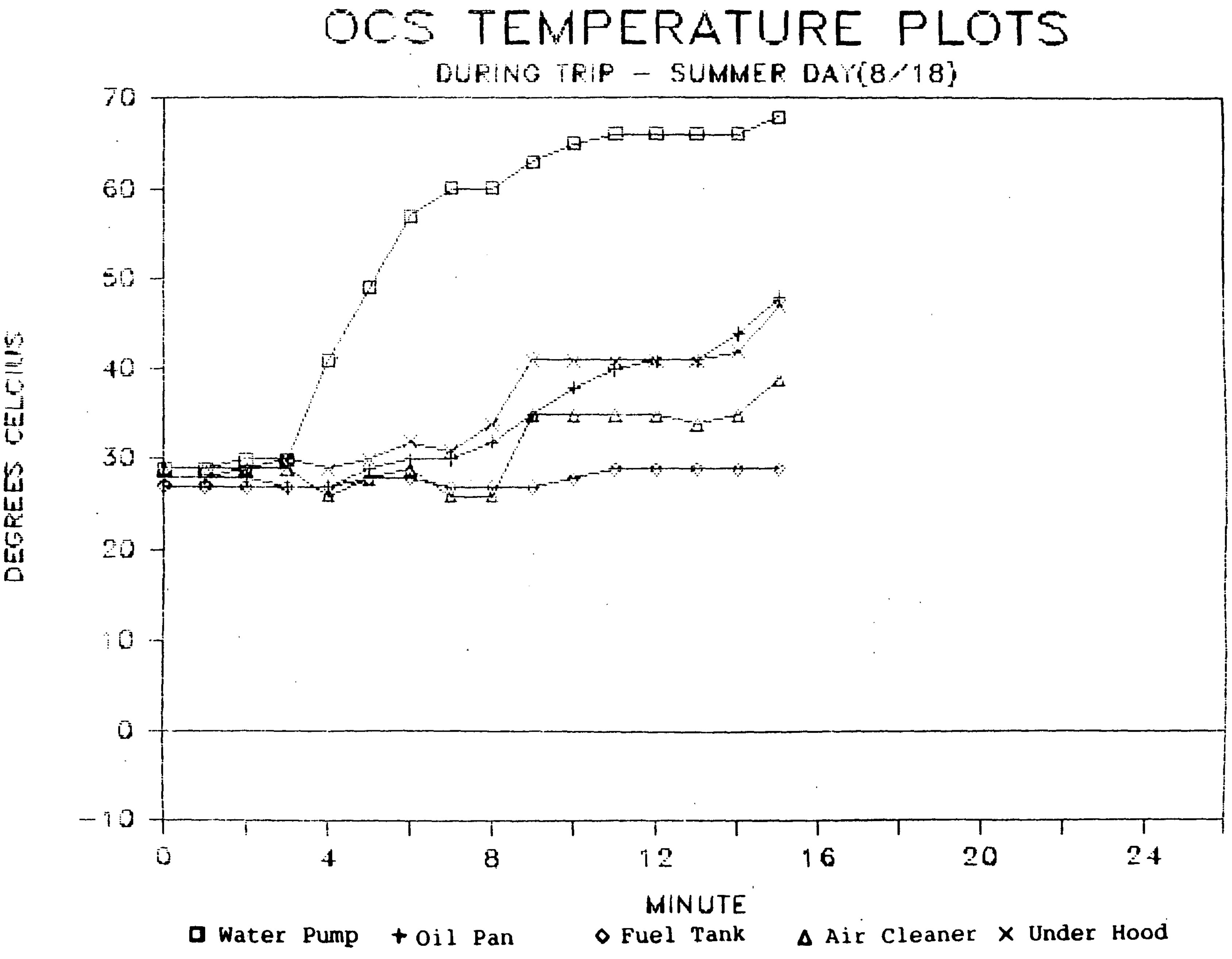

OCS TEMPERATURE PLOTS
DURING TRIP – SUMMER DAY(8/18)
DEGREES CELCIUS
70
60
50
40
30
20
10
0
-10
0
4
8
12
16
20
24
MINUTE
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-4

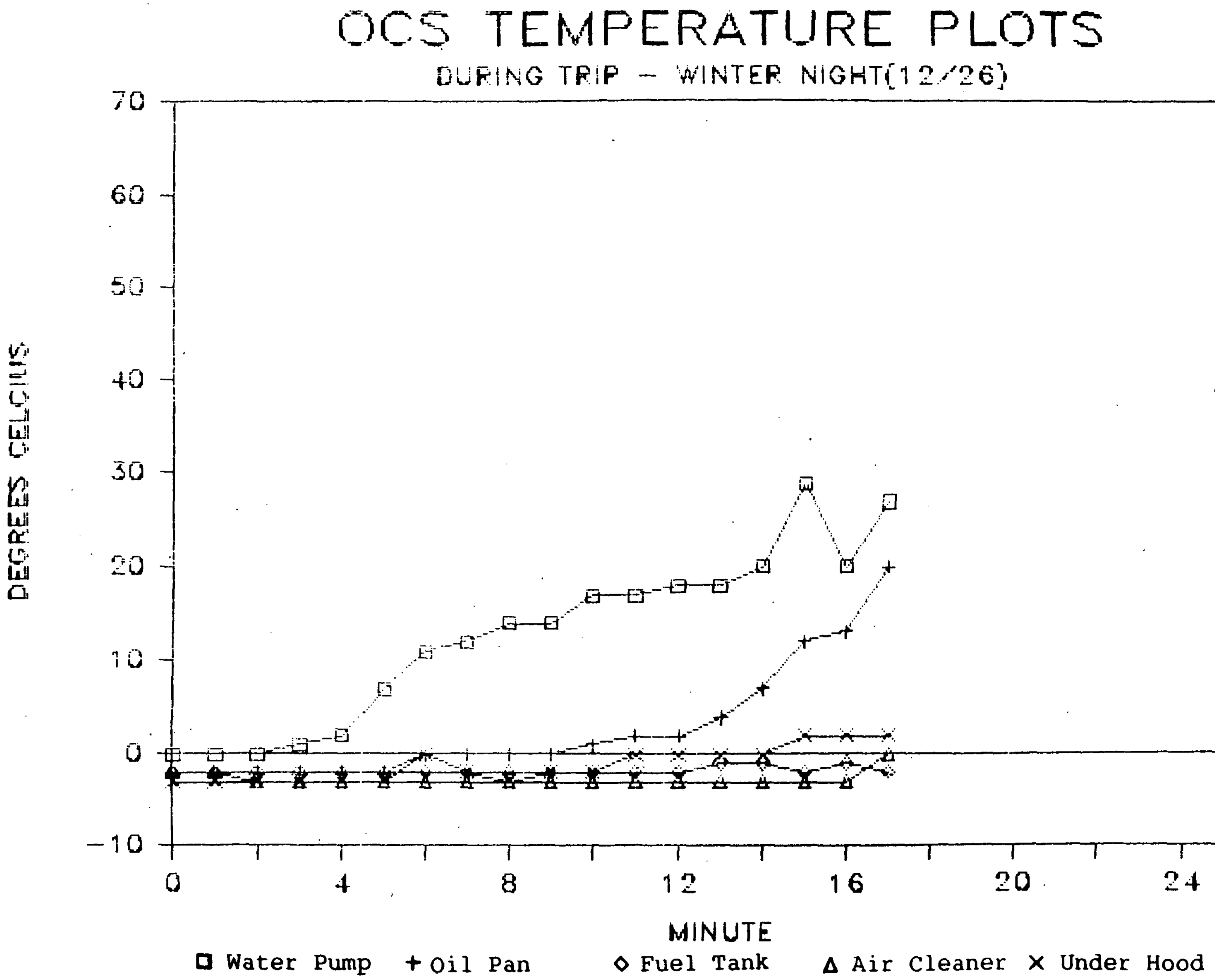

OCS TEMPERATURE PLOTS
DURING TRIP – WINTER NIGHT(12/26)
DEGREES CELCIUS
70
60
50
40
30
20
10
0
-10
0
4
8
12
16
20
24
MINUTE
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-5

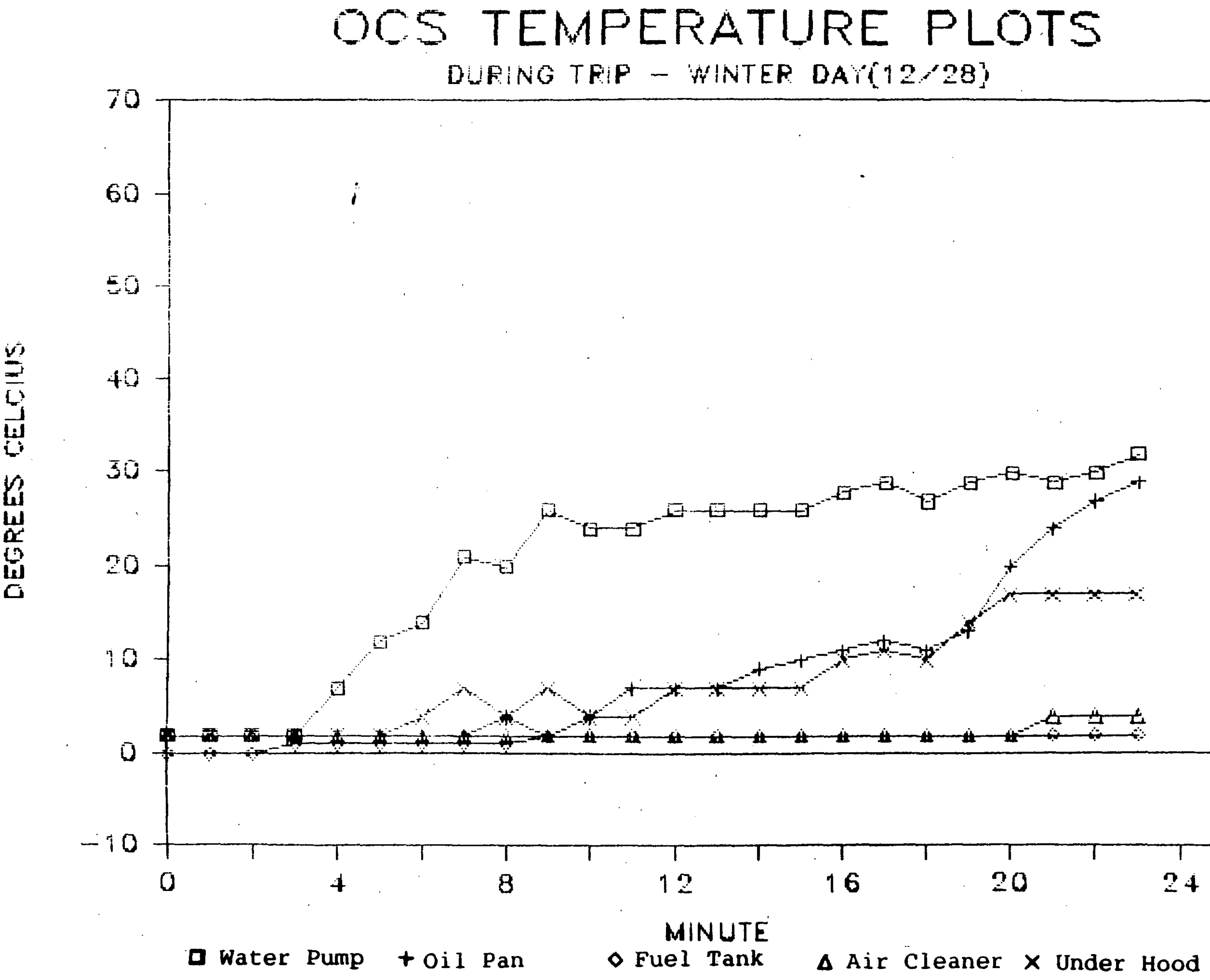

OCS TEMPERATURE PLOTS
DURING TRIP - WINTER DAY(12/28)
DEGREES CELCIUS
70
60
50
40
30
20
10
0
-10
0
4
8
12
16
20
24
MINUTE
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-6

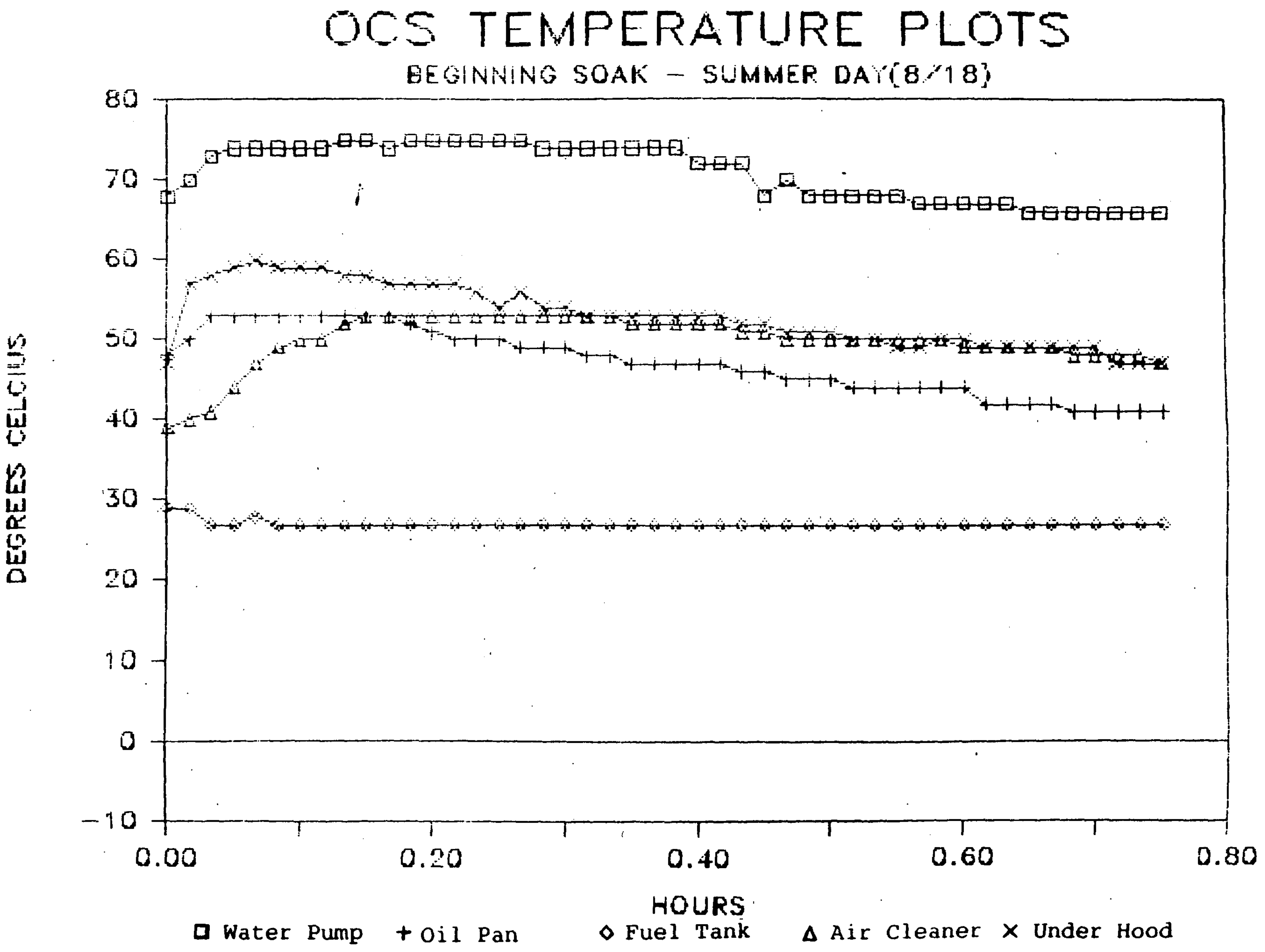

OCS TEMPERATURE PLOTS
BEGINNING SOAK - SUMMER DAY(8/18)
DEGREES CELCIUS
80
70
60
50
40
30
20
10
0
-10
0.00
0.20
0.40
0.60
0.80
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-7

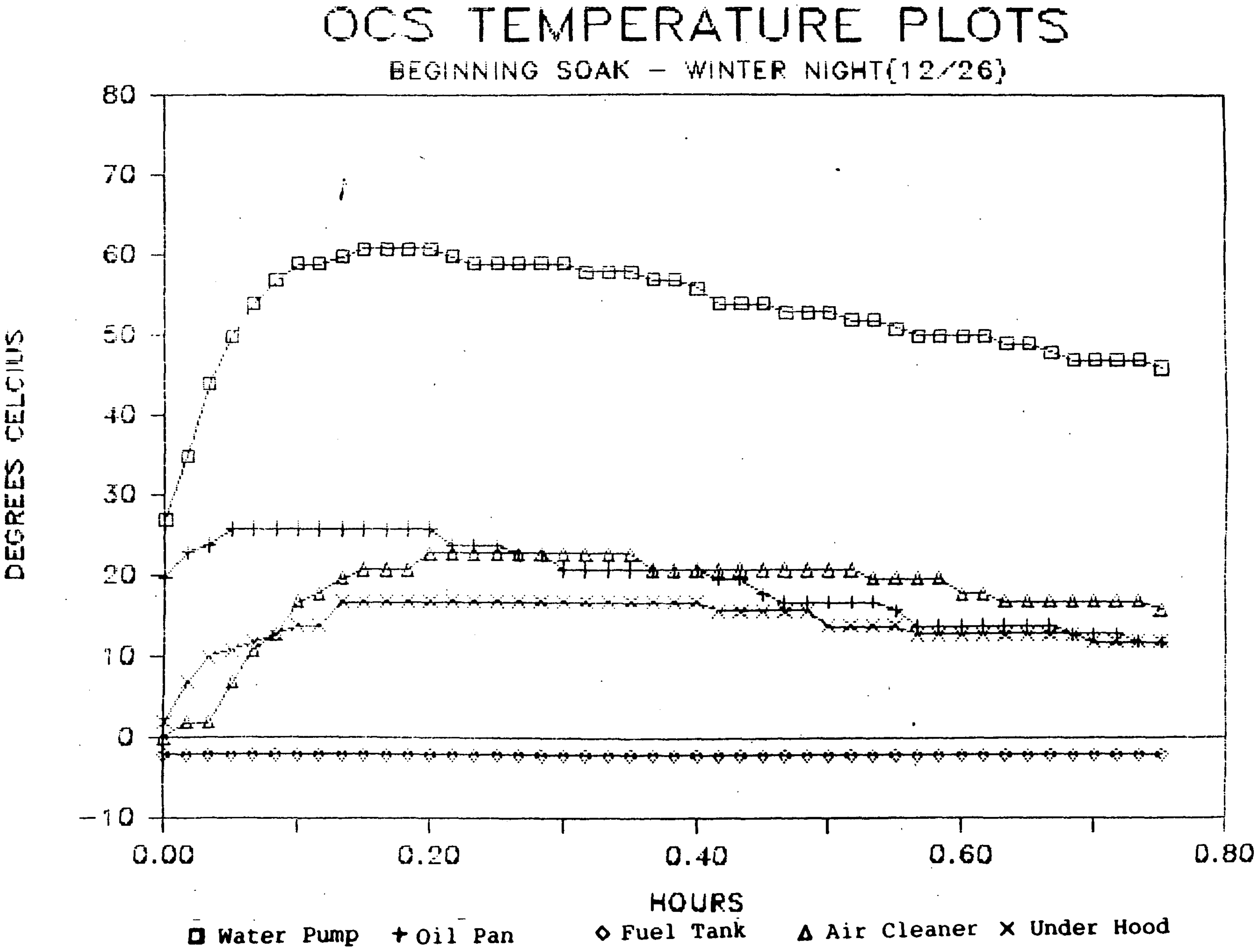

OCS TEMPERATURE PLOTS
BEGINNING SOAK - WINTER NIGHT(12/26)
DEGREES CELCIUS
80
70
60
50
40
30
20
10
0
-10
0.00
0.20
0.40
0.60
0.80
HOURS
Water Pump
Oil Pan
Fuel Tank
Air Cleaner
Under Hood

Figure H-8

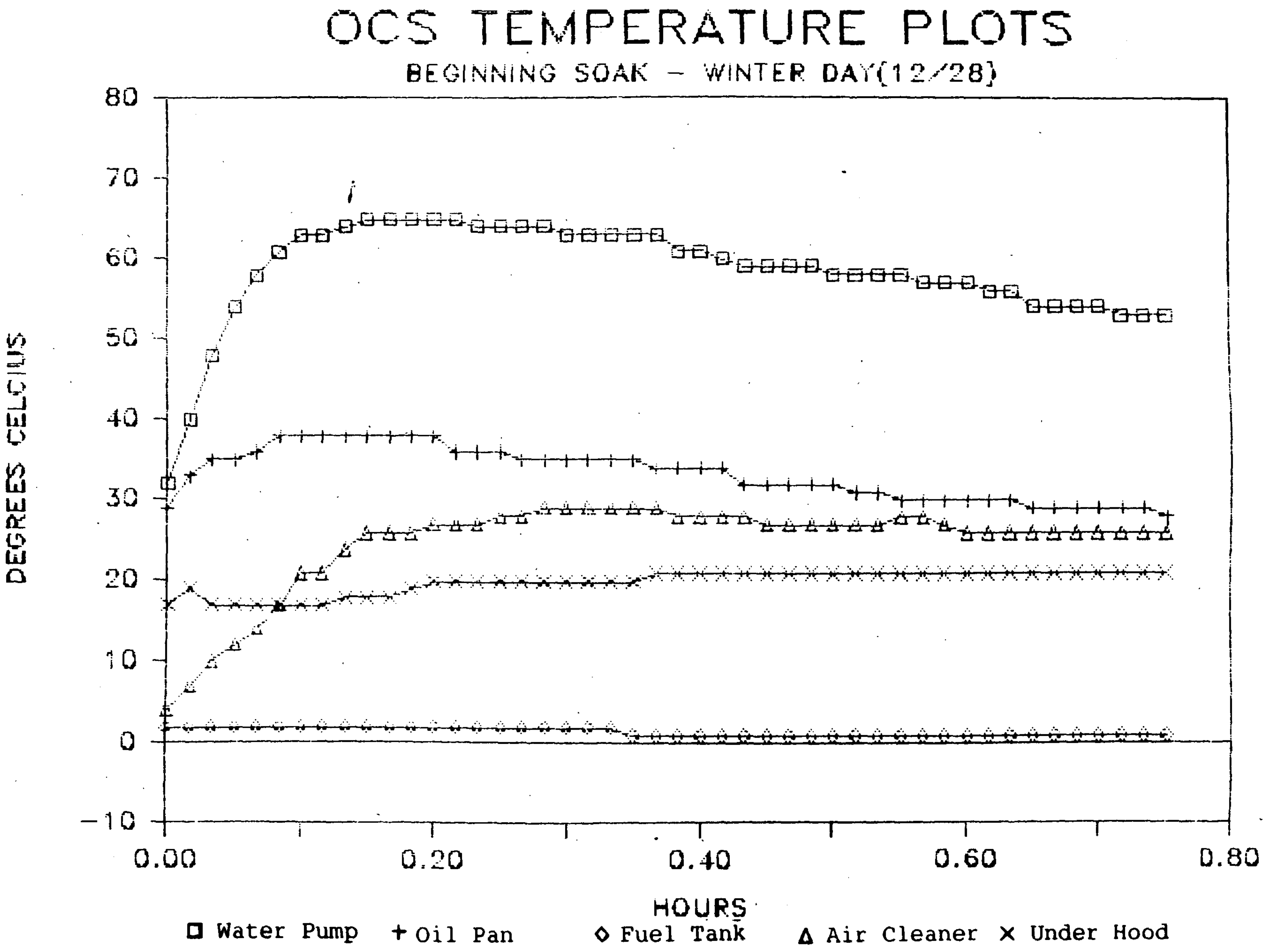

OCS TEMPERATURE PLOTS
BEGINNING SOAK - WINTER DAY(12/28)
DEGREES CELCIUS
80
70
60
50
40
30
20
10
0
-10
0.00
0.20
0.40
0.60
0.80
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
△ Air Cleaner
× Under Hood

Figure H-9

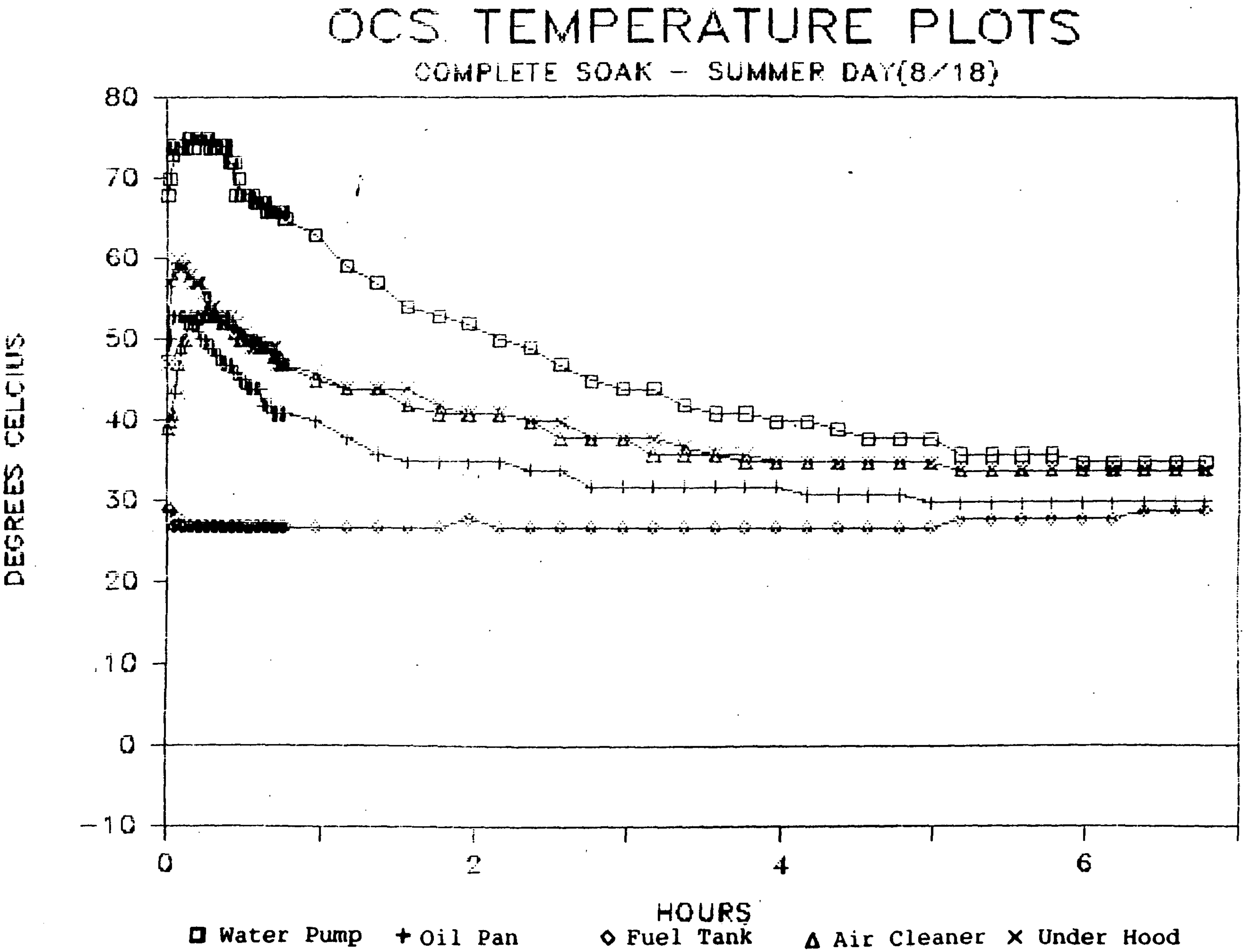

OCS TEMPERATURE PLOTS
COMPLETE SOAK - SUMMER DAY(8/18)
DEGREES CELCIUS
80
70
60
50
40
30
20
10
0
-10
0
2
4
6
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-10

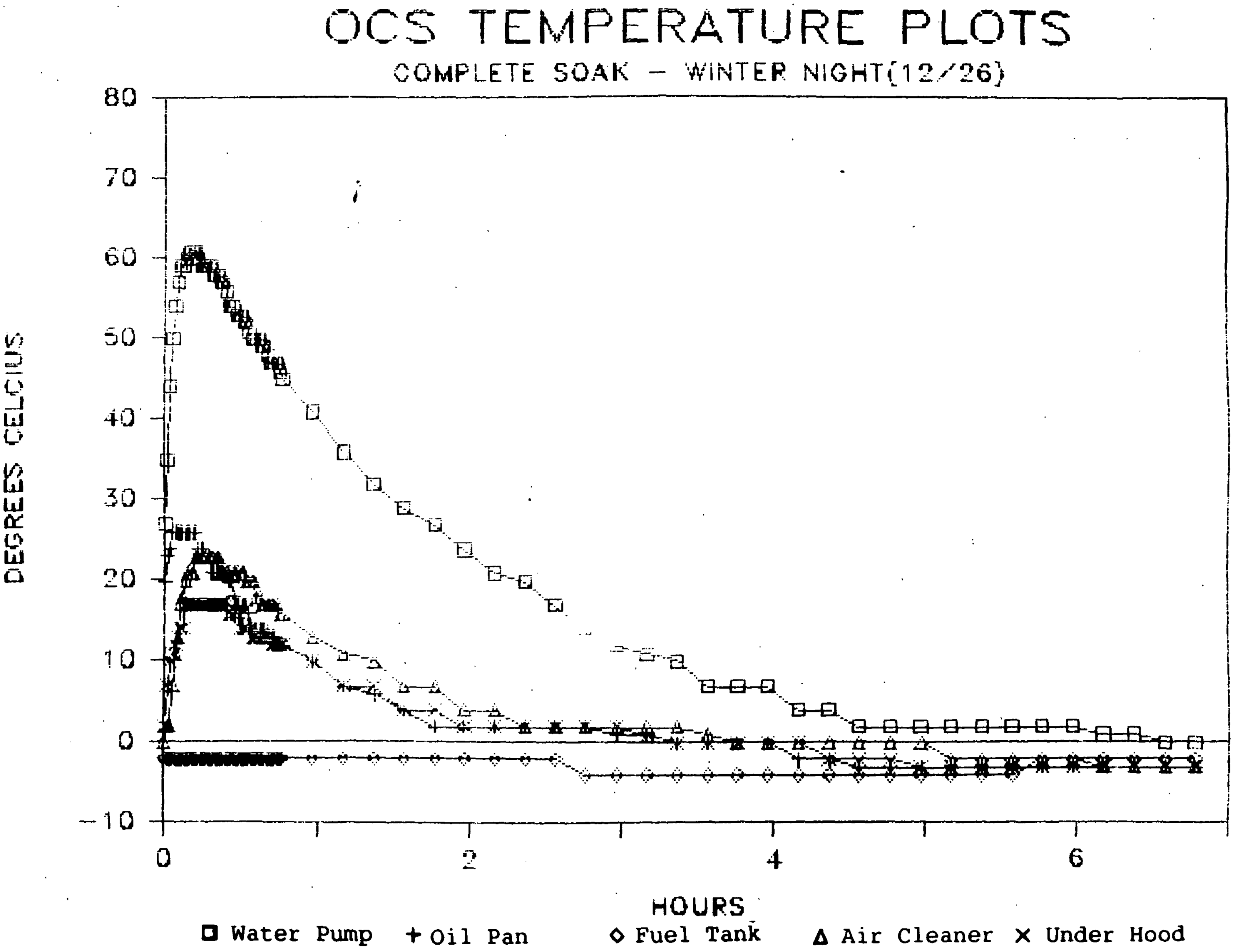

OCS TEMPERATURE PLOTS
COMPLETE SOAK – WINTER NIGHT(12/26)
DEGREES CELCIUS
80
70
60
50
40
30
20
10
0
-10
0
2
4
6
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
△ Air Cleaner
× Under Hood

Figure H-11

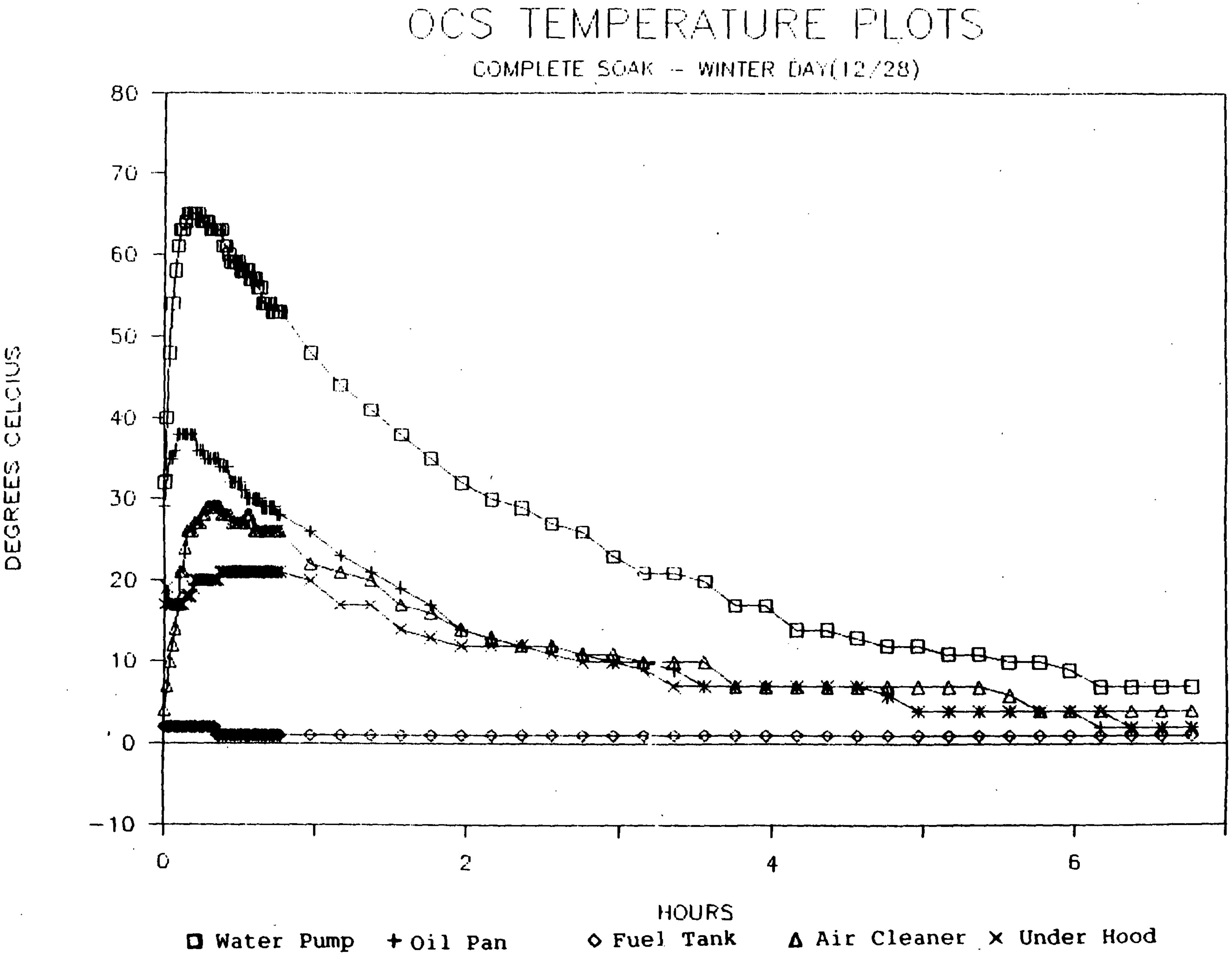

OCS TEMPERATURE PLOTS
COMPLETE SOAK - WINTER DAY(12/28)
80
70
60
50
40
30
20
10
0
-10
DEGREES CELCIUS
0
2
4
6
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-12

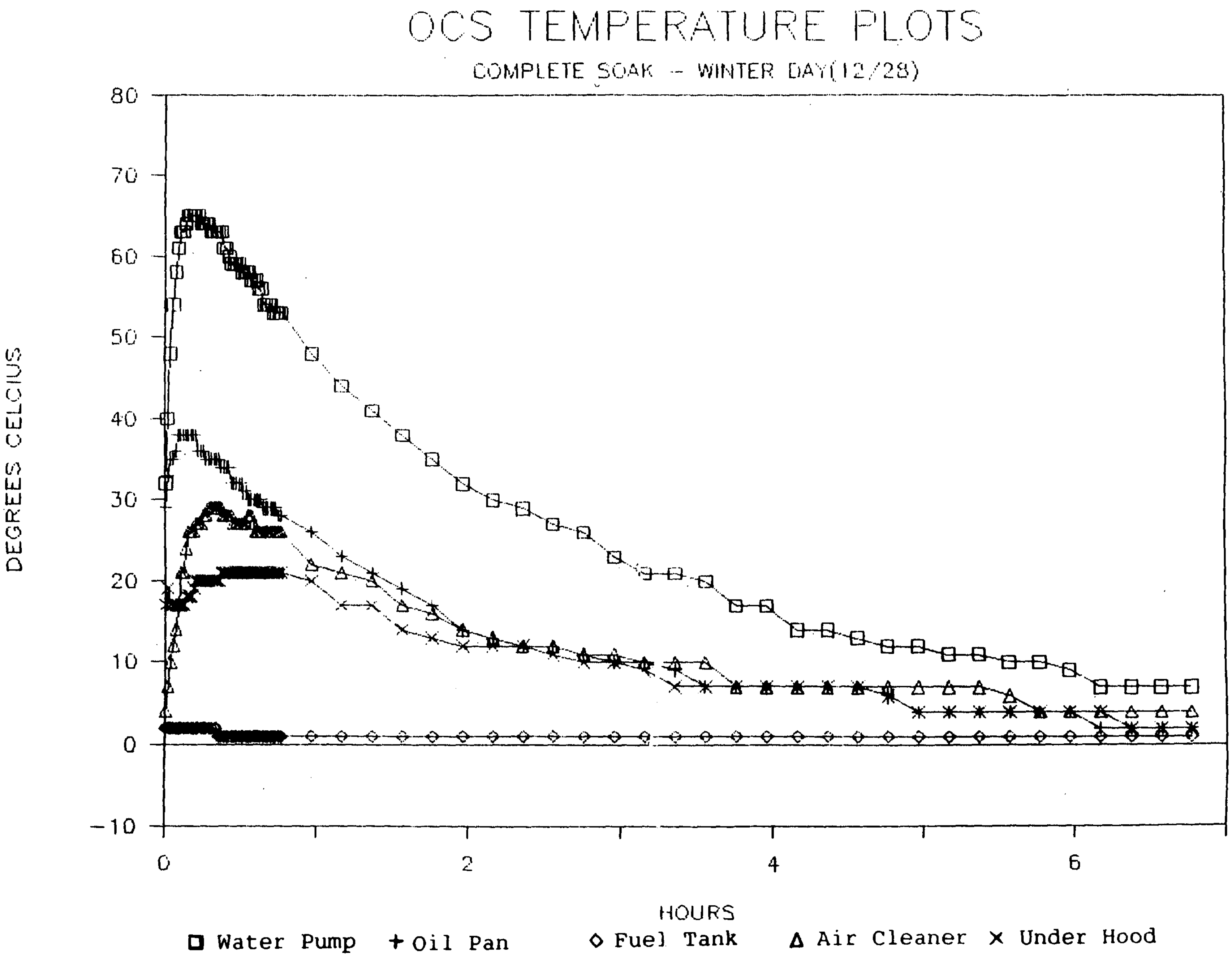

OCS TEMPERATURE PLOTS
COMPLETE SOAK – WINTER DAY(12/28)
DEGREES CELCIUS
80
70
60
50
40
30
20
10
0
−10
0
2
4
6
HOURS
□ Water Pump
+ Oil Pan
◇ Fuel Tank
Δ Air Cleaner
× Under Hood

Figure H-12