U.S. Department
of Transportation

**National Highway
Traffic Safety
Administration**

DOT-HS-807-312
DOT-TSC-NHTSA-88-2
Final Report

September 1988

Waveform Generator Signal Processing Software

MGA Research Corporation
12790 Main Road
P.O. Box 71
Akron, NY 14001-0071

Prepared for

Research and Development
Office of Crashworthiness Research
Washington, DC 20590

Technical Report Documentation Page

1. Report No.	2 Government Accession No.	3. Recipient's Catalog No.
DOT-HS-807-312		

4. Title and Subtitle	5. Report Date
WAVEFORM GENERATOR SIGNAL PROCESSING SOFTWARE	September 1988
	6. Performing Organization Code: DTS-74
7. Author(s)	8. Performing Organization Report No.: DOT-TSC-NHTSA-88-2
9. Performing Organization Name and Address: MGA Research Corporation* 12790 P.O. Box 71 Akron, NY 14001-0071	10. Work Unit No. (TRAIS): HS876/S8013
	11. Contract or Grant No.: DTRS-57-84-C-00003
12. Sponsoring Agency Name and Address: U.S. Department of Transportation National Highway Traffic Safety Administration Research and Development Washington, DC 20590	13. Type of Report and Period Covered: Final Report Jan 85 - Feb 88
	14. Sponsoring Agency Code: NRD-11

15. Supplementary Notes

*Under contract to: U.S. Department of Transportation
Research and Special Programs Administration
Transportation Systems Center
Cambridge, MA 02142

16. Abstract

This report describes the software that was developed to process test waveforms that were recorded by crash test data acquisition systems. The test waveforms are generated by an electronic waveform generator developed by MGA Research Corporation under contract to the National Highway Traffic Safety Administration. This waveform generator provides precise, repeatable signals designed to test the performance characteristics (amplitude and timing accuracy, and frequency response) of a data acquisition system. The waveform characteristics, processing algorithms, and instructions for use of the software are described in this report. Complete source listings are provided.

17. Key Words	18. Distribution Statement
Signal Waveform Generator Data Acquisition Testing Test Signals Crash Testing Data Processing Analysis Software	DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD VIRGINIA 22161

19. Security Classif. (of this report)	20 Security Classif (of this page)	21. No. of Pages	22. Price
UNCLASSIFIED	UNCLASSIFIED	172	

METRIC / ENGLISH CONVERSION FACTORS

ENGLISH TO METRIC

LENGTH (APPROXIMATE)

1 inch (in) = 2.5 centimeters (cm)
1 foot (ft) = 30 centimeters (cm)
1 yard (yd) = 0.9 meter (m)
1 mile (mi) = 1.6 kilometers (km)

AREA (APPROXIMATE)

1 square inch (sq in, in^2) = 6.5 square centimeters (cm^2)
1 square foot (sq ft, ft^2) = 0.09 square meter (m^2)
1 square yard (sq yd, yd^2) = 0.8 square meter (m^2)
1 square mile (sq mi, mi^2) = 2.6 square kilometers (km^2)
1 acre = 0.4 hectares (he) = 4,000 square meters (m^2)

MASS - WEIGHT (APPROXIMATE)

1 ounce (oz) = 28 grams (gr)
1 pound (lb) = .45 kilogram (kg)
1 short ton = 2,000 pounds (lb) = 0.9 tonne (t)

VOLUME (APPROXIMATE)

1 teaspoon (tsp) = 5 milliliters (ml)
1 tablespoon (tbsp) = 15 milliliters (ml)
1 fluid ounce (fl oz) = 30 milliliters (ml)
1 cup (c) = 0.24 liter (l)
1 pint (pt) = 0.47 liter (l)
1 quart (qt) = 0.96 liter (l)
1 gallon (gal) = 3.8 liters (l)
1 cubic foot (cu ft, ft^3) = 0.03 cubic meter (m^3)
1 cubic yard (cu yd, yd^3) = 0.76 cubic meter (m^3)

TEMPERATURE (EXACT)

[(x - 32)(5/9)] °F = y °C

METRIC TO ENGLISH

LENGTH (APPROXIMATE)

1 millimeter (mm) = 0.04 inch (in)
1 centimeter (cm) = 0.4 inch (in)
1 meter (m) = 3.3 feet (ft)
1 meter (m) = 1.1 yards (yd)
1 kilometer (km) = 0.6 mile (mi)

AREA (APPROXIMATE)

1 square centimeter (cm^2) = 0.16 square inch (sq in, in^2)
1 square meter (m^2) = 1.2 square yards (sq yd, yd^2)
1 square kilometer (km^2) = 0.4 square mile (sq mi, mi^2)
1 hectare (he) = 10,000 square meters (m^2) = 2.5 acres

MASS - WEIGHT (APPROXIMATE)

1 gram (gr) = 0.036 ounce (oz)
1 kilogram (kg) = 2.2 pounds (lb)
1 tonne (t) = 1,000 kilograms (kg) = 1.1 short tons

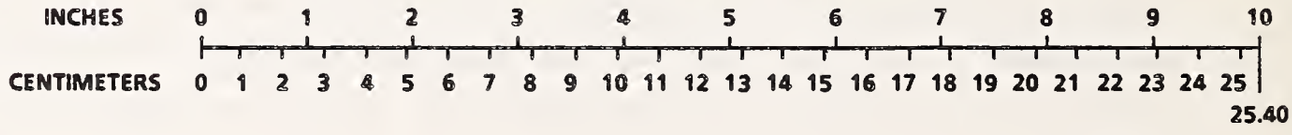
VOLUME (APPROXIMATE)

1 milliliter (ml) = 0.03 fluid ounce (fl oz)
1 liter (l) = 2.1 pints (pt)
1 liter (l) = 1.06 quarts (qt)
1 liter (l) = 0.26 gallon (gal)
1 cubic meter (m^3) = 36 cubic feet (cu ft, ft^3)
1 cubic meter (m^3) = 1.3 cubic yards (cu yd, yd^3)

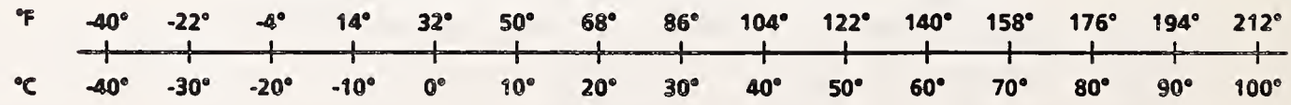
TEMPERATURE (EXACT)

[(9/5) y + 32] °C = x °F

QUICK INCH-CENTIMETER LENGTH CONVERSION



QUICK FAHRENHEIT-CELCIUS TEMPERATURE CONVERSION



For more exact and/or other conversion factors, see NBS Miscellaneous Publication 286, Units of Weights and Measures. Price $2.50. SD Catalog No. C13 10 286.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

FIGURE PAGE

1. INTRODUCTION

A waveform generator has been developed for the National Highway Traffic Safety Administration (NHTSA) that will provide standard signals for testing the performance characteristics of data acquisition channels at the facilities of NHTSA crash test contractors. Initial development of the waveform generator was carried out under the Test-Site Instrumentation Study (Contract No. DOT-HS-8-01936, Task Order No. 3). Modifications were subsequently made under Phase II (Contract No. DTNH22-82-C-07041) and Phase III (Contract No. DTRS-57-84-C-00003, Task Order Nos. 3, 3A, 8, 8A, and Purchase Order No. DTRS-57-86-P-81655) of the study. This report presents a description and operating manual for the processing software package that was initially developed under Phase II and modified under Phase III of this study. Under Phase III, the software package was modified to account for the changes that have been made to the format of the waveforms generated by the waveform generator instrument. The software package was also integrated into a "user friendly" package. The software is implemented on the VAX 11/780 minicomputer at NHTSA.

This report consists of five sections. The first section discusses the characteristics of the waveforms and the results that are generated by the software analysis. The second section is a user's manual for the software package. The third section contains a narrative description and flow diagrams of the software. The fourth section consists of a listing of the software. The fifth section describes the procedure for programming the test waveforms into EPROMs (Erasable Programmable Read Only Memory). Also included in this section is a narrative description, a flow diagram, and a source listing of the software which was developed to compare the actual content of the EPROMs to idealized test waveforms is presented. For further information on the waveform generator instrument, the reader is directed to the Operator's Manual for Waveform Generator Model RPG-6236-A.

2. WAVEFORM CHARACTERISTICS

This section describes the characteristic of each waveform that is generated by the waveform generator instrument. A discussion of the results produced by the processing software for each waveform follows the discussion of each waveform characteristic.

2.1 RECTANGLE WAVEFORM CHARACTERISTICS

The characteristics of the rectangle waveform include positive and negative full-scale amplitudes of 200 g's in magnitude. The period of one rectangle cycle is 10 msec. The are 10 cycles present in the rectangle waveform with each cycle starting at a level of +200 g's and continuing for 6 msec then dropping to a level of -200 g's and continuing for 4 msec and finally returning to the starting position level of +200 g's. (It should be noted that the 10th cycle returns only to a level of 0 g's.) The duration of the rectangle waveform is 100 msec.

Figure 2-1 shows a typical rectangle waveform sequence.

2.2 RECTANGLE WAVEFORM PROCESSING RESULTS

The rectangle waveform is used to check the following characteristics of a data channel:

a. Time deviation from theoretical time
b. Time linearity
c. Time offset
d. Steady-state amplitude response
e. Amplitude overshoot in response to a step input
f. Channel-to-channel time deviation

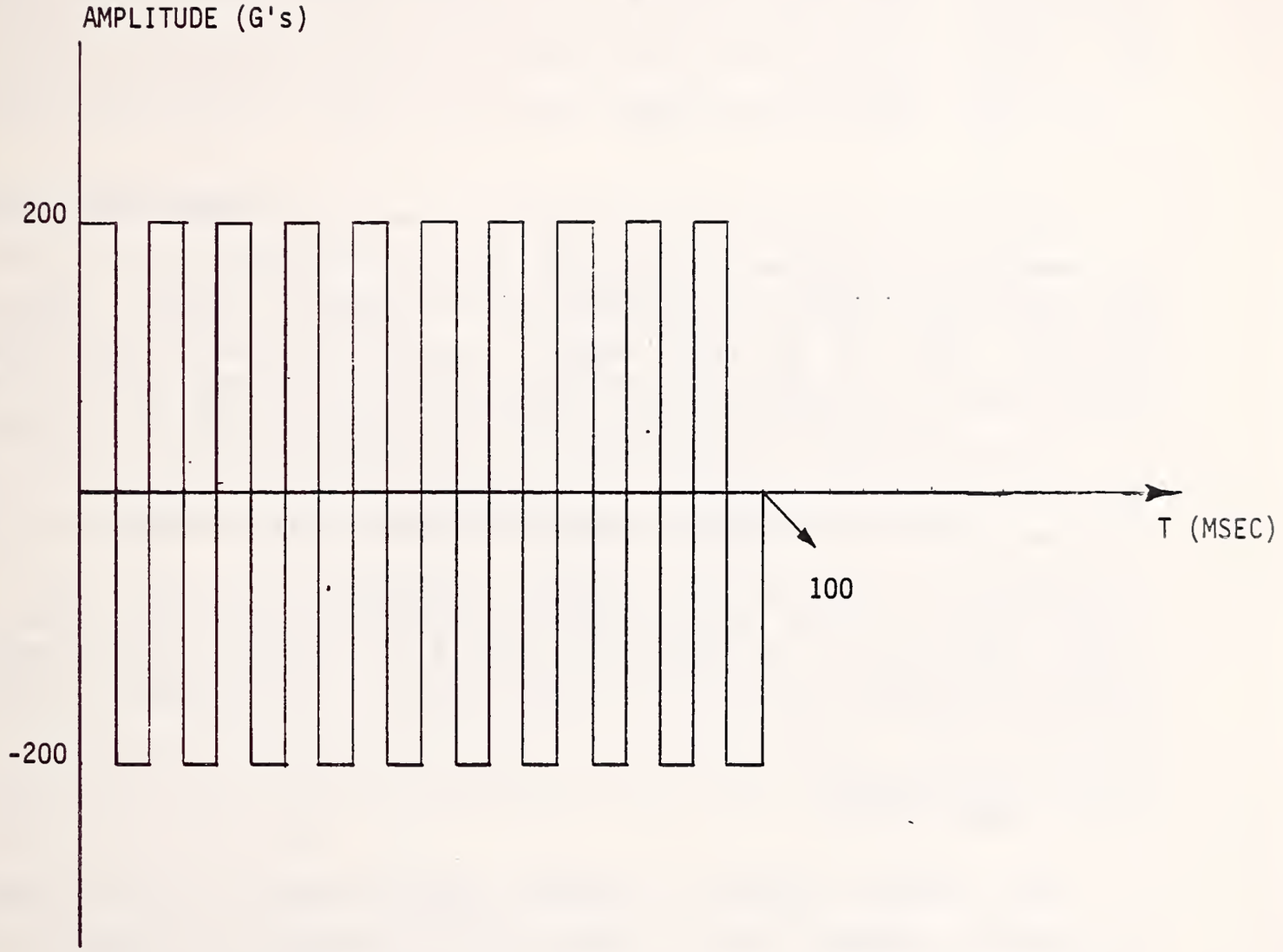
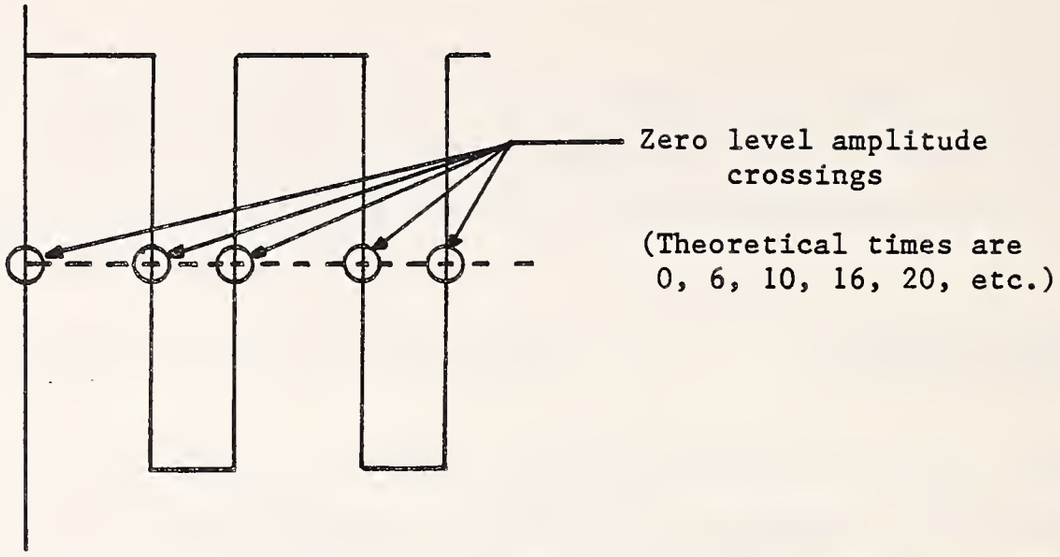


Figure 2-1 Rectangle Waveform

2.2.1 Time Deviation from Theoretical Time



The actual time of the zero crossing of the recorded rectangle waveform is measured to determine the magnitude of variations in the data time base. The theoretical time of zero crossing is established by the time reference waveform. That is, the first sample of the time reference waveform that exceeds 50% of its maximum value determines the value of time for which time = 0. Each subsequent zero crossing of the rectangle waveform is supposed to occur at 6, 10, 16, 20, 26, 30, etc. milliseconds after T0 (time = 0). The time deviation is the difference between the measured and the theoretical zero crossing times. In the present rectangle processing software, the time of the initial rise (at T0) is not measured. It is extrapolated to occur 10 milliseconds before the first measured zero crossing of the first positive going edge of the waveform.

2.2.2 Time Linearity

The rectangle waveform is examined using the method of least squares. By fitting a straight line to a number of X-Y coordinates, the slope and the Y-intercept of that line give an indication of the quality of the data.

The X coordinate used in this analysis is the theoretical time the waveform begins a complete period as defined by a 0 g's amplitude level. The theoretical time value begins at zero milliseconds and increments by 10 msec for each cycle. The Y coordinate used in this analysis, called the time

deviation from theoretical, is the difference between the actual time at which the waveform crosses the 0 g's amplitude level and theoretical time of the 0 g's amplitude level crossing. This value can range from zero in the best case to infinity for the worst case.

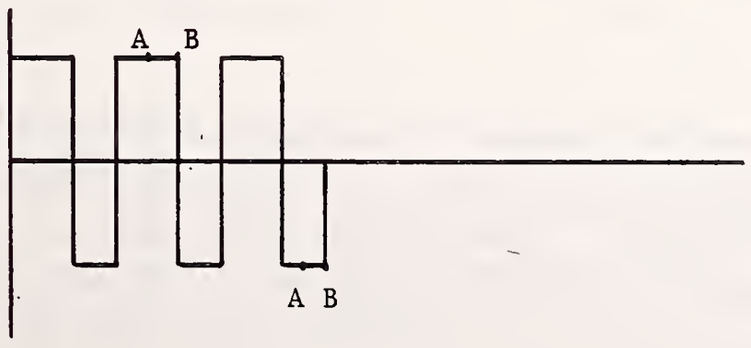
The slope of the least squares straight line fit is an indication of the time linearity of the data. A deviation from zero of + 0.01 is allowed while acceptability is maintained. This tolerance translates to + 1% variation in the data acquisition sample rate when the sample rate conforms to the minimum requirements for SAE class 1000 data (8000 samples per second). The ideal case would be a slope of zero which would indicate perfect linearity.

2.2.3 Time Offset

The Y-axis intercept of the least squares straight line fit is an indication of the time offset. A deviation of + 1.0 msec is allowed while still maintaining acceptability. The ideal value for the Y-axis intercept would be zero which would indicate perfect timing synchronization.

2.2.4 Steady-State Amplitude Deviation from Theoretical Amplitude

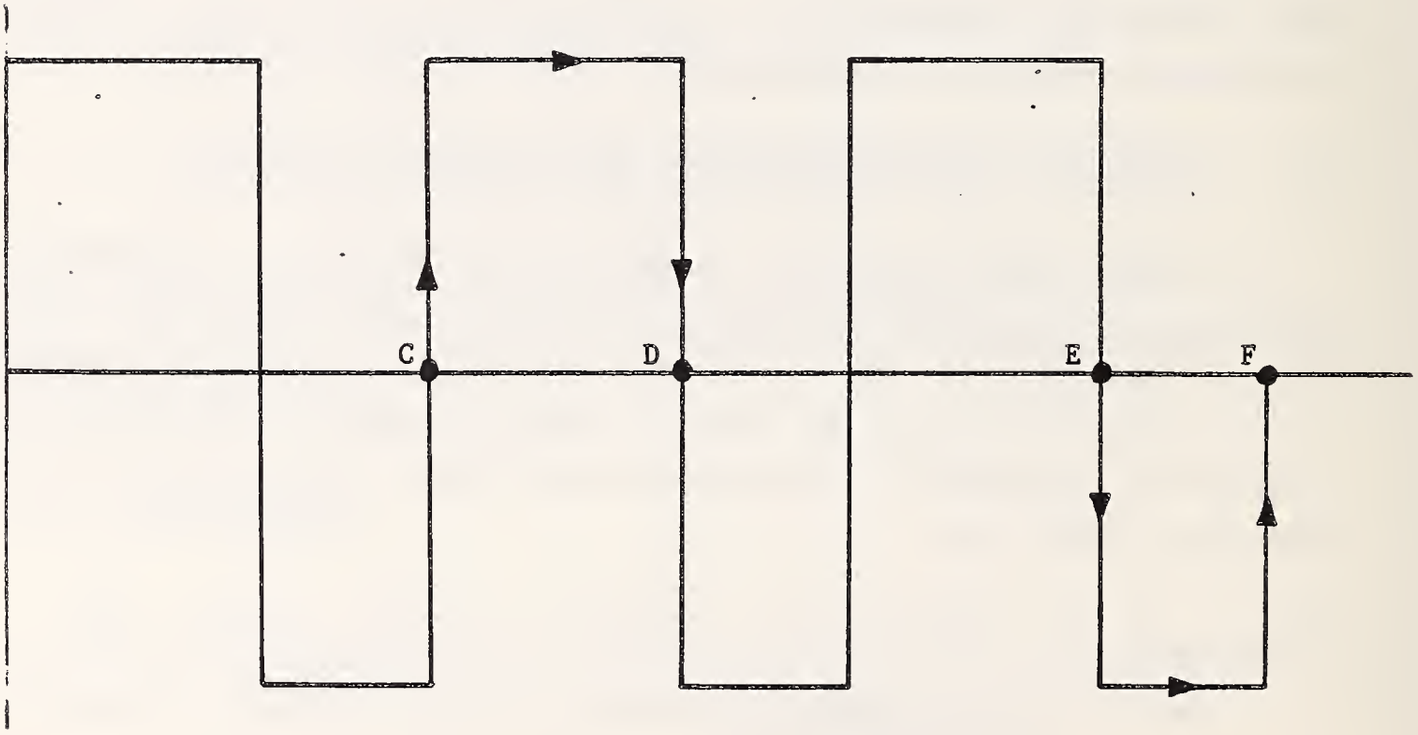
The steady-state amplitude deviation from theoretical amplitude is determined by comparing the steady state average to the theoretical value of ±200 g's (depending upon the sense of the data) for each of the 20 occurrences of the extremes throughout the entire rectangle waveform. The steady-state amplitude is calculated by averaging the data values over the closed interval from A to B shown below.



Point A is the center of the constant amplitude level duration and point B is one millisecond before the next zero crossing. The duration of the averaging is no less than 2 msec for the positive data levels and 1 msec for the negative data levels. Deviation from the theoretical steady-state amplitude is expressed as a percentage of the theoretical level and is determined by dividing the difference between the actual and theoretical amplitudes by the value of 200 (the absolute value of the theoretical amplitude).

2.2.5 Amplitude Overshoot Relative to the Calculated Steady-State Amplitude

The relative overshoot is determined by obtaining the data value extreme which occurs over the closed interval from C to D for the positive intervals and from E to F for the negative intervals.



The value of relative overshoot is expressed as a percentage of the change in steady-state amplitude by dividing the calculated overshoot by the change in steady-state amplitude. (Theoretically, all amplitude changes are 400 g's with the exception of the start of the first cycle which rises from 0 to 200 g's and the end of the last cycle which only returns to 0 g's from the -200 g's amplitude level.)

2.2.6 Channel-to-Channel Time Difference for the Rectangle Waveform

The channel-to-channel time deviation is a determination of how each individual channel is performing relative to a common standard. The value which is used to make this determination is the actual time of occurrence of each rectangle cycle beginning at zero amplitude. The common standard which is used is the actual times of occurrence of each rectangle cycle for the first channel processed. The channel-to-channel performance is measured by the difference between the time of occurrence of corresponding rectangle cycles for the first channel and the channel being examined.

2.3 STAIR WAVEFORM CHARACTERISTICS

The characteristics of the stair waveform include amplitude levels ranging from 0 to +200 g's and from -200 to 0 g's with a 0 g's level between the +200 and -200 g's extremes. The time duration of each level is 10 msec and there are a total of 13 amplitude levels (40 g increments) in the waveform, of which three are the zero level. The duration of the waveform is 120 msec; a 5 msec zero level is added to the end of the waveform to extend it to 125 msec.

Figure 2-2 shows a typical stair waveform.

2.4 STAIR WAVEFORM PROCESSING RESULTS

The stair waveform is used to check the following characteristics of the data channel:

a. Time deviation from theoretical
b. Time linearity
c. Time offset
d. Amplitude linearity
e. Amplitude offset
f. Steady-state amplitude deviation from theoretical levels
g. Amplitude overshoot for different amplitude levels
h. Channel-to-channel time deviation

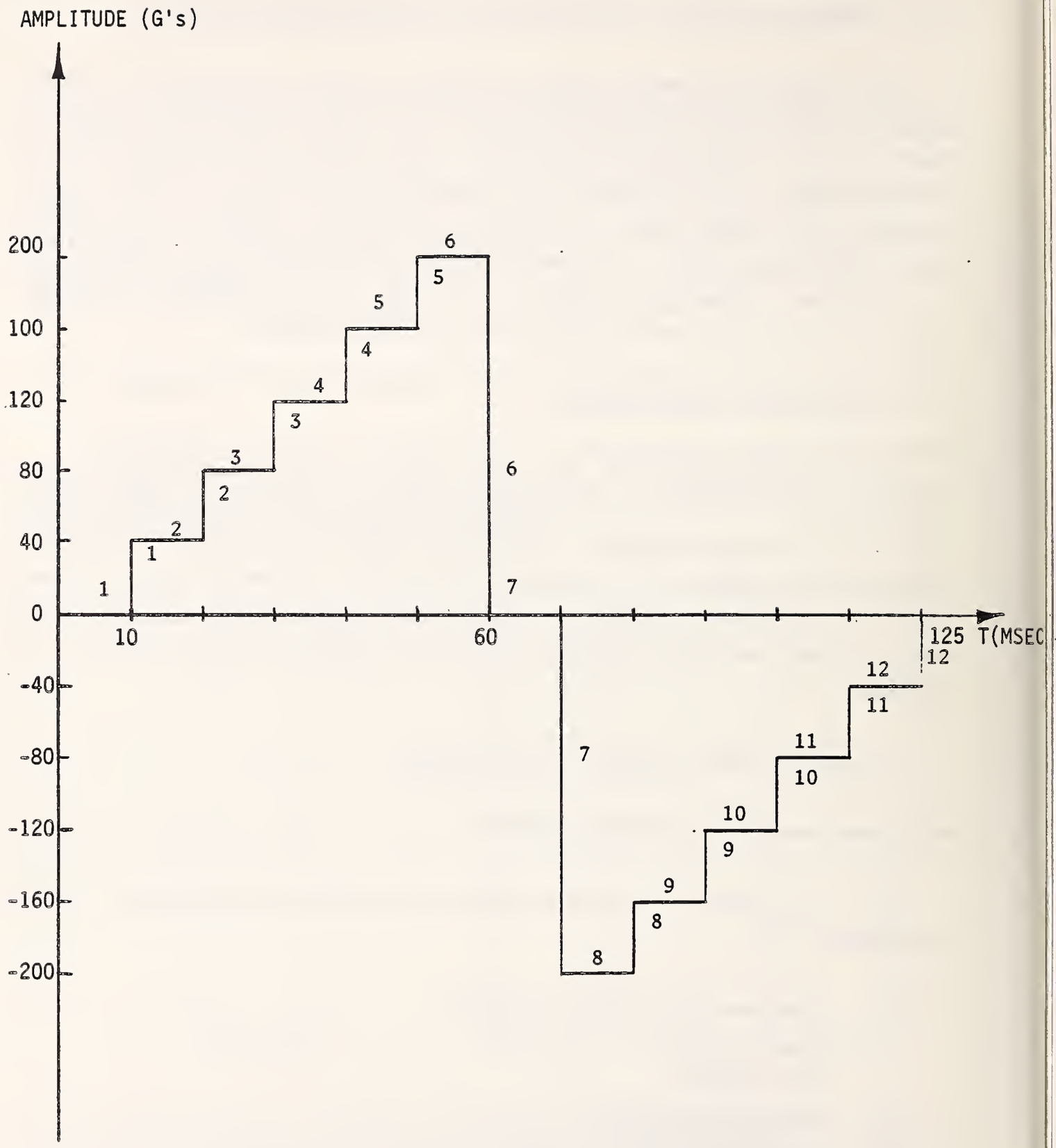
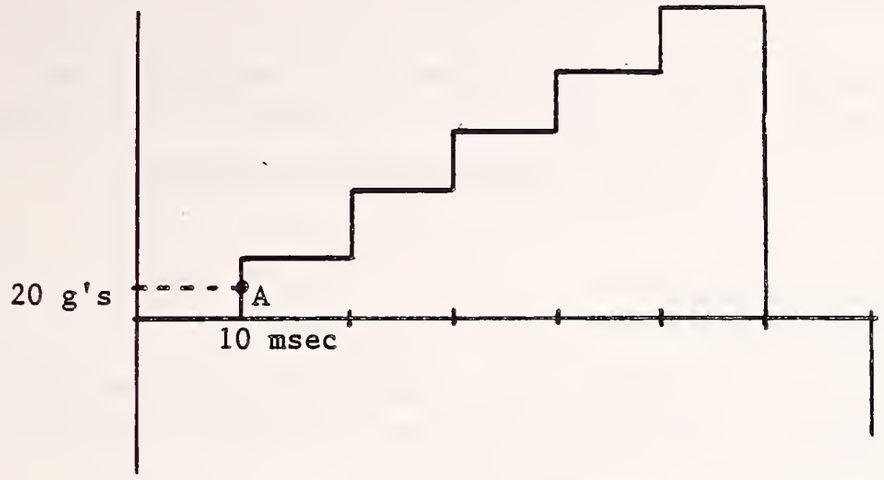


Figure 2-2 Stair Waveform

2.4.1. Time Deviation from Theoretical

Shown below is an example of the calculation of time deviation from theoretical.



The time deviation from theoretical is determined by subtracting the theoretical time the waveform should reach an amplitude which is 50% of the change in amplitude for the interval being considered from the actual time the waveform reaches that amplitude. In this example, the theoretical time of reaching the amplitude of 20 g's (50% of the amplitude change from 0 g's to 40 g's) is 10 msec. In the STAIR summary report, the "TIME DEVIATION FROM THEORETICAL TIME" reported for each interval corresponds to the measured time of occurrence of each numbered rise shown in Figure 2-2.

2.4.2 Time Linearity

The stair waveform is examined for time linearity and offset using the method of least squares. When a straight line is fitted through a number of X-Y coordinate pairs, the slope and Y-intercept of the line give an indication of the quality of the data. The X-coordinates used are the theoretical times at which the waveform changes amplitude levels. The values for the X-coordinate range from 10 to 120 msec in increments of 10 msec. The Y-coordinates used in the analysis are the time deviation from theoretical values determined for the waveform as described above. The values can range from zero for the best case to infinity for the worst case.

The slope of the least squares fit straight line is an indication of the time linearity of the data. The value for the slope is allowed a tolerance of

±0.01. This corresponds to an allowable deviation of ±1%. The ideal case would have a slope of zero, indicating perfect linearity.

2.4.3 Time Offset

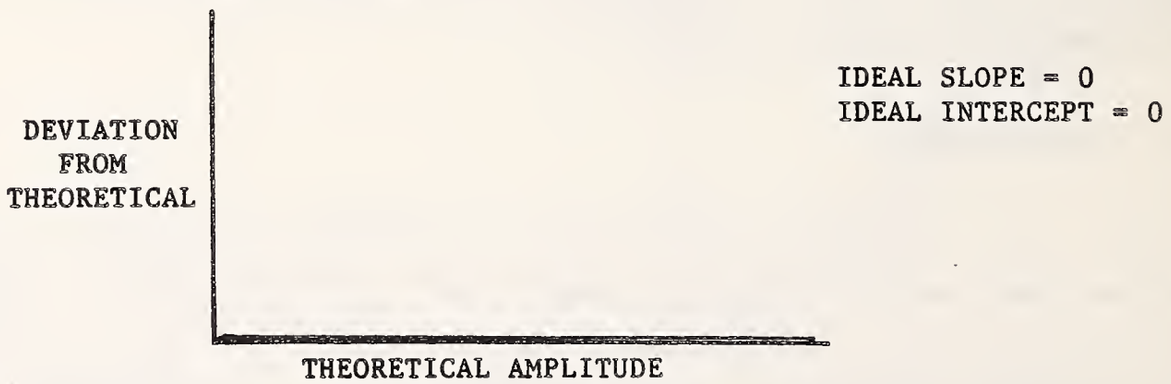
The Y-intercept of the least squares fit straight line is an indication of the time offset. The tolerance for this value is ±1.0 msec. The ideal time offset would be zero msec indicating perfect timing.

2.4.4 Amplitude Linearity

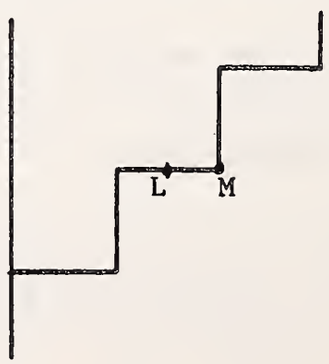
The amplitude linearity is determined using the same approach that is used for the time linearity calculation described in Section 2.4.2. The tolerance for amplitude linearity is ±2.5%.

2.4.5 Amplitude Offset

The amplitude offset is determined using a similar approach to that used for the time offset calculation described in Section 2.4.3. The tolerance for amplitude offset is ±2.5%.



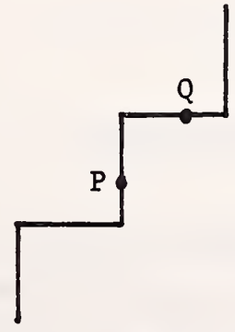
2.4.6 Steady-State Amplitude Deviation from Theoretical Amplitude



The data used to determine the steady-state amplitude occurs in the closed interval from L to M shown above. L corresponds to the center of the constant amplitude levels and M is a point which occurs one millisecond before the next rise in amplitude. The steady-state amplitude is calculated by taking the average of the amplitude values in this interval. Once the actual steady-state value is determined, it is then compared to the theoretical steady-state value. The difference between the theoretical and actual values is divided by the full-scale amplitude (200 g's) in order to express the deviation from theoretical amplitude as a percentage of the full-scale amplitude. The tolerance for this portion of the analysis is ±2.5%. In the STAIR summary report, the "STEADY STATE AMPLITUDE DEVIATION FROM THEORETICAL AMPLITUDE" reported for each interval corresponds to the percentage difference, described above, for each numbered level shown in Figure 2-2.

2.4.7 Amplitude Overshoot Relative to Calculated Steady-State Value

The maximum data point in the closed interval from P to Q shown below is used to determine the relative amplitude overshoot by comparing the value of the point to the calculated (average) steady-state amplitude. The difference between the maximum and steady-state values is divided by the difference between the steady-state levels occurring before and after the step and expressed as a percentage of the step height. Overshoot of ±13% is considered acceptable. In the STAIR summary report, the "AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED STEADY STATE AMPLITUDE" reported for each interval corresponds to the overshoot calculated for each numbered rise shown in Figure 2-2.



2.4.8 Channel-to-Channel Time Difference for the Stair Waveform

The channel-to-channel time deviation is an indication of how each individual channel is performing relative to a common standard. The value

which is examined is the actual time of occurrence of each amplitude change during the stair waveform. Each value is compared to the actual time of occurrence of the corresponding amplitude change of the waveform that is contained in the first channel processed thus making the first channel which is processed the standard. The tolerance for this part of the analysis is ±0.25 msec.

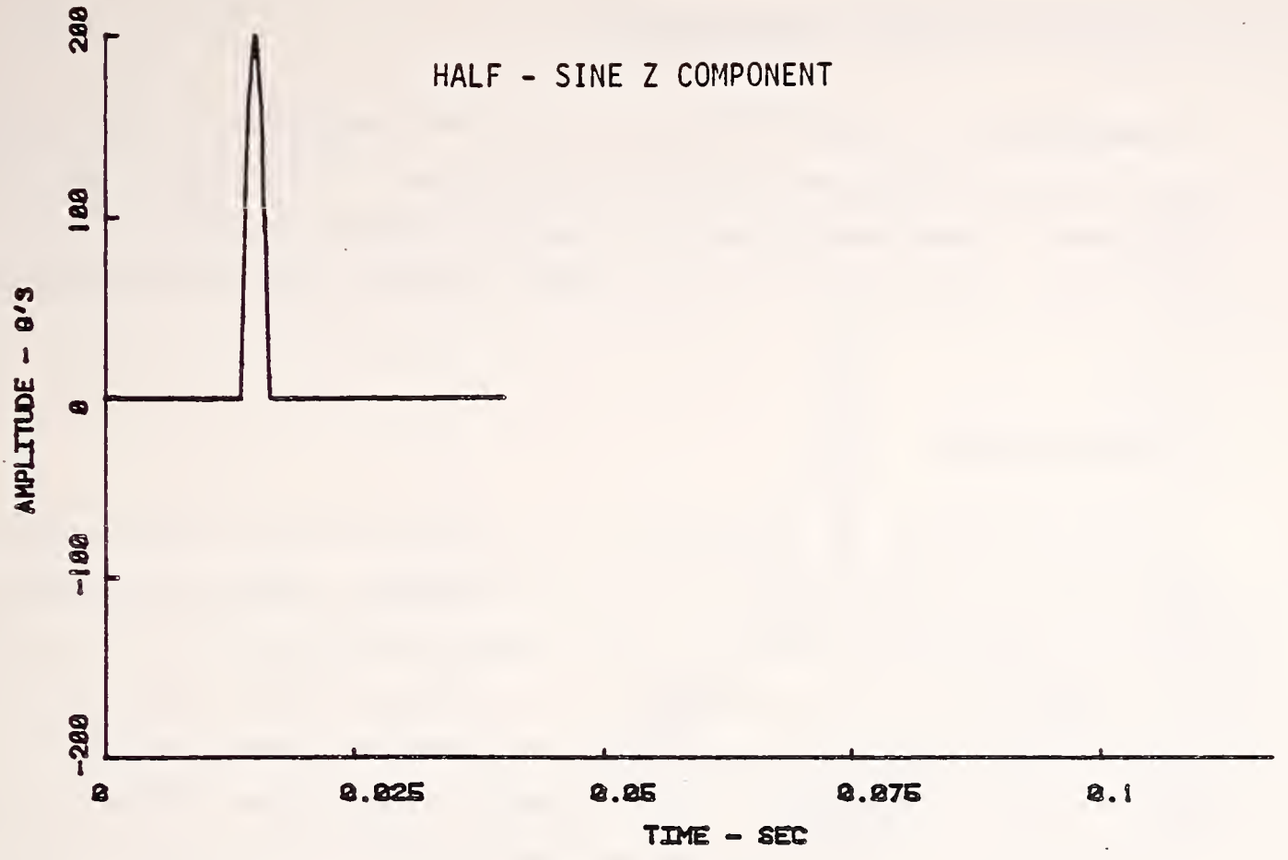
2.5 HALF-SINE WAVEFORM CHARACTERISTICS

The half-sine waveform is broken into two components - an X-axis and a Z-axis. The two components are produced on separate channels of the waveform generator instrument. (This forces the requirement of processing the data in multiples of two channels per execution of the processing software.) Both components consist of half-sine pulses which have a time duration of 2.985 msec. The X-axis component has a peak amplitude of -200 g's and the Z-axis component has a peak amplitude of +200 g's. Together, these two components form a resultant waveform which, when analyzed in terms of HIC (Head Injury Criterion), give a HIC value of 1665.6 with a HIC calculation time duration of 2.0 msec. It should be noted that the components are synchronized in time. The half-sine waveform was designed to simulate a short duration impact which is often found in actual crash test data. The half-sine waveform components are used to provide a means of checking the HIC computation accuracy and the channel to channel time difference between the two channels containing the waveform components. Figure 2-3 shows the two components of the half-sine waveform.

2.6 HALF-SINE WAVEFORM PROCESSING RESULTS

2.6.1 Channel-to-Channel Time Difference

The actual times for the peak amplitude of the X and Z components are utilized to determine the channel-to-channel time difference. The times of the peak amplitudes are determined by calculating the time at which each waveform crosses an amplitude level which is 20% of full scale. It is then assumed that the peak amplitude occurs at the time which is exactly halfway between the two 20% level crossings. The difference between the X-axis and Z-axis time-of-peak amplitude values is the channel-to-channel time difference. The tolerance allowed for this difference is ±0.1 msec.



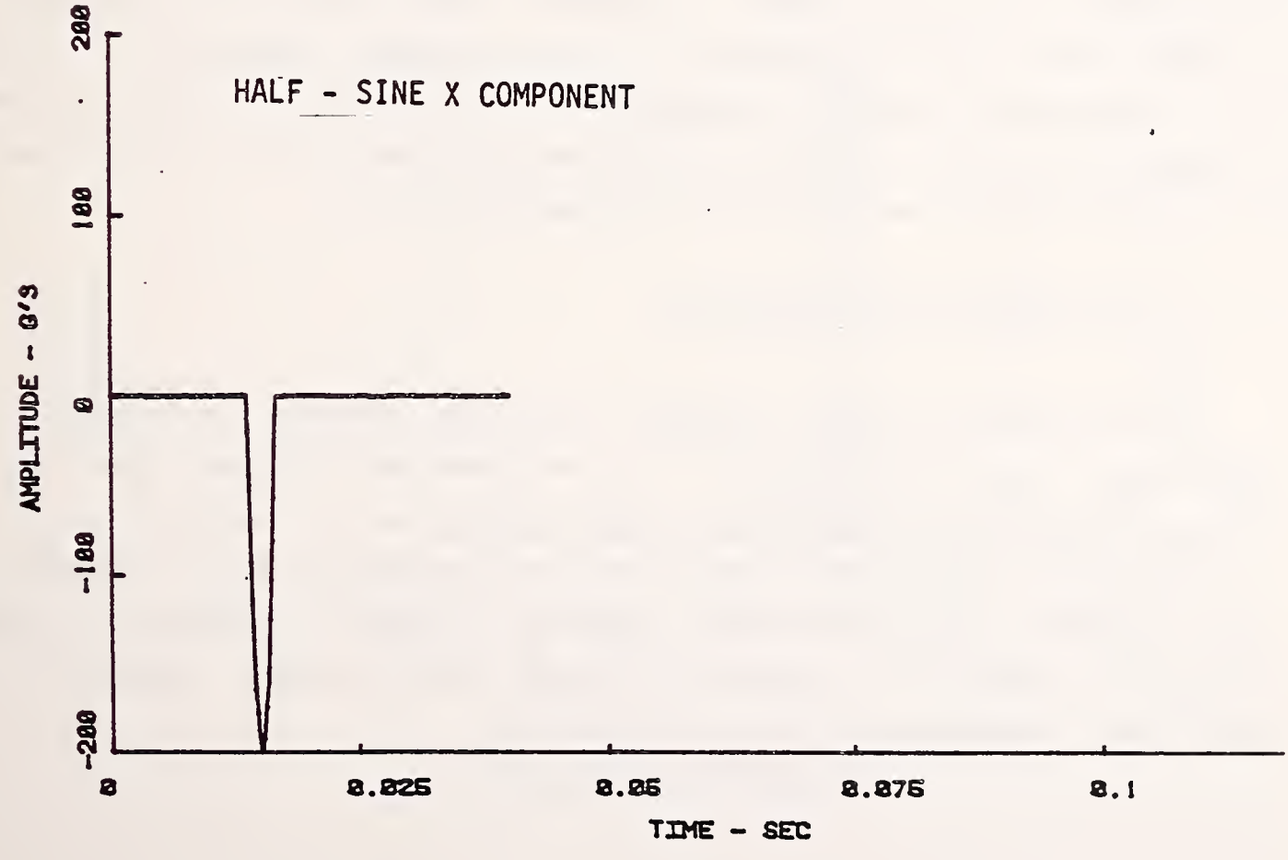


Figure 2-3 Half-Sine Waveform

2.6.2 Waveform Component Time Deviation

To determine the time deviation for the two components of the waveform, the times of peak amplitude occurrence calculated for the channel-to-channel time difference (described above) are compared against theoretical times at which the peaks should occur. The allowable deviation for each component is ±1.0 msec.

2.6.3 HIC Deviation

The HIC deviation is determined by comparing the HIC value determined from the resultant of the recorded X and Z components against the theoretical value. The theoretical value in this case may not be the value 1665.6 mentioned above since the theoretical is determined from curves which are generated so that they have the same amplitude and time duration as the recorded waveform components. To accomplish this, true half-sine curves ($a = A * \sin(wt)$) are fitted to the peak amplitude and time values calculated during the channel-to-channel time difference determination (Section 2.6.1). This method is used in order to separate the errors in time and amplitude deviation from the HIC calculation. (If the waveform components do not match the time duration and/or the amplitude of the theoretical waveform components, then the chances that the HIC calculation will fall within the ±6% limit set for the HIC calculation tolerance are greatly reduced.)

2.7 CRASH WAVEFORM CHARACTERISTICS

As with the half-sine waveform, the crash waveform is composed of two components: X-axis and Z-axis. The waveform was obtained from the head accelerations which occurred in a crash test which was assigned test number 206 by NHTSA. The resultant of the X and Z components yield a computed HIC value of 930. The crash waveform is used as a means of checking the accuracy of the data channels with respect to actual crash test data. Figure 2-4 shows the X and Z components of the crash waveform.

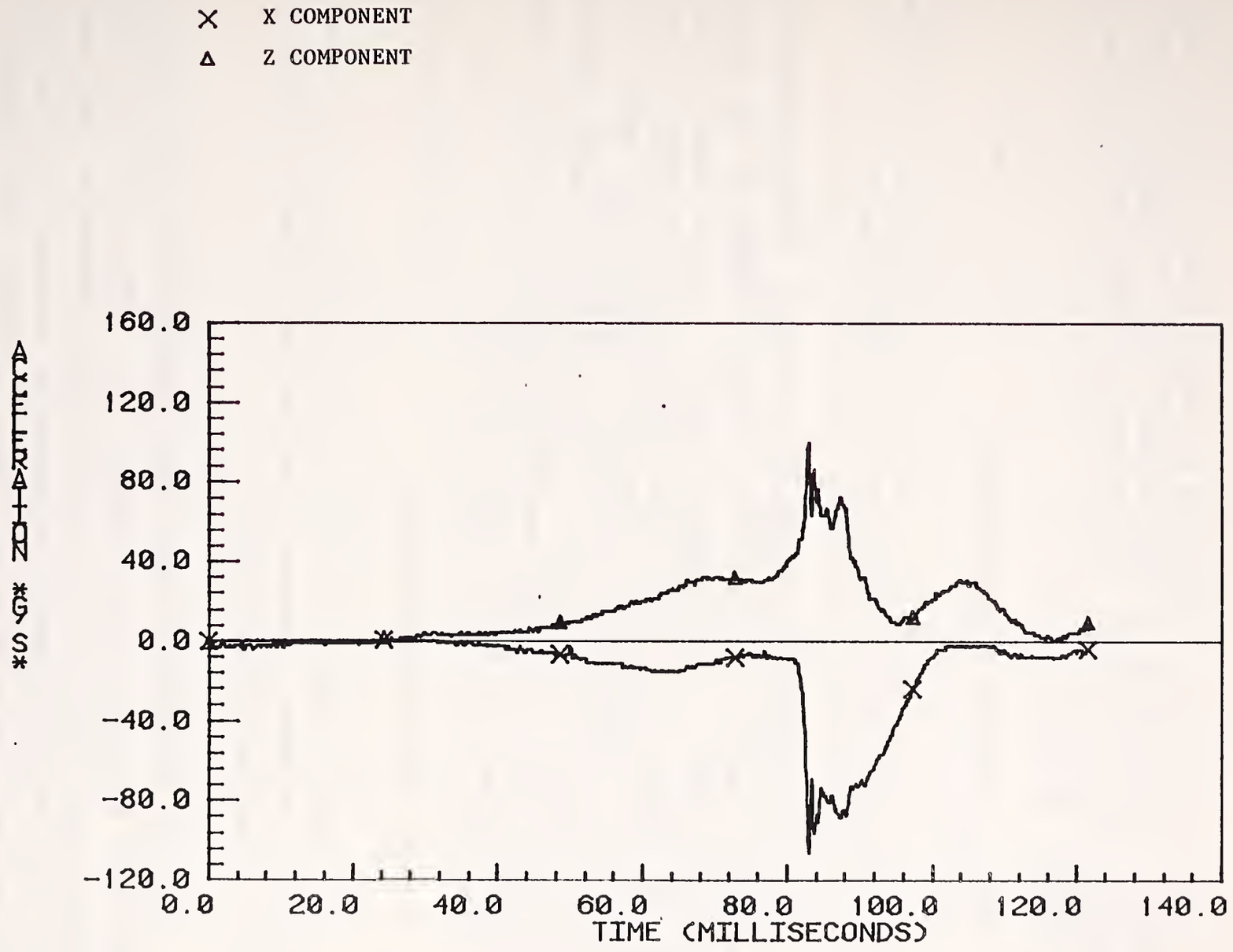


Figure 2-4 Crash Waveform

2.8 CRASH WAVEFORM PROCESSING WAVEFORM RESULTS

2.8.1 HIC Deviation

HIC value deviation is determined by comparing the HIC value obtained by processing the resultant of the recorded X and Z components against the theoretical HIC value of 930. As with the half-sine HIC deviation analysis, the allowable difference is ±6%.

2.9 SUM-OF-SINES WAVEFORM CHARACTERISTICS

The sum-of-sines waveform is a test signal which is used to determine the amplitude frequency response of a data acquisition channel.

The sum-of-sines waveform consists of the sum of equal amplitude sine waves whose frequencies are successive integer harmonics of the fundamental (lowest) frequency of the composite signal. For the Phase III waveform generator instrument, the sum-of-sines composite waveform consists of the sum of the first 14 harmonics of 273.4375 Hz. The peak level of the composite output is scaled to be the full-scale input of the data acquisition channel being tested. The sum-of-sines composite produced by the Phase III waveform generator instrument is shown in Figure 2-5.

2.10 SUM-OF-SINES WAVEFORM PROCESSING

The recorded data acquisition channel response to the sum-of-sines waveform is processed to determine the amplitude frequency response of the channel at the test signal frequencies.

This data processing produces three types of output files: plot files; a statistical data file; and a summary file. The plot file consists of fourteen ordered data pairs for plotting signal frequency vs. amplitude ratio in dB. The statistical data file consists of sequences of fourteen values of amplitude ratio (one value at each signal frequency), (one sequence for each data channel analyzed). For each signal frequency, the summary file

provides the signal frequency, the DFT line number, the DFT output frequency, the input DFT value, the output DFT value, the scaled DFT ratio, and the amplitude ratio in dB. One table, of this description, is provided for each data channel tested.

It is appropriate to discuss the information provided in the summary file. The discrete Fourier transform (DFT) plays a central role in the determination of amplitude frequency response. The DFT output consists of a sequence of numbers. As the DFT is used here, each number in the DFT output can be considered the energy level in the signal analyzed, at a particular frequency. The frequency, to which each DFT output value corresponds, starts at zero for the first DFT output value and increases monotonically, with a constant increment, for each subsequent member of the DFT output sequence. The "DFT line number" is the number of frequency increments contained in the "DFT output frequency" closest to the "signal frequency" listed. The input DFT value is the magnitude of the output value of the DFT (of the input waveform) whose corresponding frequency is closest to a signal frequency. The output DFT value is the magnitude of the output value of the DFT (of the channel output signal) whose corresponding frequency is closest to a signal frequency. The DFT ratio is the ratio of the output DFT value to the input DFT value at the same signal frequency. This ratio is multiplied by a scale factor, which is 0.993 multiplied by the ratio of the input DFT value to the output DFT value at the lowest signal frequency. The scaled DFT ratio is the numeric amplitude ratio.

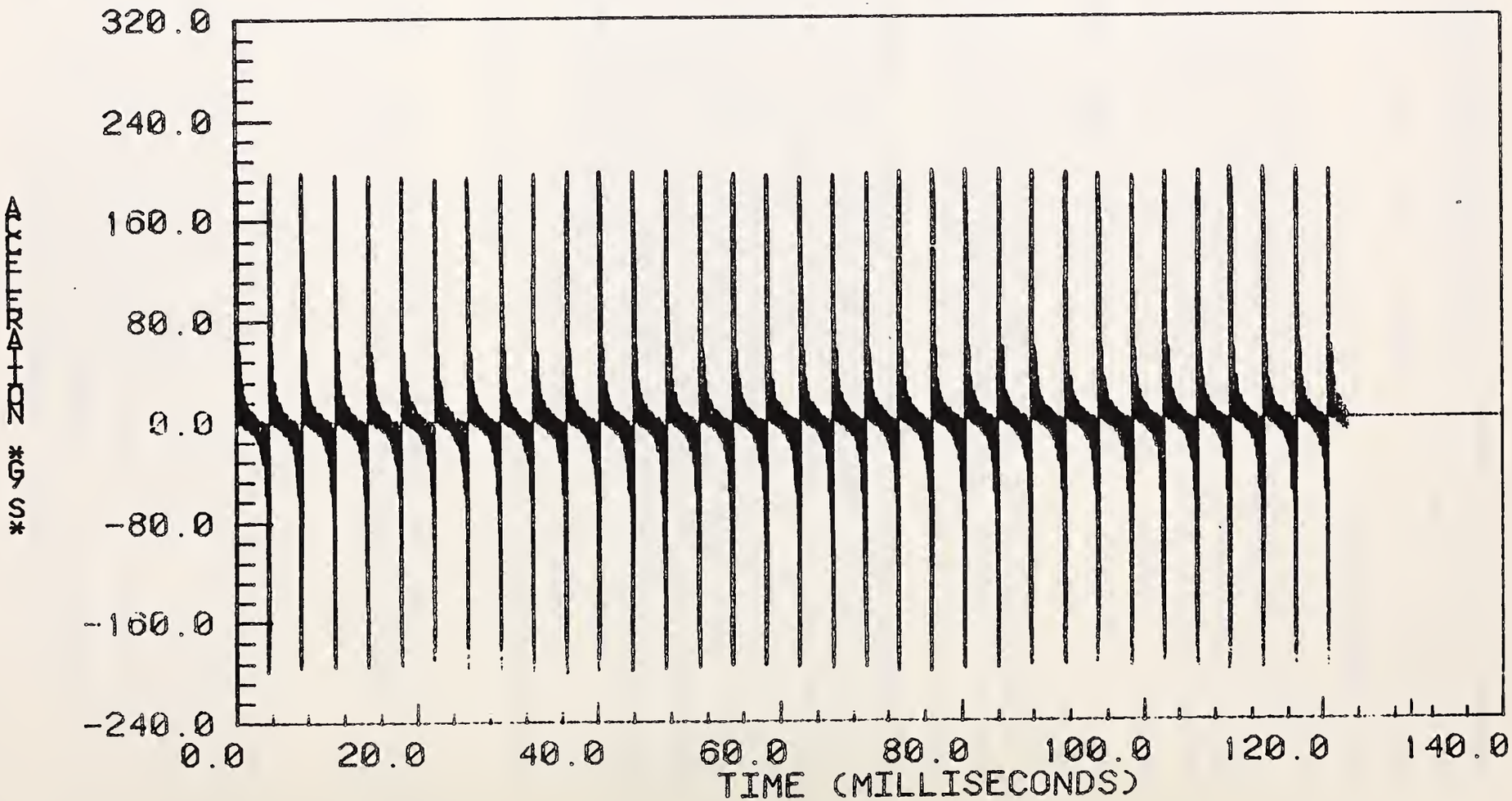


Figure 2-5 Sum-of-Sines Waveform

3. USER'S MANUAL

This manual presents the procedure for processing the recorded waveform generator instrument output data. This software package performs two general functions. The first function is to prepare the channel output data files so that individual waveforms can be processed. The second function is to process the data. This software package is capable of processing from 2 to 18 channels in a single execution pass. (Note: If half-sine and crash waveforms are to be processed, the sequence of data files to be processed must alternate between X (group 1 waveforms) and Z (group 2 waveforms) axis data, starting with an X-axis data file.)

Actual data recorded from the output of a waveform generator instrument will be used for examples throughout this manual. The data was recorded at Calspan Corporation, Buffalo, NY. The data files which will be utilized are:

UDS$WORK:[S000]S0002DA00.015	Time synchronization
UDS$WORK:[S000]S0002AA00.012	Channel 12 (X-axis)
UDS$WORK:[S000]S0002AA00.014	Channel 14 (Z-axis)

The examples to follow will be for data processed on the NHTSA VAX 11/780.

The remainder of this user's manual will assume that the reader has a working knowledge of the use of the NHTSA VAX 11/780 computer and the Digital Command Language (DCL). Program prompts will be shown as bold characters in this manual and user responses shown will be underlined.

3.1 PROGRAM EXECUTION

To begin the processing procedure the user must type:

RUN WGPROC

The program will then respond with:

```
*** WAVEFORM GENERATOR PROCESSING SOFTWARE ***

        10-MAR-88          11:27:51

DO YOU WANT ANALYSIS OF HALF-SINE AND CRASH PULSES IN THE PROCESSING? (Y or N)
>>Y

OPTION 1 : FINDING DATA POINT INDEX NUMBERS CORRESPONDING TO EDGES
OPTION 2 : ENTER CORRECT DATA POINT INDEX NUMBERS
ENTER DESIRED OPTION NUMBER (1/2)
>>>
```

Figure 4-1 (see Section 4) presents a flow diagram of the processing package. Not shown in Figure 4-1 is the first option provided to the user, the selection of full or partial processing. The option of eliminating the half-sine and crash pulse processing is provided so that when the appropriate group 1 and group 2 signals are not used as inputs to the data acquisition system under test, only the meaningful processing (RECTANGLE, STAIR, AND SUM-OF-SINES) will be performed.

Option 1 takes the time synchronization file and detects the rising or falling edge data point index numbers. If there are more than four edges for the time zero and inverted time zero data file types or more than two edges for the delayed time zero and inverted delayed time zero data file types, the program will warn the user of the occurrence of the extra edges and end program execution. The occurrence of extra edges stems primarily from two sources.
These sources are:

1) The waveform generator instrument produces a pre-trigger pulse prior to the output of the actual test waveforms sequence. If the time synchronization channel on the data acquisition system records the pre-trigger pulse, an extra edge will result.

2) Noise on the time synchronization channel on the data acquisition system can be of sufficient magnitude such that it is mistaken for extra edges.

In case of an occurrence of extra edges, the user should obtain a plot of the time synchronization file in order to ascertain the indices of the correct edges. (This can be done by using the "VTEKPLT" program which resides in the [ASGPROG] account.) Upon determining which of the indices represent the correct edges, the user can then re-execute the program and make use of Option 2. An example of the occurrence of extra edges is given by the data chosen for use in the following sample processing run.

Starting from the initial execution of the processing program (assuming the occurrence of extra edges is not known), the user enters:

```
RUN WGPROC

 *** WAVEFORM GENERATOR PROCESSING SOFTWARE ***


         10-MAR-88          11:27:51


DO YOU WANT ANALYSIS OF HALF-SINE AND CRASH PULSES IN THE PROCESSING? (Y or N)
>>Y


OPTION 1 : FINDING DATA POINT INDEX NUMBERS CORRESPONDING TO EDGES
OPTION 2 : ENTER CORRECT DATA POINT INDEX NUMBERS
ENTER DESIRED OPTION NUMBER (1/2)
>>> 1

  ENTER TIME SYNCHRONIZATION FILE NAME
>>UDS$WORK:[S000]S0002DA00.015
```

```
**NOTE : TIME ZERO HAS BEEN RECORDED**
 LEADING EDGE AT  -400 th DATA POINT
 LEADING EDGE AT     3 rd DATA POINT
 LEADING EDGE AT  1676 th DATA POINT
 LEADING EDGE AT  3350 th DATA POINT
 LEADING EDGE AT  5024 th DATA POINT

WARNING!
  ADDITIONAL EDGES IN TIME ZERO FILE HAVE BEEN DETECTED
  OBTAIN A PLOT OF TIME ZERO FILE
  AND USE OPTION 2 TO ENTER THE CORRECT DATA POINT NUMBERS


END OF PROCESSING RUN
```

Upon running the program "VTEKPLT" in the [ASGPROG] account and obtaining a plot of the time synchronization file, it becomes clear that the first edge is not an indicator for the leading edge of recorded data (see Figure 2-2). Knowing this, the user should then proceed as shown below:

```
RUN WGPROC
```

The program will then respond with:

```
*** WAVEFORM GENERATOR PROCESSING SOFTWARE ***


        10-MAR-88          11:29:22


DO YOU WANT ANALYSIS OF HALF-SINE AND CRASH PULSES IN THE PROCESSING? (Y or N)
>>Y
```

```
OPTION 1 : FINDING DATA POINT INDEX NUMBERS CORRESPONDING TO EDGES
OPTION 2 : ENTER CORRECT DATA POINT INDEX NUMBERS
ENTER DESIRED OPTION NUMBER (1/2)
>>> 2

ENTER  1  IF TIMING FILE IS A STANDARD TIME ZERO FILE
       2  IF TIMING FILE IS A DELAYED TIME ZERO FILE
       3  IF TIMING FILE IS AN INVERTED TIME ZERO FILE
       4  IF TIMING FILE IS A DELAYED & INVERTED TIME ZERO FILE
1

ENTER DATA POINT INDEX NUMBER  1
>>> 3

ENTER DATA POINT INDEX NUMBER  2
>>> 1676

ENTER DATA POINT INDEX NUMBER  3
>>> 3350

ENTER DATA POINT INDEX NUMBER  4
>>> 5024

**  BEGINNING OF INDIVIDUAL WAVEFORM SEPARATION  **


NOTE:  THE TOTAL NUMBER OF FILE NAMES YOU ENTER
       MUST BE A MULTIPLE OF TWO. THE FIRST, THIRD,
       FIFTH, etc., FILE MUST BE AN X-COMPONENT FILE
       AND THE SECOND, FOURTH, SIXTH, etc., FILE
       MUST BE A Z-COMPONENT FILE.
```

```
ENTER DATA FILE NAME (XXXXXXXXX.XXX)
IF NO MORE FILES, TYPE "CNTR Z"
>>UDS$WORK:[S000]S0002AA00.012

 DEL=  7.4686315E-05

ENTER DATA FILE NAME (XXXXXXXXX.XXX)
IF NO MORE FILES, TYPE "CNTR Z"
>>UDS$WORK:[S000]S0002AA00.014

 DEL=  7.4686315E-05

ENTER DATA FILE NAME (XXXXXXXXX.XXX)
IF NO MORE FILES, TYPE "CNTR Z"
>> Exit                              (system response to a "CNTR Z")


***   END OF PHASE 1   ***

TEST (  1) = 0002
TEST (  2) = 0002

PROCESSING SEQUENCE:
        RECTANGLE WAVEFORM
        STAIR WAVEFORM
        HALF-SINE WAVEFORM
        CRASH WAVEFORM
        SUM-OF-SINE WAVEFORM

**   RECTANGLE WAVEFORM PROCESSOR   **
PROCESSING COMPLETED FOR FILE S0002AA00.F12
PROCESSING COMPLETED FOR FILE S0002AA00.F14

**   STAIR WAVEFORM PROCESSOR   **
PROCESSING COMPLETED FOR FILE S0002AA00.E12
PROCESSING COMPLETED FOR FILE S0002AA00.E14
```

```
STEADY-STATE AMPLITUDE DEVIATIONS FROM BEST FIT STRAIGHT LINE (% FULL SCALE)
 DEV( 2)=  -0.184  DEV( 3)=   0.008  DEV( 4)=   0.057  DEV( 5)=   0.144
 DEV( 6)=   0.014  DEV( 7)=   0.095  DEV( 8)=   0.193  DEV( 9)=   0.140
 DEV(10)=  -0.155  DEV(11)=  -0.095  DEV(12)=  -0.217  DEV(
STEADY-STATE AMPLITUDE DEVIATIONS FROM BEST FIT STRAIGHT LINE (% FULL SCALE)
 DEV( 2)=  -0.076  DEV( 3)=  -0.004  DEV( 4)=   0.210  DEV( 5)=   0.034
 DEV( 6)=  -0.063  DEV( 7)=   0.148  DEV( 8)=   0.159  DEV( 9)=   0.116
 DEV(10)=  -0.098  DEV(11)=  -0.168  DEV(12)=  -0.258  DEV(

 **  HALF-SINE WAVEFORM PROCESSOR **

RESULT FILE FOR SET NUMBER    1
CALCULATING  *HIC*  FOR SET NUMBER          1
PROCESSING SET NUMBER            1

 ** CRASH WAVEFORM PROCESSOR **

RESULT FILE FOR SET NUMBER    1
CALCULATING  *HIC*  FOR SET NUMBER          1
PROCESSING SET NUMBER            1

 ** SUM-OF-SINE WAVEFORM PROCESSOR **

PROCESSING COMPLETED FOR FILE      S0002AA00.V12
PROCESSING COMPLETED FOR FILE      S0002AA00.V14


END OF PROCESSING RUN
```

As indicated above, once the user specifies all of the files to be processed, there is no further interaction with the user. One point should be noted. For the processing of the rectangular, staircase and sum-of-sines waveforms, the program indicates that processing is completed for files which the user did not specify. The file names which are indicated are hold-overs from a previous version of the processing software. In the previous version, each waveform of each channel was separated from the others of the channel

and placed into a separate file. This feature was removed from the software in order to reduce the number of calls to the system storage devices and the processing time. The file names no longer have any physical meaning outside of the program. The use of the file names has been retained in order to allow the user to track the progress of the program. These "files" can now be thought of as "pseudo-files."

3.2 SUMMARY REPORTS AND STATISTICAL DATA FILES

During the execution of the program, a number of files are generated. The files which are generated can be divided into two groups. The first group is the summary files. The summary files are files which report on the quality of the data in terms of the "properties" investigated in a clear formatted manner. The second group are the statistical data files which contain information which can later be utilized in other ways such as in statistical programs. (The other statistical programs are not part of the signal processing software package.) These statistical data files are generally not in a format which offers clearly meaningful data just by visual inspection. Unless the user intends to conduct a statistical analysis on a number of program runs, all questions relating to the quality of the recorded data can be answered from the summary files.

In Table 3-1, the names of the summary and statistical data files generated during program execution are presented. Also presented in this table is a brief description of the contents of each of the files. Examples of these files are presented following the table. It should be noted that in order to obtain easily readable summary reports, the user must have access to a device (terminal or printer) which can operate in 132-column mode.

Table 3-1 Files Generated During the Execution of WGPROC

Waveform	File name	Description
Rectangle	REC####.SUM	Summary report
	REC####.001	Data file containing time deviation from theoretical time
	REC####.002	Data file containing time linearity slope and time offset intercept
	REC####.003	Data file containing steady-state amplitude deviation from theoretical amplitude
	REC####.004	Data file containing amplitude overshoot relative to calculated steady-state amplitude
Staircase	STAIR####.SUM	Summary report
	STAIR####.001	Data file containing time deviation from theoretical time
	STAIR####.002	Data file containing time linearity slope and time offset intercept
	STAIR####.003	Data file containing amplitude linearity slope and amplitude offset intercept
	STAIR####.004	Data file containing steady-state amplitude deviation from theoretical amplitude
	STAIR####.005	Data file containing amplitude overshoot relative to calculated steady-state amplitude
Half-sine	HSNE####.SUM	Summary report
	HSNE####.001	Data file containing HIC deviation
	HSNE####.002	Data file containing time deviation of X and Z components from theoretical
Crash	CRASH####.SUM	Summary report
	CRASH####.001	Data file containing HIC deviation
Sum-of-sines	SUMSN####.SUM	Summary report
	SUMSN####.001	Data file containing frequency of actual signal and amplitude ratio
	FR####.Vnn	Frequency response plot data files
Half-sine and Crash	RESULT.SUM	File listing waveform component combinations used in the calculation of the X-Z resultants

Note: #### is the test number assigned to the data by NHTSA. This corresponds to the four-digit sequences displayed after the "END OF PHASE 1" message given during the execution of WGPROC. nn is the channel number of the data. This corresponds to the last two digits in the extension of the input data file names.

***** REC0002.SUM *****

** RECTANGLE WAVEFORM PROCESSOR DATA SUMMARY **

A VALUE OF +/- 10000 INDICATES THAT THE QUANTITY COULD NOT BE DETERMINED
THE * SYMBOL INDICATES THAT THE ALLOWABLE LIMIT HAS BEEN EXCEEDED.

TIME QUANTITIES EXPRESSED IN MILLISECONDS
AMPLITUDE QUANTITIES EXPRESSED AS PERCENT OF FULL SCALE

10-MAR-88 11:31:33

RESULTS FOR FILE : S0002AA00.F12 CHANNEL NO. 12

FACILITY : CALSPAN

1. TIME DEVIATION FROM THEORETICAL TIME (ALLOWABLE LIMIT = +/- 1.0 MSEC)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
0.000	-0.025	0.008	-0.017	0.016	-0.009	0.024	-0.001	0.032	0.007

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
0.040	0.015	0.048	0.023	0.056	0.031	0.064	0.039	-0.003	-0.028

2. TIME LINEARITY (ALLOWABLE LIMIT = +/- 1 %)

SLOPE = 0.03

3. TIME OFFSET (ALLOWABLE LIMIT = +/- 1.0 MSEC)

INTERCEPT = 0.00

4. STEADY-STATE AMPLITUDE DEVIATION FROM THEORETICAL AMPLITUDE (ALLOWABLE LIMIT = 2.5 %)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
-0.455	-0.096	-0.642	-0.027	-0.605	-0.131	-0.680	-0.061	-0.493	-0.096

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
-0.567	-0.061	-0.642	0.043	-0.399	-0.165	-0.474	-0.051	-0.549	-0.089

Figure 3-3 RECTANGLE WAVEFORM Sample Summary Report and Statistical Data Files

5. AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED STEADY-STATE AMPLITUDE (ALLOWABLE LIMIT +/- 13 %)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
9.147	8.331	8.595	8.823	8.572	8.356	8.356	8.321	9.036	8.333

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
8.555	8.317	8.857	8.524	8.457	8.627	8.507	8.308	8.806	9.116

RESULTS FOR FILE : S0002AA00.F14 CHANNEL NO. 14

FACILITY : CALSPAN

1. TIME DEVIATION FROM THEORETICAL TIME (ALLOWABLE LIMIT = +/- 1.0 MSEC)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
0.000	-0.025	0.008	-0.017	0.016	-0.009	0.024	-0.001	0.032	0.007

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
0.040	0.015	0.048	0.023	0.056	0.031	0.064	0.039	0.072	-0.028

2. TIME LINEARITY (ALLOWABLE LIMIT = +/- 1 %)

SLOPE = 0.05

3. TIME OFFSET (ALLOWABLE LIMIT = +/- 1.0 MSEC)

INTERCEPT = -0.01

4. STEADY-STATE AMPLITUDE DEVIATION FROM THEORETICAL AMPLITUDE (ALLOWABLE LIMIT = 2.5 %)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
-0.443	0.110	-0.667	0.250	-0.611	0.145	-0.705	0.215	-0.331	0.180

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
-0.536	0.285	-0.780	0.390	-0.574	0.250	-0.461	0.285	-0.583	0.285

Figure 3-3 continued

5. AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED STEADY-STATE AMPLITUDE (ALLOWABLE LIMIT +/- 13 %)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
9.048	8.477	8.304	8.673	8.531	8.465	8.323	8.431	8.640	8.434

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
8.230	8.386	8.358	8.602	8.505	8.406	8.187	8.382	8.251	8.913

6. CHANNEL-TO-CHANNEL TIME DIFFERENCE - ALL CHANNELS ARE COMPARED TO THE FIRST CHANNEL PROCESSED (ALLOWABLE LIMIT = +/- 0.25 MSEC)

RESULTS FOR FILE : S0002AA00.F12 CHANNEL NO. 12

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

RESULTS FOR FILE : S0002AA00.F14 CHANNEL NO. 14

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

INTERVAL NUMBER

11	12	13	14	15	16	17	18	19	20
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.075	0.000

FILE NAME	CHANNEL NO.	STATUS
S0002AA00.F12	12	PASS
S0002AA00.F14	14	PASS

Figure 3-3 continued

```
   TOTAL NO. OF CHANNELS PROCESSED =  2
   NUMBER OF ACCEPTABLE CHANNELS =  2
NUMBER OF UNACCEPTABLE CHANNELS =  0


*****   REC0002.001   *****

      0.000    -0.025     0.008    -0.017     0.016    -0.009     0.024    -0.001     0.032     0.007
      0.040     0.015     0.048     0.023     0.056     0.031     0.064     0.039    -0.003    -0.028
      0.000    -0.025     0.008    -0.017     0.016    -0.009     0.024    -0.001     0.032     0.007
      0.040     0.015     0.048     0.023     0.056     0.031     0.064     0.039     0.072    -0.028


*****   REC0002.002   *****

   0.03       0.00
   0.05      -0.01


*****   REC0002.003   *****

     -0.455    -0.096    -0.642    -0.027    -0.605    -0.131    -0.680    -0.061    -0.493    -0.096
     -0.567    -0.061    -0.642     0.043    -0.399    -0.165    -0.474    -0.051    -0.549    -0.089
     -0.443     0.110    -0.667     0.250    -0.611     0.145    -0.705     0.215    -0.331     0.180
     -0.536     0.285    -0.780     0.390    -0.574     0.250    -0.461     0.285    -0.583     0.285


*****   REC0002.004   *****

      9.147     8.331     8.595     8.823     8.572     8.356     8.356     8.321     9.036     8.333
      8.555     8.317     8.857     8.524     8.457     8.627     8.507     8.308     8.806     9.116
      9.048     8.477     8.304     8.673     8.531     8.465     8.323     8.431     8.640     8.434
      8.230     8.386     8.358     8.602     8.505     8.406     8.187     8.382     8.251     8.913
```

Figure 3-3 continued

***** STAIR0002.SUM *****

** STAIR WAVEFORM PROCESSOR DATA SUMMARY **

A VALUE OF +/- 10000 INDICATES THAT THE QUANTITY COULD.NOT BE DETERMINED
THE * SYMBOL INDICATES THAT THE ALLOWABLE LIMIT HAS BEEN EXCEEDED.

TIME QUANTITIES EXPRESSED IN MILLISECONDS
AMPLITUDE QUANTITIES EXPRESSED AS PERCENT OF FULL SCALE

10-MAR-88 11:31:41

RESULTS FOR FILE : S0002AA00.E12 CHANNEL NO. 12

FACILITY : CALSPAN

1. TIME DEVIATION FROM THEORETICAL TIME (ALLOWABLE LIMIT +/- 1.0 MSEC)

INTERVAL NUMBER											
1	2	3	4	5	6	7	8	9	10	11	12
0.008	0.016	0.024	0.032	0.040	0.048	-0.019	-0.011	-0.003	0.005	0.013	0.02

2. TIME LINEARITY (ALLOWABLE LIMIT = +/- 1 %)

SLOPE = -0.01

3. TIME OFFSET (ALLOWABLE LIMIT = +/- 1.0 MSEC)

INTERCEPT = 0.02

4. AMPLITUDE LINEARITY (ALLOWABLE LIMIT = +/- 2.5 %)

SLOPE = -0.27

5. AMPLITUDE OFFSET (ALLOWABLE LIMIT = +/- 2.5 %)

INTERCEPT = -0.29

6. STEADY-STATE AMPLITUDE DEVIATION FROM THEORETICAL AMPLITUDE (ALLOWABLE LIMIT = +/- 2.5 %)

INTERVAL NUMBER											
1	2	3	4	5	6	7	8	9	10	11	12
-0.291	-0.534	-0.397	-0.402	-0.369	-0.555	-0.199	-0.172	-0.064	0.286	0.281	0.45

7. AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED STEADY-STATE AMPLITUDE (ALLOWABLE LIMIT +/- 13 %)

INTERVAL NUMBER											
1	2	3	4	5	6	7	8	9	10	11	12
7.162	8.209	10.160	9.256	9.536	9.777	8.885	7.822	8.936	8.023	10.878	9.66

Figure 3-4 STAIR WAVEFORM Sample Summary Report and Statistical Data Files

RESULTS FOR FILE : S0002AA00.E14 CHANNEL NO. 14

FACILITY : CALSPAN

1. TIME DEVIATION FROM THEORETICAL TIME (ALLOWABLE LIMIT +/- 1.0 MSEC)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10	11	12
0.008	0.016	0.024	0.032	0.040	0.048	-0.019	-0.011	-0.003	0.005	0.013	0.02

2. TIME LINEARITY (ALLOWABLE LIMIT = +/- 1 %)

SLOPE = -0.01

3. TIME OFFSET (ALLOWABLE LIMIT = +/- 1.0 MSEC)

INTERCEPT = 0.02

4. AMPLITUDE LINEARITY (ALLOWABLE LIMIT = +/- 2.5 %)

SLOPE = -0.05

5. AMPLITUDE OFFSET (ALLOWABLE LIMIT = +/- 2.5 %)

INTERCEPT = -0.39

6. STEADY-STATE AMPLITUDE DEVIATION FROM THEORETICAL AMPLITUDE (ALLOWABLE LIMIT = +/- 2.5 %)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10	11	12
-0.185	-0.475	-0.413	-0.210	-0.395	-0.502	-0.241	0.180	0.233	0.457	0.537	0.63

7. AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED STEADY-STATE AMPLITUDE (ALLOWABLE LIMIT +/- 13 %)

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10	11	12
8.258	7.259	8.239	8.808	8.758	9.514	9.010	9.689	7.698	10.127	10.092	7.46

8. CHANNEL-TO-CHANNEL TIME DIFFERENCE, ALL CHANNELS ARE COMPARED TO THE FIRST CHANNEL PROCESSED (ALLOWABLE LIMIT = +/- 0.25 MSEC)

RESULTS FOR FILE : S0002AA00.E12 CHANNEL NO. 12

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10	11	12
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.00

Figure 3-4 continued

RESULTS FOR FILE : S0002AA00.E14 CHANNEL NO. 14

INTERVAL NUMBER

1	2	3	4	5	6	7	8	9	10	11	12
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.00

FILE NAME	CHANNEL NO.	STATUS
S0002AA00.E12	12	PASS
S0002AA00.E14	14	PASS

TOTAL NO. OF CHANNELS PROCESSED = 2
NUMBER OF ACCEPTABLE CHANNELS = 2
NUMBER OF UNACCEPTABLE CHANNELS = 0

***** STAIR0002.001 *****

0.008	0.016	0.024	0.032	0.040	0.048	-0.019	-0.011	-0.003	0.005	0.013	0.02
0.008	0.016	0.024	0.032	0.040	0.048	-0.019	-0.011	-0.003	0.005	0.013	0.02

***** STAIR0002.002 *****

-0.01	0.02
-0.01	0.02

***** STAIR0002.003 *****

-0.27	-0.29
-0.05	-0.39

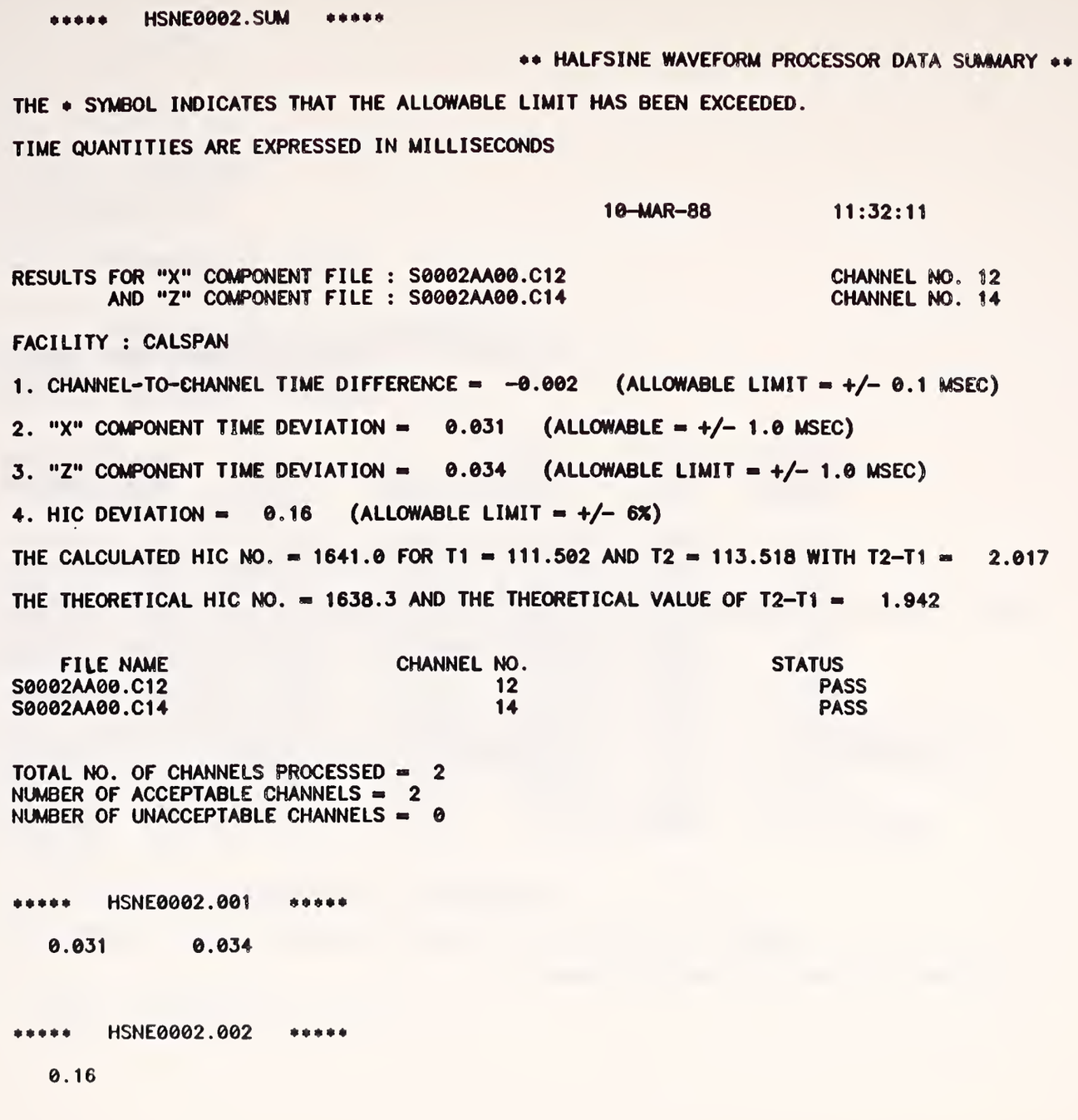
***** STAIR0002.004 *****

-0.291	-0.534	-0.397	-0.402	-0.369	-0.555	-0.199	-0.172	-0.064	0.286	0.281	0.45
-0.185	-0.475	-0.413	-0.210	-0.395	-0.502	-0.241	0.180	0.233	0.457	0.537	0.63

***** STAIR0002.005 *****

7.162	8.209	10.160	9.256	9.536	9.777	8.885	7.822	8.936	8.023	10.878	9.66
8.258	7.259	8.239	8.808	8.758	9.514	9.010	9.689	7.698	10.127	10.092	7.46

Figure 3-4 continued



```
*****   HSNE0002.SUM   *****

                                        ** HALFSINE WAVEFORM PROCESSOR DATA SUMMARY **

THE * SYMBOL INDICATES THAT THE ALLOWABLE LIMIT HAS BEEN EXCEEDED.

TIME QUANTITIES ARE EXPRESSED IN MILLISECONDS

                                                10-MAR-88          11:32:11

RESULTS FOR "X" COMPONENT FILE : S0002AA00.C12                    CHANNEL NO. 12
        AND "Z" COMPONENT FILE : S0002AA00.C14                    CHANNEL NO. 14

FACILITY : CALSPAN

1. CHANNEL-TO-CHANNEL TIME DIFFERENCE =  -0.002   (ALLOWABLE LIMIT = +/- 0.1 MSEC)

2. "X" COMPONENT TIME DEVIATION =   0.031   (ALLOWABLE = +/- 1.0 MSEC)

3. "Z" COMPONENT TIME DEVIATION =   0.034   (ALLOWABLE LIMIT = +/- 1.0 MSEC)

4. HIC DEVIATION =   0.16   (ALLOWABLE LIMIT = +/- 6%)

THE CALCULATED HIC NO. = 1641.0 FOR T1 = 111.502 AND T2 = 113.518 WITH T2-T1 =   2.017

THE THEORETICAL HIC NO. = 1638.3 AND THE THEORETICAL VALUE OF T2-T1 =   1.942

    FILE NAME                  CHANNEL NO.                  STATUS
S0002AA00.C12                        12                      PASS
S0002AA00.C14                        14                      PASS

TOTAL NO. OF CHANNELS PROCESSED =  2
NUMBER OF ACCEPTABLE CHANNELS =  2
NUMBER OF UNACCEPTABLE CHANNELS =  0


*****   HSNE0002.001   *****

   0.031       0.034


*****   HSNE0002.002   *****

   0.16
```

Figure 3-5 HALF-SINE WAVEFORM Sample Summary Report and Statistical Data Files

```
*****   CRASH0002.SUM   *****

                                        ** CRASH WAVEFORM PROCESSOR DATA SUMMARY **

THE * SYMBOL INDICATES THAT THE ALLOWABLE LIMIT HAS BEEN EXCEEDED.

TIME QUANTITIES ARE EXPRESSED IN MILLISECONDS

                                                10-MAR-88          11:32:22

RESULTS FOR "X" COMPONENT FILE : S0002AA00.D12                    CHANNEL NO. 12
        AND "Z" COMPONENT FILE : S0002AA00.D14                    CHANNEL NO. 14

FACILITY : CALSPAN

1. HIC DEVIATION =       -0.43    (ALLOWABLE LIMIT = +/- 6%)

THE CALCULATED HIC NO. =  926.0 FOR T1 = 459.246 AND T2 = 471.047 WITH T2-T1 =  11.800

THE THEORETICAL HIC NO. =  930.0

    FILE NAME                   CHANNEL NO.                   STATUS
S0002AA00.D12                         12                        PASS
S0002AA00.D14                         14                        PASS

TOTAL NO. OF CHANNELS PROCESSED =  2
NUMBER OF ACCEPTABLE CHANNELS =  2
NUMBER OF UNACCEPTABLE CHANNELS =  0


*****   CRASH0002.001   *****

      -0.43
```

Figure 3-6 CRASH WAVEFORM Sample Summary Report and Statistical Data Files

***** SUMSN0002.SUM *****

10-MAR-88 11:32:26

RESULT FOR FILE S0002AA00.V12
FREQUENCY RESPONSE FILE FR0002.V12

SIGNAL FREQUENCY (HZ)	DFT LINE NO.	DFT OUTPUT FREQUENCY	INPUT DFT VALUE	OUTPUT DFT VALUE	SCALED DFT RATIO	AMPLITUDE RATIO (DB)
273.4375	42	274.5859	116.4118	2239.0781	0.99300	-0.06
546.8750	84	549.1719	115.9737	2152.7881	0.95834	-0.37
820.3125	125	817.2201	115.5020	2176.7891	0.97298	-0.24
1093.7500	167	1091.8060	116.1409	2108.5037	0.93727	-0.56
1367.1875	209	1366.3920	116.4876	1987.4216	0.88082	-1.10
1640.6250	251	1640.9779	116.5444	1672.7551	0.74100	-2.60
1914.0625	293	1915.5638	116.3090	1230.7350	0.54630	-5.25
2187.5000	335	2190.1499	115.7811	810.4747	0.36139	-8.84
2460.9375	376	2458.1980	115.7289	506.3800	0.22590	-12.92
2734.3750	418	2732.7839	116.2774	330.1055	0.14657	-16.68
3007.8125	460	3007.3699	116.5355	232.7119	0.10309	-19.74
3281.2500	502	3281.9558	116.5022	150.9221	0.06688	-23.49
3554.6875	544	3556.5417	116.1769	99.2410	0.04410	-27.11
3828.1250	586	3831.1277	115.5613	80.4261	0.03593	-28.89

Figure 3-7 SUM-OF-SINES WAVEFORM

Sample Summary Report and Statistical Data Files

RESULT FOR FILE S0002AA00.V14
FREQUENCY RESPONSE FILE FR0002.V14

SIGNAL FREQUENCY (HZ)	DFT LINE NO.	DFT OUTPUT FREQUENCY	INPUT DFT VALUE	OUTPUT DFT VALUE	SCALED DFT RATIO	AMPLITUDE RATIO (DB)
273.4375	42	274.5859	116.4118	2229.0823	0.99300	-0.06
546.8750	84	549.1719	115.9737	2168.9026	0.96984	-0.27
820.3125	125	817.2201	115.5020	2170.0696	0.97433	-0.23
1093.7500	167	1091.8060	116.1409	2107.5266	0.94104	-0.53
1367.1875	209	1366.3920	116.4876	1965.8407	0.87516	-1.16
1640.6250	251	1640.9779	116.5444	1637.4161	0.72860	-2.75
1914.0625	293	1915.5638	116.3090	1191.5800	0.53129	-5.49
2187.5000	335	2190.1499	115.7811	770.1146	0.34494	-9.25
2460.9375	376	2458.1980	115.7289	502.5081	0.22518	-12.95
2734.3750	418	2732.7839	116.2774	334.4904	0.14918	-16.53
3007.8125	460	3007.3699	116.5355	221.5809	0.09860	-20.12
3281.2500	502	3281.9558	116.5022	154.1666	0.06862	-23.27
3554.6875	544	3556.5417	116.1769	104.2890	0.04655	-26.64
3828.1250	586	3831.1277	115.5613	81.4986	0.03657	-28.74

***** SUMSN0002.001 *****

-0.0610 -0.3696 -0.2379 -0.5627 -1.1023 -2.6036 -5.2514 -8.8404 -12.9218 -16.6793 -19.7353 -23.4941 -27.1110 -28.8908
-0.0610 -0.2660 -0.2259 -0.5278 -1.1582 -2.7502 -5.4934 -9.2452 -12.9496 -16.5258 -20.1221 -23.2705 -26.6412 -28.7368

***** FR0002.V12 *****

0.27344E+03 -0.61015E-01
0.54688E+03 -0.36962E+00
0.82031E+03 -0.23793E+00
0.10938E+04 -0.56268E+00
0.13672E+04 -0.11023E+01
0.16406E+04 -0.26036E+01
0.19141E+04 -0.52514E+01
0.21875E+04 -0.88404E+01
0.24609E+04 -0.12922E+02
0.27344E+04 -0.16679E+02
0.30078E+04 -0.19735E+02
0.32813E+04 -0.23494E+02
0.35547E+04 -0.27111E+02
0.38281E+04 -0.28891E+02

Figure 3-7 continued

```
*****   FR0002.V14   *****

 0.27344E+03 -0.61016E-01
 0.54688E+03 -0.26598E+00
 0.82031E+03 -0.22592E+00
 0.10938E+04 -0.52784E+00
 0.13672E+04 -0.11582E+01
 0.16406E+04 -0.27502E+01
 0.19141E+04 -0.54934E+01
 0.21875E+04 -0.92452E+01
 0.24609E+04 -0.12950E+02
 0.27344E+04 -0.16526E+02
 0.30078E+04 -0.20122E+02
 0.32813E+04 -0.23270E+02
 0.35547E+04 -0.26641E+02
 0.38281E+04 -0.28737E+02
```

Figure 3-7 continued

```
*****   RESULT.SUM   *****

                        ** RESULTANT SUMMARY REPORT **

SET NO :      1
INPUT FILES(X AND Z COMPONENT) :      S0002AA00.C12                S0002AA00.C14
RESULTANT FILE :      S0002AA00.RES

                        ** RESULTANT SUMMARY REPORT **

SET NO :      1
INPUT FILES(X AND Z COMPONENT) :      S0002AA00.D12                S0002AA00.D14
RESULTANT FILE :      S0002AA00.RES
```

Figure 3-8 Resultant Summary Report

4. SOFTWARE MODULE DESCRIPTIONS

This section presents a narrative description and flow diagram for each program module incorporated in the waveform signal processing software package.

4.1 PROGRAM "WGPROC"

"WGPROC" is the main program module for the signal processing software package. Its main purpose is to interact with the user and, based upon the response of the user, control the overall flow of the processing procedure. The flow diagram of this module is shown in Figure 4-1. Not shown in Figure 4-1 is the option to select full or partial processing.

4.2 SUBROUTINE "LED"

Subroutine "LED" contains the routine which is used to detect the type of time synchronization file. If the time synchronization file is either a standard time zero file or a delayed time zero file, subroutine "VWGEN1" is called. If the time synchronization file is inverted (either a full or delayed time zero file), subroutine "VWGEN2" is called. The flow diagram of this module is shown in Figure 4-2.

4.3 SUBROUTINES "VWGEN1" AND "VWGEN2"

Subroutines "VWGEN1" and "VWGEN2" are used to detect the data point index number corresponding to the edges in the time synchronization file. ("VWGEN1" is used to detect the leading edges and "VWGEN2" is used to detect the trailing edges.) As indicated in Section 4.2, "VWGEN1" is used for the standard and delayed time zero files and "VWGEN2" is used for the inverted standard and inverted delayed time zero files. The flow diagrams for "VWGEN1" and "VWGEN2" are shown in Figures 4-3 and 4-4 respectively.

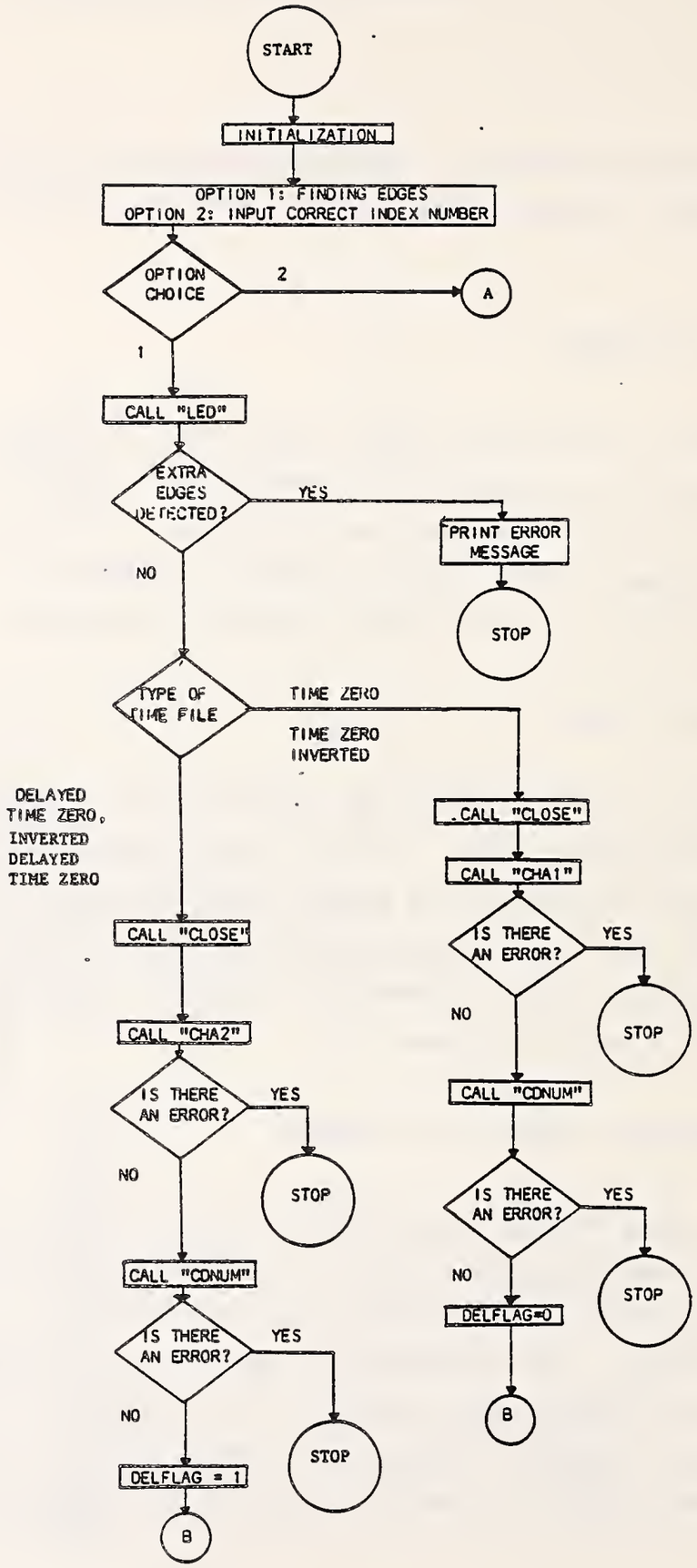


Figure 4-1 Flow Diagram of "WGPROC"

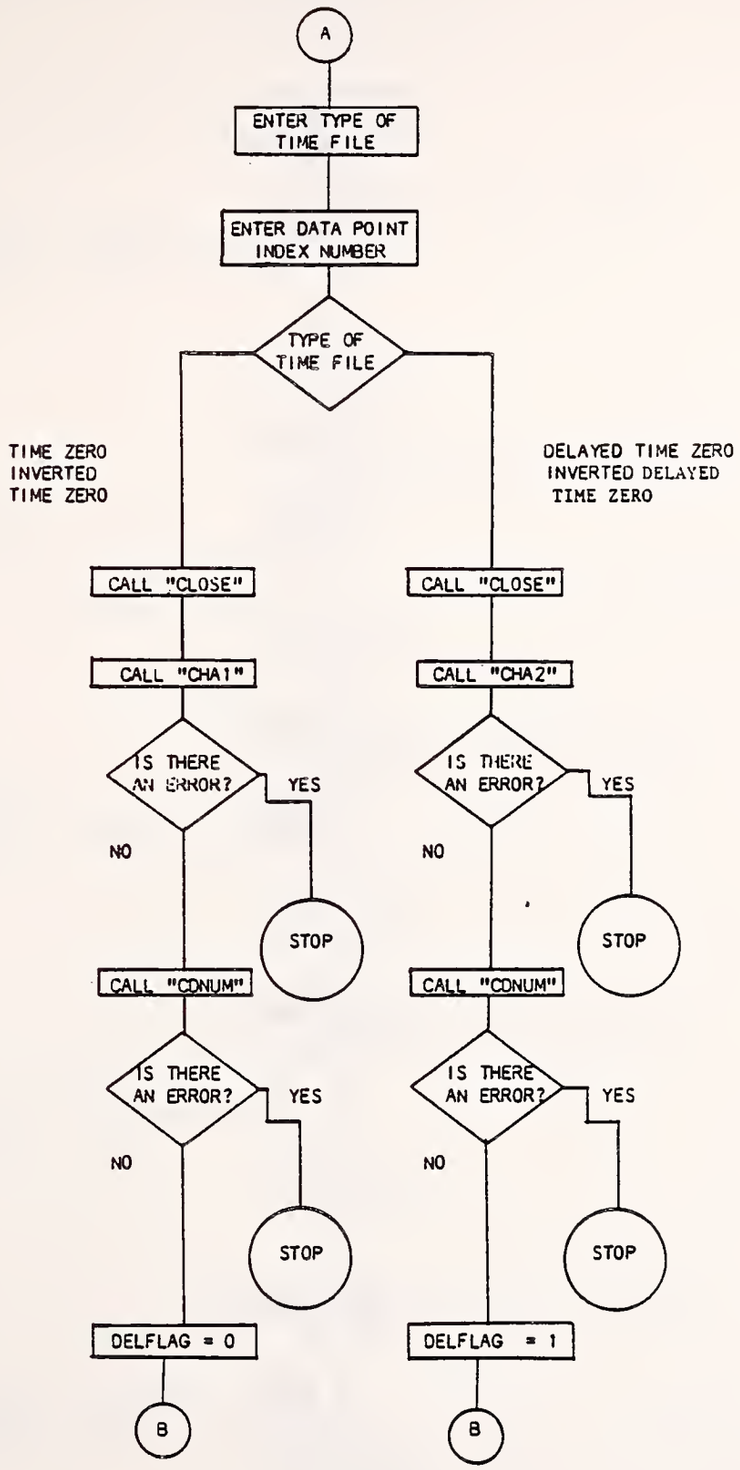


Figure 4-1 continued

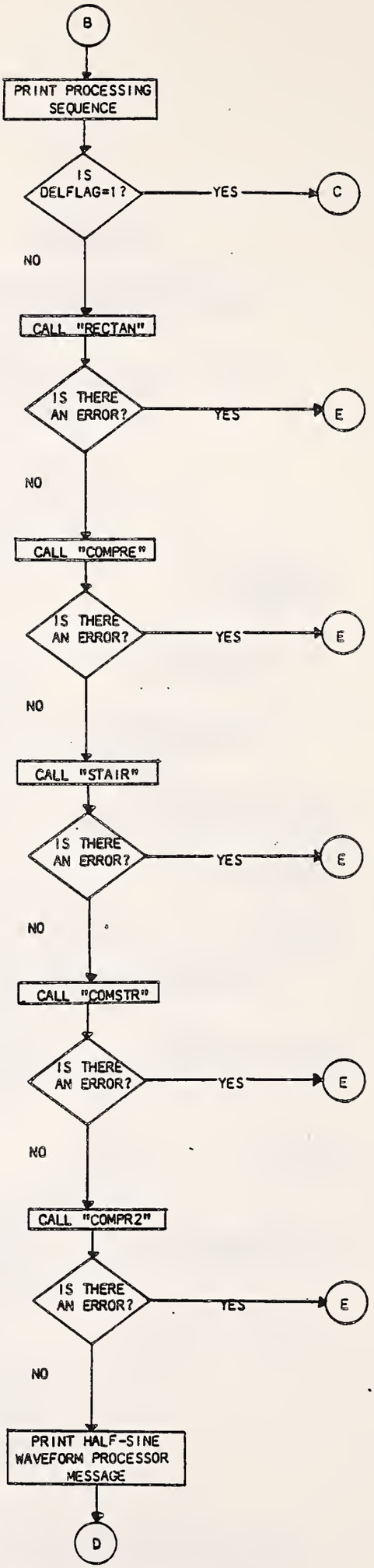


Figure 4-1 continued

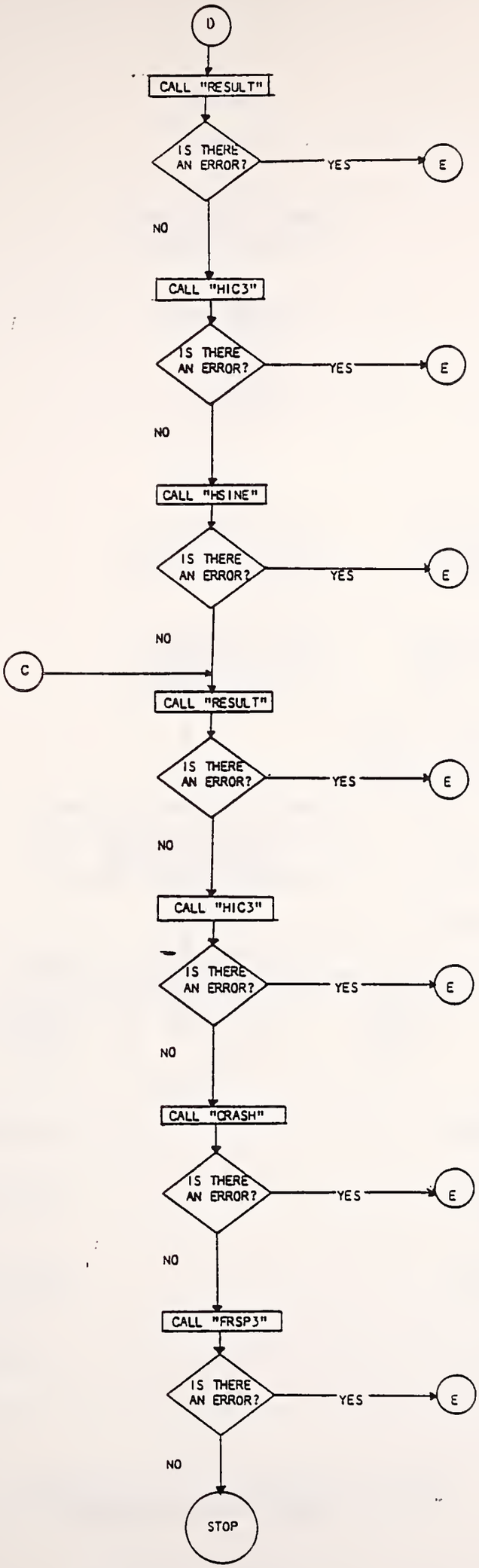


Figure 4-1 continued

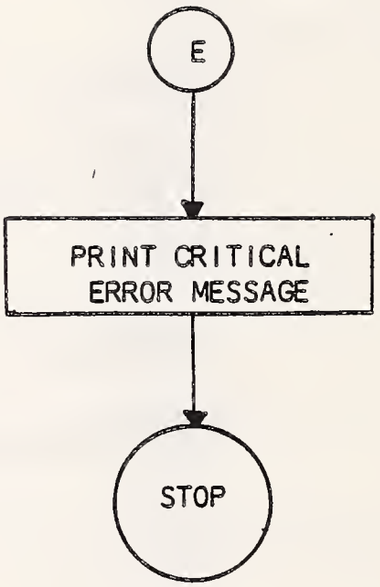


Figure 4-1 continued

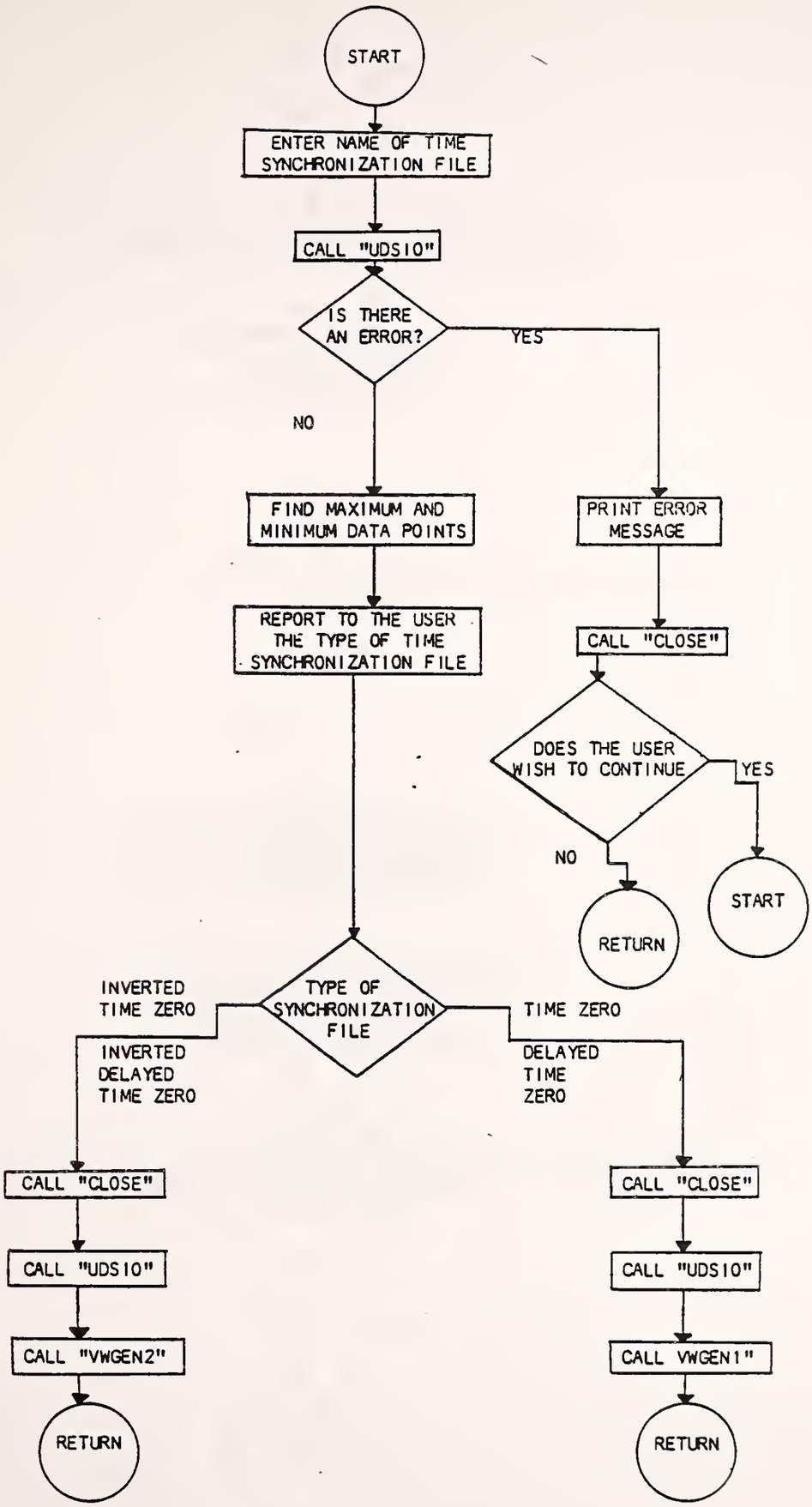


Figure 4-2 Flow Diagram of "LED"

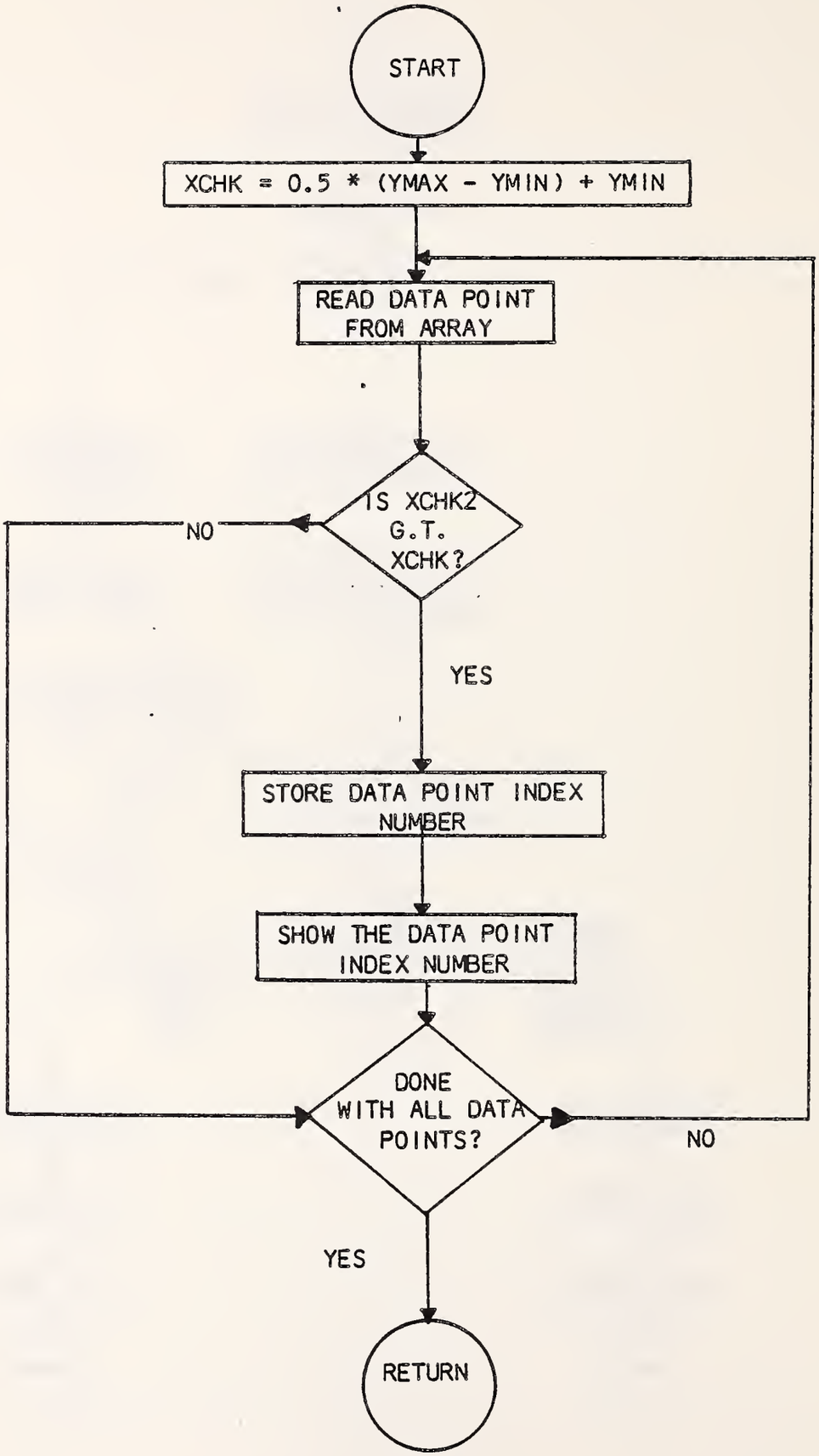


Figure 4-3 Flow Diagram of "VWGEN1"

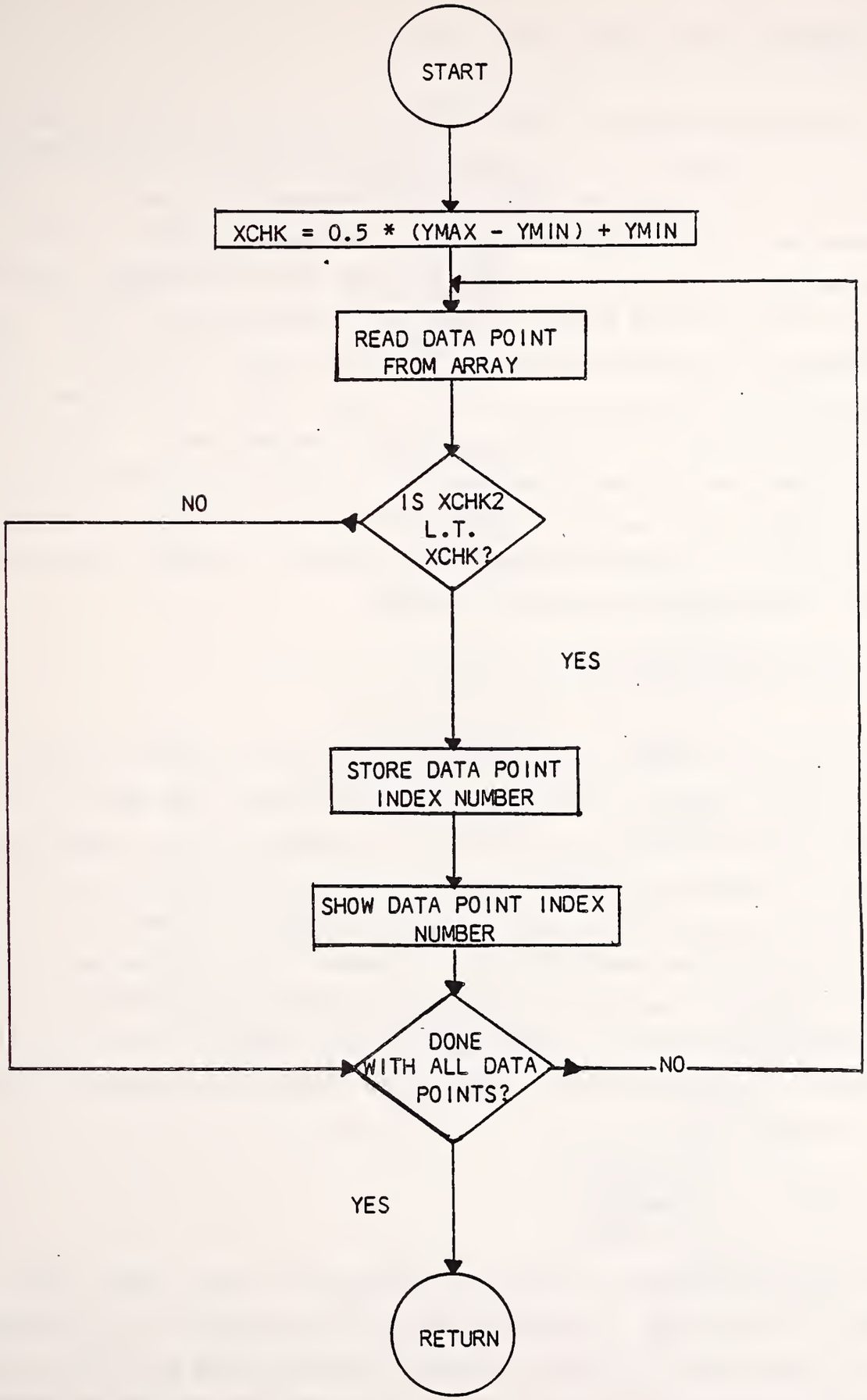


Figure 4-4 Flow Diagram of "VWGEN2"

4.4 SUBROUTINES "CHA1" AND "CHA2"

Subroutines "CHA1" and "CHA2" are used to obtain the file names for the data to be processed and to mark the starting points of the individual waveforms. In previous versions of the processing software, the individual waveforms were physically separated into individual files. In this version of the software, the data is stored in large arrays in memory with the index numbers of the starting points marking the "separations." "CHA1" is used for the standard and inverted time zero files and "CHA2" is used for the delayed standard and inverted delayed time zero files. The flow diagrams for "CHA1" and "CHA2" are shown in Figures 4-5 and 4-6 respectively. The E1(I) used in CHA1 and CHA2 are the array indices of the data points in the time reference waveform that first exceed one-half the distance between the minimum and maximum values of that waveform for a positive (negative) going edge in a standard (inverted) time reference waveform.

4.5 SUBROUTINES "OPNFIL" AND "OPNFIL1"

Subroutines "OPNFIL" and "OPNFIL1" are used to create the names for the "pseudo-files" which are displayed in order to allow the user to monitor the progress of the processing procedure. The names for the "pseudo-files" are created by changing the first character of the extension of the data channel file name (entered by the user in either "CHA1" or "CHA2") to a character which represents the waveform being processed. (The characters used are "F" for the rectangle waveform; "C" for the half-sine waveform; "E" for the staircase waveform; "V" for the sum-of-sines waveform; and "D" for the crash waveform.) Flow diagrams for "OPNFIL" and "OPNFIL1" are shown in Figures 4-7 and 4-8 respectively.

4.6 SUBROUTINE "CDNUM"

Subroutine "CDNUM" is used to extract the test number from the data channel file name and to ensure that all of the files to be processed are from the same test. If data files are specified from more than one test, a message is given and a flag is set which causes the program to cease execution. The flow diagram of this module is shown in Figure 4-9. In this

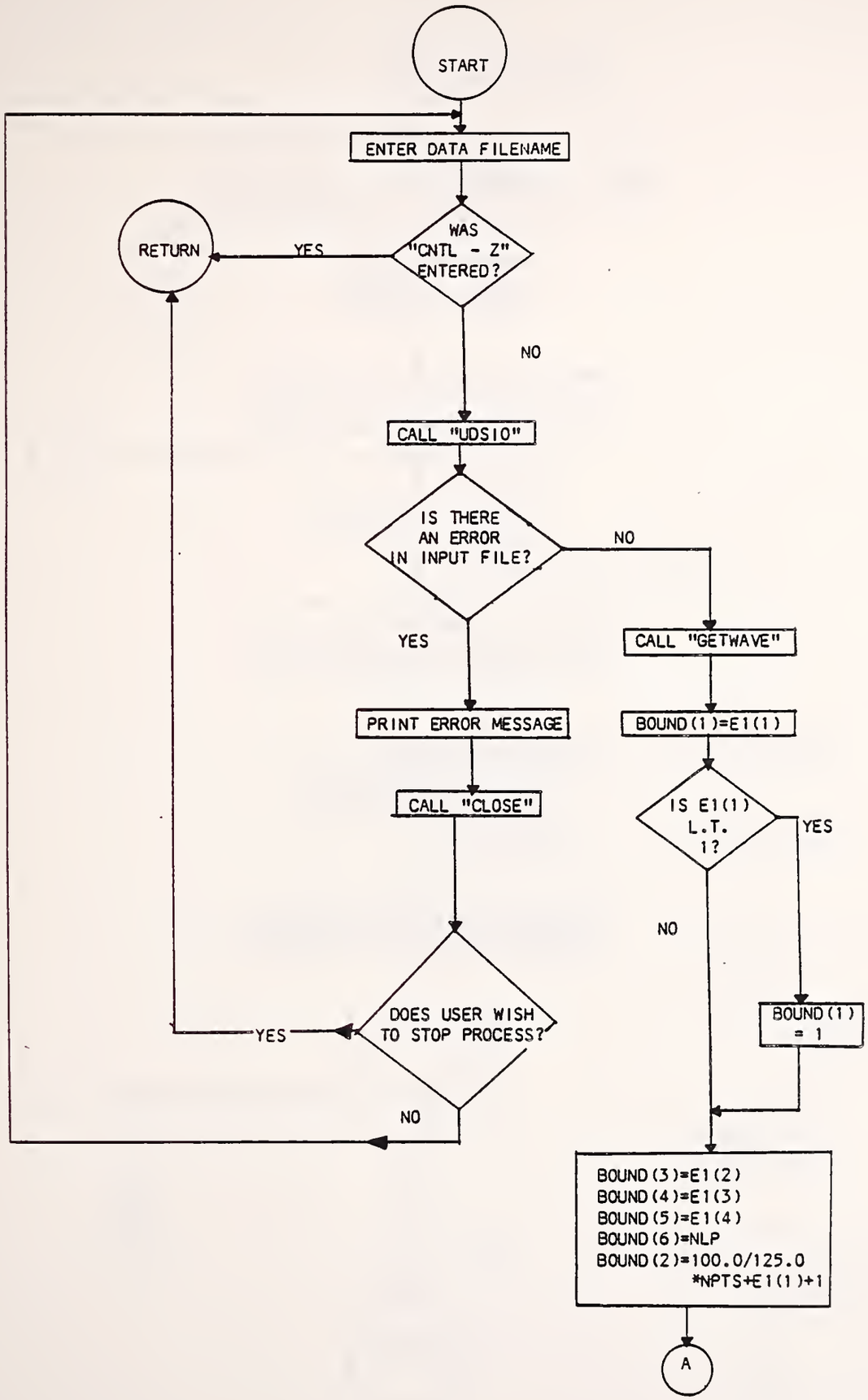


Figure 4-5 Flow Diagram of "CHA1"

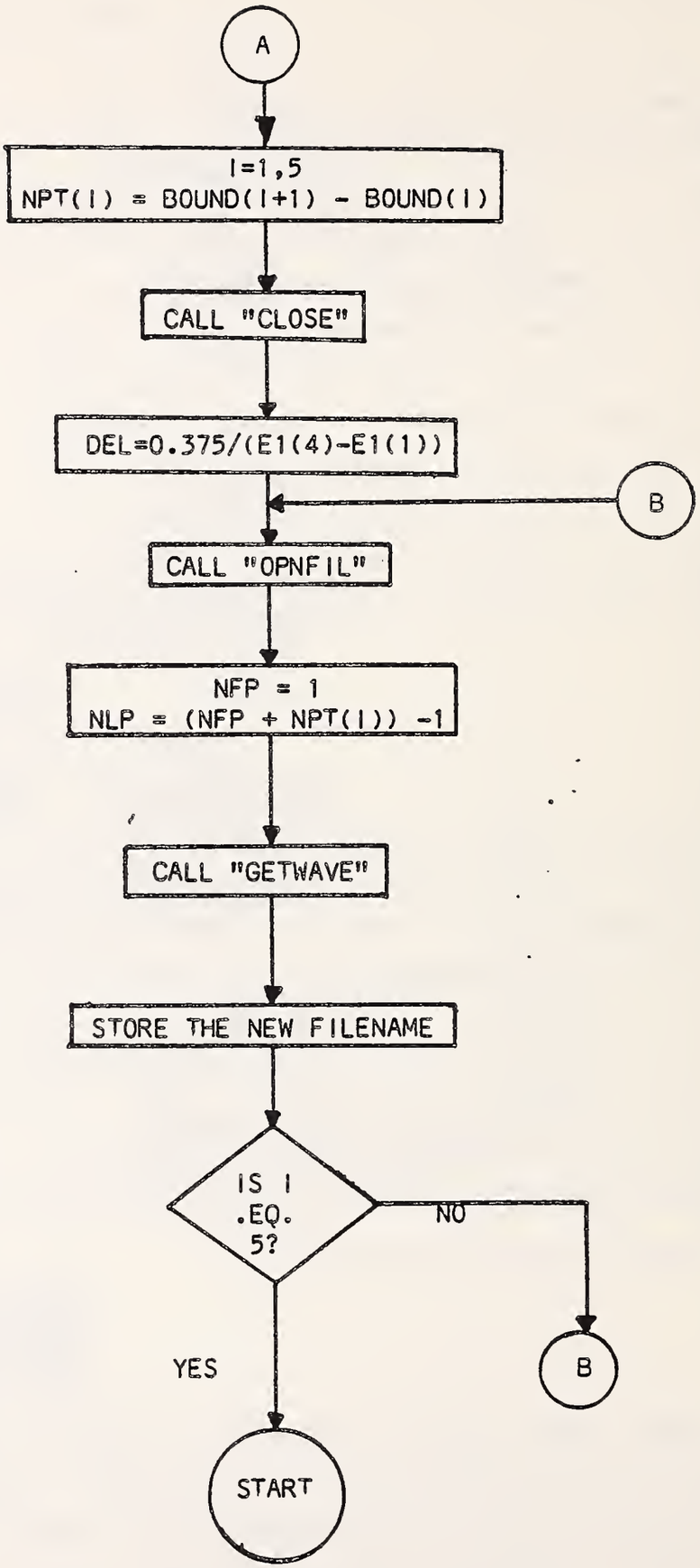


Figure 4-5 continued

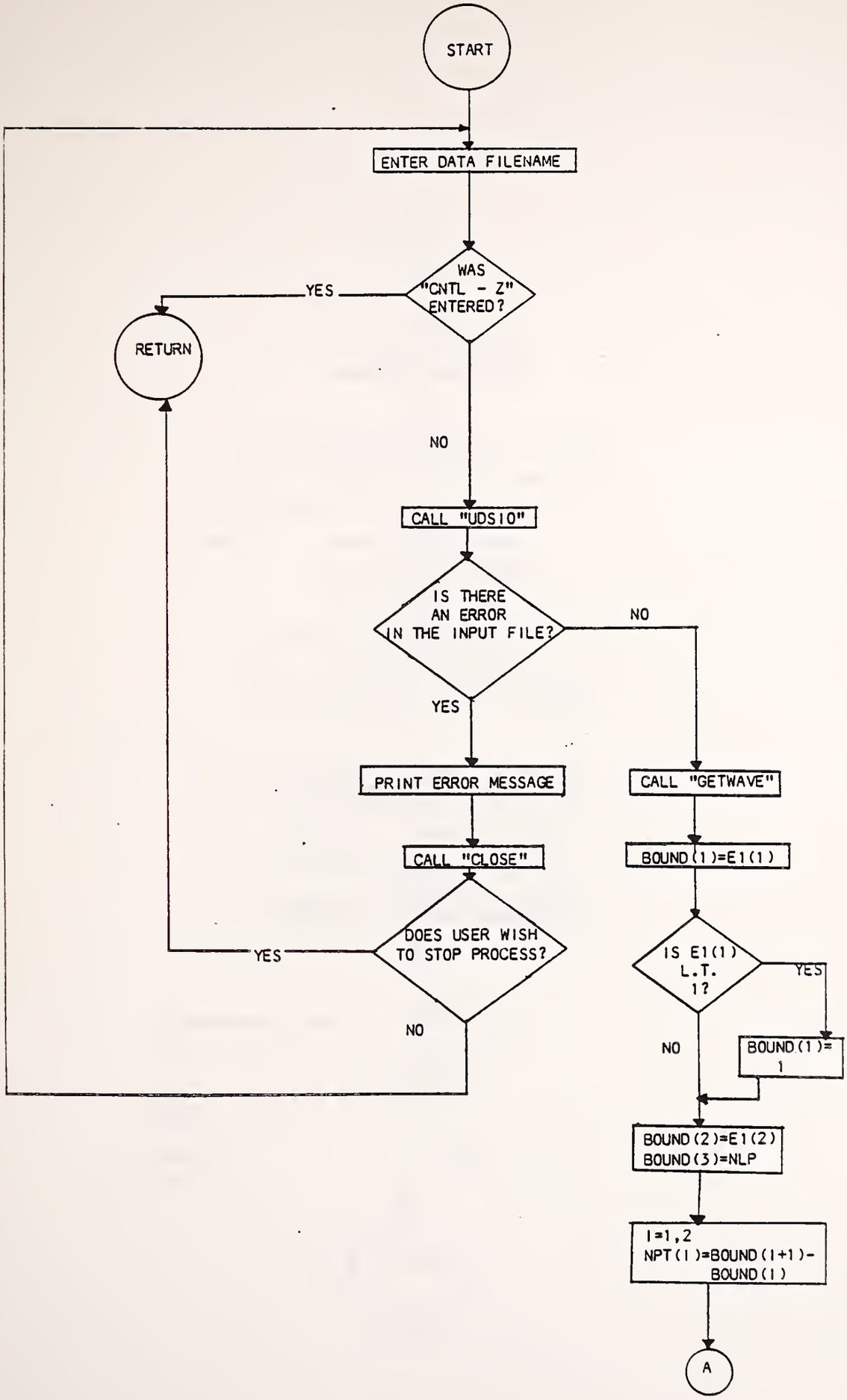


Figure 4-6 Flow Diagram of "CHA2"

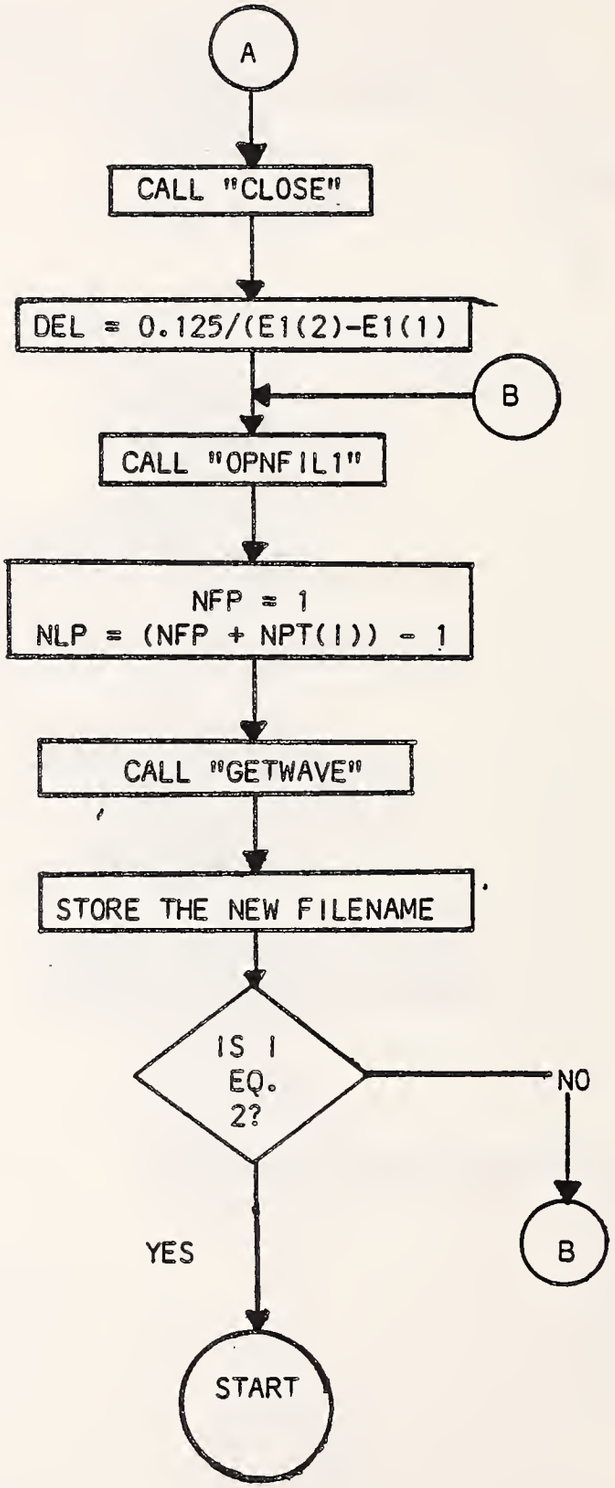


Figure 4-6 continued

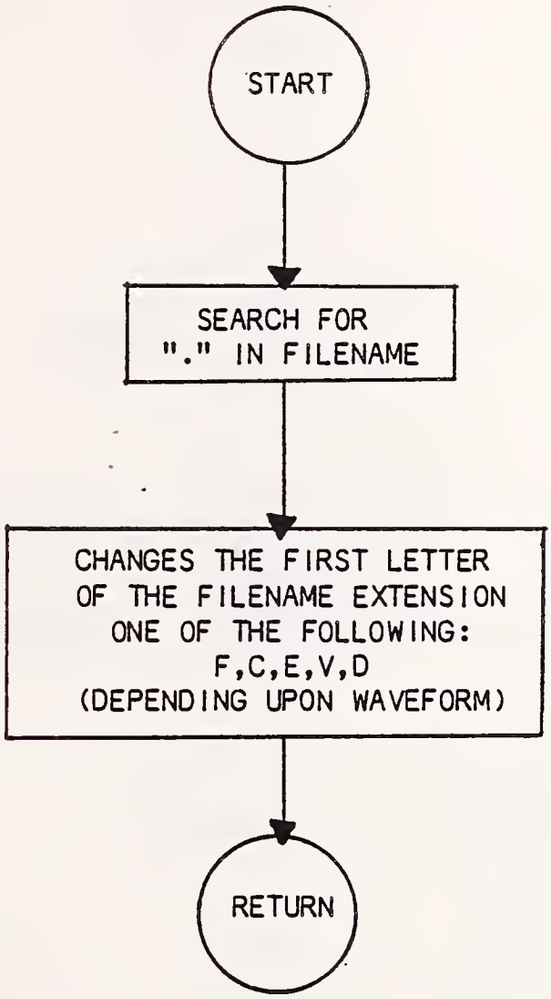


Figure 4-7 Flow Diagram of "OPNFIL".

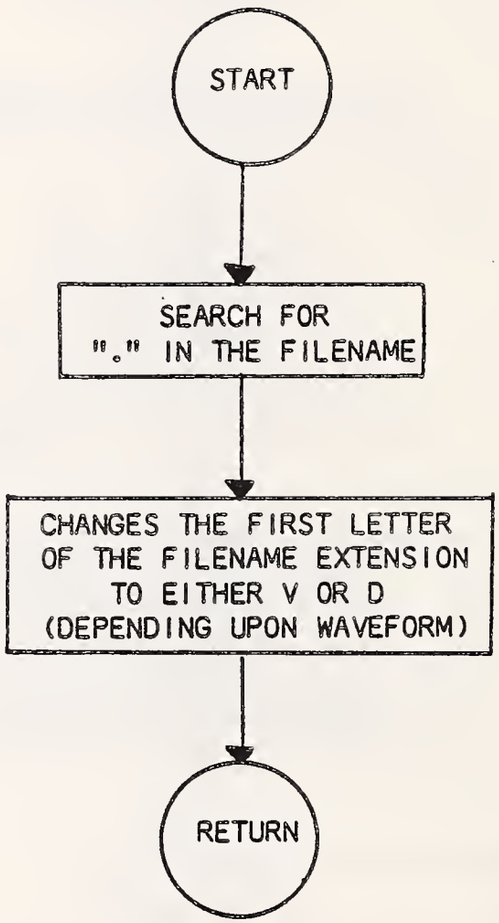


Figure 4-8 Flow Diagram of "OPNFIL1"

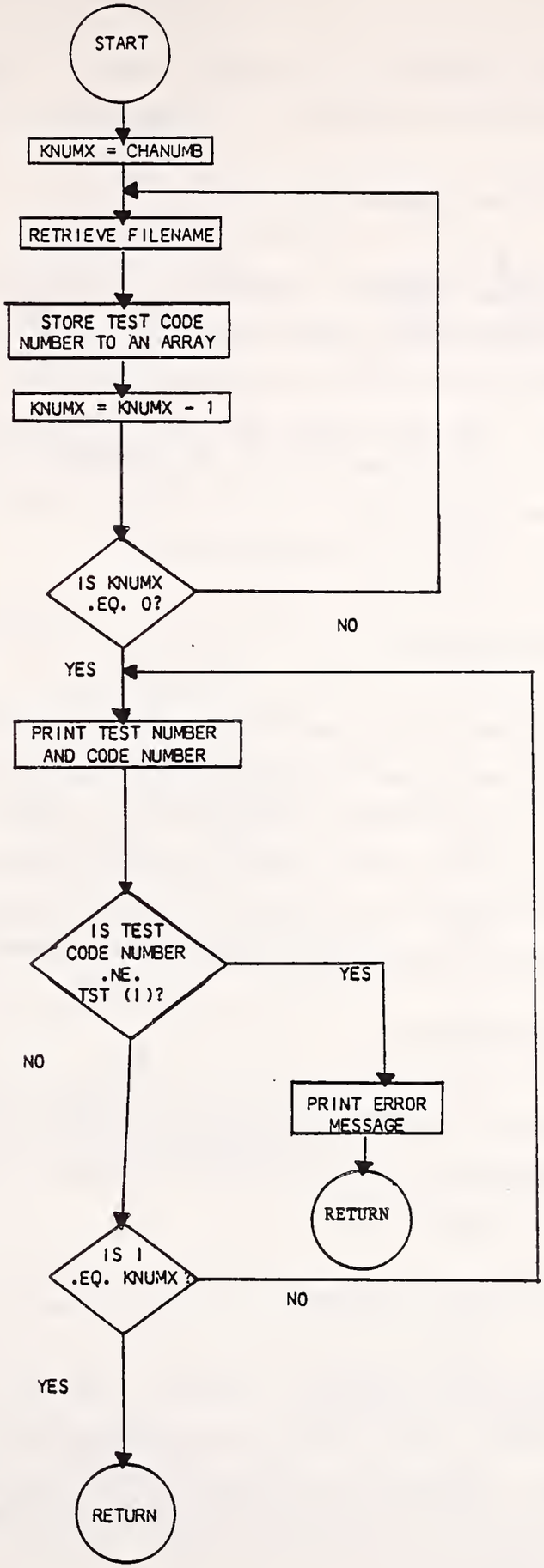


Figure 4-9 Flow Diagram of "CDNUM"

subroutine, CHANUMB is an integer that is incremented by one each time an input data file is read into a storage array in CHA1 or CHA2.

4.7 SUBROUTINE "RESULT"

Subroutine "RESULT" generates a data array which is the resultant of X and Z component input data files. This module also creates an output file called "RESULT.SUM" which contains a listing of the pairings of data files which were used to create the resultant data arrays. As with the files "created" in the separation of the individual waveforms, the output resultant files do not physically exist but, are "pseudo-files." The flow diagram of this module is shown in Figure 4-10.

4.8 SUBROUTINE "HIC3"

Subroutine "HIC3" calculates the HIC value of a resultant data array. This module is a modified version of the HIC3 routine written for NHTSA by S. Mentzer of Automated Sciences Group. The modifications made remove the read/write calls to the system storage device and the creation of a report file which stated the value of the HIC calculated for the data file. The flow diagram of this module is shown in Figure 4-11.

4.9 SUBROUTINE "PART"

Subroutine "PART" performs partitioning of possible optimal regions found during the execution of the HIC3 subroutine. The flow diagram of this module is shown in Figure 4-12.

4.10 SUBROUTINE "EXTRACT"

Subroutine "EXTRACT" extracts the file name out of the data file specification. (The data file specification can include any or all of the following: communications node, logical unit device name, account name, file name, extension, and version number.) The flow diagram of this module is shown in Figure 4-13.

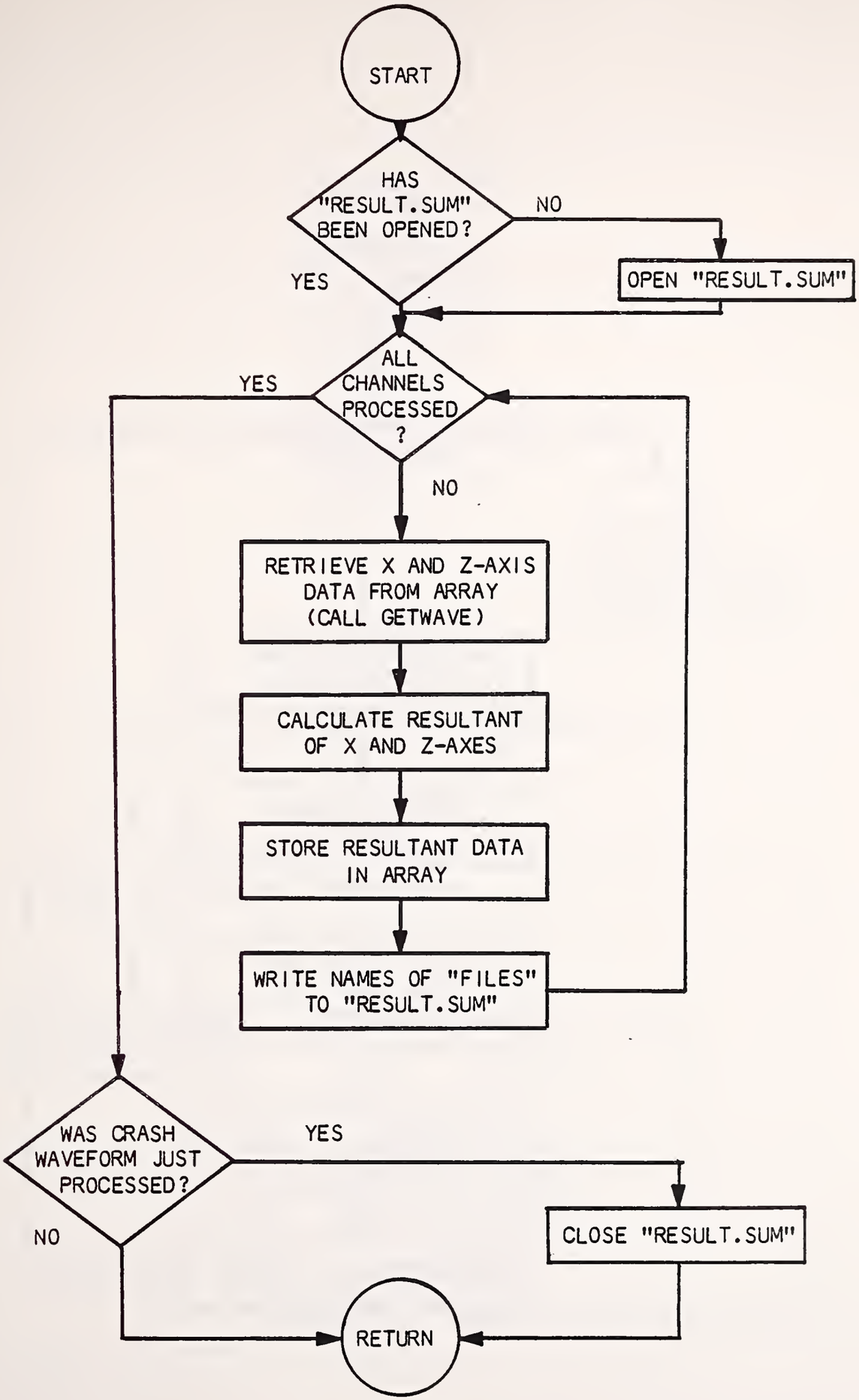


Figure 4-10 Flow Diagram of "RESULT"

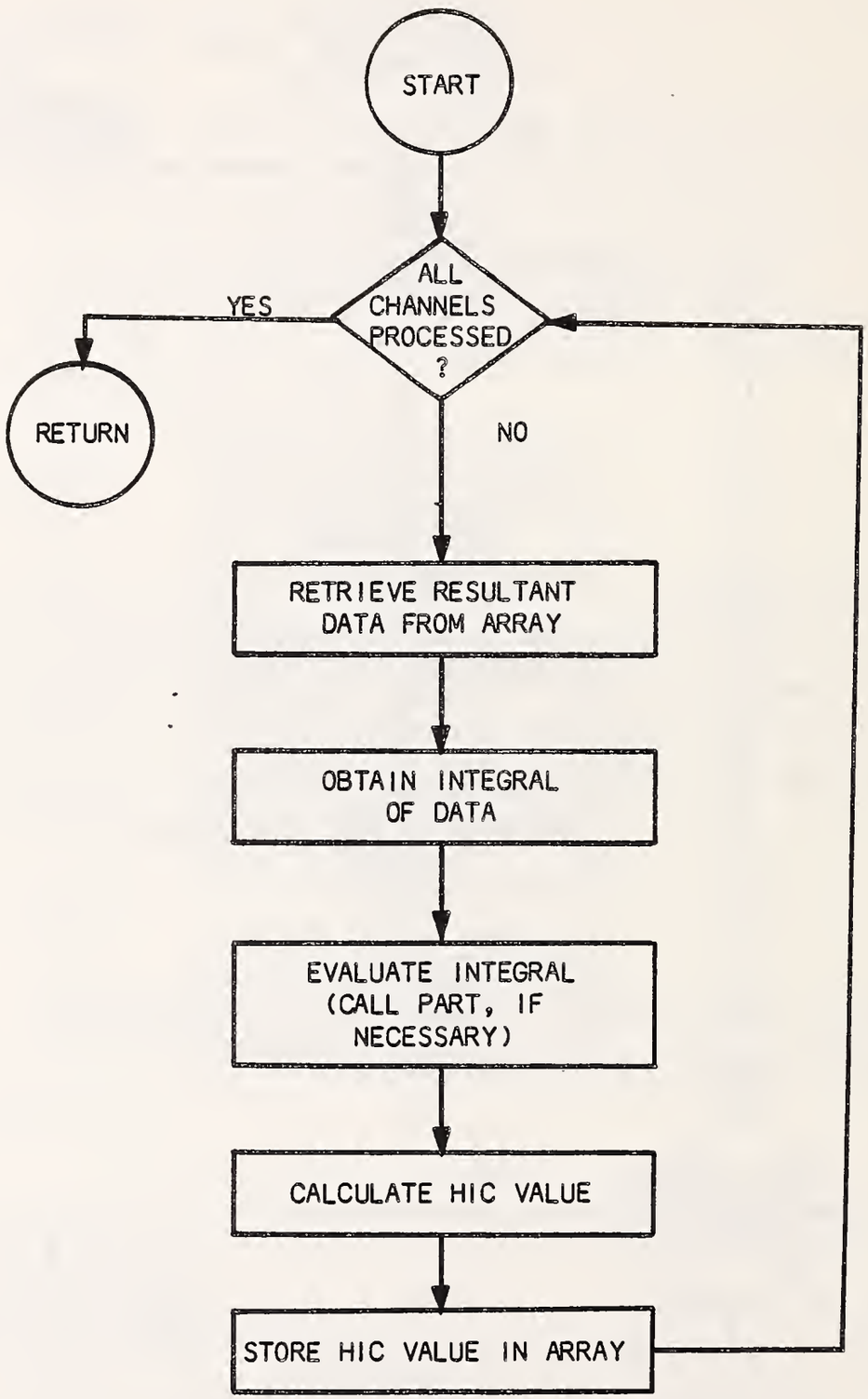


Figure 4-11 Flow Diagram of "HIC3"

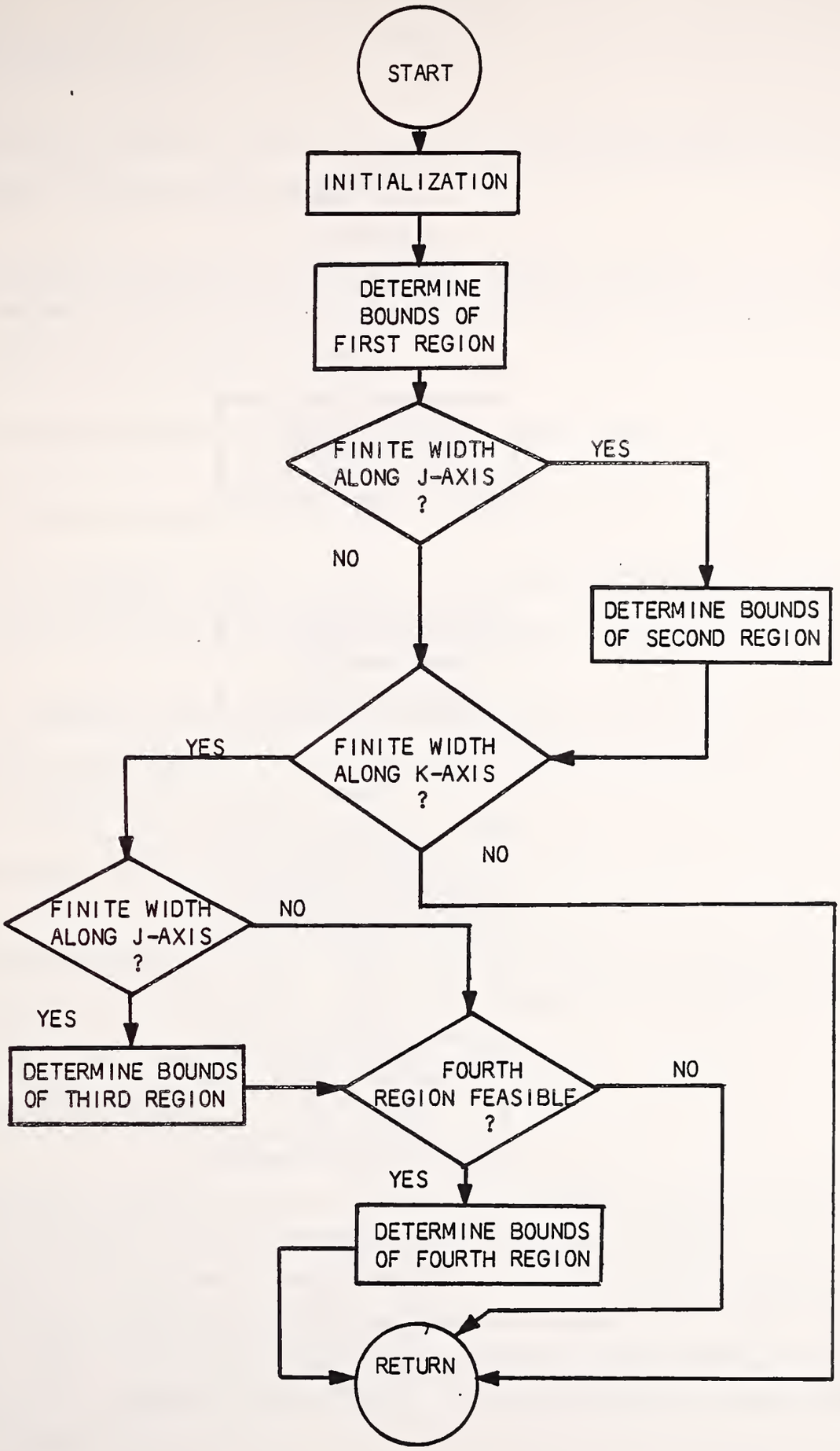


Figure 4-12 Flow Diagram of "PART"

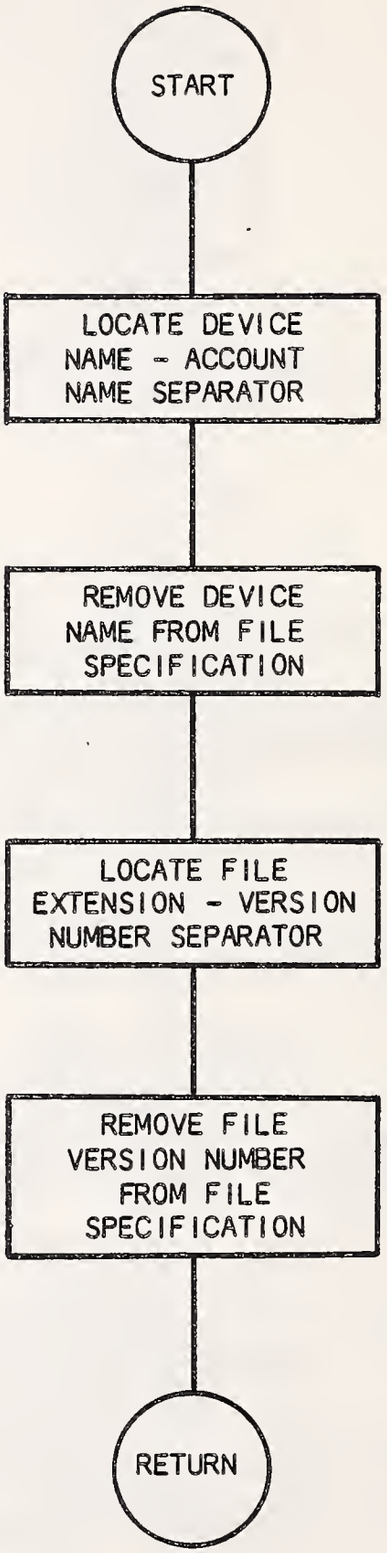


Figure 4-13 Flow Diagram of "EXTRACT"

4.11 SUBROUTINE "HSINE"

Subroutine "HSINE" performs four calculations/comparisons on the half-sine waveform. These calculations are:

1. Channel-to-channel time deviation. This comparison is carried out between the two data channels which contain the X-axis and Z-axis half-sine waveform components.

2. X-axis component time deviation from theoretical.

3. Z-axis component time deviation from theoretical.

4. HIC deviation. This comparison is carried out against a calculated theoretical HIC value which is calculated in subroutine "THCALC."

The results of the comparisons made in this module are written to a summary file (see Table 3-1 for the file naming convention). The flow diagram of this module is shown in Figure 4-14.

4.12 SUBROUTINE "ERR2"

Subroutine "ERR2" is used to interact with the user if an error occurs while a data file is being read. The flow diagram of this module is shown in Figure 4-15.

4.13 SUBROUTINE "CRASH"

Subroutine "CRASH" performs a comparison between the HIC value obtained from the resultant of the recorded X-axis and Z-axis crash data and a value of 930 which was calculated from the original crash test data using HIC3. The results of this comparison are then written to a summary file and a statistical data file. (See Table 3-1 for the file naming convention and the content of the statistical data file.) The flow diagram of this module is shown in Figure 4-16.

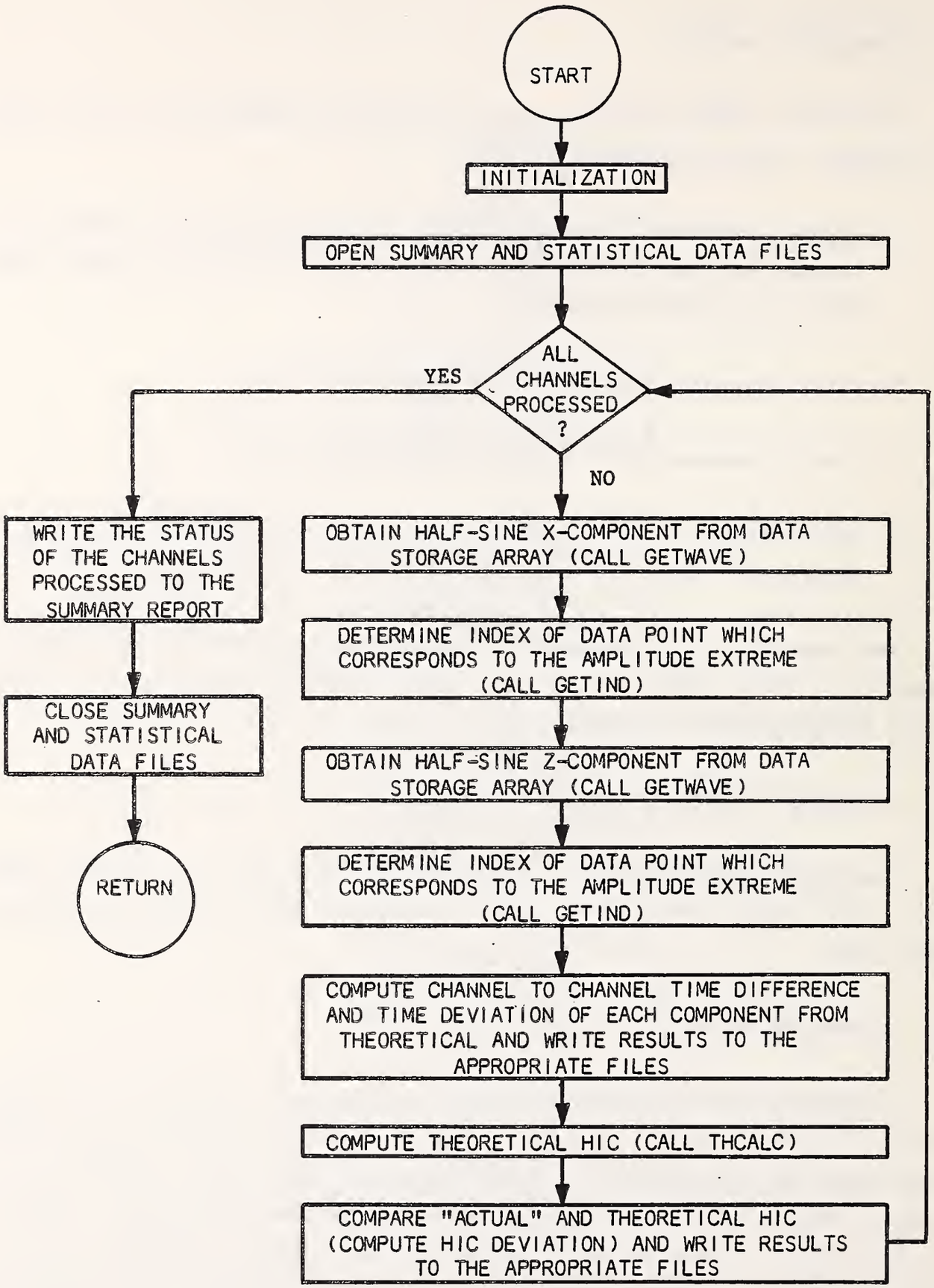


Figure 4-14 Flow Diagram of "HSINE"

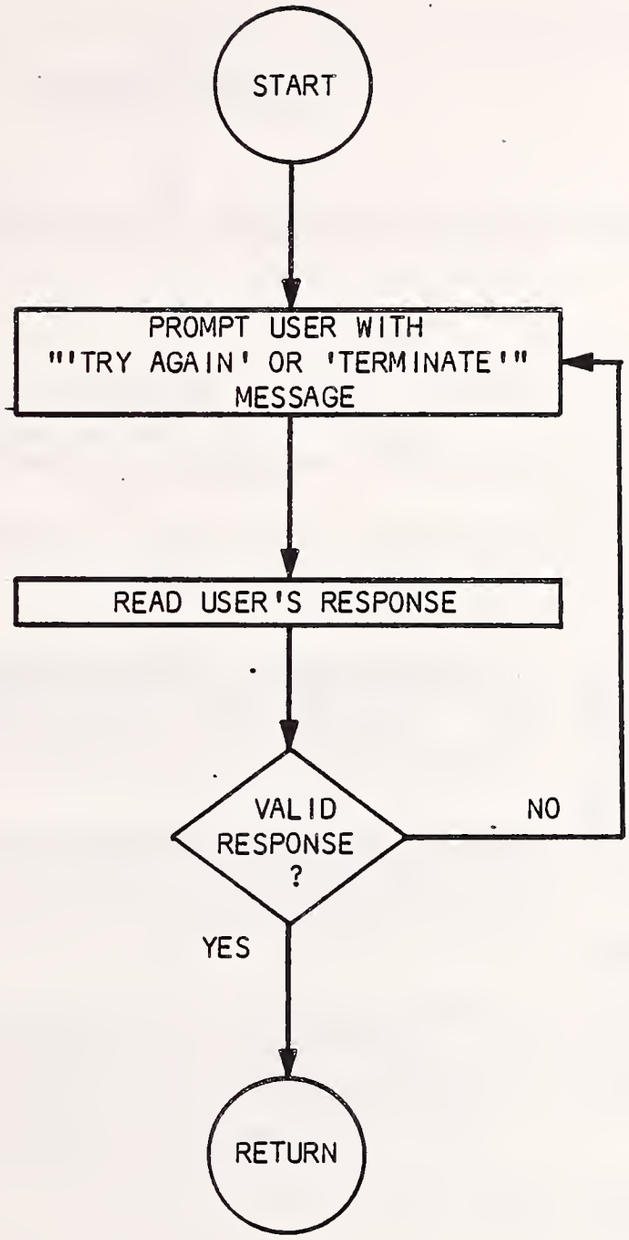


Figure 4-15 Flow Diagram of "ERR2"

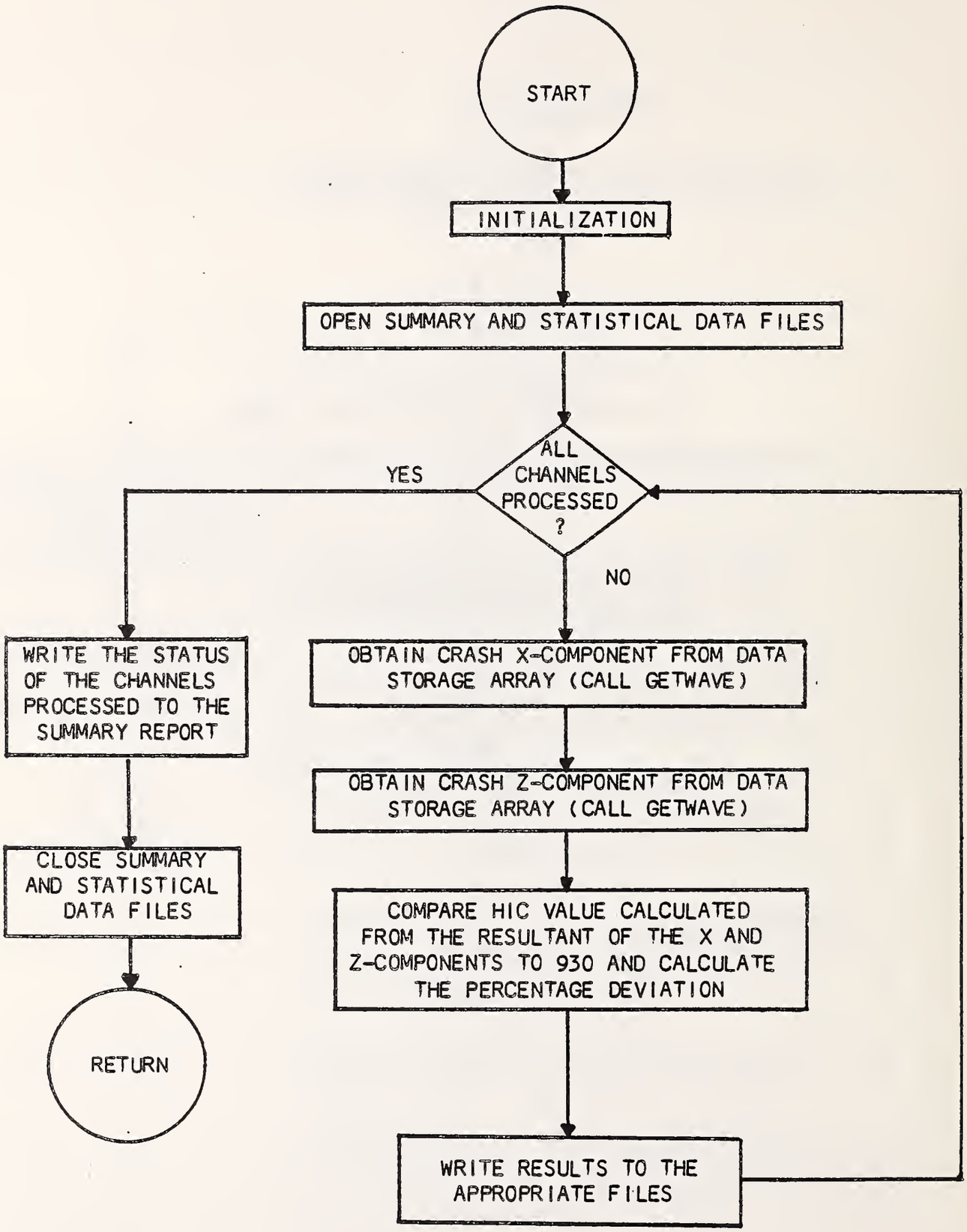


Figure 4-16 Flow Diagram of "CRASH"

4.14 SUBROUTINE "FRSP3"

Subroutine "FRSP3" computes the amplitude frequency response of a data acquisition channel in the following sequence:

1. The data file of the recorded data acquisition channel response to the sum-of-sines waveform is read into an array.

2. The index of the response array is displaced by 1 from "0 to N+1" to "1 to N+2" for compatibility with the fast Fourier transform (FFT) processing.

3. The input waveform data is recreated, sampled at the same frequency as was the response data.

4. Subroutine "SGAUSS" is called to generate an N point Gaussian data window.

5. The waveform and response arrays are multiplied by the Gaussian window weighting function.

6. The discrete Fourier transforms (DFT) of the weighted waveform and response arrays are computed using the FAST* fast Fourier transform subroutine.

7. The index of each DFT output array element whose corresponding frequency is closest to each signal frequency is determined.

8. The magnitude of each DFT output array element that corresponds to a signal frequency is determined.

9. The scaling factor is computed.

*The FAST FFT algorithm and implementing software were developed by G.D. Bergland and M.T. Dolan of Bell Laboratories, Murray Hill, N.J. and published in Programs for Digital Signal Processing by the IEEE Press, New York, © 1979 IEEE.

10. The scaled ratio of the DFT of the response to the DFT of the waveform at each signal frequency is computed. This is the numeric amplitude ratio at each signal frequency.

11. The amplitude ratio (in dB) is computed at each signal frequency.

12. The frequency response data files are written.

The flow diagram of this module is shown in Figure 4-17.

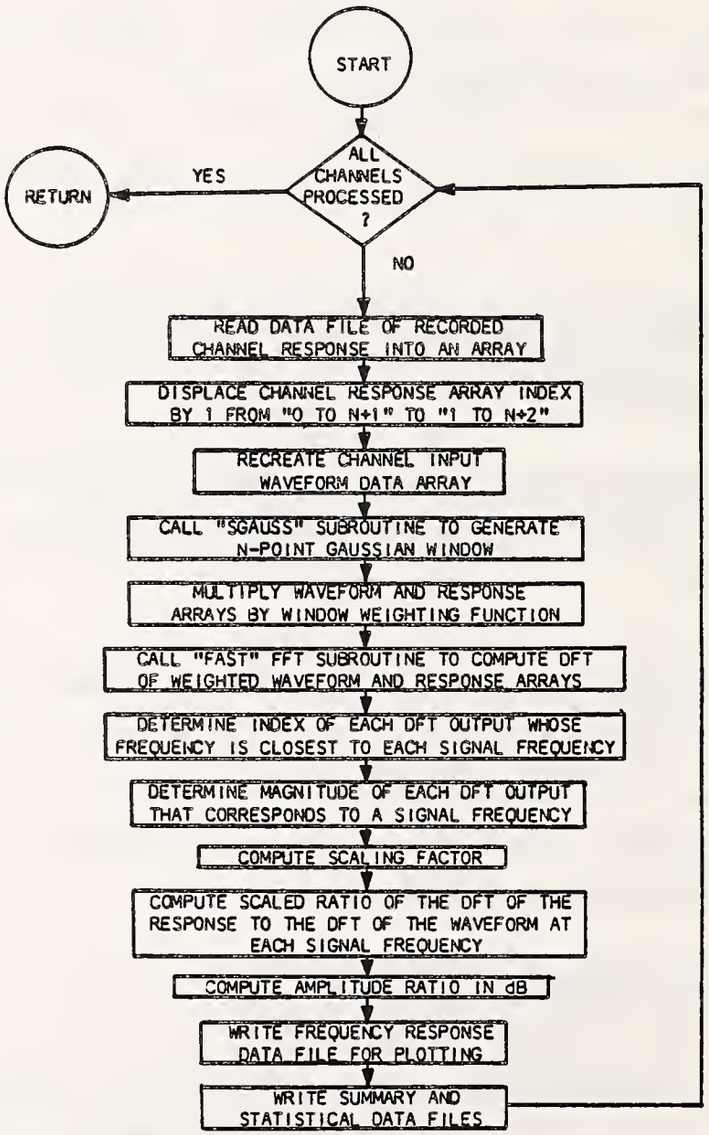


Figure 4-17 Flow Diagram of "FRSP3"

4.15 SUBROUTINE "SGAUSS"

Subroutine "SGAUSS" generates an N point Gaussian data window.

4.16 SUBROUTINE "FAST"

Subroutine "FAST" is the fast Fourier transform routine called by "FRSP3."

4.17 SUBROUTINE "FR2TR"

Subroutine "FR2TR" is a radix 2 iteration subroutine.

4.18 SUBROUTINE "FR4TR"

Subroutine "FR4TR" is a radix 4 iteration subroutine.

4.19 SUBROUTINE "FORD1"

Subroutine "FORD1" is the first of two in-place reordering subroutines called by "FAST."

4.20 SUBROUTINE "FORD2"

Subroutine "FORD2" is the second of two in-place reordering subroutines called by "FAST."

4.21 SUBROUTINE "RECTAN"

Subroutine "RECTAN" is used to process the rectangular waveform. "RECTAN" separates the rectangular waveform into ten cycles. Each of the cycles is then inspected with respect to the steady-state amplitude and amplitude overshoot on both the positive and negative sections of the cycle as well as for time shift within the channel. The flow diagram of this module is shown in Figure 4-18.

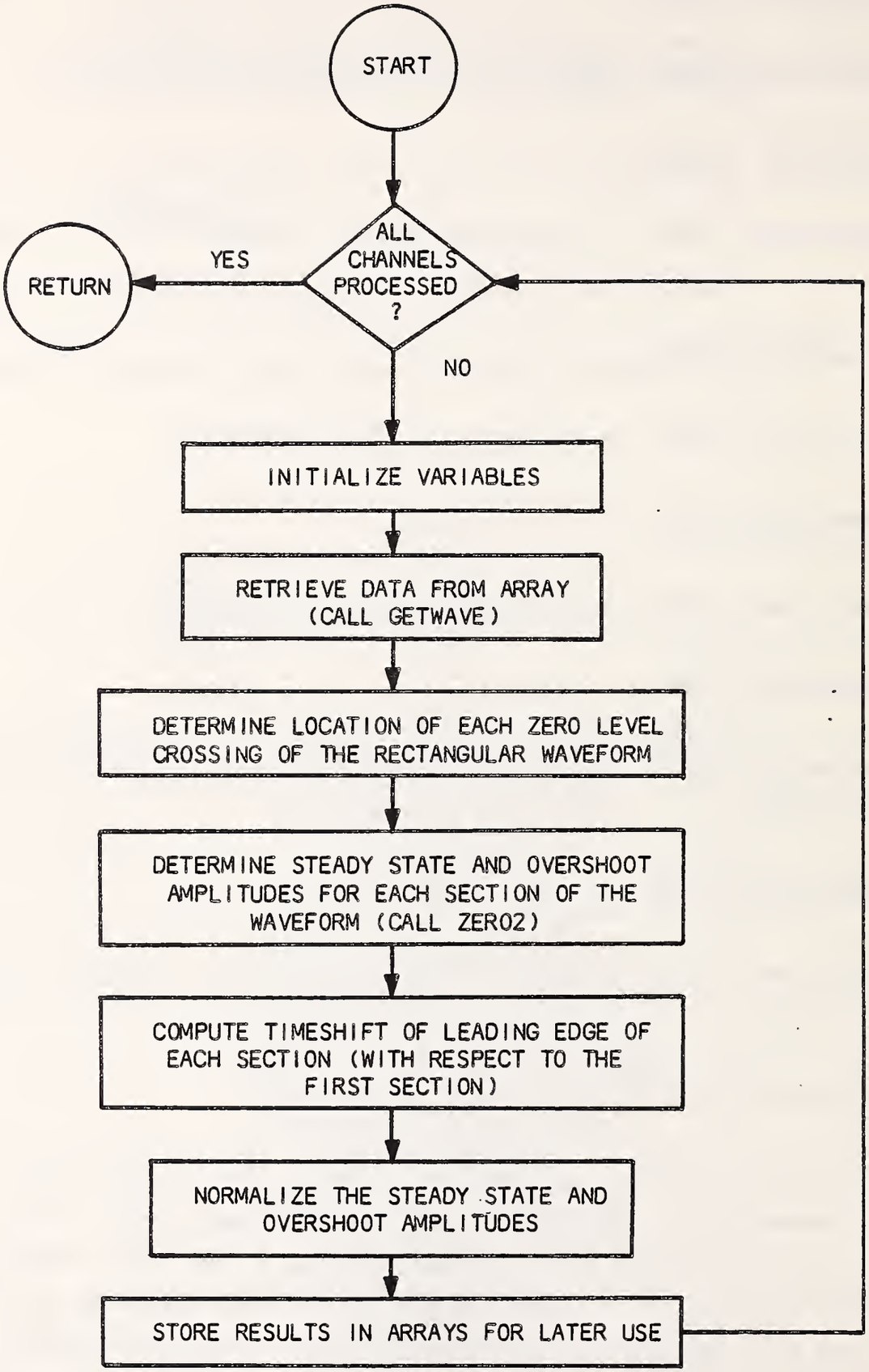


Figure 4-18 Flow Diagram of "RECTAN"

4.22 SUBROUTINE "ZER02"

Subroutine "ZER02" calculates the maximum overshoot and the steady-state amplitude values for any given cycle of the rectangular waveform. The flow diagram of this module is shown in Figure 4-19.

4.23 SUBROUTINE "COMPRE"

Subroutine "COMPRE" performs comparisons between the results of the steady-state amplitude and overshoot and time shift calculations and predefined tolerances. There are four time comparisons and two amplitude comparisons. The time comparisons are:

1. Deviation between actual times and corresponding theoretical times and between the deviations and the tolerance.

2. Time linearity. (The slope of a least squares fit straight line through data pairs consisting of theoretical time as the X coordinate and calculated time shift as the Y coordinate.)

3. Time offset. (The Y-axis intercept of a least squares fit straight line through data pairs consisting of theoretical time as the X coordinate and calculated time shift as the Y coordinate.)

4. Channel-to-channel time deviation.

The amplitude comparisons are:

1. Steady-state amplitude deviation from theoretical amplitude.

2. Amplitude overshoot relative to the calculated steady-state amplitude.

The flow diagram of this module is shown in Figure 4-20.

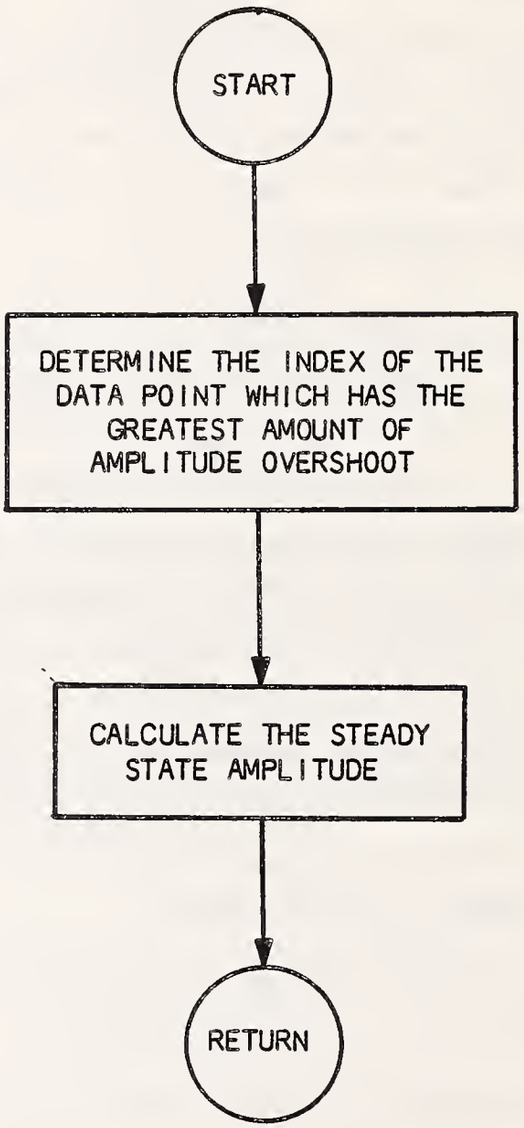


Figure 4-19 Flow Diagram of "ZERO2"

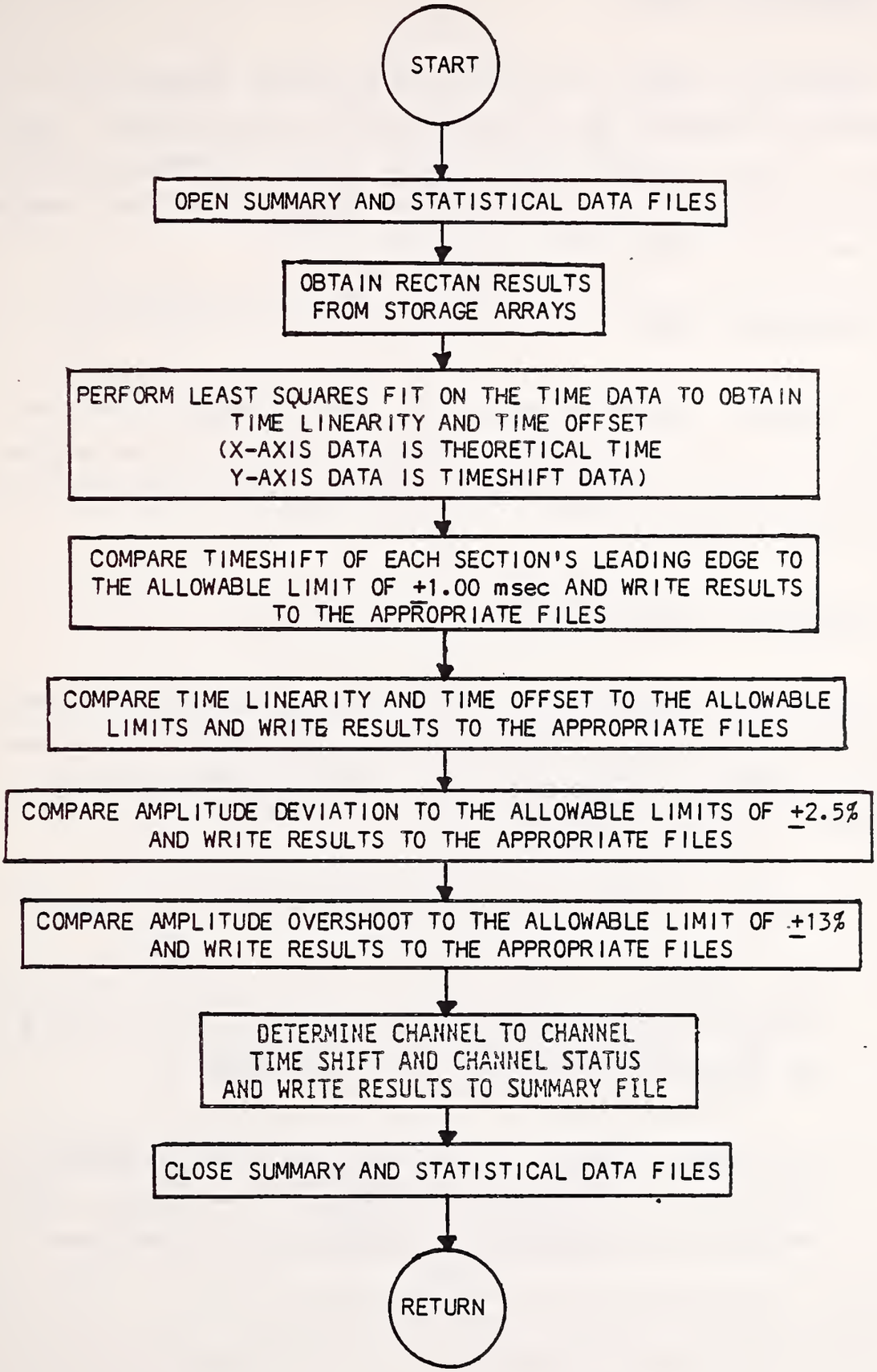


Figure 4-20 Flow Diagram of "COMPRE"

4.24 SUBROUTINE "STAIR"

Subroutine "STAIR" separates the staircase waveform into 12 periods (sections) by detecting the 12 leading edges of the waveform. Each of the 12 periods is then inspected with regards to the steady-state amplitude and overshoot and the time shift within the channel. The flow diagram of this module is shown in Figure 4-21.

4.25 SUBROUTINE "ZER03"

Subroutine "ZER03" performs the same functions for the staircase waveform processing as subroutine "ZER02" does for the rectangular waveform processing. For this reason, no discussion or flow diagram will be presented for subroutine "ZER03".

4.26 SUBROUTINE "COMSTR"

Subroutine "COMSTR" performs comparisons between the results of the steady-state amplitude and overshoot and time shift calculations and predefined tolerances. There are three time comparisons and four amplitude comparisons. The time comparisons are:

1. Deviation between the recorded waveform and the theoretical time.

2. Time linearity. (The slope of a least squares fit straight line through data pairs consisting of theoretical time as the X coordinate and calculated time shift as the Y coordinate.)

3. Time offset. (The Y-axis intercept of a least squares fit straight line through data pairs consisting of theoretical time as the X coordinate and calculated time shift as the Y coordinate.)

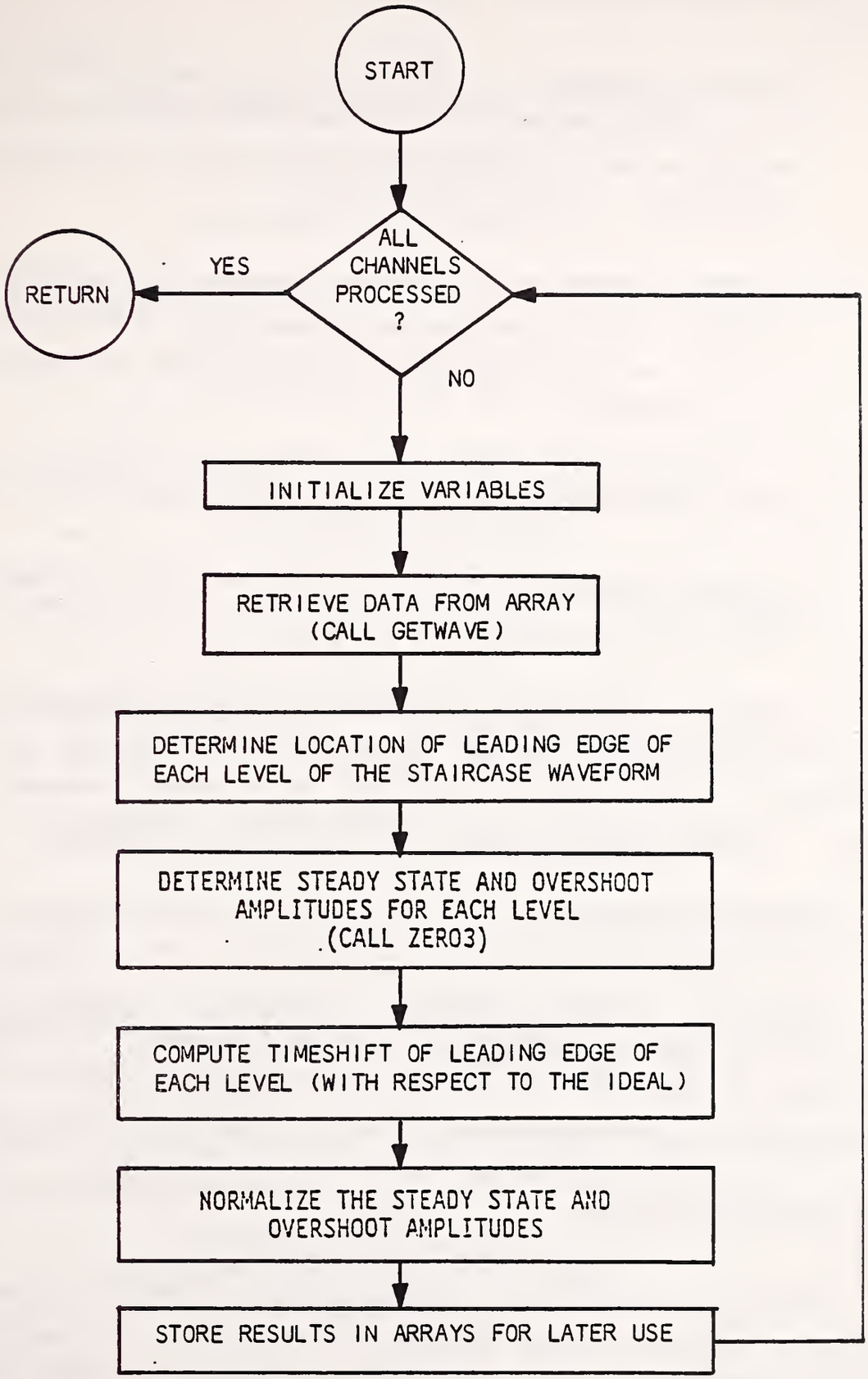


Figure 4-21 Flow Diagram of "STAIR"

The amplitude comparisons are:

1. Amplitude linearity. (The slope of a least squares fit straight line through data pairs consisting of theoretical amplitude as the X coordinate and measured steady-state amplitude deviation as the Y coordinate.)

2. Amplitude offset. (The Y-axis intercept of a least squares fit straight line through data pairs consisting of theoretical amplitude as the X coordinate and measured steady-state amplitude deviation as the Y coordinate.)

3. Steady-state amplitude deviation from theoretical amplitude.

4. Amplitude overshoot relative to the calculated steady-state amplitude.

In addition to the comparisons, this subroutine also generates a summary file and five statistical data files which contain the results of the comparisons. (See Table 2-1 for the file naming convention for these files.) The flow diagram of this module is shown in Figure 4-22.

4.27 SUBROUTINE "COMPR2"

Subroutine "COMPR2" performs a comparison of channel-to-channel time deviation for each period of the staircase waveform. The results of this comparison is added to the summary file originally created in subroutine "COMSTR". The flow diagram of this module is shown in Figure 4-23.

4.28 SUBROUTINE "UDSIO"

Subroutine "UDSIO" is a modification of the UDS (User Data Set) read/write utility routine developed for NHTSA. The modifications made relate to the error handling routines found within the subroutine. The flow diagram of this module is shown in Figure 4-24.

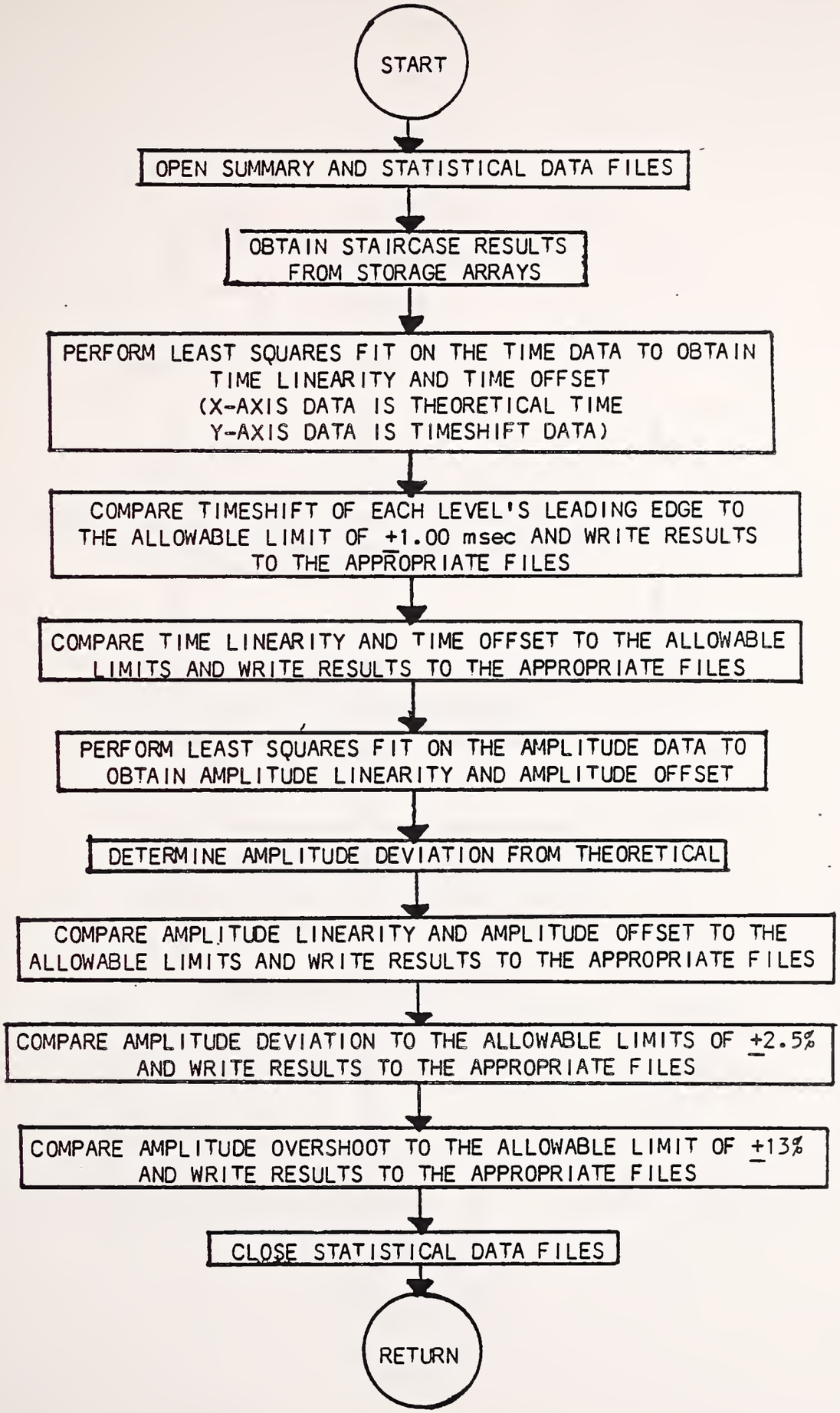


Figure 4-22 Flow Diagram of "COMSTR"

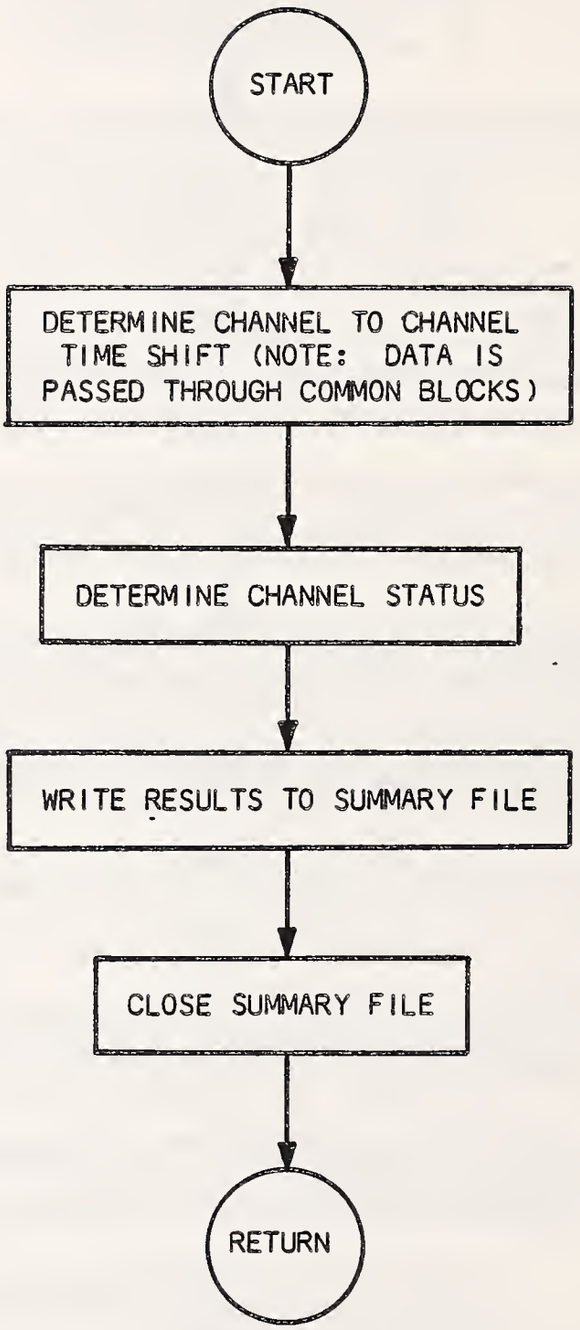


Figure 4-23 Flow Diagram of "COMPR2"

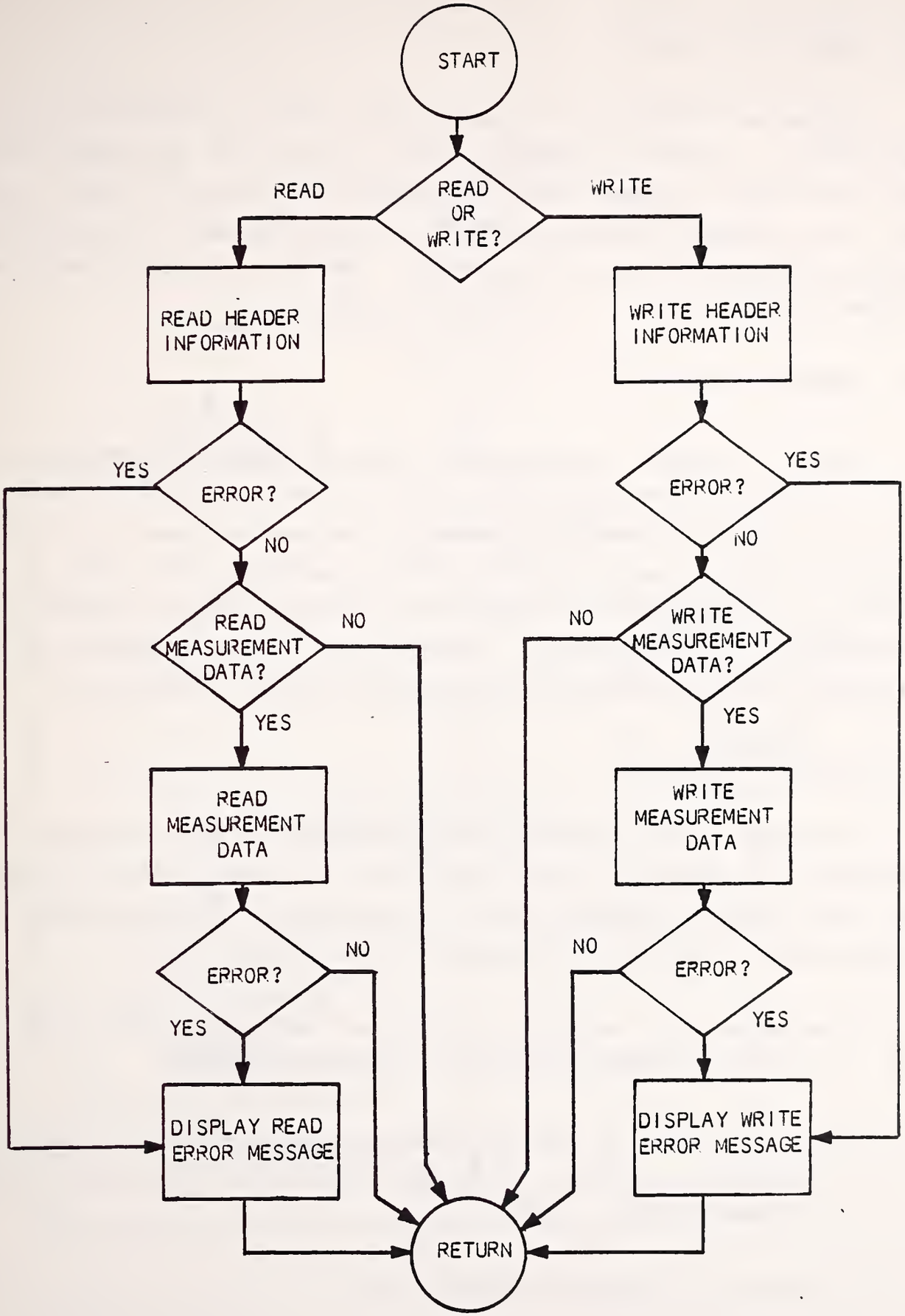


Figure 4-24 Flow Diagram of "UDSIO"

4.29 SUBROUTINE "GETWAVE"

Subroutine "GETWAVE" is a routine which allows the reduction of system input/output. Instead of writing to and reading from the system storage device, the data is placed into large arrays thus forming a crude "virtual disk." All necessary functions pertinent to the replacement of the read/write process are handled within this subroutine. The flow diagram of this module is shown in Figure 4-25.

4.30 SUBROUTINE "GETIND"

Subroutine "GETIND" determines the index of the point in the data array which is closest to the calculated time of occurrence of the maximum data value in the half-sine waveform components. The approach taken is to average the times the data passes through a certain amplitude level which is expressed as a percentage of the theoretical maximum (+200 g's depending upon the component being considered). Interpolation is used to improve the time calculation. The flow diagram of this module is shown in Figure 4-26.

4.31 SUBROUTINE "THCALC"

Subroutine "THCALC" obtains, through multiple steps, the value of the theoretical HIC if the curve were to be made up of two components (X and Z) which have the same amplitudes and duration as the actual (recorded) data. The steps taken to obtain the theoretical HIC value are:

1. Determine the equations of the two half-sine curves which pass through the points determined in subroutine "GETIND".

2. Generate arrays of data which are the equivalent of the equations by sampling the equations at the same sample rate as was used to record the waveform data.

3. Obtain the resultant of the two arrays.

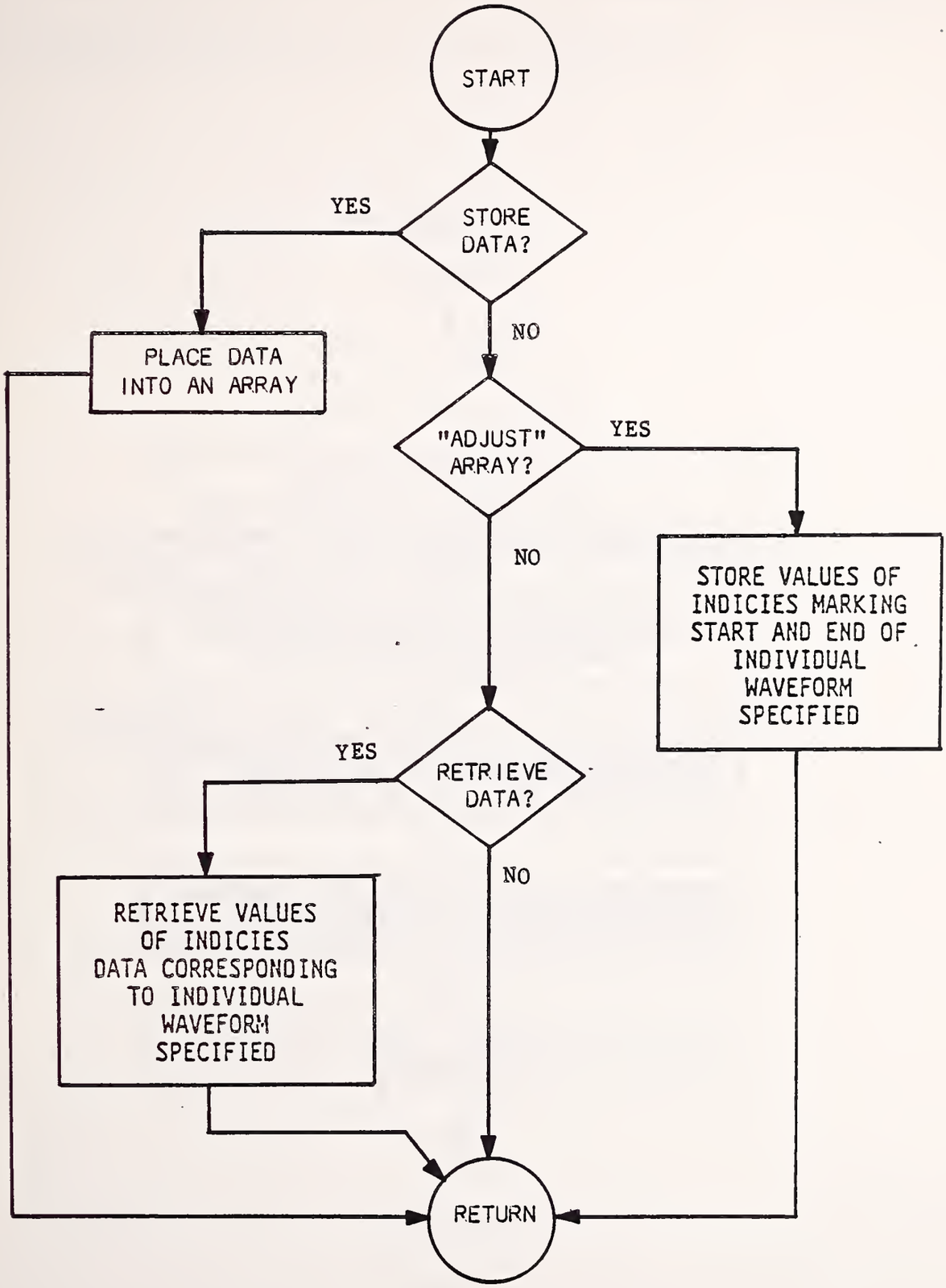


Figure 4-25 Flow Diagram of "GETWAVE"

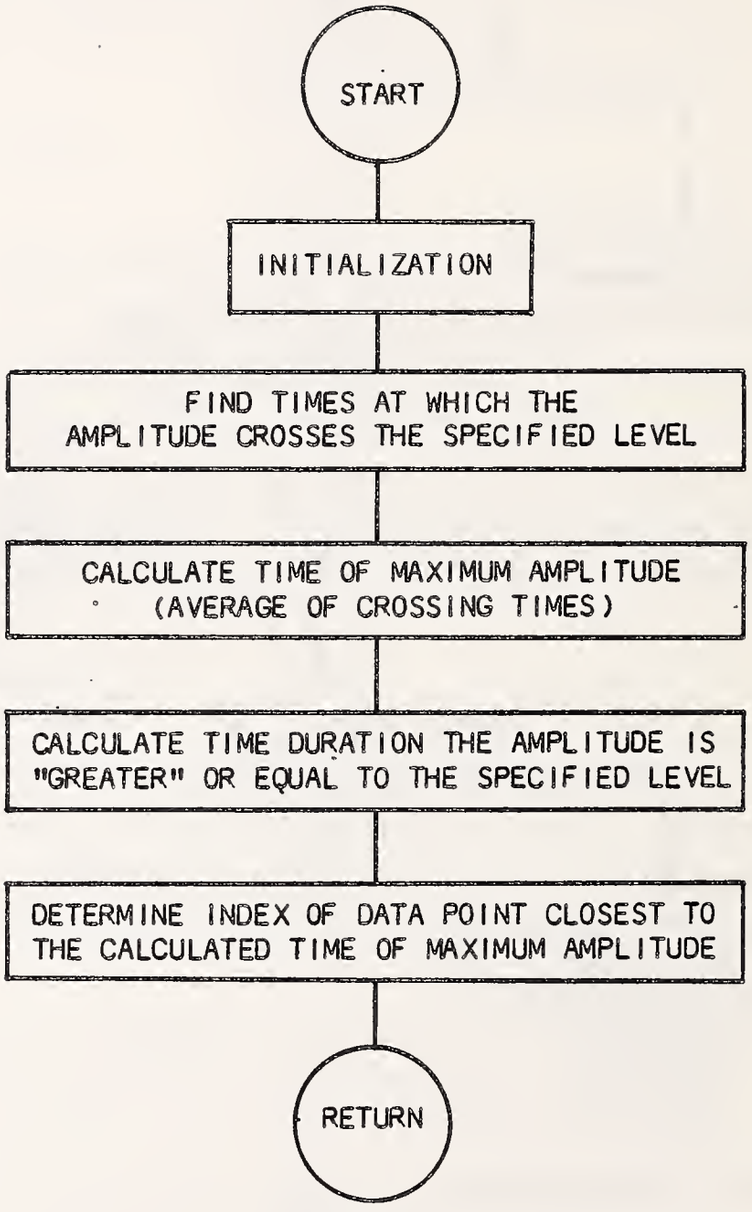


Figure 4-26 Flow Diagram of "GETIND"

4. Calculate HIC and the upper and lower time boundaries (T1 and T2) using a modified version of the HIC3 subroutine.

The flow diagram of this module is shown in Figure 4-27.

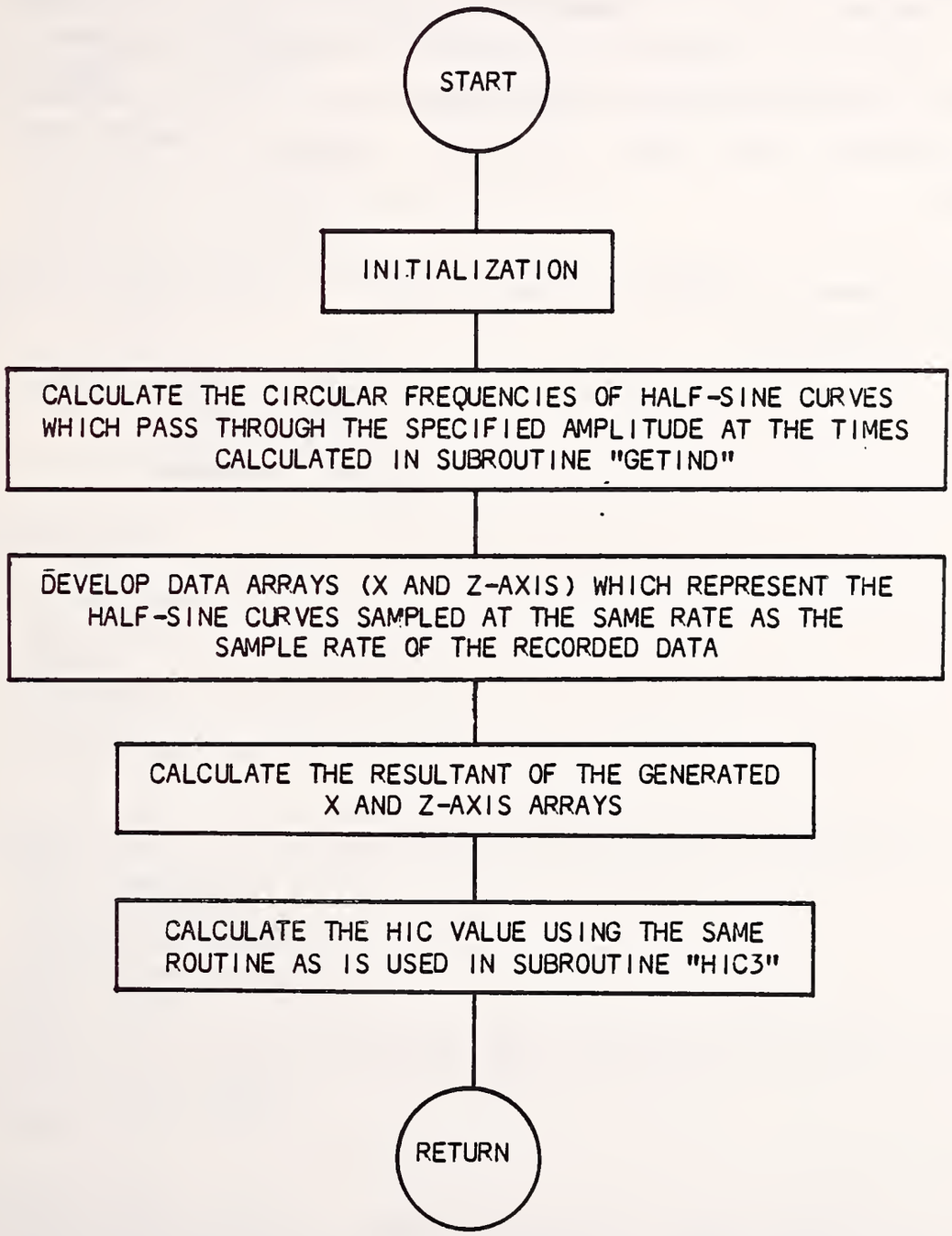


Figure 4-27 Flow Diagram of "THCALC"

5. PROGRAM LISTINGS

This section presents the source code listings for the waveform generator signal processing software package. The package is divided into five segments with each segment containing one or more program modules. The program listings will be presented in the order as they appear in the segments. For those readers interested in locating the source code on the NHTSA VAX 11/780, the segments are located in user area [JOHNN.WGPROG] and have the following file names: WGPROC.FOR, FILMGT.FOR, WGSUBS.FOR, REC.FOR, and STR.FOR.

The first segment, "WGPROC", contains only the main program module. The source code listing is:

```
C   ********************
C   **   MAIN PROGRAM **
C   ********************
        PROGRAM WGPROC
C
C  VARIABLE DECLARATION
C
        DIMENSION Y(-1000:10000)
        INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
        CHARACTER ITIMEZ *30,DUM7 *1
        DIMENSION IEDGE(10)
        INTEGER  OPTION,CHANUMB,CNTFL,EFLAG(9,18),DFLAG,DELFLAG
        REAL T(18,23)
        CHARACTER *4 TSTCDNUM
        CHARACTER *30 CHAFIL(108),RESFIL(72)
        CHARACTER *9 DAY,HOUR
        Character*1 ANSWER
C
C  COMMON BLOCKS
C
        COMMON  /CBFIL/ CHAFIL
     *          /CBCH/ CHANUMB
     *          /CBRES/ RESFIL
     *          /CBCDNUM/ TSTCDNUM
C
C Initialization
C
      DO 1 I=1,108
        CHAFIL(I)='                                                '
    1 CONTINUE
      DO 2 I=1,72
        RESFIL(I)='                                                '
    2 CONTINUE
C
      WRITE(6,10)
   10 FORMAT(//1X,' *** WAVEFORM GENERATOR PROCESSING SOFTWARE ***'//)
        CALL DATE(DAY)
        CALL TIME(HOUR)
        WRITE(6,12)DAY,HOUR
   12   FORMAT(10X,A9,10X,A9)
      WRITE (6,101)
  101 FORMAT (//3x, 'DO YOU WANT ANALYSIS OF HALF-SINE AND CRASH PULSES
     * IN THE PROCESSING? (Y or N)', /1x,'>>',$)
      READ (5,102) ANSWER
  102 FORMAT (A)

 1000 WRITE(6,100)
  100 FORMAT(//1X,'OPTION 1 : FINDING DATA POINT INDEX NUMBERS ',
     & 'CORRESPONDING TO EDGES',/1X,'OPTION 2 : ENTER CORRECT DATA ',
     & 'POINT INDEX NUMBERS',/1X,'ENTER DESIRED OPTION NUMBER (1/2)',
     & /1X,'>>> ',$)
      READ(5,200)  OPTION
  200 FORMAT(I1)
```

```
      IF (OPTION.NE.1.AND.OPTION.NE.2) THEN
        WRITE(6,3)
    3   FORMAT(/1X,'  *** ERROR - INCORRECT OPTION NUMBER *** ',
     &         /1X,'  *** TRY AGAIN ***')
        GOTO 1000
       END IF
       IF (OPTION.EQ.1) THEN
C
C  OPTION 1
C
        CALL LED(IFLAG2,IEDGE,ITFL)
        IF (IFLAG2  .EQ. 0 ) GOTO 2000
        IF (IFLAG2  .EQ. 10) GOTO 9999
C
C  plot of time zero is necessary
C
        WRITE(6,4)
    4   FORMAT(/1X,' WARNING!',
     &         /1X,'   ADDITIONAL EDGES IN TIME ZERO FILE HAVE BEEN',
     &       ' DETECTED',/1X,'   OBTAIN A PLOT OF TIME ZERO FILE '
     &         /1X,'   AND USE OPTION 2 TO ENTER THE CORRECT',
     &      ' DATA POINT NUMBERS')
        GOTO 9999
C
C  correct number of edges detected
C
 2000   IF (ITFL.EQ.1.OR.ITFL.EQ.3) GOTO 2010
C
C  delayed TIMEZ / inverted delayed TIMEZ
C
        CALL CLOSE(1)
        CALL CHA2(IEDGE,IFLAG2)
        IF (IFLAG2 .EQ. 10 ) GOTO 9999
        CALL CDNUM(IFLAG2)
        IF (IFLAG2 .EQ. 10) GOTO 9999
        DELFLAG =1
        GOTO 4000
C
C   TIMEZ / inverted TIMEZ
C
 2010   CALL CLOSE(1)
        CALL CHA1(IEDGE,IFLAG2)
        IF (IFLAG2 .EQ. 10) GOTO 9999
        CALL CDNUM(IFLAG2)
        IF (IFLAG2  .EQ. 10) GOTO 9999
        DELFLAG = 0
        GOTO 4000
       END IF
C
C OPTION 2
C
       IF (OPTION.EQ.2) THEN
 5000   WRITE(5,9000)
 9000   FORMAT(/1X,'ENTER  1  IF TIMING FILE IS A STANDARD TIME ZERO',
```

```
     &  ' FILE',/1X,'       2  IF TIMING FILE IS A DELAYED TIME ZERO',
     &  ' FILE',/1X,'       3  IF TIMING FILE IS AN INVERTED TIME ZERO',
     &  ' FILE',/1X,'       4  IF TIMING FILE IS A DELAYED & INVERTED',
     &  ' TIME ZERO FILE')
        READ(5,9001) ID1
 9001   FORMAT(I2)
        IF (ID1.EQ.2.OR.ID1.EQ.4) ICOUNT = 2
        IF (ID1.EQ.1.OR.ID1.EQ.3) ICOUNT = 4
        DO 5010  I = 1,ICOUNT
          WRITE(6,130)  I
  130     FORMAT(/1X,'ENTER DATA POINT INDEX NUMBER ',I2,
     &           /1X,'>>> ',$)
          READ(5,210)  IEDGE(I)
  210     FORMAT(I4)
 5010   CONTINUE
        IF (ID1.EQ.1.OR.ID1.EQ.3) GOTO 5050
C
C    delayed / inverted delayed TIMEZ
C
        CALL CLOSE(1)
        CALL CHA2(IEDGE,IFLAG2)
        IF (IFLAG2 .EQ. 10) GOTO 9999
        CALL CDNUM(IFLAG2)
        IF (IFLAG2 .EQ. 10) GOTO 9999
        DELFLAG = 1
        GOTO 4000
C
C   TIMEZ-/ inverted TIMEZ
C
 5050   CALL CLOSE(1)
        CALL CHA1(IEDGE,IFLAG2)
        IF (IFLAG2 .EQ. 10) GOTO 9999
        CALL CDNUM(IFLAG2)
        IF (IFLAG2 .EQ. 10) GOTO 9999
        DELFLAG = 0
      END IF
 4000   WRITE(6,135)
 135    FORMAT(/1X,'PROCESSING SEQUENCE:',
     & /10X,'RECTANGLE WAVEFORM',/10X,'STAIR WAVEFORM',
     & /10X,'HALF-SINE WAVEFORM',/10X,'CRASH WAVEFORM',
     & /10X,'SUM-OF-SINE WAVEFORM')
        CNTFL = 1
C
C    AUTOMATIC PROCESSING
C
 4010   DFLAG = 0
        IF (DELFLAG.EQ.1) GOTO 4500
C
C   RECTANGLE PROCESSOR
C
        CALL RECTAN(ICHAN,DFLAG,CNTFL)
        IF (DFLAG  .EQ.  1) GOTO 9990
        CALL COMPRE(ICHAN,DFLAG)
        IF (DFLAG  .EQ. 1) GOTO 9990
```

```
C
C   STAIR PROCESSOR
C
          DFLAG = 0
          CALL STAIR(ICHAN,DFLAG,CNTFL)
          IF (DFLAG  .EQ. 1) GOTO 9990
          CALL COMSTR(T,ICHAN,EFLAG,DFLAG)
          IF (DFLAG  .EQ. 1) GOTO 9990
          CALL COMPR2(T,ICHAN,EFLAG)
          IF (DFLAG  .EQ.  1) GOTO 9990
          IF (ANSWER .EQ. 'Y') GO TO 4601
          IF (ANSWER .EQ. 'y') GO TO 4601
          GO TO 4600
 4601     CONTINUE
C    HALF-SINE PROCESSOR
C
          DFLAG = 0
          WRITE(6,6)
    6     FORMAT(/1X,' **  HALF-SINE WAVEFORM PROCESSOR ** '/)
          IKJNUM = -3
          IWF=2
          ISLT=1
          CALL RESULT(DFLAG,IKJNUM,CNTFL,IWF,DELFLAG)
          IF (DFLAG .EQ. 1) GOTO 9990
          CALL HIC3(DFLAG,CNTFL,ISLT)
          IF (DFLAG .EQ. 1) GOTO 9990
          CALL HSINE(DFLAG,CNTFL)
          IF (DFLAG .EQ. 1) GOTO 9990
C
C     CRASH WAVEFORM PROCESSOR
C
 4500     DFLAG = 0
          WRITE(6,7)
    7     FORMAT(/1X,' ** CRASH WAVEFORM PROCESSOR ** '/)
          IKJNUM = 0
          IWF=5
          ISLT=2
          CALL RESULT(DFLAG,IKJNUM,CNTFL,IWF,DELFLAG)
          IF (DFLAG .EQ. 1) GOTO 9990
          CALL HIC3(DFLAG,CNTFL,ISLT)
          IF (DFLAG .EQ. 1) GOTO 9990
          CALL CRASH(DFLAG,CNTFL)
          IF (DFLAG .EQ. 1) GOTO 9990
C
C    SUM-OF-SINE WAVEFORM PROCESSOR
C
 4600     WRITE(6,8)
    8     FORMAT(/1X,' ** SUM-OF-SINE WAVEFORM PROCESSOR **'/)
          DFLAG = 0
          CALL FRSP3(DFLAG,CNTFL)
          IF (DFLAG  .EQ. 1) GOTO 9990
          GOTO 9999
C
C  ERROR
```

```
C
 9990   WRITE(6,9)
    9   FORMAT(//1X,'***** CRITICAL ERROR *****'/)
 9999   CONTINUE
        WRITE(6,11)
   11   FORMAT(/1X/)
        STOP 'END OF PROCESSING RUN'
        END
```

The second segment, "FILMGT" contains the following modules:

LED
VWGEN1
VWGEN2
CHA1
OPNFIL
CHA2
OPNFIL1
CDNUM
UDSIO
GETWAVE

The source code listing for these modules is:

```
C  ***************************************************
C  *  SUBROUTINE LED                                 *
C  ***************************************************
        SUBROUTINE LED(IFLAG2,IED1,ITFL)
        CHARACTER  IFLNM1*60,DUM *1
        DIMENSION IED1(10)
        INTEGER BLNK,IUNITS(2)
        INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
        REAL Y(-1000:10000),YMA,YMI
C
C
        DATA BLNK/'  '/,NO/'NO'/
 10     IFLAG2 = 0
        WRITE(6,20)
 20     FORMAT(/3X,'ENTER TIME SYNCHRONIZATION FILE NAME',/1X,'>>',$)
        READ(5,25)IFLNM1
 25     FORMAT(A60)
        WRITE(6,*)' '
        CALL UDSIO('R','F',IFLNM1,1,Y,IERR)
        IF (IERR .EQ. 0)  GOTO 30
C
C   ERROR
C
 27     WRITE(6,*)  '  ERROR!!!'
        WRITE(6,*)  '  ***  FILE CANNOT BE ACCESSED *** '
        GOTO 1000
 30     NOPTS = (NLP - NFP) + 1
C
C  FINDING MINIMUM AND MAXIMUM
C
        YMA = -1.E10
        YMI = 1.E10
        DO 3000 I = NFP,NLP
          YMA = AMAX1(YMA,Y(I))
          YMI = AMIN1(YMI,Y(I))
 3000   CONTINUE
        ID1 = 1
        ITFL = ID1
        IF (ID1  .EQ. 1) GOTO 100
        IF (ID1  .EQ. 2) GOTO 150
        IF (ID1  .EQ. 3) GOTO 200
        IF (ID1  .EQ. 4) GOTO 250
C
C  ERROR IN HEADING INFORMATION FOR TIME ZERO CHANNEL OPTION USED
C
        WRITE(6,*) '   ERROR!!!'
        WRITE(6,*) '  *** HEADING INFORMATION DOES NOT HAVE CORRECT',
     * ' TIME ZERO OPTION IDENTIFIER '
        GOTO 1000
C
C
C
 100    WRITE(6,*) ' **NOTE : TIME ZERO HAS BEEN RECORDED**'
        GOTO 300
```

```
 150    WRITE(6,*) ' **NOTE : DELAYED TIME ZERO HAS BEEN RECORDED**'
        GOTO 300
 200    WRITE(6,*) ' **NOTE : INVERTED TIME ZERO HAS BEEN RECORDED**'
        GOTO 400
 250    WRITE(6,*) ' **NOTE : INVERTED DELAYED TIME ZERO HAS BEEN ',
     * 'RECORDED**'
        GOTO 400
C
C   TIME ZERO AND DELAYED TIME ZERO
C
 300    CLOSE(1)
        CALL UDSIO('R','F',IFLNM1,1,Y,IERR)
        IF (IERR  .NE.  0) GOTO 27
        CALL VWGEN1(NOPTS,Y,IFLAG1,IED1,YMA,YMI,NFP,NLP)
        IF (IFLAG1  .GT.  4) IFLAG2 = 1
        IF (IFLAG1  .GT.  2 .AND.  ID1  .EQ.  2) IFLAG2 = 1
        GOTO 1100
C
C    INVERTED TIME ZERO AND INVERTED DELAYED TIME ZERO
C
 400    CLOSE(1)
        CALL UDSIO('R','F',IFLNM1,1,Y,IERR)
        IF (IERR  .NE. 0) GOTO 27
        CALL VWGEN2(NOPTS,Y,IFLAG1,IED1,YMA,YMI,NFP,NLP)
        IF (IFLAG1  .GT.  4) IFLAG2 = 1
        IF (IFLAG1  .GT.  2  .AND.  ID1  .EQ.  4 ) IFLAG2 = 1
        GOTO 1100
C
C
C
 1000   CLOSE(1)
 1002   WRITE(6,1005)
 1005   FORMAT(3X,'ENTER: [Y] TO TRY AGAIN',/10X,
     *  '[N] TO TERMINATE THE PROCESSES',/1X,'>>',$)
        READ(5,1007) DUM
 1007   FORMAT(A1)
        IF (DUM  .EQ.  'Y') GOTO 10
        IF (DUM  .EQ.  'N') GOTO 1105
        GOTO 1002
 1105   IFLAG2 = 10
 1100   CONTINUE
        RETURN
        END
C
C  ***********************************************************************
C  *  SUBROUTINE VWGEN1                                                  *
C  ***********************************************************************
        SUBROUTINE VWGEN1(NOPTS,X,IFLAG1,IED2,YMAX,YMIN,NFP,NLP)
        CHARACTER*3 C3
        DIMENSION   X(-1000:10000)
        DIMENSION IED2(10)
        REAL YMAX,YMIN
C
        XCHK = 0.5 *(YMAX-YMIN)+YMIN
```

```
      IFLAG1 = 0
      J = 1
      ISW=0
      DO 100 I = NFP,NLP
        XCHK2 = X(I)
        IF (ISW .EQ. 1 .AND. XCHK2  .LT. XCHK) ISW = 0
        IF (ISW .EQ. 1 .AND. XCHK2  .GT. XCHK) GO TO 100
        IF (ISW .EQ. 0 .AND. XCHK2  .LT. XCHK) GO TO 100
        ISW = 1
        K3 = I
        IED2(J) = K3
        J = J + 1
        RK3=ABS(FLOAT(K3)/10.0)
        RK3=RK3-ABS(FLOAT(INT(RK3)))
        KKK3=JNINT(RK3*10.0)
        C3=' th'
        IF(KKK3.EQ.1) C3=' st'
        IF(KKK3.EQ.2) C3=' nd'
        IF(KKK3.EQ.3) C3=' rd'
        WRITE(6,1) K3,C3
1       FORMAT(3X,'LEADING EDGE AT',I6,A3,' DATA POINT')
        IFLAG1 = IFLAG1 + 1
100   CONTINUE
      CLOSE(1)
      RETURN
      END
C
C  ****************************************************************
C  *   SUBROUTINE VWGEN2                                          *
C  ****************************************************************
      SUBROUTINE VWGEN2(NOPTS,X,IFLAG1,IED2,YMAX,YMIN,NFP,NLP)
      CHARACTER*3 C3
      DIMENSION  X(-1000:10000)
      DIMENSION IED2(10)
      REAL YMIN,YMAX
C
      XCHK = 0.5*(YMAX - YMIN) + YMIN
      ISW = 0
      IFLAG1 = 0
      J = 1
      DO 100 I = NFP,NLP
        XCHK2 = X(I)
        IF (ISW  .EQ.  1  .AND.  XCHK2  .GT.  XCHK)  ISW = 0
        IF (ISW  .EQ.  1  .AND.  XCHK2  .LT.  XCHK)  GOTO 100
        IF (ISW  .EQ.  0  .AND.  XCHK2  .GT.  XCHK)  GOTO 100
        ISW = 1
        K3 = I
        IED2(J) = K3
        J = J+ 1
        RK3=ABS(FLOAT(K3)/10.0)
        RK3=RK3-ABS(FLOAT(INT(RK3)))
        KKK3=JNINT(RK3*10.0)
        C3=' th'
        IF(KKK3.EQ.1) C3=' st'
```

```
          IF(KKK3.EQ.2) C3=' nd'
          IF(KKK3.EQ.3) C3=' rd'
          WRITE(6,1) K3,C3
 1        FORMAT(3X,'TRAILING EDGE AT',I6,A3,' DATA POINT')
          IFLAG1 = IFLAG1 + 1
 100    CONTINUE
        CLOSE(1)
        RETURN
        END

C
C
C   ************************************************************
C   *                                                          *
C   *       SUBROUTINE CHANNEL1                                *
C   *         TIMEZ,INVERTED TIMEZ                             *
C   ************************************************************
      SUBROUTINE  CHA1(E1,IFLAG2)
      DIMENSION Y(-1000:10000)
      INTEGER NPT(5),BOUND(6),E1(10),CHANUMB
      INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
      CHARACTER  IFL*60,DUMMY*1
      CHARACTER *30 CHAFIL(108)
      CHARACTER  JNFIL*30,BLANK,IDUM
      COMMON    /CBY/    Y
     *          /CBE  /  E,NPT,BOUND
     *          /CBNPT/  NPTS
     *          /CBFIL/ CHAFIL
     *          /CBCH /CHANUMB
C
      IFLAG2 = 0
      CHANUMB = 0
      KCH = 0
      INDX=0
      BLANK=' '
      CLOSE(1)
      WRITE(6,700)
  700 FORMAT(/1X,'**  BEGINNING OF INDIVIDUAL WAVEFORM SEPARATION  **')
      WRITE(6,701)
  701 FORMAT(//1X,'NOTE:  THE TOTAL NUMBER OF FILE NAMES YOU ENTER',
     &        /1X,'       MUST BE A MULTIPLE OF TWO. THE FIRST, THIRD,',
     &        /1X,'       FIFTH, etc. FILE MUST BE AN X-COMPONENT FILE',
     &        /1X,'       AND THE SECOND, FOURTH, SIXTH, etc., FILE',
     &        /1X,'       MUST BE A Z-COMPONENT FILE.'//)
   10 WRITE(6,710)
  710  FORMAT(//1X,'ENTER DATA FILE NAME (XXXXXXXXX.XXX)',
     *       /1X,'IF NO MORE FILES, TYPE "CNTR Z"',/1X,'>>',$)
       READ(5,520 ,END = 4000) IFL
  520  FORMAT(A60)
C
C  READING HEADING FROM THE FILE CONTAINING 5 WAVEFORMS
C
      CALL UDSIO('R','F',IFL,1,Y,IERR)
      IF (IERR  .NE.  0 ) THEN
```

```
C
C  ERROR IN INPUT FILE
C
 20     WRITE(6,*)  ' ERROR!!!'
        WRITE(6,*)  '  ***  FILE CANNOT BE ACCESSED ***'
        GOTO 8000
       END IF
       CALL EXTRACT(IFL)
C
C    CORRECT  UDSIO FILE
C     BOUND GENERATION
C
 30    INDX=INDX+1
       CALL GETWAVE(INDX,0,0,NFP,NLP,DEL,'I',Y)
       CHANUMB = CHANUMB + 1
       KCH = KCH + 1
       CHAFIL(KCH) = IFL(1:30)
       BOUND(1)=E1(1)
       IF(E1(1).LT.1) BOUND(1)=1
       BOUND(3)=E1(2)
       BOUND(4)=E1(3)
       BOUND(5)=E1(4)
       NOPTS = (NLP - NFP) + 1
       BOUND(6)=NLP
       NPTS=E1(2)-E1(1)
       BOUND(2)=100.0/125.0*NPTS+E1(1) + 1
       DEL=0.375/(E1(4)-E1(1))
       WRITE(6,*)' DEL=',DEL
       DO 40 I =1,5
         NPT(I)= BOUND(I+1) - BOUND(I)
   40  CONTINUE
       CLOSE(1)
       DO 1000  I = 1,5
         IK = I
         CALL OPNFIL(IK,IFL,JNFIL)
         NFP = 1
         NLP = (NFP + NPT(I)) - 1
C
C  STORING NECESSARY VALUES FOR INDIVIDUAL WAVEFORM
C
         CALL GETWAVE(INDX,BOUND(I),I,NFP,NLP,DEL,'A',Y)
  400    KCH = KCH + 1
         CHAFIL(KCH) = JNFIL
 1000  CONTINUE
       GOTO 10
 4000  CONTINUE
       WRITE(6,*) ' '
       WRITE(6,*) '***   END OF PHASE 1   ***'
       WRITE(6,*) ' '
       GOTO 8010
 8000  CALL CLOSE(1)
 8002  WRITE(6,8005)
 8005  FORMAT(3X,'ENTER: [Y] TO TRY AGAIN',/10X,
     *'[N] TO TERMINATE THE PROCESSES',/1X,'>>',$)
```

```
      READ(5,8007) DUMMY
 8007 FORMAT(A1)
      IF (DUMMY  .EQ. 'Y') GOTO 10
      IF (DUMMY  .EQ. 'N') GOTO 8008
      GOTO 8002
 8008 IFLAG2 = 10
 8010 CONTINUE
      RETURN
      END
C
C  ********************************************************
C  *                                                      *
C  *     SUBROUTINE OPNFIL                                *
C  *                                                      *
C  ********************************************************
C
      SUBROUTINE OPNFIL(IK,IFL,JNFIL)
      DATA CODES  /'F','C','E','V','D'/
      CHARACTER  CODES(5),IFL*60
      CHARACTER  JNFIL*30,BLANK*1
C
      JNFIL=IFL(1:30)
      IDOT=INDEX(JNFIL,'.')
      JNFIL(IDOT+1:IDOT+1)=CODES(IK)
      RETURN
      END
C
C   ********************************************************
C   *                                                      *
C   *       SUBROUTINE CHANNEL2                            *
C   *       DELAYED TIMEZ,INVERTED TIMEZ                   *
C   ********************************************************
      SUBROUTINE  CHA2(E1,IFLAG2)
      DIMENSION Y(-1000:10000)
      INTEGER NPT(3),BOUND(4),E1(10),CHANUMB
      INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
      CHARACTER  IFL*60,DUMMY *1
      CHARACTER  JNFIL*30,BLANK,IDUM
      CHARACTER *30 CHAFIL(108)
      COMMON   /CBY/    Y
     *         /CBE  /  E,NPT,BOUND
     *         /CBNPT/  NPTS
     *         /CBFIL/ CHAFIL
     *         /CBCH / CHANUMB
C
      IFLAG2 = 0
      CHANUMB = 0
      KCH = 0
      INDX=0
      BLANK=' '
      CLOSE(1)
      WRITE(6,700)
  700 FORMAT(/1X,'**  BEGINNING OF INDIVIDUAL WAVEFORM SEPARATION  **')
      WRITE(6,701)
```

```
  701 FORMAT(//1X,'NOTE:  THE TOTAL NUMBER OF FILE NAMES YOU ENTER',
     &        /1X,'       MUST BE A MULTIPLE OF TWO. THE FIRST, THIRD,',
     &        /1X,'       FIFTH, etc. FILE MUST BE AN X-COMPONENT FILE',
     &        /1X,'       AND THE SECOND, FOURTH, SIXTH, etc., FILE',
     &        /1X,'       MUST BE A Z-COMPONENT FILE.'//)
   10 WRITE(6,710)
  710  FORMAT(//1X,'ENTER DATA FILE NAME (XXXXXXXXX.XXX)',
     *       /1X,'IF NO MORE FILES, TYPE "CNTR Z"',/1X,'>>',$)
       READ(5,520 ,END = 4000) IFL
  520  FORMAT(A60)
C
C  READING HEADING FROM THE FILE CONTAINING 5 WAVEFORMS
C
       CALL UDSIO('R','F',IFL,1,Y,IERR)
       IF (IERR  .EQ.  0 ) THEN
C
C  ERROR IN INPUT FILE
C
  20     WRITE(6,*)  ' ERROR!!!'
         WRITE(6,*)  '  ***  FILE CANNOT BE ACCESSED ***'
         GOTO 8000
       END IF
       CALL EXTRACT(IFL)
C
C    CORRECT  UDSIO FILE
C     BOUND GENERATION
C
  30   INDX=INDX+1
       CALL GETWAVE(INDX,0,0,NFP,NLP,DEL,'I',Y)
       CHANUMB = CHANUMB + 1
       KCH = KCH + 1
       CHAFIL(KCH) = IFL(1:30)
       KCH=KCH+3
       BOUND(1)=E1(1)
       IF(E1(1).LT.1) BOUND(1)=1
       BOUND(2)=E1(2)
       NOPTS = (NLP - NFP) + 1
       BOUND(3)=NLP
       DEL=0.125/(E1(2)-E1(1))
       WRITE(6,*)' DEL=',DEL
       DO 40 I =1,2
         NPT(I)= BOUND(I+1) - BOUND(I)
   40 CONTINUE
       CLOSE(1)
       DO 1000  I = 1,2
         IK = I
         CALL OPNFIL1(IK,IFL,JNFIL)
         NFP = 1
         NLP = (NFP + NPT(I)) - 1
C
C  STORING NECESSARY VALUES FOR INDIVIDUAL WAVEFORM
C
         CALL GETWAVE(INDX,BOUND(I),I,NFP,NLP,DEL,'A',Y)
 400     KCH = KCH + 1
```

```
        CHAFIL(KCH) = JNFIL
 1000 CONTINUE
      GOTO 10
 4000 CONTINUE
      WRITE(6,*) '***   END OF PHASE 1   ***'
      GOTO 8010
 8000 CLOSE(1)
 8002 WRITE(6,8005)
 8005 FORMAT(3X,'ENTER: [Y] TO TRY AGAIN',/10X,
     *'[N] TO TERMINATE THE PROCESSES',/1X,'>>',$)
      READ(5,8007) DUMMY
 8007 FORMAT(A1)
      IF (DUMMY  .EQ.  'Y') GOTO 10
      IF (DUMMY  .EQ.  'N') GOTO  8008
      GOTO 8002
 8008 IFLAG2 = 10
 8010 CONTINUE
      RETURN
      END
C
C  ************************************************************
C  *                                                          *
C  *     SUBROUTINE OPNFIL1                                   *
C  *                                                          *
C  ************************************************************
C
      SUBROUTINE OPNFIL1(IK,IFL,JNFIL)
      DATA CODES  /'V','D'/
      CHARACTER  CODES(2),IFL*60
      CHARACTER  JNFIL*30,BLANK*1
C
      JNFIL=IFL(1:30)
      IDOT=INDEX(JNFIL,'.')
      JNFIL(IDOT+1:IDOT+1)=CODES(IK).
      RETURN
      END
C
C    ********************************
C    *                              *
C    *   SUBROUTINE CDNUM           *
C    *                              *
C    ********************************
C
      SUBROUTINE CDNUM(IFLAG2)
      CHARACTER *4 TSTCDNUM
      CHARACTER *30  CHAFIL(108),XFIL
      CHARACTER *4 TST(18)
      CHARACTER BLANK
      INTEGER CHANUMB
C
C  COMMON BLOCKS
C
      COMMON  /CBCH/CHANUMB
     *        /CBFIL/ CHAFIL
```

```
     *        /CBCDNUM/ TSTCDNUM
C
      IFLAG2 = 0
      KX = 1
      KXCH =1
      BLANK = ' '
      KNUMX = CHANUMB
 5    XFIL = CHAFIL(KXCH)
      DO 10 J =1,30
        IF (XFIL(J:J)  .NE.  BLANK) GOTO 20
 10   CONTINUE
 20    TSTCDNUM(1:4) = XFIL(J+1 :J+4)
      TST(KX) =TSTCDNUM
      KX = KX + 1
      KNUMX = KNUMX - 1
      IF (KNUMX  .EQ. 0) GOTO 100
      KXCH = KXCH + 6
      GOTO 5
C
C   COMPARING ALL THE TEST CODE NUMBERS
C
 100  KNUMX =CHANUMB
      TSTCDNUM = TST(1)
      DO 200 I =1,KNUMX
           WRITE(6,101)I,TST(I)
 101       FORMAT(1X,'TEST (',I3,') = ',A4)
           IF (TSTCDNUM  .NE.  TST(I)) GOTO 220
 200  CONTINUE
C
C  NO ERROR
C
      GOTO 300
C
C ERROR
C
 220  WRITE(6,*) 'ERROR!!!'
      WRITE(6,*) '*** DATA FILE NAMES ARE FROM MORE THAN ONE TEST ***'
      WRITE(6,*) '**            PROGRAM TERMINATED                ***'
      IFLAG2 = 10
 300  RETURN
      END


        SUBROUTINE UDSIO(ROW,SOF,FILNAM,LUN,Y,IOS)

C-----------------------------------------------------------------------
C       UDSIO - UDS File Read / Write Utility Routine
C
C       Principal Variables:
C               ROW     -Read or Write flag
C               SOF     -Specs only or Full flag
C               FILNAM  -UDS File Name
C               LUN     -Logical Unit for UDS I/O
C               Y       -Measurement data array
```

```
C               IOS     -Status and message suppression flag
C-----------------------------------------------------------------------

        INTEGER SYS$GETMSG
        DIMENSION Y(-1000:*)
        CHARACTER ROW*1,SOF*1,FILNAM*(*),M*256

        INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'

        IMSG=IOS
        IOS=0

        IF (ROW.EQ.'R') THEN
C           Read a UDS file
            IF (FILNAM.NE.' ') OPEN(LUN,FILE=FILNAM,STATUS='OLD',
     *      FORM='UNFORMATTED',READONLY,IOSTAT=IOS,ERR=104)

            READ(LUN,ERR=105)TSTSRC,TSTNUM,TSTPRF,TSTREF,TSTCFN,CLSSPD,
     *      VEHNO,MAKE,MODEL,YEAR,BODY,ENGINE,VEHTWT,
     *      OCCTYP,OCCAGE,OCCSEX,OCCWT,DUMSIZ,RESTR1,RESTR2,
     *      HIC,T1,T2,CLIP3M,CSI,AIS,
     *      CURNUM,DATTYP,UNITS,AXIS,SENLOC,SENATT,STATUS,FORM,DESC,
     *      NFP,NLP,DEL,INIVEL,PREF,FCUT,FCOR,FSTP,YO,YD,YCAL,
     *      ID1,ID2,ID3,ID4,ID5,RD1,RD2,RD3,RD4,RD5,CD1,CD2

            IF (SOF.NE.'S') THEN
C               Read Measurement Data Also
                IF ((NFP.LT.-1000).OR.(NLP.LT.NFP)) THEN
                    IF (IMSG.NE.-99) WRITE(6,993)
                    IOS=-3
                    RETURN
                END IF

                IF ((NSIZE.GT.0).AND.(NSIZE.LT.NLP)) THEN
                    NLP=NSIZE
                END IF

                IF (NLP.GT.10000) THEN
                    NLP=10000
                END IF

C               Read Measurement Data from Input File
                DO 101 IR=1,(NLP-NFP)/1000+1
                    IS=NFP+1000*(IR-1)
                    READ(LUN,ERR=106)(Y(I),I=IS,MIN0(IS+999,NLP))
  101           CONTINUE

                IF (NFP.GT.0) THEN
                    DO 102 I=0,NFP-1
                        Y(I)=0.0
  102               CONTINUE
                    NFP=0
                ELSE IF (NLP.LT.0) THEN
                    DO 103 I=NLP+1,0
```

```
                    Y(I)=0.0
103             CONTINUE
                NLP=0
            END IF

            IF ((FORM.EQ.'DFT').AND.(NFP.LT.0)) THEN
                NFP=0
            END IF
        END IF
        RETURN

104     IF (IMSG.NE.-99) THEN
            WRITE(6,991)
            CALL ERRSNS(,IRMS,,,)
            IF (IRMS.NE.0) THEN
                IST=SYS$GETMSG(%VAL(IRMS),LEN,M,,)
                IF (LEN.GT.0) WRITE(6,*)M(:LEN)
            END IF
        END IF
        RETURN

105     IF (IMSG.NE.-99) WRITE(6,992)
        IOS=-2
        RETURN

106     IF (IMSG.NE.-99) WRITE(6,995)
        IOS=-4
        RETURN
    ELSE IF (ROW.EQ.'W') THEN
C       Write a UDS File
        IF (FILNAM.NE.' ') THEN
            IF (SOF.NE.'S') THEN
                OPEN(LUN,FILE=FILNAM,STATUS='NEW',FORM='UNFORMATTED',
     *          IOSTAT=IOS,ERR=110)
            ELSE
                OPEN(LUN,FILE=FILNAM,STATUS='UNKNOWN',FORM='UNFORMATTED',
     *          IOSTAT=IOS,ERR=110)
            END IF
        END IF

        IF (SOF.NE.'S') THEN
            IF (NFP.LT.-1000) THEN
                NFP=-1000
            END IF

            IF (NLP.LT.NFP) THEN
                NLP=NFP
            END IF

            IF ((NSIZE.GT.0).AND.(NSIZE.LT.NLP)) THEN
                NLP=NSIZE
            END IF

            IF (NLP.GT.10000) THEN
```

```
                NLP=10000
            END IF

            IF (NFP.GT.0) THEN
                DO 107 I=0,NFP-1
                    Y(I)=0.0
107             CONTINUE
                NFP=0
            ELSE IF (NLP.LT.0) THEN
                DO 108 I=NLP+1,0
                    Y(I)=0.0
108             CONTINUE
                NLP=0
            END IF

            IF ((FORM.EQ.'DFT').AND.(NFP.LT.0)) THEN
                NFP=0
            END IF
        END IF

        WRITE(LUN)TSTSRC,TSTNUM,TSTPRF,TSTREF,TSTCFN,CLSSPD,
     *  VEHNO,MAKE,MODEL,YEAR,BODY,ENGINE,VEHTWT,
     *  OCCTYP,OCCAGE,OCCSEX,OCCWT,DUMSIZ,RESTR1,RESTR2,
     *  HIC,T1,T2,CLIP3M,CSI,AIS,
     *  CURNUM,DATTYP,UNITS,AXIS,SENLOC,SENATT,STATUS,FORM,DESC,
     *  NFP,NLP,DEL,INIVEL,PREF,FCUT,FCOR,FSTP,Y0,YD,YCAL,
     *  ID1,ID2,ID3,ID4,ID5,RD1,RD2,RD3,RD4,RD5,CD1,CD2

        IF (SOF.NE.'S') THEN
            DO 109 IR=1,(NLP-NFP)/1000+1
                IS=NFP+1000*(IR-1)
                WRITE(LUN)(Y(I),I=IS,MIN0(IS+999,NLP))
109         CONTINUE
        END IF
        RETURN

110     IF (IMSG.NE.-99) THEN
            WRITE(6,991)
            CALL ERRSNS(,IRMS,,,)
            IF (IRMS.NE.0) THEN
                IST=SYS$GETMSG(%VAL(IRMS),LEN,M,,)
                IF (LEN.GT.0) WRITE(6,*)M(:LEN)
            END IF
        END IF
        RETURN
    ELSE
        IF (IMSG.NE.-99) WRITE(6,996)
        IOS=-5
        RETURN
    END IF

991 FORMAT(/1X,'*** UDSIO ERROR - File open failure'/)
992 FORMAT(/1X,'*** UDSIO ERROR - Specification data read error'/)
993 FORMAT(/1X,'*** UDSIO ERROR - Illegal NFP and/or NLP value'/)
```

```
  995   FORMAT(/1X,'*** UDSIO ERROR - Measurement data read error'/)
  996   FORMAT(/1X,'*** UDSIO ERROR - Bad Read or Write flag value'/)

        END
C
C  Subroutine GETWAVE is a routine which stores the full (unseparated)
C  set of waveforms and then directs the portion of interest into
C  the array Y when called to do so
      SUBROUTINE GETWAVE(INDX,BOUND,IWF,NFP,NLP,DEL,OT,Y)
      DIMENSION IPT(18,5),INFP(18,5),INLP(18,5),TDEL(18,5)
      DIMENSION YFULL(18,-1000:10000)
      DIMENSION Y(-1000:10000)
      INTEGER BOUND
      CHARACTER*1 OT
      IF (OT.EQ.'I'.OR.OT.EQ.'i') THEN
        DO 20 I=NFP,NLP
          YFULL(INDX,I)=Y(I)
   20   CONTINUE
        GOTO 99
      END IF
      IF (OT.EQ.'A'.OR.OT.EQ.'a') THEN
        IPT(INDX,IWF)=BOUND
        INFP(INDX,IWF)=NFP
        INLP(INDX,IWF)=NLP
        TDEL(INDX,IWF)=DEL
        GOTO 99
      END IF
      IF (OT.EQ.'O'.OR.OT.EQ.'o') THEN
        J=IPT(INDX,IWF)-1
        NFP=INFP(INDX,IWF)
        NLP=INLP(INDX,IWF)
        DEL=TDEL(INDX,IWF)
        DO 30 I=NFP,NLP
          Y(I)=YFULL(INDX,J+I)
   30   CONTINUE
      END IF
   99 RETURN
      END
```

The third segment ,"WGSUBS", contains the following modules:

RESULT
HIC3
PART
EXTRACT
HSINE
ERR2
CRASH
FRSP3
SGAUSS
FAST
FR2TR
FR4TR
FORD1
FORD2
GETIND
THCALC

The source code listing for these modules is:

```
      SUBROUTINE RESULT(DFLAG,IKJNUM,CNTFL,IWF,DELFLAG)
C
C  ***************************************************
C  *                                                 *
C  *   SUBROUTINE RESULT                             *
C  *    OBJECTIVE :TAKE THE SQRT (SUM OF SQURES OF X *
C  *               AND Z COMPONENT OF HALF-SINE AND  *
C  *               CRASH DATA FILE) TO PRODUCE THE   *
C  *               RESULTANT FILE.                   *
C  * PRINCIPAL VARIABLES:                            *
C  *    NOPTS  - NO. OF POINTS FOR THE DATA SET      *
C  *    RES    - RESULTANT VECTOR                    *
C  *    FILNAM - INPUT FILE NAMES (X,Y,Z)            *
C  *    OUTFIL - OUTPUT FILE NAME (RESULTANT)        *
C  ***************************************************
C
C   DECLARATION OF VARIABLES
C
        INCLUDE '[ASGPROG]COMVAR.LIS'
        REAL RES(-1000:4500),Y(-1000 :10000),Y1(-1000:10000)
        CHARACTER *30 FILNAM(3)
        CHARACTER BLANK*1,OUTFIL *30,DUMFILE *30,DUMMY *1
        CHARACTER DUMFIL1*30
        CHARACTER AXIS3(2)
        INTEGER NOPTS,SETNUM,CHANUMB,CNTFL,DELFLAG
        REAL XMIN,XMAX,YMIN,YMAX
        CHARACTER *30 CHAFIL(108),RESFIL(72)
        CHARACTER DUMFILE1 *30,RESF*30
        DATA AXIS3/'X','Z'/
C
C    COMMON BLOCKS
C
        COMMON  /CBFIL/ CHAFIL
     *          /CBRES /RESFIL
     *          /CBCH/CHANUMB
        COMMON/RSLTNTS/RESPF(2,9,-1000:10000),NFPRES(2,9),NLPRES(2,9)
C
C   START OF PROGRAM
C
        DFLAG = 0
        KJ = 0
        KNUM = CHANUMB
        SETNUM = 0
        BLANK =' '
        KKJNUM=0
        ISLT=1
        IF (IWF.EQ.5) ISLT=2
C
C    SUMMARY REPORT FILE
C
        IF ((DELFLAG.EQ.1).OR.((DELFLAG.EQ.0).AND.(IWF.EQ.2)))
     &       OPEN(4,FILE='RESULT.SUM',STATUS='NEW',ERR=1000)
        WRITE (4,400)
  400   FORMAT(/34X,'** RESULTANT SUMMARY REPORT **')
```

```
C
C   WRITE PROGRAM OUTPUT HEADER
C
 20     CONTINUE
        IF (KNUM  .EQ. 0  .AND.  CNTFL  .EQ. 1) GOTO 300
        SETNUM = SETNUM +1
        WRITE(6,21)SETNUM
 21     FORMAT(1X,'RESULT FILE FOR SET NUMBER ',I4)
C
C   BEGIN PROCESSING
C
        DO 30 I=-1000,10000
                IF (I.LT.4501) RES(I) = 0.0
                RESPF(ISLT,SETNUM,I)=0.0
 30     CONTINUE
C
C   INPUT FILE NAMES
C
        DO 50 L=1,2
          IKJNUM = IKJNUM + 6
          FILNAM(L) = CHAFIL(IKJNUM)
          KJ = KJ + 1
          RESFIL(KJ) = FILNAM(L)
          DUMFIL1 = FILNAM(L)
 522      DO 523  I = -1000,10000
            Y(I) = 0.0
 523      CONTINUE
C
C   READ DATA FROM ARRAY (X-AXIS ON THE FIRST PASS,
C                         Z-AXIS ON THE SECOND PASS)
C
          KKJNUM=KKJNUM+1
          CALL GETWAVE(KKJNUM,K,IWF,NFP,NLP,DEL,'O',Y)
 525      NOPTS = (NLP - NFP) + 1
          KNUM = KNUM - 1
          IF(NLP .GT. 4500) NLP=4500
          DO 70 I=NFP,NLP
            RES(I) = RES(I)+(Y(I)*Y(I))
 70       CONTINUE
          CALL CLOSE(1)
 50     CONTINUE
C
C   PERFORM CALCULATION
C
        DO 100 I=NFP,NLP
                RES(I) = SQRT(RES(I))
                RESPF(ISLT,SETNUM,I) = RES(I)
 100    CONTINUE
        NLPRES(ISLT,SETNUM)=NLP
        NFPRES(ISLT,SETNUM)=NFP
C
C   OUTPUT RESULTANT FILE
C
 540  K = 0
```

```
      DO 545 J=1,26
       IF(DUMFIL1(J:J)  .EQ. ' ') GOTO 545
      K = K + 1
      RESF(K:K) = DUMFIL1(J:J)
 545  CONTINUE
      DO 550 J =1,26
      IF (RESF(J:J) .EQ. '.') GOTO 555
 550  CONTINUE
 555  RESF(J+1 : J+1) = 'R'
      RESF(J+2 : J+2) = 'E'
      RESF(J+3 : J+3) = 'S'
      OUTFIL = RESF
      KJ = KJ + 1
      RESFIL(KJ) = OUTFIL
C
C   ADD ENTRIES TO REPORT
C
        WRITE(4,1100)  SETNUM
 1100   FORMAT(/2X,'SET NO : ',2X,I3)
        WRITE(4,1200)  FILNAM(1),FILNAM(2)
 1200   FORMAT(2X,'INPUT FILES(X AND Z COMPONENT) : ',5X,
     1  A30,10X,A30)
        WRITE(4,1300)  OUTFIL
 1300   FORMAT(2X,'RESULTANT FILE : ',5X,A30)
        GO TO 20
 300    CONTINUE
        IF (IWF.EQ.5) CALL CLOSE(4)
        GOTO 2000
 1000   WRITE(6,*) 'ERROR !!!'
        WRITE(6,*) ' ** "RESULT.SUM" CANNOT BE CREATED **'
        WRITE(6,*) 'PLEASE TRY AGAIN'
        DFLAG = 1
 2000   CONTINUE
        RETURN
        END
C
        SUBROUTINE HIC3(DFLAG,CNTFL,ISLT)
C-----------------------------------------------------------------------
C       Objective:
C               To find the maximum Head Injury Criterion value
C               and the corresponding time interval using an
C               efficient global branch and bound algorithm.
C
C       Programmer:
C               ASGI - S. Mentzer       7/83
C               Latest update           3/84
C               UDSIO calls removed    11/86
C
C       Major variables:
C               R(I)    - Resultant acceleration array, I=0,1,...,NLP
C               S(I)    - Summed resultant array (integral)
C               JL(I)   - Lower J boundary index array
C               KL(I)   - Lower K boundary index array
C               FMAX    - Incumbent/max value of HIC subfunction F
```

```
C               JMAX    - J index of FMAX value
C               KMAX    - K index of FMAX value
C-----------------------------------------------------------------------
        DIMENSION R(-1000:10000),S(0:10000),JL(15000),KL(15000)
        DOUBLE PRECISION RC,RM,SC,SM
        CHARACTER INFIL*30,CURDAT*9,DUMMY*1
        INTEGER CNTFL,DFLAG,KNUM,CHANUMB,SETNUM
        INCLUDE '[ASGPROG]COMVAR.LIS'
        CHARACTER *30 RESFIL(72)
        COMMON   /CBRES/RESFIL
     *            /CBCH/CHANUMB
        COMMON/RSLTNTS/RESPF(2,9,-1000:10000),NFPRES(2,9),NLPRES(2,9)
        COMMON/RHICS/RHIC(2,9),RT1(2,9),RT2(2,9)
        DATA NAR/15000/
C
        DFLAG = 0
        IKJ = 0
        KNUM = CHANUMB
        SETNUM = 0
        CALL DATE(CURDAT)
C       OPEN(3,FILE='HIC3.REP',STATUS='NEW',ERR = 1000)
C       WRITE(3,931)CURDAT
C  931  FORMAT(//1X,55X,'HIC3 REPORT',30X,A/1X,54X,13(1H-)///
C     * 1X,2X,'TSTREF',5X,'VEHICLE',22X,'OCCUPANT',3X,'HIC',
C     * 4X,'T1 (msec)',1X,'T2 (msec)',2X,'T2 - T1',3X,'FILE'/
C     * 1X,10(1H-),2X,29(1H-),1X,8(1H-),1X,8(1H-),1X,9(1H-),1X,
C     * 9(1H-),1X,9(1H-),2X,40(1H-))
 10     CONTINUE
        IF (KNUM .EQ. 0 .AND. CNTFL  .EQ. 1) GOTO 400
        SETNUM = SETNUM + 1
        WRITE(6,*) 'CALCULATING  *HIC*  FOR SET NUMBER',SETNUM
 509    IKJ = IKJ + 3
        INFIL = RESFIL(IKJ)
        DO 516 I = -1000,10000
         R(I) = RESPF(ISLT,SETNUM,I)
 516    CONTINUE
 512    KNUM = KNUM -2
        NFP=NFPRES(ISLT,SETNUM)
        NLP=NLPRES(ISLT,SETNUM)
C  Compute sum (integral) array using trapezoid rule
        S(0)=0.0
        SM=0.D0
        RM=ABS(R(0))
        DO 102 I=1,NLP
           RC=ABS(R(I))
           SC=SM+.5*(RC+RM)
           S(I)=SC
           SM=SC
           RM=RC
  102   CONTINUE
C       Initialize variables
        FMAX=0.0
        JMAX=0
        KMAX=0
```

```
      JL(1)=0
      KL(1)=-1
      NWU=NLP
      IBR=1
      IER=1
      INC=-1
      IBL=NAR
C  Begin a round of region evaluations
  103 IL=IBL-INC
      DO 104 IR=IBR,IER,INC
         JLI=JL(IR)
         IF (JLI.GT.0) THEN
            NJ=NWU
         ELSE
            JLI=ABS(JLI)
            NJ=MAX0(NWU-1,0)
         END IF
         KLI=KL(IR)
         IF (KLI.GT.0) THEN
            NK=NWU
         ELSE
            KLI=ABS(KLI)
            NK=MAX0(NWU-1,0)
         END IF
         KUI=KLI+NK
         IF ((NJ.EQ.0).AND.(NK.EQ.0)) THEN
C  This is a single point region - Evaluate
            F=(S(KUI)-S(JLI))*(KUI-JLI)**(-.6)
            IF (F.GT.FMAX) THEN
               FMAX=F
               JMAX=JLI
               KMAX=KUI
            END IF
         ELSE
C           This is a finite region with feasible points - Bound
            JUI=JLI+NJ
            FS=S(KUI)-S(JLI)
            IF (FS*MAX0(KLI-JUI,1)**(-.6).GT.FMAX) THEN
C              Region is possibly optimal - Evaluate and partition
               F=FS*(KUI-JLI)**(-.6)
               IF (F.GT.FMAX) THEN
                  FMAX=F
                  JMAX=JLI
                  KMAX=KUI
               END IF
               IF ((IL+4*INC)*INC.GT.IR*INC) THEN
                  WRITE(6,993)
  993             FORMAT(/1X,'*** ERROR - HIC3 array size exceeded'/)
                  CALL CLOSE(3)
                  DFLAG =1
                  GOTO 670
               END IF
               CALL PART(JLI,JUI,NJ,NWU-NJ,KLI,KUI,NK,NWU-NK,IL,INC,
     *           JL,KL)
```

```
            END IF
          END IF
  104   CONTINUE
        IF (ABS(IBL-IL-INC).GT.0) THEN
C  Some regions remain - Set up boundary index arrays
          ITMP=IBL
          IBL=IER
          IER=ITMP
          IBR=IL
          INC=-1*INC
          NWU=NWU/2
          GO TO 103
        ELSE
C  Done - No regions remain - Report results
          HIC=DEL*FMAX**2.5
          T1=1000*JMAX*DEL
          T2=1000*KMAX*DEL
          RHIC(ISLT,SETNUM)=HIC
          RT1(ISLT,SETNUM)=T1
          RT2(ISLT,SETNUM)=T2
C    Reporting of HIC (normally done here) has been removed
          CALL EXTRACT(INFIL)
          CLOSE(2)
          GOTO 10
        END IF
  400   CONTINUE
        CALL CLOSE(3)
        GOTO 670
 1000   WRITE(6,*) 'ERROR !!!!'
        WRITE(6,*) '*** "HIC.REP" CANNOT BE OPENED ***'
        DFLAG = 1
  670   CONTINUE
        RETURN
        END
C
C
        SUBROUTINE PART(JLI,JUI,NJ,NJD,KLI,KUI,NK,NKD,IL,INC,JL,KL)
C
C       Performs partitioning of possibly optimal regions
C
        DIMENSION JL(*),KL(*)
C
        J2M=2*MOD(NJ,2)
        NJ2=NJD*J2M
        JMU=(JUI-NJ/2)*(1-NJ2)
        K2M=2*MOD(NK,2)
        NK2=NKD*K2M
        KMU=(KUI-NK/2)*(1-NK2)
C  First region
        IL=IL+INC
        JL(IL)=JMU
        KL(IL)=KMU
C  Second region if finite width along J axis
        IF (NJ.NE.0) THEN
```

```
            JLI=JLI*(J2M-NJ2-1)
            IL=IL+INC
            JL(IL)=JLI
            KL(IL)=KMU
         END IF
         IF (NK.NE.0) THEN
            KLI=KLI*(K2M-NK2-1)
C  Third region if finite width along J and K axes
            IF (NJ.NE.0) THEN
               IL=IL+INC
               JL(IL)=JLI
               KL(IL)=KLI
            END IF
C  Fourth region if feasible and finite width along K axis
            IF (ABS(KMU)-1.GT.ABS(JMU)) THEN
               IL=IL+INC
               JL(IL)=JMU
               KL(IL)=KLI
            END IF
         END IF
         RETURN
         END
C
C
         SUBROUTINE EXTRACT(FILNAM)
         CHARACTER*(*) FILNAM
C
         LOC=INDEX(FILNAM,':')+1
         FILNAM=FILNAM(LOC:)
         LOC=INDEX(FILNAM,']')+1
         FILNAM=FILNAM(LOC:)
         LOC=INDEX(FILNAM,';')
         IF (LOC.GT.0) FILNAM=FILNAM(:LOC-1)
         RETURN
         END

C
C     ****************************************
C     *   HALF-SINE PROCESSING               *
C     ****************************************
C
       SUBROUTINE HSINE(DFLAG,CNTFL)
C
C    DECLERATION OF VARIABLES
C
       INTEGER CNTFL,CHANUMB,DFALG,SETNUM
       INTEGER EFLAG(18,4),IND(4),NOPTS
       CHARACTER *4 TSTCDNUM
       CHARACTER *30 HSNE1,HSNE2,HSNE3
       CHARACTER *30 IFLNM2(18)
       CHARACTER * 2 C(18)
       CHARACTER DAY*9,HOUR*9
       CHARACTER *30 RESFIL(72)
       CHARACTER IFLNM1*30,CHANUM*2,DUMMY *1
```

```
      REAL Y(-1000 :10000),NPMS,YMN,YMX
C
C  COMMON BLOCKS
C
      INCLUDE '[ASGPROG]COMVAR.LIS'
      COMMON /CBRES/ RESFIL
     *       /CBCH/CHANUMB
     *       /CBCDNUM/TSTCDNUM
      COMMON /HSTMX/TOMX,YMXX,TOMZ,YMNZ,TCLEV1,TCLEV2,PCT,TDACLX,TDACLZ
      COMMON/RSLTNTS/RESPF(2,9,-1000:10000),NFPRES(2,9),NLPRES(2,9)
      COMMON/RHICS/RHIC(2,9),RT1(2,9),RT2(2,9)
      COMMON/HSSTAT2/HSDEVX(18),HSDEVZ(18)
      COMMON/HSSTAT3/HSHDEV(18)
C
C HALF-SINE WAVEFORM PROCESSOR
C
      DFLAG = 0
      SETNUM = 0
      KNUM = CHANUMB
      N=0
      IKJ = 0
      KKJ=0
      DO 30 I=1,18
      DO 30 J=1,4
      EFLAG(I,J)=0
   30 CONTINUE
C
C   REPORT FILE
C
      HSNE1 ='HSNE'//TSTCDNUM(1:4)//'.SUM'
      HSNE2 ='HSNE'//TSTCDNUM(1:4)//'.001'
      HSNE3 ='HSNE'//TSTCDNUM(1:4)//'.002'
      OPEN(3,FILE=HSNE1,STATUS='NEW',ERR=3000)
      OPEN(7,FILE=HSNE2,STATUS='NEW',ERR=3000)
      OPEN(8,FILE=HSNE3,STATUS='NEW',ERR=3000)
      WRITE(3,210)
  210 FORMAT(43X,'** HALF-SINE WAVEFORM PROCESSOR DATA SUMMARY **'//
     1       1X,'THE * SYMBOL INDICATES THAT THE ALLOWABLE LIMIT HAS '
     2       ,'BEEN EXCEEDED.'//1X,'TIME QUANTITIES ARE EXPRESSED IN '
     3       ,'MILLISECONDS')
      CALL DATE(DAY)
      CALL TIME(HOUR)
      WRITE(3,9995) DAY,HOUR
 9995 FORMAT(//,50X,A9,10X,A9)
  700 CONTINUE
      IF (KNUM .EQ. 0 .AND. CNTFL .EQ. 1) GOTO 800
      SETNUM = SETNUM + 1
      WRITE(6,*) 'PROCESSING SET NUMBER ',SETNUM
      DO 35 I=1,4
   35 IND(I)='  '
C
C PROCESS HALF-SINE "X" COMPONENT
C
 701  IKJ = IKJ + 1
```

```
      KKJ=KKJ+1
      IFLNM1 = RESFIL(IKJ)
 705  DO 706  I=-1000,10000
       Y(I) = 0.0
 706  CONTINUE
C
C  READING "X" COMPONENT INPUT FILE(DATA,HEAD)
C
      CALL GETWAVE(KKJ,0,2,NFP,NLP,DEL,'O',Y)
 6    N=N+2
      KNUM = KNUM - 1
      IFLNM2(N-1)=IFLNM1
C
C   GETTING THE CHANNEL NUMBER
C
      DO 900 I = 1,30
      IF (IFLNM1(I:I)  .EQ.  '.') GOTO 910
 900  CONTINUE
 910  CHANUM(1:1) = IFLNM1(I+2 : I+2)
      CHANUM(2:2) = IFLNM1(I+3 : I+3)
      C(N-1) = CHANUM
      NPMS=1.0/DEL/1000.
C
C DETERMINE ARRAY NO. WHICH CORRESPONDS TO THE MINIMUM VALUE IN DATA
C
      NOPTS = (NLP - NFP) + 1
      PCT=20.0
      CALL GETIND(NFP,NLP,PCT,Y,DEL,NX,TOMX,DNX,YMXX,TDACLX)
C
C PROCESS HALF-SINE "Z" COMPONENT
C
      CALL CLOSE(1)
 801  IKJ = IKJ + 1
      KKJ=KKJ+1
      IFLNM1 = RESFIL(IKJ)
 805  DO 807 I =-1000,10000
       Y(I) = 0.0
 807  CONTINUE
C
C  READING "Z" COMPONENT INPUT FILE(DATA,HEAD)
C
      CALL GETWAVE(KKJ,0,2,NFP,NLP,DEL,'O',Y)
 18   IFLNM2(N)=IFLNM1
      KNUM = KNUM - 1
C
C  GETTING THE CHANNEL NUMBER
C
      I=INDEX(IFLNM1,'.')
 960  CHANUM(1:1) = IFLNM1(I+2 : I+2)
      CHANUM(2:2) = IFLNM1(I+3 : I+3)
      C(N) =CHANUM
C
C DETERMINE ARRAY NO. WHICH CORRESPONDS TO THE MAXIMUM VALUE IN DATA
C
```

```
      NOPTS = (NLP - NFP) +1
      PCT=20.0
      CALL GETIND(NFP,NLP,PCT,Y,DEL,NZ,TOMZ,DNZ,YMNZ,TDACLZ)
C
C COMPUTE TIME DEVIATION FROM THEORETICAL AND CHANNEL-TO-CHANNEL TIME DIFFERENCE
C
      CALL CLOSE(1)
      DIF=TOMX-TOMZ
      CCTD=DIF/1000.
      IF(ABS(DIF).LE.0.1) GOTO 200
      EFLAG(N-1,1)=1
      EFLAG(N,1)=1
      IND(1)=' *'
  200 DEVX=TOMX-12.5
      DEVZ=TOMZ-12.5
      IF(ABS(DEVX).GT.1.0) EFLAG(N-1,2)=1
      IF(ABS(DEVZ).GT.1.0) EFLAG(N,3)=1
      IF(ABS(DEVX).GT.1.0) IND(2)=' *'
      IF(ABS(DEVZ).GT.1.0) IND(3)=' *'
      WRITE(3,11) IFLNM2(N-1),C(N-1),IFLNM2(N),C(N)
   11 FORMAT(//1X,'RESULTS FOR "X" COMPONENT FILE : ',A30,5X,
     1       'CHANNEL NO. ',A2,/9X,'AND "Z" COMPONENT FILE : ',
     2       A30,5X,'CHANNEL NO. ',A2)
      WRITE(3,13)   TSTPRF
   13 FORMAT(/1X,'FACILITY : ',A25)
      WRITE(3,1000) DIF,IND(1)
 1000 FORMAT(/1X,'1. CHANNEL-TO-CHANNEL TIME DIFFERENCE = ',F7.3,1A2,
     1       1X,'(ALLOWABLE LIMIT = +/- 0.1 MSEC)')
      WRITE(3,1010) DEVX,IND(2)
 1010 FORMAT(/1X,'2. "X" COMPONENT TIME DEVIATION = ',F7.3,1A2,
     1       1X,'(ALLOWABLE = +/- 1.0 MSEC)')
      WRITE(3,1020) DEVZ,IND(3)
 1020 FORMAT(/1X,'3. "Z" COMPONENT TIME DEVIATION = ',F7.3,1A2,
     1       1X,'(ALLOWABLE LIMIT = +/- 1.0 MSEC)')
      HSDEVX(SETNUM)=DEVX
      HSDEVZ(SETNUM)=DEVZ
      WRITE(7,1001)DEVX,DEVZ
 1001 FORMAT(2X,F7.3,5X,F7.3)
 810  IKJ = IKJ +1
      IFLNM1 = RESFIL(IKJ)
 820  DO 825  I=-1000,10000
      Y(I) = RESPF(1,SETNUM,I)
 825  CONTINUE
      HIC=RHIC(1,SETNUM)
      T1=RT1(1,SETNUM)+100
      T2=RT2(1,SETNUM)+100
      T2T1=T2-T1
      HICEXP=0.0
      TT1=0.0
      TT2=0.0
      CALL THCALC(CCTD,HICEXP,TT1,TT2)
      T21EXP=TT2-TT1
      HICDEV=(HIC/HICEXP-1)*100.0
      IF(ABS(HICDEV).LE.6.0) GOTO 51
```

```
      EFLAG(N-1,4)=1
      EFLAG(N,4)=1
      IND(4)=' *'
   51 WRITE(3,31) HICDEV,IND(4)
   31 FORMAT(/1X,'4. HIC DEVIATION = ',F6.2,1A2,
     1       ' (ALLOWABLE LIMIT = +/- 6%)')
      HSHDEV(SETNUM)=HICDEV
      WRITE(8,1002)HICDEV
 1002 FORMAT(2X,F6.2)
      WRITE(3,110) HIC,T1,T2,T2T1
  110 FORMAT(/1X,'THE CALCULATED HIC NO. = ',F6.1,' FOR T1 = ',F7.3,
     1       ' AND T2 = ',F7.3,' WITH T2-T1 = ',F7.3)
      WRITE(3,20) HICEXP,T21EXP
   20 FORMAT(/1X,'THE THEORETICAL HIC NO. = ',F6.1,' AND THE ',
     1       'THEORETICAL VALUE OF T2-T1 = ',F7.3)
      GOTO 700
  800 WRITE(3,1060)
 1060 FORMAT(//5X,'FILE NAME',19X,'CHANNEL NO.',20X,'STATUS')
      NA=0
      NU=0
      DO 160 I=1,N
      KND=0
      DO 170 J=1,4
      IF(EFLAG(I,J).EQ.0) GOTO 170
      KND=1
  170 CONTINUE
      IF(KND.EQ.0) NA=NA+1
      IF(KND.EQ.1) NU=NU+1
      IF(KND.EQ.0) WRITE(3,1070)  IFLNM2(I),C(I)
      IF(KND.EQ.1) WRITE(3,1080)  IFLNM2(I),C(I)
 1070 FORMAT(1X,A30,10X,A2,25X,'PASS')
 1080 FORMAT(1X,A30,10X,A2,25X,'FAIL')
  160 CONTINUE
      WRITE(3,1050) N,NA,NU
 1050 FORMAT(//1X,'TOTAL NO. OF CHANNELS PROCESSED = ',I2
     1       /1X,'NUMBER OF ACCEPTABLE CHANNELS = ',I2
     2       /1X,'NUMBER OF UNACCEPTABLE CHANNELS = ',I2)
      CALL CLOSE(3)
      GOTO 9999
 3000 WRITE(6,*) 'ERROR !!!'
      WRITE(6,*) ' ** "HSINE.SUM" CANNOT BE CREATED **'
 9999 CONTINUE
      CALL CLOSE(7)
      CALL CLOSE(8)
      RETURN
      END
C
        SUBROUTINE ERR2(DUMMY)
C
        CHARACTER DUMMY*1
C
        CALL CLOSE(1)
 8002   WRITE(6,8005)
 8005   FORMAT(3X,'ENTER: [Y] TO TRY AGAIN',/10X,
```

```
     *  '[N] TO TERMINATE THE PROCESSES',/1X,'>>',$)
        READ(5,8007) DUMMY
 8007   FORMAT(A1)
        IF (DUMMY  .EQ.  'Y'  .OR.  DUMMY  .EQ.  'N') GOTO 8010
        GOTO 8002
 8010   RETURN
        END
C
C     ************************************************************
C     *   SUBROUTINE CRASH                                       *
C     ************************************************************
C
        SUBROUTINE CRASH(DFLAG,CNTFL)
C
C  VARIABLE DECLARATION
C
      INTEGER EFLAG(18),IND,NOPTS,DFLAG,CNTFL
      INTEGER SETNUM,CHANUMB
      CHARACTER *30 CRASHFI1,CRASHFI2
      CHARACTER *4 TSTCDNUM
      CHARACTER *30 IFLNM2(18)
      CHARACTER *2 C(18)
      CHARACTER  IFLNM1 *30,CHANUM *2,DUMMY *1
      REAL Y(-1000 :10000),NPMS,YMN,YMX
      CHARACTER *30 RESFIL(72)
      CHARACTER DAY*9,HOUR*9
C
      INCLUDE '[ASGPROG]COMVAR.LIS'
      COMMON    /CBRES/RESFIL
     *          /CBCH/CHANUMB
     *          /CBCDNUM/TSTCDNUM
      COMMON/RSLTNTS/RESPF(2,9,-1000:10000),NFPRES(2,9),NLPRES(2,9)
      COMMON/RHICS/RHIC(2,9),RT1(2,9),RT2(2,9)
      COMMON/CSTAT2/CHDEV(18)
C
C CRASH WAVEFORM PROCESSOR
C
      DFLAG = 0
      SETNUM = 0
      KNUM = CHANUMB
      IKJ = 0
      KKJ=0
      N=0
      DO 30 I=1,18
      EFLAG(I)=0
   30 CONTINUE
C
C   REPORT FILE
C
      CRASHFI1 ='CRASH'//TSTCDNUM(1:4)//'.SUM'
      CRASHFI2 ='CRASH'//TSTCDNUM(1:4)//'.001'
      OPEN(3,FILE=CRASHFI1,STATUS='NEW',ERR= 3000)
      OPEN(7,FILE=CRASHFI2,STATUS='NEW',ERR= 3000)
      WRITE(3,210)
```

```
  210 FORMAT(43X,'** CRASH WAVEFORM PROCESSOR DATA SUMMARY **'//
     1       1X,'THE * SYMBOL INDICATES THAT THE ALLOWABLE LIMIT HAS '
     2       ,'BEEN EXCEEDED.'//1X,'TIME QUANTITIES ARE EXPRESSED IN '
     3       ,'MILLISECONDS')
       CALL DATE(DAY)
       CALL TIME(HOUR)
       WRITE(3,9995) DAY,HOUR
 9995 FORMAT(//,50X,A9,10X,A9)
  700 CONTINUE
       IF (KNUM .EQ. 0 .AND. CNTFL .EQ. 1) GOTO 800
       SETNUM = SETNUM + 1
       WRITE(6,*) 'PROCESSING SET NUMBER ',SETNUM
   35 IND='  '
C
C READ CRASH "X" COMPONENT DATA
C
 701   IKJ = IKJ + 1
       KKJ=KKJ+1
       IFLNM1 = RESFIL(IKJ)
 705   DO 706 I =-1000,10000
        Y(I) = 0.0
 706   CONTINUE
C
C   READ X-AXIS DATA FROM ARRAY
C
       CALL GETWAVE(KKJ,0,5,NFP,NLP,DEL,'O',Y)
 6     N=N+2
       KNUM = KNUM -1
       IFLNM2(N-1)=IFLNM1
C
C  GETTING THE CHANNEL NUMBER FROM INPUT FILE
C
       I=INDEX(IFLNM1,'.')
 910   CHANUM(1:1) = IFLNM1(I+2 : I+ 2)
       CHANUM(2:2) = IFLNM1(I+3 : I+ 3)
       C(N-1) = CHANUM
       CALL CLOSE(1)
       NPMS=1.0/DEL/1000.
C
C READ CRASH "Z" COMPONENT DATA
C
 921   IKJ = IKJ + 1
       KKJ=KKJ+1
       IFLNM1 = RESFIL(IKJ)
 925   DO 926 I=-1000,10000
        Y(I) = 0.0
 926   CONTINUE
C
C   READ Z-AXIS DATA FROM ARRAY
C
       CALL GETWAVE(KKJ,0,5,NFP,NLP,DEL,'O',Y)
   60  IFLNM2(N)=IFLNM1
       KNUM = KNUM -1
C
```

```
C   GETTING THE CHANNEL NUMBER
C
      I=INDEX(IFLNM1,'.')
 9600 CHANUM(1:1) = IFLNM1(I+2 : I+2)
      CHANUM(2:2) = IFLNM1(I+3 : I+3)
      C(N) = CHANUM
 980  CONTINUE
      CALL CLOSE(1)
      WRITE(3,11) IFLNM2(N-1),C(N-1),IFLNM2(N),C(N)
   11 FORMAT(//1X,'RESULTS FOR "X" COMPONENT FILE : ',A30,5X,
     1       'CHANNEL NO. ',A2/9X,'AND "Z" COMPONENT FILE : ',
     2       A30,5X,'CHANNEL NO. ',A2)
      WRITE(3,13)  TSTPRF
   13 FORMAT(/1X,'FACILITY : ',A25)
 950  IKJ = IKJ + 1
      IFLNM1= RESFIL(IKJ)
 960  DO 961 I =-1000,1000
       Y(I) = RESPF(2,SETNUM,I)
 961  CONTINUE
      HIC=RHIC(2,SETNUM)
      T1=RT1(2,SETNUM)+375
      T2=RT2(2,SETNUM)+375
      T2T1=T2-T1
      HICEXP=930
      HICDEV=(HIC/HICEXP-1)*100.0
      IF(ABS(HICDEV).LE.6.0) GOTO 51
      EFLAG(N-1)=1
      EFLAG(N)=1
      IND=' *'
   51 WRITE(3,31) HICDEV,IND
   31 FORMAT(/1X,'1. HIC DEVIATION = ',F10.2,1A2,
     1       ' (ALLOWABLE LIMIT = +/- 6%)')
      CHDEV(SETNUM)=HICDEV
      WRITE(7,1061)HICDEV
 1061 FORMAT(2X,F10.2)
      WRITE(3,110) HIC,T1,T2,T2T1
  110 FORMAT(/1X,'THE CALCULATED HIC NO. = ',F6.1,' FOR T1 = ',F7.3,
     1       ' AND T2 = ',F7.3,' WITH T2-T1 = ',F7.3)
      WRITE(3,20) HICEXP
   20 FORMAT(/1X,'THE THEORETICAL HIC NO. = ',F6.1)
      GOTO 700
  800 WRITE(3,1060)
 1060 FORMAT(//5X,'FILE NAME',19X,'CHANNEL NO.',20X,'STATUS')
      NA=0
      NU=0
      DO 160 I=1,N
      KND=0
      IF(EFLAG(I).EQ.0) GOTO 170
      KND=1
  170 IF(KND.EQ.0) NA=NA+1
      IF(KND.EQ.1) NU=NU+1
      IF(KND.EQ.0) WRITE(3,1070)   IFLNM2(I),C(I)
      IF(KND.EQ.1) WRITE(3,1080)   IFLNM2(I),C(I)
 1070 FORMAT(1X,A30,10X,A2,25X,'PASS')
```

```
 1080 FORMAT(1X,A30,10X,A2,25X,'FAIL')
  160 CONTINUE
      WRITE(3,1050) N,NA,NU
 1050 FORMAT(//1X,'TOTAL NO. OF CHANNELS PROCESSED = ',I2
     1       /1X,'NUMBER OF ACCEPTABLE CHANNELS = ',I2
     2       /1X,'NUMBER OF UNACCEPTABLE CHANNELS = ',I2)
      CALL CLOSE(3)
      CALL CLOSE(7)
      GOTO 9999
 3000 WRITE(6,*)  'ERROR !!!'
      WRITE(6,*)  ' *** "CRASH.SUM" CANNOT BE CREATED ***'
 9999 CONTINUE
      RETURN
      END
C
        SUBROUTINE FRSP3(DFLAG,CNTFL)
C-------------------------------------------------------------------------
C       PROGRAM FRSP3:                          John E. Nickles
C                                               Transportation Systems Center
C                                               23-October-1984
C
C       Last Revision Date: 3-October-1985
C
C       Abstract:
C
C               This program calculates the frequency response of a system
C       when the input to the system is a sum-of-sines waveform.  The
C       frequency response is determined by calculating the ratio of two
C       DFT outputs at the signal frequencies.  The two DFTs are the DFT
C       of the recorded system output and the DFT of the reconstructed
C       system input.  The input to the program is a raw, time series data
C       set  of the recorded system output signal.  The system input data can
C       be recreated if the waveform specifications and the sampling fre-
C       quency, with which the system output data set was made, are known.
C       The data is windowed before the DFTs are calculated.  The program
C       uses the FAST FFT algorithm to compute the DFTs.
C
C       Subroutines Called:
C
C       FAST            FFT
C       SGAUSS          Gaussian window
C
C       Revision History:
C
C       11-September-1985  Added scale factor, CSC
C       3-October-1985     Integrated FRSP3 with waveform generator signal
C                          processing software from MGA Research Corp.
C
C-------------------------------------------------------------------------
C
        REAL A(-1000:10000),B(2050),C(2050),AR(20),F(20),Y(20),Z(20),
     *  FDFT(20),ARDB(20),U(0:1024),V(0:1024),W(2048)
        INTEGER LN(20)
    CHARACTER*30 OUTFIL,FRFIL,SUMFIL,STTFIL,CHAFIL(108)
```

```
      CHARACTER*9 DAY,HOUR,DUMMY*1
      INTEGER CNTFL,DFLAG,CHANUMB,SETNUM
C
      COMMON  /CBFIL/CHAFIL
     *        /CBCH/CHANUMB
     * COMMON/SSSTAT2/SSF(18,14),SSARDB(18,14)
C
      INCLUDE '[ASGPROG]COMVAR.LIS'
C
      IKJ = -1
      KNUM = CHANUMB
      SETNUM = 0
      NN = 2048
C
      IF(KNUM.EQ.0.AND.CNTFL.EQ.1)GOTO 999
 610  IKJ=IKJ+6
      SUMFIL='SUMSN'//CHAFIL(IKJ)(2:5)//'.SUM'
      STTFIL='SUMSN'//CHAFIL(IKJ)(2:5)//'.001'
      IKJ=IKJ-6
C
 650  OPEN(3,STATUS='NEW',FILE=SUMFIL)
      OPEN(7,STATUS='NEW',FILE=STTFIL)
C
      CALL DATE(DAY)
      CALL TIME(HOUR)
      WRITE(3,1)DAY,HOUR
   1  FORMAT(1X,A9,2X,A9,/)
C
 700  CONTINUE
      IF (KNUM  .EQ. 0  .AND.  CNTFL  .EQ. 1) GOTO 800
 701  IKJ = IKJ + 6
      OUTFIL = CHAFIL(IKJ)
      J=INDEX(OUTFIL,'.')
 712  FRFIL ='FR'//CHAFIL(IKJ)(2:5)//'.'//OUTFIL(J+1:J+3)
  750 OPEN(21,STATUS='NEW',FILE=FRFIL)
C
C     Read output signal data set.
C
      INDX=1+INT(FLOAT(IKJ)/6.0)
      CALL GETWAVE(INDX,0,4,NFP,NLP,DEL,'O',A)
 760  KNUM = KNUM -1
      N =(NLP - NFP) + 1
      DT = DEL
      IF (N .GT. NN)  N=NN
      L = 14
      FSG = 273.4375
      ALPHA = 9
      PI=4.*ATAN(1.)
C
C     Index translation
C
      DO 5 I=0,NN+1
   5    B(I+1)=A(I)
C
```

```
C       Recreate input waveform data file.
C
        OMEGA=2.*PI*FSG
        DO 4 I=1,N
          T=DT*FLOAT(I-1)
          C(I)=0.
          DO 2 J=1,L
            X=SIN(J*OMEGA*T)
    2       C(I)=C(I)+X
    4     CONTINUE
C
C       Zero fill
C
        IF(N.LT.NN)GO TO 8
        GO TO 12
    8   DO 10 I=N+1,NN+2
          B(I)=0.
   10     C(I)=0.
C
C       Window the data
C
   12   CALL SGAUSS(W,N,ALPHA)
        DO 14 I=1,N
          B(I)=W(I)*B(I)
   14     C(I)=W(I)*C(I)
C
C       Compute DFTs
C
        CALL FAST(B,NN)
        CALL FAST(C,NN)
        DO 20 I=0,NN/2
          U(I)=SQRT(C(2*I+1)*C(2*I+1)+C(2*I+2)*C(2*I+2))
   20     V(I)=SQRT(B(2*I+1)*B(2*I+1)+B(2*I+2)*B(2*I+2))
C
C       Compute frequency response
C
        DF=1./(NN*DT)           ! DF=frequency increment of analyzing DFT output
        DO 30 J=1,L             ! J-th harmonic is analyzed during J-th pass.
          FS=J*FSG              ! FS = frequency of J-th harmonic
          RN=FS/DF
          K=INT(RN)             ! K = DFT output line no. corresponding to FS
          R=RN-K                ! R = remainder fraction after truncation
          IF(R.GT.0.5)K=K+1
          LN(J)=K
          Y(J)=U(K)
          Z(J)=V(K)
          F(J)=FS
          FDFT(J)=K*DF
   30     CONTINUE
        CSC=(0.993)*Y(1)/Z(1)
        DO 40 J=1,L
          AR(J)=CSC*Z(J)/Y(J)
                    ! AR = ratio of output DFT to input DFT at J-th harmonic.
   40     ARDB(J)=20.*ALOG10(AR(J))     ! frequency response in dB at FS
```

```
        WRITE(21,150)(F(J),ARDB(J),J=1,L)
C
C       Write summary file.
C
        WRITE(3,500)  OUTFIL,FRFIL
 500    FORMAT(////10X,'RESULT FOR FILE ',5X,A30,/10X,
     *  'FREQUENCY RESPONSE FILE ',5X,A30,///)
        WRITE(3,220)
        WRITE(3,221)
        WRITE(3,222)
        WRITE(3,230)(F(I),LN(I),FDFT(I),Y(I),Z(I),AR(I),ARDB(I),I=1,L)
C
C       Store statistical data file.
C
        WRITE(7,509)(ARDB(I),I=1,L)
  509   FORMAT(1X,<L>F9.4)
        DO 510 I=1,L
          SSF(INDX,I)=F(I)
          SSARDB(INDX,I)=ARDB(I)
 510    CONTINUE
C
        WRITE(6,601) OUTFIL
 601    FORMAT(1X,'PROCESSING COMPLETED FOR FILE ',5X,A30)
        CLOSE(21)
        GOTO 700
C
C       End of loop
C
 800    CLOSE(3)
        CLOSE(7)
 999    RETURN
 120    FORMAT(A)
 150    FORMAT(2E13.5)
 220    FORMAT(/,3X,'SIGNAL',4X,'DFT',6X,'DFT',7X,'INPUT',6X,'OUTPUT',
     *  4X,'SCALED',2X,'AMPLITUDE')
 221    FORMAT(' FREQUENCY',2X,'LINE',4X,'OUTPUT',7X,'DFT',9X,'DFT',
     *  7X,'DFT',5X,'RATIO')
 222    FORMAT(4X,'(HZ)',5X,'NO.',2X,'FREQUENCY',5X,'VALUE',7X,'VALUE',
     *  5X,'RATIO',5X,'(DB)',/)
 230    FORMAT(20(1X,F9.4,2X,I4,2X,F9.4,2X,2(F10.4,2X),F7.5,3X,F7.2,:/))
        END
C
C-----------------------------------------------------------------------
C
C       Subroutine SGAUSS                 John E. Nickles
C                                         Transportation Systems Center
C                                         14-August-1984
C
C       Subroutine to generate a Gaussian data window
C
C-----------------------------------------------------------------------
C
        SUBROUTINE SGAUSS(W,L,ALPHA)
        DIMENSION W(L)
```

```
        DO 5 I=1,L
          D=(2.*(I-1)/L)-1.
    5     W(I)=EXP(-(ALPHA*D*ALPHA*D)/2)
        RETURN
        END
C
C-----------------------------------------------------------------------
C SUBROUTINE:  FAST
C REPLACES THE REAL VECTOR B(K), FOR K=1,2,...,N,
C WITH ITS FINITE DISCRETE FOURIER TRANSFORM
C-----------------------------------------------------------------------
C
      SUBROUTINE FAST(B, N)
C
C THE DC TERM IS RETURNED IN LOCATION B(1) WITH B(2) SET TO 0.
C THEREAFTER THE JTH HARMONIC IS RETURNED AS A COMPLEX
C NUMBER STORED AS  B(2*J+1) + I B(2*J+2).
C THE N/2 HARMONIC IS RETURNED IN B(N+1) WITH B(N+2) SET TO 0.
C HENCE, B MUST BE DIMENSIONED TO SIZE N+2.
C THE SUBROUTINE IS CALLED AS  FAST(B,N) WHERE N=2**M AND
C B IS THE REAL ARRAY DESCRIBED ABOVE.
C
      DIMENSION B(N+2)
      COMMON /CONS/ PII, P7, P7TWO, C22, S22, PI2
C
C IW IS A MACHINE DEPENDENT WRITE DEVICE NUMBER
C
      IW = 6
C
      PII = 4.*ATAN(1.)
      PI8 = PII/8.
      P7 = 1./SQRT(2.)
      P7TWO = 2.*P7
      C22 = COS(PI8)
      S22 = SIN(PI8)
      PI2 = 2.*PII
      DO 10 I=1,15
        M = I
        NT = 2**I
        IF (N.EQ.NT) GO TO 20
   10 CONTINUE
      WRITE (IW,9999)
 9999 FORMAT (33H N IS NOT A POWER OF TWO FOR FAST)
      STOP
   20 N4POW = M/2
C
C DO A RADIX 2 ITERATION FIRST IF ONE IS REQUIRED.
C
      IF (M-N4POW*2) 40, 40, 30
   30 NN = 2
      INT = N/NN
      CALL FR2TR(INT, B(1), B(INT+1))
      GO TO 50
   40 NN = 1
```

```
C
C PERFORM RADIX 4 ITERATIONS.
C
  50  IF (N4POW.EQ.0) GO TO 70
      DO 60 IT=1,N4POW
        NN = NN*4
        INT = N/NN
        CALL FR4TR(INT, NN, B(1), B(INT+1), B(2*INT+1), B(3*INT+1),
     *      B(1), B(INT+1), B(2*INT+1), B(3*INT+1))
  60  CONTINUE
C
C PERFORM IN-PLACE REORDERING.
C
  70  CALL FORD1(M, B)
      CALL FORD2(M, B)
      T = B(2)
      B(2) = 0.
      B(N+1) = T
      B(N+2) = 0.
      DO 80 IT=4,N,2
        B(IT) = -B(IT)
  80  CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------
C SUBROUTINE:  FR2TR
C RADIX 2 ITERATION SUBROUTINE
C-------------------------------------------------------------------
C
      SUBROUTINE FR2TR(INT, B0, B1)
      DIMENSION B0(INT+1), B1(INT+1)
      DO 10 K=1,INT
        T = B0(K) + B1(K)
        B1(K) = B0(K) - B1(K)
        B0(K) = T
  10  CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------
C SUBROUTINE:  FR4TR
C RADIX 4 ITERATION SUBROUTINE
C-------------------------------------------------------------------
C
      SUBROUTINE FR4TR(INT, NN, B0, B1, B2, B3, B4, B5, B6, B7)
      DIMENSION L(15), B0(NN*INT+2), B1(NN*INT+2), B2(NN*INT+2),
     *B3(NN*INT+2), B4(NN*INT+2), B5(NN*INT+2), B6(NN*INT+2),
     *B7(NN*INT+2)
      COMMON /CONS/ PII, P7, P7TWO, C22, S22, PI2
      EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
     *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
     *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
     *    (L1,L(15))
```

```
C
C JTHET IS A REVERSED BINARY COUNTER, JR STEPS TWO AT A TIME TO
C LOCATE THE REAL PARTS OF INTERMEDIATE RESULTS, AND JI LOCATES
C THE IMAGINARY PART CORRESPONDING TO JR.
C
      L(1) = NN/4
      DO 40 K=2,15
        IF (L(K-1)-2) 10, 20, 30
   10   L(K-1) = 2
   20   L(K) = 2
        GO TO 40
   30   L(K) = L(K-1)/2
   40 CONTINUE
      PIOVN = PII/FLOAT(NN)
      JI = 3
      JL = 2
      JR = 2
      DO 120 J1=2,L1,2
      DO 120 J2=J1,L2,L1
      DO 120 J3=J2,L3,L2
      DO 120 J4=J3,L4,L3
      DO 120 J5=J4,L5,L4
      DO 120 J6=J5,L6,L5
      DO 120 J7=J6,L7,L6
      DO 120 J8=J7,L8,L7
      DO 120 J9=J8,L9,L8
      DO 120 J10=J9,L10,L9
      DO 120 J11=J10,L11,L10
      DO 120 J12=J11,L12,L11
      DO 120 J13=J12,L13,L12
      DO 120 J14=J13,L14,L13
      DO 120 JTHET=J14,L15,L14
        TH2 = JTHET - 2
        IF (TH2) 50, 50, 90
   50   DO 60 K=1,INT
          T0 = B0(K) + B2(K)
          T1 = B1(K) + B3(K)
          B2(K) = B0(K) - B2(K)
          B3(K) = B1(K) - B3(K)
          B0(K) = T0 + T1
          B1(K) = T0 - T1
   60   CONTINUE
        IF (NN-4) 120, 120, 70
   70   K0 = INT*4 + 1
        KL = K0 + INT - 1
        DO 80 K=K0,KL
          PR = P7*(B1(K)-B3(K))
          PI = P7*(B1(K)+B3(K))
          B3(K) = B2(K) + PI
          B1(K) = PI - B2(K)
          B2(K) = B0(K) - PR
          B0(K) = B0(K) + PR
   80   CONTINUE
        GO TO 120
```

```
 90     ARG = TH2*PIOVN
        C1 = COS(ARG)
        S1 = SIN(ARG)
        C2 = C1**2 - S1**2
        S2 = C1*S1 + C1*S1
        C3 = C1*C2 - S1*S2
        S3 = C2*S1 + S2*C1
        INT4 = INT*4
        J0 = JR*INT4 + 1
        K0 = JI*INT4 + 1
        JLAST = J0 + INT - 1
        DO 100 J=J0,JLAST
          K = K0 + J - J0
          R1 = B1(J)*C1 - B5(K)*S1
          R5 = B1(J)*S1 + B5(K)*C1
          T2 = B2(J)*C2 - B6(K)*S2
          T6 = B2(J)*S2 + B6(K)*C2
          T3 = B3(J)*C3 - B7(K)*S3
          T7 = B3(J)*S3 + B7(K)*C3
          T0 = B0(J) + T2
          T4 = B4(K) + T6
          T2 = B0(J) - T2
          T6 = B4(K) - T6
          T1 = R1 + T3
          T5 = R5 + T7
          T3 = R1 - T3
          T7 = R5 - T7
          B0(J) = T0 + T1
          B7(K) = T4 + T5
          B6(K) = T0 - T1
          B1(J) = T5 - T4
          B2(J) = T2 - T7
          B5(K) = T6 + T3
          B4(K) = T2 + T7
          B3(J) = T3 - T6
 100    CONTINUE
        JR = JR + 2
        JI = JI - 2
        IF (JI-JL) 110, 110, 120
 110    JI = 2*JR - 1
        JL = JR
 120  CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------
C SUBROUTINE:  FORD1
C IN-PLACE REORDERING SUBROUTINE
C-------------------------------------------------------------------
C
      SUBROUTINE FORD1(M, B)
      DIMENSION B(2**M+2)
C
      K = 4
```

```
      KL = 2
      N = 2**M
      DO 40 J=4,N,2
        IF (K-J) 20, 20, 10
   10   T = B(J)
        B(J) = B(K)
        B(K) = T
   20   K = K - 2
        IF (K-KL) 30, 30, 40
   30   K = 2*J
        KL = J
   40 CONTINUE
      RETURN
      END
C
C-------------------------------------------------------------------------
C SUBROUTINE:  FORD2
C IN-PLACE REORDERING SUBROUTINE
C-------------------------------------------------------------------------
C
      SUBROUTINE FORD2(M, B)
      DIMENSION L(15), B(2**M+2)
      EQUIVALENCE (L15,L(1)), (L14,L(2)), (L13,L(3)), (L12,L(4)),
     *    (L11,L(5)), (L10,L(6)), (L9,L(7)), (L8,L(8)), (L7,L(9)),
     *    (L6,L(10)), (L5,L(11)), (L4,L(12)), (L3,L(13)), (L2,L(14)),
     *    (L1,L(15))
      N = 2**M
      L(1) = N
      DO 10 K=2,M
        L(K) = L(K-1)/2
   10 CONTINUE
      DO 20 K=M,14
        L(K+1) = 2
   20 CONTINUE
      IJ = 2
      DO 40 J1=2,L1,2
      DO 40 J2=J1,L2,L1
      DO 40 J3=J2,L3,L2
      DO 40 J4=J3,L4,L3
      DO 40 J5=J4,L5,L4
      DO 40 J6=J5,L6,L5
      DO 40 J7=J6,L7,L6
      DO 40 J8=J7,L8,L7
      DO 40 J9=J8,L9,L8
      DO 40 J10=J9,L10,L9
      DO 40 J11=J10,L11,L10
      DO 40 J12=J11,L12,L11
      DO 40 J13=J12,L13,L12
      DO 40 J14=J13,L14,L13
      DO 40 JI=J14,L15,L14
        IF (IJ-JI) 30, 40, 40
   30   T = B(IJ-1)
        B(IJ-1) = B(JI-1)
        B(JI-1) = T
```

```
        T = B(IJ)
        B(IJ) = B(JI)
        B(JI) = T
 40     IJ = IJ + 2
      RETURN
      END
C  Subroutine GETIND determines the index of the data array which is
C  closest to the calculated time of occurrence of the maximum data
C  value.  The approach taken is to average the times the data passes
C  through a certain level which is expressed as a percentage of the
C  theoretical maximum (200 g's).  Interpolation is used to improve
C  the time calculation.  It is assumed that noise riding on the data
C  will not affect this calculation.  (This should be true as long as
C  the percentage chosen places the level in an area of the curve
C  which has a steep slope.)
C
      SUBROUTINE GETIND(NFP,NLP,PCT,Y,DEL,IOMAX,TOMAX,DIOMAX,VMAX,TDACL)
      REAL Y(-1000 :10000)
C
      chklev=200.0*pct/100.0
      tclev1=0.0
      tclev2=0.0
      do 100 I=nfp,nlp-1
        til=(I+1)*del*1000.
        ti=I*del*1000.
        if (abs(y(I)).le.chklev.and.abs(y(I+1)).ge.chklev) then
          tclev1=ti+(((chklev-abs(y(I)))/(abs(y(I+1))-abs(y(I))))
     &          *(til-ti))
        end if
        if (abs(y(I+1)).le.chklev.and.abs(y(I)).ge.chklev) then
          tclev2=ti+(((abs(y(I))-chklev)/(abs(y(I))-abs(y(I+1))))
     &          *(til-ti))
        end if
 100  continue
      tomax=(tclev1+tclev2)/2.0
      tdacl=tclev2-tclev1
      do 200 J=nfp,nlp-1
        tjl=(J+1)*del*1000.
        tj=J*del*1000.
        if (tj.le.tomax.and.tjl.ge.tomax) then
          dtj=tomax-tj
          dtjl=tjl-tomax
C  If the calculated time of maximum falls exactly between two samples,
C  the index of maximum is arbitrarily assigned to the earlier sample.
          if (dtj.le.dtjl) then
            iomax=J
            diomax=dtj
          end if
          if (dtjl.lt.dtj) then
            iomax=J+1
            diomax=-dtjl
          end if
          vmax=y(j)+((dtj/(tjl-tj))*(y(j+1)-y(j)))
        end if
```

```
  200 continue
      return
      end
C
C  Subroutine THCALC obtains, through multiple steps, the value of the
C  theoretical HIC if the curve were to be made up of two components
C  (x and z) which have the same amplitude and duration of the actual
C  data.  (The parameters necessary for determining the duration and
C  amplitude are obtained in subroutine GETIND.)  The steps taken to
C  obtain the theoretical HIC value are:
C     1) determine the equations of the two half-sine curves which pass
C        through the points determined via GETIND
C     2) generate arrays of data which are the equivalent of the
C        equations should they be sampled at the interval of DEL
C     3) obtain the resultant of the two arrays
C     4) calculate HIC, T1, and T2 using a modified version of the HIC3
C        subroutine
C
      SUBROUTINE THCALC(CCTD,THIC,TT1,TT2)
      COMMON /HSTMX/TOMX,YMXX,TOMZ,YMNZ,TCLEV1,TCLEV2,PCT,TDACLX,TDACLZ
      DIMENSION YX(-1000:10000),YZ(-1000:10000),R(-1000:10000)
      DIMENSION S(0:10000),JL(15000),KL(15000)
      DOUBLE PRECISION RC,RM,SC,SM
      INCLUDE '[ASGPROG]COMVAR.LIS'
      DATA NAR/15000/
C
      pi=3.1415927
      do 10 I=-1000,10000
        yx(I)=0.0
        yz(I)=0.0
        r(I)=0.0
   10 continue
      ix01=INT(.01/del)
      tx01=del*ix01
      tz01=tx01+cctd
C  From A = X * Sin(w * t) and knowing the times and values of the
C  points at which a certain level is crossed,
C  w = (2 * arc-cosine(pct/100))/dt
      wx=(2.0*ACOS(pct/100.0))/(tdaclx/1000.)
      wz=(2.0*ACOS(pct/100.0))/(tdaclz/1000.)
      icflagx=0
      icflagz=0
      do 20 I=0,nlp
        t=FLOAT(I)*del
        yx(I)=0.0
        yz(I)=0.0
        if (t.ge.tx01.and.icflagx.eq.0) then
          yx(I)=ymxx*sin(wx*(t-tx01))
          if (yx(I).gt.0.0) then
            yx(I)=0.0
            icflagx=1
          end if
        end if
        if (t.ge.tz01.and.icflagz.eq.0) then
```

```
            yz(I)=ymnz*sin(wz*(t-tz01))
            if (yz(I).lt.0.0) then
              yz(I)=0.0
              icflagz=1
            end if
          end if
          r(I)=SQRT((yx(I)**2)+(yz(I)**2))
   20 continue
C  The following routine is taken from the subroutine HIC3
C  The code is repeated here to eliminate the need to write UDS files
C
C  Compute sum (integral) array using trapezoid rule
          S(0)=0.0
          SM=0.D0
          RM=ABS(R(0))
          DO 102 I=1,NLP
             RC=ABS(R(I))
             SC=SM+.5*(RC+RM)
             S(I)=SC
             SM=SC
             RM=RC
  102     CONTINUE
C  Initialize variables
          FMAX=0.0
          JMAX=0
          KMAX=0
          JL(1)=0
          KL(1)=-1
          NWU=NLP
          IBR=1
          IER=1
          INC=-1
          IBL=NAR
C  Begin a round of region evaluations
  103     IL=IBL-INC
          DO 104 IR=IBR,IER,INC
             JLI=JL(IR)
             IF (JLI.GT.0) THEN
                NJ=NWU
             ELSE
                JLI=ABS(JLI)
                NJ=MAX0(NWU-1,0)
             END IF
             KLI=KL(IR)
             IF (KLI.GT.0) THEN
                NK=NWU
             ELSE
                KLI=ABS(KLI)
                NK=MAX0(NWU-1,0)
             END IF
             KUI=KLI+NK
             IF ((NJ.EQ.0).AND.(NK.EQ.0)) THEN
C  This is a single point region - Evaluate
                F=(S(KUI)-S(JLI))*(KUI-JLI)**(-.6)
```

```
               IF (F.GT.FMAX) THEN
                  FMAX=F
                  JMAX=JLI
                  KMAX=KUI
               END IF
            ELSE
C  This is a finite region with feasible points - Bound
               JUI=JLI+NJ
               FS=S(KUI)-S(JLI)
               IF (FS*MAX0(KLI-JUI,1)**(-.6).GT.FMAX) THEN
C  Region is possibly optimal - Evaluate and partition
                  F=FS*(KUI-JLI)**(-.6)
                  IF (F.GT.FMAX) THEN
                     FMAX=F
                     JMAX=JLI
                     KMAX=KUI
                  END IF
                  IF ((IL+4*INC)*INC.GT.IR*INC) THEN
                     WRITE(6,993)
  993                FORMAT(/1X,'*** ERROR - HIC3 array size exceeded'/)
                     CALL CLOSE(3)
                     DFLAG =1
                     GOTO 670
                  END IF
                  CALL PART(JLI,JUI,NJ,NWU-NJ,KLI,KUI,NK,NWU-NK,IL,INC,
     *            JL,KL)
               END IF
            END IF
  104    CONTINUE
         IF (ABS(IBL-IL-INC).GT.0) THEN
C  Some regions remain - Set up boundary index arrays
            ITMP=IBL
            IBL=IER
            IER=ITMP
            IBR=IL
            INC=-1*INC
            NWU=NWU/2
            GO TO 103
         ELSE
C  Done - No regions remain - Report results
            THIC=DEL*FMAX**2.5
            TT1=1000*JMAX*DEL
            TT2=1000*KMAX*DEL
         END IF
  670 CONTINUE
      RETURN
      END
```

The fourth segment, "REC", contains the following modules:

RECTAN
ZERO2
COMPRE

The source code listing for these modules is:

```
C     ********************************************************
C     *                                                      *
C     *          SUBROUTINE FOR RECTANGLE WAVEFORM           *
C     *                                                      *
C     ********************************************************
C
C     Subroutine rectan performs signal processing calculations
C     upon the rectangular waveform that has been recorded.  This
C     program performs four time checks and two amplitude checks on
C     the rectangle waveform output.  All rectangle waveforms are
C     processed simultaneously.
C
C     Rectan then outputs both upper and lower overshoot, attenuation,
C     and timeshift.  Overshoots and steady-state values are determined
C     by subroutine zer02.
C
      SUBROUTINE RECTAN(ICHAN,DFLAG,CNTFL)
C
C     define data types
C
      CHARACTER  IFLNM1 *30,CHANUM *2,DUMMY *1
      INTEGER N1(13),N2(13),N3(13),N4(13),NOPTS,DFLAG
      INTEGER NCHAN,ZIC,ZDC,ZERINC(100),ZERDEC(100),ZFLAG,
     *        OLFLAG,LB(14),TZIC,UBND,LBND,START,FINISH,
     *        INGOV(13),IPSOV(13),TB,TR,CHANUMB,CNTFL
      REAL XMIN,XMAX,YMIN,YMAX,TO
      REAL Y(-1000 :10000),DAN(10000)
      REAL NPMS,POSOV(13),NEGOV(13),YMN,YMX,PAVGOV,NAVGOV,
     *     SSPAV(13),SSNAV(13),AVG,TL(13),TU(13)
      CHARACTER *30 CHAFIL(108)

      INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
C
C     define common block symbolic names
C
      COMMON /CBXY/   XMIN,XMAX,YMIN,YMAX
     *       /CBTO/   TO
     *       /CBNPTS/ NOPTS
     *       /CBY/    Y
      COMMON /OVER/   POSOV,NEGOV,IPSOV,INGOV,SSPAV,SSNAV
     *       /RANGE/  FINISH
     *       /SHFT/   TL,TU
     *       /STPT/   N1,N2,N3,N4
     *       /PTN/    START,LB
     *       /COUNT/  ZIC,ZDC,ZERINC,ZERDEC,ZFLAG
     *       /FLAG/   EFLAG
     *       /CBFIL/ CHAFIL
     *       /CBCH/ CHANUMB
      COMMON/RSTAT1/RPOS(18,10),RNOS(18,10),RSSPAV(18,10),
     &       RSSNAV(18,10),RTU(18,10),RTL(18,10),RFLNM1(18),
     &       RSATT(18),RCNUM(18),RTPERF(18)
      CHARACTER*30 RFLNM1
      CHARACTER*25 RSATT,RTPERF
      CHARACTER*2 RCNUM
```

```
      DIMENSION  SSNMP(13),SSNMN(13)
C
C     Rectangle waveform processor.
C     prepare to input data and begin processing.
C
      IKJ = -4
      KNUM =CHANUMB
      DFLAG = 0
      WRITE(6,1)
    1 FORMAT(/1X,'**   RECTANGLE WAVEFORM PROCESSOR   **')
      ICHAN = 0
C
C     START OF THE LOOP FOR PROCESSING
C
  700 CONTINUE
      IF (KNUM  .EQ. 0  .AND.  CNTFL .EQ. 1) GOTO 800
      IKJ = IKJ + 6
      YMAX = 200.0
      YMIN = -200.0
      DO 8700 I=-1000,10000
       Y(I) = 0.0
 8700 CONTINUE
 5000 IFLNM1 = CHAFIL(IKJ)
C
C     READING FROM DATA FILE CONTAINING WAVEFORM
C
 5001 INDX=1+INT(FLOAT(IKJ)/6.0)
      CALL GETWAVE(INDX,0,1,NFP,NLP,DEL,'O',Y)
      J=1
      DO 8705 I=NFP,NLP
        DAN(J)=Y(I)
        J=J+1
 8705 CONTINUE
      DO 8710 I=1,NLP+1
        Y(I)=DAN(I)
 8710 CONTINUE
      KNUM = KNUM - 1
      ICHAN = ICHAN + 1
      NPMS=1.0/DEL/1000.
C
C     The first processing performed determines the zero crossing
C     of the waveform.  When the waveform crosses zero from negative to
C     positive the point number is stored in the zerinc(I) array.
C     When the waveform crosses zero from positive to negative the point
C     number is stored in the zerdec(I) array.  The point numbers are only
C     assigned to the two arrays above if there is a definite sign change.
C     It is not assigned for a positive to zero to positive change
C     in the waveform. All crossings are stored sequencially
C     in their respective arrays.
C
      ZIC=0
      ZDC=0
      NOPTS=(NLP-NFP)+1
      DO 55 I=1,100
```

```
      ZERINC(I)=0
   55 ZERDEC(I)=0

      DO 11 I=1,NOPTS
        IF(Y(I)) 20,30,40
   20   ZFLAG=-1
        GO TO 50
   30   ZFLAG=0
        GO TO 50
   40   ZFLAG=1
   50   IF (I.EQ.1) GOTO 100
   60   IF(ZFLAG.EQ.OLFLAG) GOTO 100
        IF(OLFLAG.EQ.0) GOTO 100
        IF(OLFLAG.EQ.-1) GOTO 90
        ZDC=ZDC+1
        ZERDEC(ZDC)=I
        GOTO 100
   90   ZIC=ZIC+1
        ZERINC(ZIC)=I
  100   OLFLAG=ZFLAG
   11 CONTINUE
        IF (Y(NOPTS)  .LT. 0.0 .AND.  Y(NOPTS)  .GT. -250.0) GOTO 8715
        GOTO 8720
 8715   ZIC = ZIC + 1
      ZERINC(ZIC)=NOPTS
 8720 CONTINUE
      IF (ZIC  .GT. 10) ZIC = 10
C
C     The second processing performed is the division of the waveform
C     into 10 separate sections.  The point numbers are assigned to the
C     LB(I) array.  LB(1) is assumed to occur EXACTLY 10 msec before LB(2).
C     LB(2) and each successive LB(I) corresponds to the end of a complete
C     period of a  rectangle waveform.
C
      DO 15 I=1,13
        LB(I)=0
   15 CONTINUE
      IMAX=NOPTS
      TZIC=ZIC
      IMIN=ZERINC(TZIC)
      YTST=.5*YMAX
   31 DO 12 I=IMIN,IMAX
        IF(Y(I).GT.YTST) GOTO 21
   12 CONTINUE
      IMAX=ZERINC(TZIC)-1
      TZIC=TZIC-1
      IMIN=ZERINC(TZIC)
      GOTO 31
   21 UBND=TZIC
      TZIC=1
      IMIN=1
      IMAX=ZERINC(TZIC)
      YTST=.5*YMIN
   61 DO 41 I=IMAX,IMIN,-1
```

```
        IF(Y(I).LT.YTST) GOTO 51
   41 CONTINUE
      IMIN=ZERINC(TZIC)+1
      TZIC=TZIC+1
      IMAX=ZERINC(TZIC)
      GOTO 61
   51 LBND=TZIC
      J=LBND-1
C      START=10-(UBND-LBND)
C      FINISH=10
      START=2
      FINISH=ZIC
      DO 70 I=START,FINISH+1
        J=J+1
        LB(I)=ZERINC(J)
   70 CONTINUE
      LB(12)=NOPTS
C
C     The third processing performed on the rectangle waveform determines
C     the steady-state averages and overshoots for each of the 20
C     sections of the rectangle waveform.  The positive steady-state values
C     are stored in the SSPAV(I) array.  The negative steady-state values
C     are stored in the SSNAV(I) array.  The positive overshoot values are
C     stored in the POSOV(I) array.  The negative overshoot values are
C     stored in the NEGOV(I) array.  The average positive and negative
C     values for overshoot are calculated and stored as PAVGOV and
C     NAVGOV respectively.
C
C     Initialize outputs as default values
C
      DO 105 I=1,13
        SSPAV(I)=1.0E+4
        SSNAV(I)=-1.0E+4
        POSOV(I)=1.0E+4
        NEGOV(I)=-1.0E+4
  105 CONTINUE
      LB(1)=LB(2)-(10.0*NPMS)
      DO 6 I=1,ZDC
        IF(ZERDEC(I).GT.LB(START)) GOTO 6
        KK=I
    6 CONTINUE
      K=KK
      DO 110 I=1,FINISH+1
        LL=LB(I)
        LU=ZERDEC(K)
        IF(LU.EQ.0) LU=NOPTS
        K=K+1
        NPT=LU-LL+1
        CALL ZER02(Y(LL),NPT,SSPAV(I),POSOV(I),0,NPMS)
  110 CONTINUE
      K=KK
      JK=FINISH
      DO 120 I=1,JK
        LL=ZERDEC(K)
```

```
        K=K+1
        LU=LB(I+1)
        NPT=LU-LL+1
        CALL ZERO2(Y(LL),NPT,SSNAV(I),NEGOV(I),1,NPMS)
  120 CONTINUE
      PAVGOV=0.0
      NAVGOV=0.0
      DO 130 I=START-1,FINISH
        PAVGOV=PAVGOV+POSOV(I)
        IF(I.EQ.FINISH) GOTO 130
        NAVGOV=NAVGOV+NEGOV(I)
  130 CONTINUE
      PAVGOV=PAVGOV/(FINISH-START+2)
      NAVGOV=NAVGOV/(FINISH-START+2)
  150 NEGOV(13)=-1.0E+4
C
C     The fourth processing performed determines the time shifts in the
C     waveforms leading edges.  All edges are compared to the first
C     leading edge which is the first rectangle waveform of the 10 rectangle
C     waveforms expected for each channel recorded. Time shifts are not
C     compared between channel-to-channel variations.
C
C     The timeshifts of the positive section are stored in the TU(I) array.
C     Timeshifts for the negative sections are stored in the TL(I) array.
C
      DO 7 I=1,10
        TL(I)=-1.0E+4
        TU(I)=1.0E+4
    7 CONTINUE
      YTST1=0.9*YMIN
      YTST2=0.9*YMAX
      TL(1)=(LB(1)-1)/NPMS
      LSTART=START-1
      TU(LSTART)=(ZERDEC(LSTART)-1)/NPMS-(6+(LSTART-1)*10)
      DO 300 I=START,FINISH
        IF(POSOV(I).EQ.1.0E+4) GOTO 310
        TL(I)=((LB(I)-1)/NPMS)-(10*(I-1))
  310   IF(NEGOV(I).EQ.-1.0E+4) GOTO 300
        K1=KK+I-1
        TU(I)=((ZERDEC(K1)-1)/NPMS)-(6+(I-1)*10)
  300 CONTINUE
C
C    The fifth processing performed determines the normalized steady-state
C    and overshoot values for each of the 20 sections.  These values are
C    stored over the values in the already existing NEGOV,POSOV,
C    SSNAV and SSPAV arrays.
C
C  SSNM is a "normalization factor" which takes into account the
C  fact that the starting point for most changes in the waveform
C  is not at zero but rather at the last steady-state level.
      SSNMP(1)=0.0
      SSNMN(1)=SSPAV(1)
      DO 409 I=2,FINISH
```

```
        SSNMP(I)=SSNAV(I-1)
        SSNMN(I)=SSPAV(I)
  409 CONTINUE
      DO 420 I=1,FINISH
        IF (POSOV(I).EQ.1.0E+4) GOTO 410
        IF (SSNMP(I).EQ.-1.0E+4) SSNMP(I)=0.0
        POSOV(I)=(POSOV(I)-SSPAV(I))/(SSPAV(I)-SSNMP(I))*100.0
        SSPAV(I)=(SSPAV(I)-200.0)/2.0
  410   IF(NEGOV(I).EQ.-1.0E+4) GOTO 420
        SSNM=SSPAV(I)
        IF (SSNMN(I).EQ.1.0E+4) SSNMN(I)=0.0
        NEGOV(I)=(NEGOV(I)-SSNAV(I))/(SSNAV(I)-SSNMN(I))*100.0
        SSNAV(I)=(SSNAV(I)+200.0)/-2.0
  420 CONTINUE
C
C     The data is now stored until it is needed for comparisons
C     with allowable limits.  The data are the upper and lower
C     overshoots, the upper and lower attenuations, and the upper
C     and lower timeshifts.
C
      RFLNM1(INDX)=IFLNM1
      RSATT(INDX)=SENATT
      I=INDEX(IFLNM1,'.')
  910 CHANUM=IFLNM1(I+2:I+3)
      RCNUM(INDX)=CHANUM
      RTPERF(INDX)=TSTPRF
      DO 520 K=1,10
        RPOS(INDX,K)=POSOV(K)
        RNOS(INDX,K)=NEGOV(K)
        RSSPAV(INDX,K)=SSPAV(K)
        RSSNAV(INDX,K)=SSNAV(K)
        RTU(INDX,K)=TU(K)
        RTL(INDX,K)=TL(K)
  520 CONTINUE
      CALL CLOSE(1)
      WRITE(6,*)  'PROCESSING COMPLETED FOR FILE ',IFLNM1
      GOTO 700
  800 CONTINUE
      RETURN
      END
C
        SUBROUTINE ZERO2(X,NPTS,Z,AVG,IFL,XNPM)
C
        DIMENSION X(2000),Y(2000)
        K=NPTS-XNPM
        IF(IFL.EQ.1) GOTO 520
        AVG=-1.E+4
        DO 500 I=1,K
  500 AVG=AMAX1(X(I),AVG)
        GOTO 510
  520 AVG=1.E+4
        DO 530 I=1,K
  530 AVG=AMIN1(X(I),AVG)
  510 GTST=ABS(X(1))
```

```
      IGTST=NPTS
      Y(1)=X(1)
      DO 30 I=2,IGTST
      DO 40 INNER=1,I-1
      IF(X(I).GT.Y(INNER)) GOTO 50
   40 CONTINUE
      Y(I)=X(I)
      GOTO 70
   50 CONTINUE
      DO 60 K=I,INNER+1,-1
      Y(K)=Y(K-1)
   60 CONTINUE
      Y(INNER)=X(I)
   70 CONTINUE
   30 CONTINUE
      TEST=1.E10
      IW=IGTST/2
      NEND=IGTST-IW
      DO 10 I=1,NEND
      D=ABS(Y(I)-Y(I+IW))
      IF(D.GT.TEST) GOTO 10
      TEST=D
      INDEX=I
   10 CONTINUE
      Z=Y(INDEX+(IW+1)/2)
      K=NPTS-XNPM
      LEA=NPTS/2
      SUM=0.
      DO 707 I=LEA,K
      SUM=SUM+X(I)
  707 CONTINUE
      Z=SUM/(K-LEA+1)
      RETURN
     END
C
C   ***************************************************************
C   *                                                             *
C   *    SUBROUTINE FOR COMPARING RECTANGLE WAVEFORM RESULTS      *
C   *                                                             *
C   ***************************************************************
C
C   Subroutine Compre receives as input the overshoots, attenuations, and
C   timeshifts from the rectangle waveform that was processed.  These values
C   were output from the rectan subroutine.  These values are compared
C   to their allowable limits and the results determine if a particular
C   channel is acceptable for recording.  All channels are processed
C   sequencially.
C
C   The first comparison performed compares the deviation of actual time
C   from theoretical time and allowable limits.
C   The result is stored as EFLAG(1,I) where I=channel number.  There
C   are a total of five more comparisons for each channel.  A table is
C   provided for clarity.
```

```
C
C  ************************************************************************
C  ************************************************************************
C  **             *                       *                                **
C  ** comparison  * result of status *   description of comparison    **
C  **    number   *   is stored as   *                                **
C  **             *                       *                                **
C  ************************************************************************
C  ************************************************************************
C   *             *                       *                                *
C   *     1       *    EFLAG (1,I)        *  deviation of actual time from *
C   *             *                       *  theoretical time and allowable*
C   *             *                       *  limit                         *
C   ************************************************************************
C   *             *                       *                                *
C   *     2       *    EFLAG (2,I)        *  deviation of slope from ideal *
C   *             *                       *  slope of zero                 *
C   ************************************************************************
C   *             *                       *                                *
C   *     3       *    EFLAG (3,I)        *  deviation of intercept from   *
C   *             *                       *  ideal of zero                 *
C   ************************************************************************
C   *             *                       *                                *
C   *     4       *    EFLAG (4,I)        *  attenuation deviation         *
C   *             *                       *                                *
C   ************************************************************************
C   *             *                       *                                *
C   *     5       *    EFLAG (5,I)        *   overshoot deviation          *
C   *             *                       *                                *
C   ************************************************************************
C   *             *                       *                                *
C   *     6       *    EFLAG (6,I)        *  channel-to-channel time lag   *
C   *             *                       *                                *
C   ************************************************************************
      SUBROUTINE COMPRE(ICHAN,DFLAG)
C
C     identify data types
C
      INTEGER EFLAG(6,18),IND(25),DFLAG
      REAL O(18,26),A(18,26),T(18,26),SLOPE,INTCP,DEV(26)
      CHARACTER *30 RECFI1,RECFI2,RECFI3,RECFI4,RECFI5
      CHARACTER *4 TSTCDNUM
      CHARACTER *2  C(18)
      CHARACTER *9 DAY,HOUR
C
C    DEFINE COMMON BLOCKS
C
      COMMON   /CBCDNUM/TSTCDNUM
      COMMON/RSTAT1/RPOS(18,10),RNOS(18,10),RSSPAV(18,10),
     &       RSSNAV(18,10),RTU(18,10),RTL(18,10),RFLNM1(18),
     &       RSATT(18),RCNUM(18),RTPERF(18)
      COMMON/RSTAT2/RT(18,20)
      COMMON/RSTAT3/RSLPT(18),RINTCT(18)
      COMMON/RSTAT4/RA(18,20)
```

```
      COMMON/RSTAT5/ROS(18,20)
      CHARACTER*30 RFLNM1
      CHARACTER*25 RSATT,RTPERF
      CHARACTER*2 RCNUM
C
C
      INCLUDE '[ASGPROG]COMVAR.LIS'
C
C     Begin output to data file with title of results
C
      DFLAG = 0
      RECFI1='REC'//TSTCDNUM(1:4)//'.SUM'
      RECFI2='REC'//TSTCDNUM(1:4)//'.001'
      RECFI3='REC'//TSTCDNUM(1:4)//'.002'
      RECFI4='REC'//TSTCDNUM(1:4)//'.003'
      RECFI5='REC'//TSTCDNUM(1:4)//'.004'
      OPEN(3,FILE=RECFI1,STATUS='NEW',ERR=3200)
      OPEN(7,FILE=RECFI2,STATUS='NEW',ERR=3200)
      OPEN(8,FILE=RECFI3,STATUS='NEW',ERR=3200)
      OPEN(9,FILE=RECFI4,STATUS='NEW',ERR=3200)
      OPEN(10,FILE=RECFI5,STATUS='NEW',ERR=3200)
      WRITE(3,10)
   10 FORMAT(43X,'** RECTANGLE WAVEFORM PROCESSOR DATA SUMMARY **'//
     1       1X,'A VALUE OF +/- 10000 INDICATES THAT THE QUANTITY ',
     2       'COULD NOT BE DETERMINED'/1X,'THE * SYMBOL INDICATES THAT'
     3       ,' THE ALLOWABLE LIMIT HAS BEEN EXCEEDED.'//1X,'TIME ',
     4        'QUANTITIES EXPRESSED IN MILLISECONDS'/1X,'AMPLITUDE ',
     5        'QUANTITIES EXPRESSED AS PERCENT OF FULL SCALE')
      CALL DATE(DAY)
      CALL TIME(HOUR)
      WRITE(3,12) DAY,HOUR
 12   FORMAT(//,50X,A9,12X,A9)
      DO 20 I=1,ICHAN
      DO 200 J=1,26
      DEV(J)=1.0E+4
  200 CONTINUE
      DO 300 J=1,6
      EFLAG(J,I)=0
  300 CONTINUE
      WRITE(3,11)  RFLNM1(I),RCNUM(I)
   11 FORMAT(//1X,'RESULTS FOR FILE : ',A30,5X,'CHANNEL NO. ',A2)
      WRITE(3,13)  RTPERF(I)
   13 FORMAT(/1X,'FACILITY : ',A25)
      DO 30 K =1,10
        L=(K-1)*2+1
        M=(K-1)*2+2
        O(I,L)=RPOS(I,K)
        O(I,M)=RNOS(I,K)
        A(I,L)=RSSPAV(I,K)
        A(I,M)=RSSNAV(I,K)
        T(I,M)=RTU(I,K)
        T(I,L)=RTL(I,K)
   30 CONTINUE
C
```

```
C     Perform least squares fit on timeshift data for each channel processed
C
      N=0
      SUMXSQ=0
      SUMXY=0
      SUMY=0
      SUMX=0
      DO 40 J=1,20,2
      IF(ABS(T(I,J)).EQ.(1.0E+4)) GOTO 40
      X=(J-1)/2*10
   55 SUMX=SUMX+X
      SUMY=SUMY+T(I,J)
      SUMXY=SUMXY+X*T(I,J)
      SUMXSQ=SUMXSQ+X**2
      N=N+1
   40 CONTINUE
      DO 41 J=2,20,2
      IF(ABS(T(I,J)).EQ.(1.0E+4)) GOTO 41
      X=(J/2-1)*10+6
      SUMX=SUMX+X
      SUMY=SUMY+T(I,J)
      SUMXY=SUMXY+X*T(I,J)
      SUMXSQ=SUMXSQ+X**2
      N=N+1
   41 CONTINUE
C
C     calculate slope and intercept for least squares fit
C
      SLOPE=(SUMX*SUMY-N*SUMXY)/(SUMX**2-N*SUMXSQ)
      INTCP=(SUMY-SLOPE*SUMX)/N
      WRITE(3,1000)
 1000 FORMAT(/1X,'1. TIME DEVIATION FROM THEORETICAL TIME
     *   (ALLOWABLE LIMIT = +/- 1.0 MSEC)')
      DO 49 J=1,20
   49 IND(J)='  '
      DO 50 J=1,20,2
      IF(ABS(T(I,J)).EQ.(1.0E+4)) GOTO 50
      X=(J-1)/2*10
   75 DEV(J)=T(I,J)-SLOPE*X-INTCP
      IF(ABS(DEV(J)).GT.(1.0)) EFLAG(1,I)=1
      IF(ABS(DEV(J)).GT.(1.0)) IND(J)=' *'
   50 CONTINUE
      DO 51 J=2,20,2
      IF(ABS(T(I,J)).EQ.(1.0E+4)) GOTO 51
      X=(J/2-1)*10+6
      DEV(J)=T(I,J)-SLOPE*X-INTCP
      IF(ABS(T(I,J)).GT.(1.0)) EFLAG(1,I)=1
      IF(ABS(T(I,J)).GT.(1.0)) IND(J)=' *'
   51 CONTINUE
      WRITE(3,1021)
      WRITE(3,1002)
 1002 FORMAT(7X,'1',9X,'2',9X,'3',9X,'4',9X,'5',9X,'6',9X,'7',9X,
     1       '8',9X,'9',8X,'10')
 1003 FORMAT(6X,'11',8X,'12',8X,'13',8X,'14',8X,'15',8X,'16',8X,'17',
```

```
     1       8X,'18',8X,'19',8X,'20')
      WRITE(3,1004)(T(I,J),J=1,10)
      WRITE(7,1004)(T(I,J),J=1,10)
 1004 FORMAT(2X,13(F10.3))
      WRITE(3,1005)(IND(J),J=1,10)
 1005 FORMAT(2X,13(4X,1A2,4X))
      WRITE(3,1021)
      WRITE(3,1003)
      WRITE(3,1004)(T(I,J),J=11,20)
      WRITE(7,1004)(T(I,J),J=11,20)
      DO 2004 J=1,20
        RT(I,J)=T(I,J)
 2004 CONTINUE
      WRITE(3,1005)(IND(J),J=11,20)
      DO 52 J=1,20
   52 IND(J)='  '
      WRITE(3,1010)
 1010 FORMAT(/1X,'2. TIME LINEARITY (ALLOWABLE LIMIT = +/- 1 %)',
     1       20X,'3. TIME OFFSET (ALLOWABLE LIMIT = +/- 1.0 MSEC)')
C
C     time deviation/slope intercept comparison to allowable limits
C
      IF(ABS(SLOPE).GT.(0.01)) EFLAG(2,I)=1
      IF(ABS(SLOPE).GT.(0.01)) IND(1)=' *'
      IF(ABS(INTCP).GT.(1.0)) EFLAG(3,I)=1
      IF(ABS(INTCP).GT.(1.0)) IND(2)=' *'
      SLOPE=SLOPE*100
      WRITE(3,1011) SLOPE,IND(1),INTCP,IND(2)
      WRITE(8,1012) SLOPE,INTCP
      RSLPT(I)=SLOPE
      RINTCT(I)=INTCP
 1012 FORMAT(2X,F6.2,5X,F6.2)
 1011 FORMAT(/10X,'SLOPE = ',F6.2,1A2,48X,'INTERCEPT = ',F6.2,1A2)
      IND(1)='  '
      IND(2)='  '
      WRITE(3,1020)
 1020 FORMAT(/1X,'4. STEADY-STATE AMPLITUDE DEVIATION
     1 FROM THEORETICAL AMPLITUDE
     1   (ALLOWABLE LIMIT = 2.5 %)')
      DO 95 J=1,20
C
C     attenuation comparison to allowable limits
C
      IF(ABS(A(I,J)).EQ.(1.0E+4)) GOTO 95
      IF(ABS(A(I,J)).GT.(2.5)) EFLAG(4,I)=1
      IF(ABS(A(I,J)).GT.(2.5)) IND(J)=' *'
   95 CONTINUE
      WRITE(3,1021)
 1021 FORMAT(/59X,'INTERVAL NUMBER'/)
      WRITE(3,1002)
      WRITE(3,1004) (A(I,J),J=1,10)
      WRITE(9,1004) (A(I,J),J=1,10)
      WRITE(3,1005)(IND(J),J=1,10)
      WRITE(3,1021)
```

```
      WRITE(3,1003)
      WRITE(3,1004) (A(I,J),J=11,20)
      WRITE(9,1004) (A(I,J),J=11,20)
      DO 3004 J=1,20
        RA(I,J)=A(I,J)
 3004 CONTINUE
      WRITE(3,1005) (IND(J),J=11,20)
      DO 96 J=1,25
   96 IND(J)='  '
      WRITE(3,1030)
 1030 FORMAT(/1X,'5. AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED
     1 STEADY-STATE AMPLITUDE (ALLOWABLE LIMIT +/- 13 %)')
C
C     overshoot comparison to allowable limits
C
      DO 85 J=1,20
      IF(ABS(O(I,J)).EQ.1.0E+4) GOTO 85
      IF(ABS(O(I,J)).GT.13.0) EFLAG(5,I)=1
      IF(ABS(O(I,J)).GT.13.0) IND(J)=' *'
   85 CONTINUE
      WRITE(3,1021)
      WRITE(3,1002)
      WRITE(3,1004) (O(I,J),J=1,10)
      WRITE(10,1004) (O(I,J),J=1,10)
      WRITE(3,1005) (IND(J),J=1,10)
      WRITE(3,1021)
      WRITE(3,1003)
      WRITE(3,1004) (O(I,J),J=11,20)
      WRITE(10,1004) (O(I,J),J=11,20)
      DO 5004 J=1,20
        ROS(I,J)=O(I,J)
 5004 CONTINUE
      WRITE(3,1005) (IND(J),J=11,20)
   20 CONTINUE
      WRITE(3,1040)
 1040 FORMAT(/1X,'6. CHANNEL-TO-CHANNEL TIME DIFFERENCE - ',
     1  'ALL CHANNELS ARE COMPARED TO THE FIRST CHANNEL PROCESSED'
     1       ' (ALLOWABLE LIMIT = +/- 0.25 MSEC)')
      K1=1
C
C     channel-to-channel time variation to allowable limits
C
      DO 110 I=1,ICHAN
      DO 101 J=1,20
      DEV(J)=1.0E+4
  101 IND(J)='  '
      WRITE(3,11)  RFLNM1(I),RCNUM(I)
      DO 100 J=1,20
      IF(ABS(T(I,J)).EQ.1.0E+4) GOTO 100
      DEV(J)=T(I,J)-T(K1,J)
      IF(ABS(DEV(J)).GT.0.25) EFLAG(6,I)=1
      IF(ABS(DEV(J)).GT.0.25) IND(J)=' *'
  100 CONTINUE
      WRITE(3,1021)
```

```
      WRITE(3,1002)
      WRITE(3,1004) (DEV(J),J=1,10)
      WRITE(3,1005) (IND(J),J=1,10)
      WRITE(3,1021)
      WRITE(3,1003)
      WRITE(3,1004) (DEV(J),J=11,20)
      WRITE(3,1005) (IND(J),J=11,20)
  110 CONTINUE
      WRITE(3,1060)
 1060 FORMAT(//5X,'FILE NAME',19X,'CHANNEL NO.',20X,'STATUS')
      NA=0
      NU=0
      DO 60 I=1,ICHAN
      KND=0
      DO 70 J=1,6
      IF(EFLAG(J,I).EQ.0) GOTO 70
      KND=1
   70 CONTINUE
      IF(KND.EQ.0) NA=NA+1
      IF(KND.EQ.1) NU=NU+1
      IF(KND.EQ.0) WRITE(3,1070)  RFLNM1(I),RCNUM(I)
      IF(KND.EQ.1) WRITE(3,1080)  RFLNM1(I),RCNUM(I)
 1070 FORMAT(1X,A30,10X,A2,25X,'PASS')
 1080 FORMAT(1X,A30,10X,A2,25X,'FAIL')
   60 CONTINUE
      WRITE(3,1050)  ICHAN,NA,NU
 1050 FORMAT(//1X,'TOTAL NO. OF CHANNELS PROCESSED = ',I2
     1       /1X,'NUMBER OF ACCEPTABLE CHANNELS = ',I2
     2       /1X,'NUMBER OF UNACCEPTABLE CHANNELS = ',I2)
      CLOSE(3)
      CLOSE(7)
      CLOSE(8)
      CLOSE(9)
      CLOSE(10)
      GOTO 3400
 3200 WRITE(6,*) 'ERROR!!!   *"RECTAN.SUM" CANNOT BE ACCESSED*'
      DFLAG = 1
 3400 CONTINUE
      RETURN
      END
```

The fifth segment, "STR", contains the following modules:

STAIR
ZERO3
COMSTR
COMPR2

The source code listing for these modules is:

```
C                  ccccccccccccccccccccccccc
C                  c                       c
C                  c    SUBROUTINE STAIR   c
C                  c                       c
C                  ccccccccccccccccccccccccc
C
C   Subroutine stair receives staircase waveform data as input and determines
C  the leading edges for each level.  The point number of each leading edge
C  is stored in the zerinc(I) array.  There are 12 values of zerinc, one
C  for each new level of amplitude encountered.  Zerinc(1) is the edge
C  with a theoretical time of occurrence of 10 msec.  Zerinc(12) is the
C  is the edge with a theoretical time of occurrence of 120 millisec after
C  the start of the staircase waveform.
C
C  Subroutine stair calls subroutine zer03 to determine the steady-state
C  and maximum overshoot values for each level of amplitude.  Subroutine
C  stair sends an address in the Y array (Y(LL)), the number of points
C  between two consecutive leading edges (NPT),  and the number of points
C  per millisec (NPMS) as input to subroutine zer03.  Subroutine zer03
C  sends as output the steady-state value for each level (SSPAV(I)) and
C  the maximum overshoot for each level (POSOV(I)).
C
C  Subroutine stair upon receiving the steady state (SSPAV) and overshoot
C  (POSOV) computes a timeshift (TL(I)) which equals the actual time
C  recorded minus the theoretical time for each respective amplitude
C  leading edge within the channel.
C
C  Subroutine stair then computes the normalized steady-state and normalized
C  overshoot values and stores these values in place of their respective
C  actual values in the SSPAV and POSOV arrays. The normalized steady-
C  state values are the STEADY STATE ERROR divided by the full-scale
C  theoretical amplitude multiplied by 100 to give a percentage of full
C  scale.  The normalized overshoot values are the OVERSHOOT beyond the
C  calculated steady-state value divided by the change in calculated
C  amplitude multiplied by 100 to give a percentage of actual change in
C  amplitude recorded.
C
C  Subroutine stair then writes the results to a file.  Outputting normalized
C  overshoot (POSOV(I)), normalized steady state (SSPAV(I)) and timeshift
C  (TL(I)) for each level.
C
C  Principal variables:
C    TL(I) = timeshift or time error for leading edges within a channel
C    POSOV(I) = overshoot for each level of amplitude encountered
C    SSPAV(I) = steady-state amplitude
C    NPMS = number of points per msec
C    G = full-scale output
C    NOPTS = number of points per data curve
C    ZERINC(I) = point number corresponding to a leading edge
C    LEVEL = amplitude of a particular portion of waveform
C    NPT = number of points between leading edges
C    Y(LL) = the address in the Y array that is passed to subroutine zer03
C            that defines the first element in the x array in subroutine
```

```
C            zer03.  The first element is the first point on any individual
C            level.
C
C
      SUBROUTINE STAIR(ICHAN,DFLAG,CNTFL)
C
C  Define data variables and arrays
C
         INTEGER NOPTS,LL,LU,NPT,TB,TR,DFLAG,CNTFL
         INTEGER N1(13),N2(13),N3(13),N4(13)
         INTEGER ZIC,ZERINC(22),
     *         LB(14),TZIC,UBND,LBND,START,FINISH,
     *         INGOV(13),IPSOV(13)
         REAL XMIN,XMAX,YMIN,YMAX,TO,G,LEVEL,N9
         REAL Y(-1000 :10000)
         REAL NPMS,POSOV(23),YMN,YMX,PAVGOV,NAVGOV,
     *      SSPAV(23),TL(23),TU(13)
         INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
         CHARACTER IFLNM1*30,CHANUM *2,DUMMY *1
         CHARACTER *30 CHAFIL(108)
         INTEGER CHANUMB
C
C  Define common block symbolic names
C
         COMMON /FILE/    IFLNM1
     *         /CBNPTS/ NOPTS
     *         /OVER/    POSOV,NEGOV,IPSOV,INGOV,SSPAV,SSNAV
     *         /RANGE/   FINISH
     *         /SHFT/    TL,TU
     *         /STPT/    N1,N2,N3,N4
     *         /PTN/     START,LB
     *         /COUNT/   ZIC,ZERINC
     *         /FLAG/    EFLAG
     *         /CBFIL/ CHAFIL
     *         /CBCH / CHANUMB
       COMMON/SSTAT1/SPOS(18,13),SSSPAV(18,13),STL(18,13),SFLNM1(18),
      &         SSATT(18),SCNUM(18),STPERF(18)
       CHARACTER*30 SFLNM1
       CHARACTER*25 SSATT,STPERF
       CHARACTER*2 SCNUM
       DIMENSION AVE(23)
C
C           stair waveform processor
C
       DFALG = 0
         WRITE(6,1)
     1 FORMAT(/1X,'**   STAIR WAVEFORM PROCESSOR   **')
       ICHAN = 0
       KNUM = CHANUMB
       IKJ = -2
    26 CONTINUE
       DO 8700 I=-1000,10000
         Y(I) = 0.0
  8700 CONTINUE
```

```
      IF (KNUM  .EQ. 0  .AND.  CNTFL  .EQ. 1) GOTO 300
      IKJ = IKJ +6
      YMAX = 200.
      YMIN = -200.
C
C  Initialize variables
C
      DO 22 I=1,13
      TL(I)=1.0E+4
      POSOV(I)=1.0E+4
   22 SSPAV(I)=1.0E+4
C
C  Input waveform data
C
 5000 IFLNM1 = CHAFIL(IKJ)
C
C     READING FROM DATA FILE ARRAY
C
    8 INDX=1+INT(FLOAT(IKJ)/6.0)
      CALL GETWAVE(INDX,0,3,NFP,NLP,DEL,'O',Y)
    6 ICHAN = ICHAN + 1
      KNUM = KNUM - 1
      NPMS=1.0/DEL/1000.
C
C  store edge locations in zerinc array
C
C  when waveform reaches 1/2 the expected amplitude for each respective
C  level, it is that point that determines the leading edge.
C
        ZIC=0
        G=200
        LEVEL=G/10
        NOPTS = (NLP - NFP) + 1
        DO 10 J=NFP,NOPTS
          IF(ZIC.GE.5) GOTO 20
          IF(Y(J).LT.LEVEL) GOTO 10
          ZIC=ZIC+1
          ZERINC(ZIC)=J
          LEVEL=LEVEL+G/5
          GOTO 10
   20     IF(ZIC.GT.5) GOTO 30
          LEVEL=G/2
          IF(Y(J).GT.LEVEL) GOTO 10
          ZIC=ZIC+1
          ZERINC(ZIC)=J
          GOTO 10
   30     IF(ZIC.GT.6) GOTO 40
          LEVEL=-G/2
          IF(Y(J).GT.LEVEL) GOTO 10
          ZIC=ZIC+1
          ZERINC(ZIC)=J
          LEVEL=-.90*G
          GOTO 10
   40     K=J-1
```

```
            IF(Y(J).LT.LEVEL.OR.Y(J).LT.Y(K)) GOTO 10
            ZIC=ZIC+1
            ZERINC(ZIC)=J
            LEVEL=LEVEL+G/5
   10    CONTINUE
C
C    call subroutine to calculate steady-state and
C    overshoot values for each level
C
C   initial zero level
C
          LL=1
          LU=ZERINC(1)
          NPT=LU-LL+1
          CALL ZER03(Y(LL),NPT,SSPAV(1),POSOV(1),0,NPMS)
C
C   positive levels
C
          DO 90 I=2,6
          J=I-1
          LL=ZERINC(J)
          LU=ZERINC(I)
          NPT=LU-LL+1
          CALL ZER03(Y(LL),NPT,SSPAV(I),POSOV(I),0,NPMS)
   90 CONTINUE
C
C   intermediate levels
C
          DO 11 I=7,8
          J=I-1
          LL=ZERINC(J)
          LU=ZERINC(I)
          NPT=LU-LL+1
          CALL ZER03(Y(LL),NPT,SSPAV(I),POSOV(I),1,NPMS)
   11 CONTINUE
C
C  negative levels
C
          DO 15 I=9,12
          J=I-1
          LL=ZERINC(J)
          LU=ZERINC(I)
          NPT=LU-LL+1
          CALL ZER03(Y(LL),NPT,SSPAV(I),POSOV(I),0,NPMS)
   15 CONTINUE
C
C   final zero level
C
          I=13
          LL=ZERINC(12)
          LU=NOPTS
          NPT=LU-LL+1
          CALL ZER03(Y(LL),NPT,SSPAV(I),POSOV(I),0,NPMS)
C
```

```
C   compute timeshift using points on the leading edges
C
      DO 110 I=1,12
        DAN=ZERINC(I)-1
        TL(I)=((DAN)/NPMS-10*I)
  110 CONTINUE
C
C  normalize overshoot and steady-state error values
C
C  AVE is the value against which the overshoot is normalized.
C  Since there is no amplitude change before the initial zero
C  level (theoretically) the value used for normalization
C  has been set to full-scale, steady-state value (SSPAV(6)).
      DO 199 I=1,13
        IF (I.EQ.1) THEN
          AVE(I)=SSPAV(6)
         ELSE
          AVE(I)=ABS(SSPAV(I)-SSPAV(I-1))
        END IF
  199 CONTINUE
      DO 200 I=1,13
C
C  sspav is redefined as deviation in percent of full scale
C
      POSOV(I)=ABS(POSOV(I)-SSPAV(I))/AVE(I)*100.0
  205 LEVEL=(I-1)*40
      IF(I.EQ.7) LEVEL=LEVEL-240
      IF(I.GE.8) LEVEL=LEVEL-480
      SSPAV(I)=((SSPAV(I)-LEVEL)/200.0)*100.0
      IF(I.GT.7.AND.I.LT.13) SSPAV(I)=-SSPAV(I)
  200 CONTINUE
C
C  store results
C
      I=INDEX(IFLNM1,'.')
  910 CHANUM(1:1) = IFLNM1(I+2:I+2)
      CHANUM(2:2) = IFLNM1(I+3:I+3)
      SFLNM1(INDX)=IFLNM1
      SSATT(INDX)=SENATT
      SCNUM(INDX)=CHANUM
      STPERF(INDX)=TSTPRF
      DO 210 K=1,13
        SPOS(INDX,K)=POSOV(K)
        SSSPAV(INDX,K)=SSPAV(K)
        STL(INDX,K)=TL(K)
  210 CONTINUE
      CLOSE(1)
      WRITE(6,*)'PROCESSING COMPLETED FOR FILE ',IFLNM1
      GOTO 26
 300  CONTINUE
      RETURN
      END
C
        SUBROUTINE ZER03(X,NPTS,AVG,PK,IFL,XNPM)
```

```
C
        DIMENSION X(1000),Y(1000)
        K=NPTS-XNPM
        IF(IFL.EQ.1) GOTO 520
        PK=-1.E+4
        DO 500 I=1,K
  500 PK=AMAX1(X(I),PK)
        GOTO 510
  520 PK=1.E+4
        DO 530 I=1,K
  530 PK=AMIN1(X(I),PK)
  510 K=NPTS-XNPM
      LEA=NPTS/2
      SUM=0.
      DO 747 LP=LEA,K
      SUM=SUM+X(LP)
  747 CONTINUE
      AVG=SUM/(K-LEA+1)
        RETURN
      END
C
C          ccccccccccccccccccccccccccc
C          c                         c
C          c   SUBROUTINE COMSTAIR   c
C          c                         c
C          ccccccccccccccccccccccccccc
C
C  Subroutine Comstair performs a comparison between processed stair
C  waveform data and the corresponding theoretical values desired.
C
C  Output of +/- 10000 indicates quantity could not be determined
C
C  The symbol * indicates the allowable limit has been exceeded
C
C  Time quantities are expressed in millisec
C
C  Amplitude quantities are expressed as percent of full scale
C
C  Subroutine Comstair receives data from program MGAPROC
C
C  NCHAN = number of channels
C  C(I,J) = channel number
C  T(I) = timeshift
C  EFLAG = channel status
C
      SUBROUTINE COMSTR(T,NCHAN,EFLAG,DFLAG)
C
C  define data types
C
      CHARACTER IFLNM1 *14
      CHARACTER DAY*9,HOUR *9
      CHARACTER *30 STARFI1,STARFI2,STARFI3,STARFI4,STARFI5,STARFI6
      CHARACTER *4 TSTCDNUM
      INTEGER  EFLAG(9,18),IND(23),DFALG
```

```
      REAL O(18,23),A(18,23),T(18,23),SLOPE,INTCP,DEV(23),SLOPEA,INTCPA
      INCLUDE 'DISK$OVR:[ASGPROG]COMVAR.LIS'
C
C  COMMON BLOCKS
C
      COMMON  /CBCDNUM/TSTCDNUM
      COMMON/SSTAT1/SPOS(18,13),SSSPAV(18,13),STL(18,13),SFLNM1(18),
     &       SSATT(18),SCNUM(18),STPERF(18)
      COMMON/SSTAT2/ST(18,13)
      COMMON/SSTAT3/SSLPT(18),SINTCT(18)
      COMMON/SSTAT4/SSLPA(18),SINTCA(18)
      COMMON/SSTAT5/SA(18,13)
      COMMON/SSTAT6/SOS(18,13)
      CHARACTER*30 SFLNM1
      CHARACTER*25 SSATT,STPERF
      CHARACTER*2 SCNUM
C
C  compare results of stair waveform processing with allowable limits
C
      DFLAG = 0
      STARFI1='STAIR'//TSTCDNUM//'.SUM'
      STARFI2='STAIR'//TSTCDNUM//'.001'
      STARFI3='STAIR'//TSTCDNUM//'.002'
      STARFI4='STAIR'//TSTCDNUM//'.003'
      STARFI5='STAIR'//TSTCDNUM//'.004'
      STARFI6='STAIR'//TSTCDNUM//'.005'
      OPEN(3,FILE=STARFI1,STATUS='NEW',SHARED,ERR=3200)
      OPEN(7,FILE=STARFI2,STATUS='NEW',SHARED,ERR=3200)
      OPEN(8,FILE=STARFI3,STATUS='NEW',SHARED,ERR=3200)
      OPEN(9,FILE=STARFI4,STATUS='NEW',SHARED,ERR=3200)
      OPEN(10,FILE=STARFI5,STATUS='NEW',SHARED,ERR=3200)
      OPEN(11,FILE=STARFI6,STATUS='NEW',SHARED,ERR=3200)
      WRITE(3,10)
   10 FORMAT(43X,'** STAIR WAVEFORM PROCESSOR DATA SUMMARY **'//
     1       1X,'A VALUE OF +/- 10000 INDICATES THAT THE QUANTITY ',
     2       'COULD NOT BE DETERMINED'/1X,'THE * SYMBOL INDICATES THAT'
     3       ,' THE ALLOWABLE LIMIT HAS BEEN EXCEEDED.'//1X,'TIME ',
     4        'QUANTITIES EXPRESSED IN MILLISECONDS'/1X,'AMPLITUDE ',
     5        'QUANTITIES EXPRESSED AS PERCENT OF FULL SCALE')
        CALL DATE(DAY)
        CALL TIME(HOUR)
        WRITE(3,12) DAY,HOUR
 12     FORMAT(//,50X,A9,10X,A9)
C
C  input data
C
C  O(I,J) = overshoot
C  A(I,J) = steady-state error
C  T(I,J) = timeshift
C
      IFLNM1 = 'STAIR.DAT'
      DO 20 I=1,NCHAN
      DO 200 J=1,13
      DEV(J)=1.0E+4
```

```
  200 CONTINUE
      DO 300 J=1,9
      EFLAG(J,I)=0
  300 CONTINUE
      WRITE(3,11)  SFLNM1(I),SCNUM(I)
   11 FORMAT(//1X,'RESULTS FOR FILE : ',A30,5X,'CHANNEL NO. ',A2)
      WRITE(3,13)  STPERF(I)
   13 FORMAT(/1X,'FACILITY : ',A25)
      DO 30 K =1,13
        O(I,K)=SPOS(I,K)
        A(I,K)=SSSPAV(I,K)
        T(I,K)=STL(I,K)
   30 CONTINUE
C
C  perform least squares using timeshift data on y-axis and theoretical
C  time on x-axis
C
C  if data correlation is perfect slope = 0.0  and y intercept = 0.0
C
      N=0
      SUMXSQ=0
      SUMXY=0
      SUMY=0
      SUMX=0
      DO 40 J=1,12
      IF(ABS(T(I,J)).EQ.(1.0E+4)) GOTO 40
      X=J*10
   55 SUMX=SUMX+X
      SUMY=SUMY+T(I,J)
      SUMXY=SUMXY+X*T(I,J)
      SUMXSQ=SUMXSQ+X**2
      N=N+1
   40 CONTINUE
      SLOPE=(SUMX*SUMY-N*SUMXY)/(SUMX**2-N*SUMXSQ)
      INTCP=(SUMY-SLOPE*SUMX)/N
      WRITE(3,1000)
 1000 FORMAT(/1X,'1. TIME DEVIATION FROM THEORETICAL TIME
     *  (ALLOWABLE LIMIT +/- 1.0 MSEC)')
      DO 49 J=1,13
   49 IND(J)='  '
      DO 50 J=1,12
      IF(ABS(T(I,J)).EQ.(1.0E+4)) GOTO 50
      X=J*10
C
C  DEV(I) = deviation from least squares fit
C  IND(I) = indicated channel status
C
   75 DEV(J)=T(I,J)-SLOPE*X-INTCP
C
C  EFLAG(L,I) = status indicator for a particular comparison
C  L = test being compared
C  I = channel number
C  there are eight test comparisons for each channel
C
```

```
C  EFLAG(1,J) = time deviation from least squares fit
C   ''  (2,J) = time slope deviation from least squares fit
C   ''  (3,J) = time intercept deviation from least squares fit
C   ''  (4,J) = amplitude deviation from least squares fit
C   ''  (5,J) = amplitude slope deviation from least squares fit
C   ''  (6,J) = amplitude intercept deviation from least squares fit
C   ''  (7,J) = amplitude deviation from theoretical
C   ''  (8,J) = amplitude overshoot
C
      IF(ABS(T(I,J)).GT.(1.0)) EFLAG(1,I)=1
      IF(ABS(T(I,J)).GT.1.0) IND(J)=' *'
   50 CONTINUE
      WRITE(3,1021)
      WRITE(3,1002)
 1002 FORMAT(7X,'1',9X,'2',9X,'3',9X,'4',9X,'5',9X,'6',9X,'7',9X,
     1       '8',9X,'9',8X,'10',8X,'11',8X,'12')
      WRITE(3,1004)(T(I,J),J=1,12)
      WRITE(7,1004)(T(I,J),J=1,12)
 1004 FORMAT(2X,12(F10.3))
      DO 2004 J=1,12
        ST(I,J)=T(I,J)
 2004 CONTINUE
      WRITE(3,1005)(IND(J),J=1,12)
 1005 FORMAT(2X,13(4X,1A2,4X))
      DO 52 J=1,13
      DEV(J)=1.0E+4
   52 IND(J)='  '
      WRITE(3,1010)
 1010 FORMAT(/1X,'2. TIME LINEARITY (ALLOWABLE LIMIT = +/- 1 %)',
     1       20X,'3. TIME OFFSET (ALLOWABLE LIMIT = +/- 1.0 MSEC)')
      IF(ABS(SLOPE).GT.(0.01)) EFLAG(2,I)=1
      IF(ABS(SLOPE).GT.(0.01)) IND(1)=' *'
      IF(ABS(INTCP).GT.(1.0)) EFLAG(3,I)=1
      IF(ABS(INTCP).GT.(1.0)) IND(2)=' *'
      SLOPE=SLOPE*100
      WRITE(3,1011) SLOPE,IND(1),INTCP,IND(2)
      WRITE(8,1009) SLOPE,INTCP
      SSLPT(I)=SLOPE
      SINTCT(I)=INTCP
 1009 FORMAT(1X,F6.2,2X,F6.2)
 1011 FORMAT(/10X,'SLOPE = ',F6.2,1A2,48X,'INTERCEPT = ',F6.2,1A2)
      IND(1)='  '
      IND(2)='  '
C
C  perform amplitude least squares fit
C
C  adjust theoretical values to be all positive
C  perform least squares fit on adjusted data
C  correct y - intercept value
C
      N=0
      SUMXSQ=0
      SUMXY=0
      SUMY=0
```

```
      SUMX=0
      DO 41 J=2,12
      IF(ABS(A(I,J)).EQ.1.0E+4) GOTO 41
      X=(J-1)*20+100
      IF(J.EQ.7) X=X-120
      IF(J.GE.8) X=X-240
      A1=A(I,J)
      IF(J.GE.8) A1=-A(I,J)
      SUMX=SUMX+X
      SUMY=SUMY+A1
      SUMXY=SUMXY+X*A1
      SUMXSQ=SUMXSQ+X**2
      N=N+1
   41 CONTINUE
      SLOPEA=(SUMX*SUMY-N*SUMXY)/(SUMX**2-N*SUMXSQ)
      INTCPA=(SUMY-SLOPEA*SUMX)/N+SLOPEA*100
      DO 51 J=2,12
      IF(ABS(A(I,J)).EQ.1.0E+4) GOTO 51
      A1=A(I,J)
      IF(J.GE.8) A1=-A(I,J)
      X=(J-1)*20
      IF(J.EQ.7) X=X-120
      IF(J.GE.8) X=X-240
C
C  amplitude deviation from least squares
C
      DEV(J)=A1-SLOPEA*X-INTCPA
      IF(ABS(DEV(J)).GT.2.5) EFLAG(8,I)=1
      IF(ABS(DEV(J)).GT.2.5) IND(J)=' *'
   51 CONTINUE
C
        WRITE(6,1252)
 1252   FORMAT(' STEADY-STATE AMPLITUDE DEVIATIONS FROM BEST FIT
     * STRAIGHT LINE (% FULL SCALE)')
        WRITE(6,1253)(J,DEV(J),J=2,12)
 1253   FORMAT(4('  DEV(',I2,')=',F8.3))
C
 1021 FORMAT(/59X,'INTERVAL NUMBER'/)
C      WRITE(3,1002)
C
C  output amplitude deviations and indicators
C
      DO 96 J=1,13
      DEV(J)=1.0E+4
   96 IND(J)='  '
      WRITE(3,1110)
 1110 FORMAT(/1X,'4. AMPLITUDE LINEARITY (ALLOWABLE LIMIT = +/- 2.5 %)'
     1       ,15X,'5. AMPLITUDE OFFSET (ALLOWABLE LIMIT = +/- 2.5 %)')
      IF(ABS(SLOPEA).GT.0.025) EFLAG(5,I)=1
      IF(ABS(SLOPEA).GT.0.025) IND(1)=' *'
      IF(ABS(INTCPA).GT.2.5) EFLAG(6,I)=1
      IF(ABS(INTCPA).GT.2.5) IND(2)=' *'
      SLOPEA=SLOPEA*100
      WRITE(3,1011) SLOPEA,IND(1),INTCPA,IND(2)
```

```
      WRITE(9,1009) SLOPEA,INTCPA
      SSLPA(I)=SLOPEA
      SINTCA(I)=INTCPA
      IND(1)='  '
      IND(2)='  '
      WRITE(3,1120)
 1120 FORMAT(/1X,'6. STEADY-STATE AMPLITUDE DEVIATION FROM ',
     1       ' THEORETICAL AMPLITUDE'
     1       ' (ALLOWABLE LIMIT = +/- 2.5 %)')
      DO 95 J=1,13
      IF(ABS(A(I,J)).EQ.1.0E+4) GOTO 95
      IF(ABS(A(I,J)).GT.2.5) EFLAG(7,I)=1
      IF(ABS(A(I,J)).GT.2.5) IND(J)=' *'
   95 CONTINUE
      WRITE(3,1021)
C
C  output steady-state results
C
      WRITE(3,1002)
      WRITE(3,1004)(A(I,J),J=1,12)
      WRITE(10,1004)(A(I,J),J=1,12)
      DO 3004 J=1,12
        SA(I,J)=A(I,J)
 3004 CONTINUE
      WRITE(3,1005)(IND(J),J=1,12)
      DO 97 J=1,13
      DEV(J)=1.0E+4
   97 IND(J)='  '
      WRITE(3,1030)
 1030 FORMAT(/1X,'7. AMPLITUDE OVERSHOOT RELATIVE TO CALCULATED
     1 STEADY-STATE AMPLITUDE (ALLOWABLE LIMIT +/- 13 %)')
      DO 85 J=2,13
      IF(ABS(O(I,J)).EQ.1.0E+4) GOTO 85
      IF(ABS(O(I,J)).GT.13.0) EFLAG(8,I)=1
      IF(ABS(O(I,J)).GT.13.0) IND(J)=' *'
   85 CONTINUE
      WRITE(3,1021)
      WRITE(3,1002)
C
C  output overshoot results
C
      WRITE(3,1004) (O(I,J),J=2,13)
      WRITE(11,1004) (O(I,J),J=2,13)
      DO 4004 J=2,13
        SOS(I,J)=O(I,J)
 4004 CONTINUE
      WRITE(3,1005) (IND(J),J=2,13)
   20 CONTINUE
      GOTO 320
 3200 CLOSE(3)
      WRITE(6,*) 'ERROR !!!'
      WRITE(6,*) ' *** "STAIR.SUM" CANNOT BE OPENED ***'
      DFLAG = 1
  320 CLOSE(7)
```

```
      CLOSE(8)
      CLOSE(9)
      CLOSE(10)
      CLOSE(11)
      RETURN
      END
C
C               ccccccccccccccccccccccccccccc
C               c                           c
C               c      SUBROUTINE COMPR2    c
C               c                           c
C               ccccccccccccccccccccccccccccc
C
C  Subroutine Compr2 checks the channel-to-channel time error.
C  All channels are compared to the first channel processed.
C
C  Input to subroutine Compr2 will be the number of channels
C  and the timeshift of each level of each channel processed.
C  Output will be the channel status of all channels processed.
C
      SUBROUTINE COMPR2(T,NCHAN,EFLAG)
C
C  define data types
C
      COMMON/SSTAT1/SPOS(18,13),SSSPAV(18,13),STL(18,13),SFLNM1(18),
     &       SSATT(18),SCNUM(18),STPERF(18)
      CHARACTER*30 SFLNM1
      CHARACTER*25 SSATT,STPERF
      CHARACTER*2 SCNUM,IND(23)
      REAL T(18,23),DEV(23)
      INTEGER EFLAG(9,18)
      WRITE(3,1040)
 1040 FORMAT(/1X,'8. CHANNEL-TO-CHANNEL TIME DIFFERENCE,'
     1       ' ALL CHANNELS ARE COMPARED TO THE FIRST'
     1       ' CHANNEL PROCESSED'
     1       ' (ALLOWABLE LIMIT = +/- 0.25 MSEC)')
      K1=1
      DO 110 I=1,NCHAN
      DO 101 J=1,13
      DEV(J)=1.0E+4
  101 IND(J)='  '
C
      WRITE(3,11)  SFLNM1(I),SCNUM(I)
      DO 100 J=1,12
      IF(ABS(T(I,J)).EQ.1.0E+4) GOTO 100
C
C  calculate channel-to-channel time deviation
C  comparing to the first channel processed
C
      DEV(J)=T(I,J)-T(K1,J)
      IF(ABS(DEV(J)).GT.0.25) EFLAG(9,I)=1
      IF(ABS(DEV(J)).GT.0.25) IND(J)=' *'
  100 CONTINUE
C
```

```
C  output deviations
C
      WRITE(3,1021)
      WRITE(3,1002)
      WRITE(3,1004) (DEV(J),J=1,12)
      WRITE(3,1005) (IND(J),J=1,12)
  110 CONTINUE
      WRITE(3,1060)
 1060 FORMAT(//5X,'FILE NAME',19X,'CHANNEL NO.',20X,'STATUS')
      NA=0
      NU=0
      DO 60 I=1,NCHAN
      KND=0
      DO 70 J=1,9
      IF(EFLAG(J,I).EQ.0) GOTO 70
      KND=1
   70 CONTINUE
C
C  determine channel status
C
      IF(KND.EQ.0) NA=NA+1
      IF(KND.EQ.1) NU=NU+1
      IF(KND.EQ.0) WRITE(3,1070)  SFLNM1(I),SCNUM(I)
      IF(KND.EQ.1) WRITE(3,1080)  SFLNM1(I),SCNUM(I)
 1070 FORMAT(1X,A30,10X,A2,25X,'PASS')
 1080 FORMAT(1X,A30,10X,A2,25X,'FAIL')
   60 CONTINUE
C
C  output channel status
C
      WRITE(3,1050) NCHAN,NA,NU
 1050 FORMAT(//1X,'TOTAL NO. OF CHANNELS PROCESSED = ',I2
     1       /1X,'NUMBER OF ACCEPTABLE CHANNELS = ',I2
     2       /1X,'NUMBER OF UNACCEPTABLE CHANNELS = ',I2)
      CLOSE(3)
   11 FORMAT(//1X,'RESULTS FOR FILE : ',A30,5X,'CHANNEL NO. ',A2)
 1021 FORMAT(/59X,'INTERVAL NUMBER'/)
 1002 FORMAT(7X,'1',9X,'2',9X,'3',9X,'4',9X,'5',9X,'6',9X,'7',9X,
     1       '8',9X,'9',8X,'10',8X,'11',8X,'12')
 1004 FORMAT(2X,13(F10.3))
 1005 FORMAT(2X,13(4X,1A2,4X))
       RETURN
      END
```

6. EPROM CHECK-OUT SOFTWARE

This section presents a narrative description, flow diagram, and program listing of the software which has been developed at MGA Research Corporation (MGA) to compare the content of an EPROM with the ideal test waveforms.

The ideal test waveform data was generated and stored in files on the VAX 11/780 at NHTSA. The data files were then transmitted to an IBM PC at MGA and converted to a binary format suitable to be burned into EPROMs. Due to the size of the waveform data files, the waveform was divided into segments such that each segment would be only as large as the capacity of one of the EPROMs in the waveform generator instrument. In total, there are eight EPROMs contained within the waveform generator instrument which contain two sequences of test waveforms. The EPROMs were burned using an IBM PC and a PROLOG EPROM programmer. Once they were burned, the contents of each EPROM was read back and stored in a data file. The data file was then compared against the ideal test waveform data using the EPROM check-out program.

The EPROM check-out program was written in the BASIC language on an IBM PC. The program first establishes the required arrays for the data and then prompts the user for the name of the data file which contains the EPROM data (the data file which was read back from the EPROMs). Once the user has responded, the program checks the data read back from the EPROMs against the data files which contain the ideal waveform data. If there are any discrepancies between the two data sources, the user is given a "FAIL" message otherwise a "PASS" message is given. Shown in Figures 6-1 and 6-2 are the flow diagram and program listing respectively, for this program.

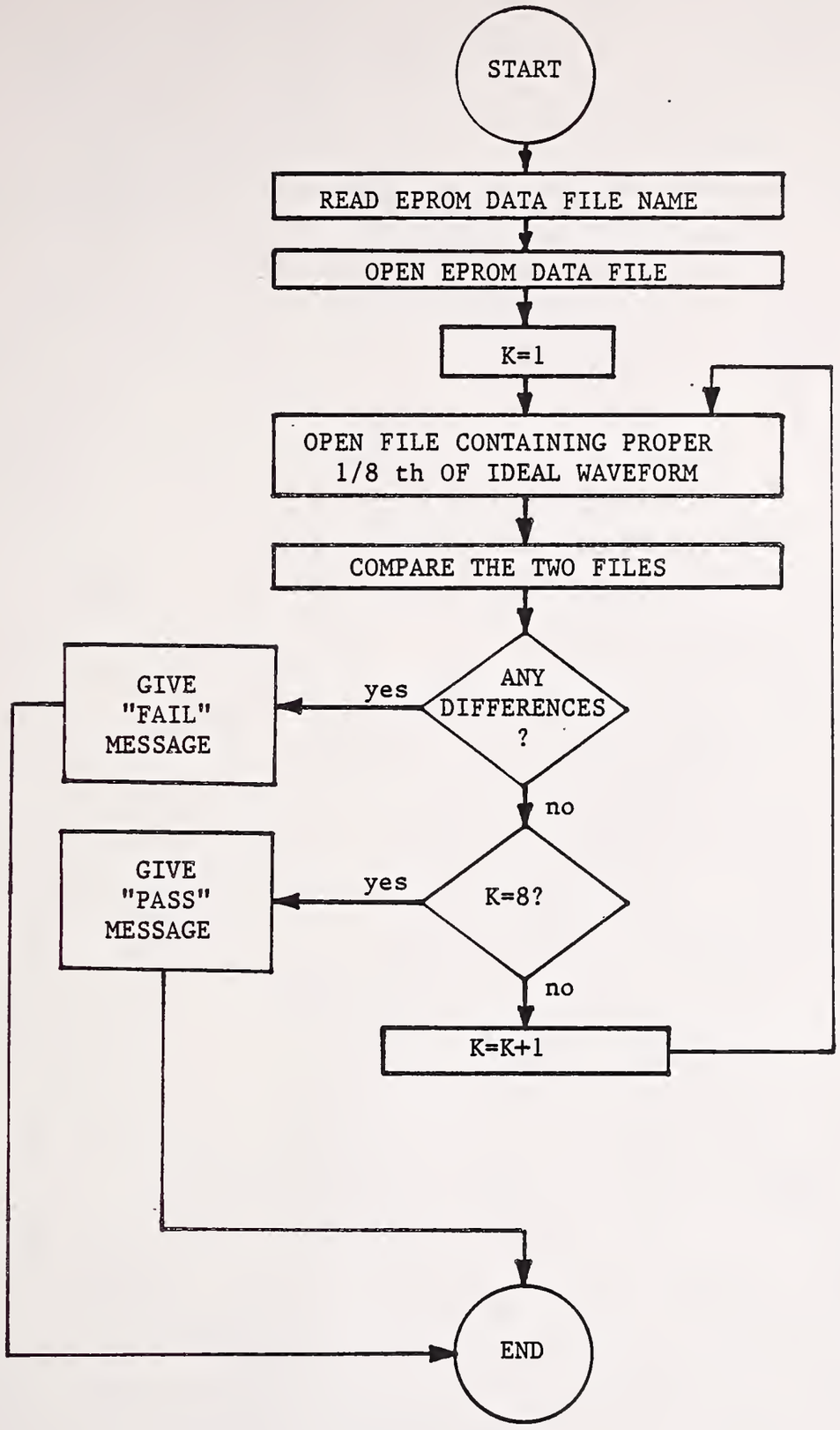


Figure 6-1 Flow Diagram of EPROM Check-out Program

```
10 CLS
15 WIDTH 80
20 DIM C$(8),X3$(16)
30 C$(1)=".001"
35 C$(2)=".002"
40 C$(3)=".003"
50 C$(4)=".004"
55 C$(5)=".005"
60 C$(6)=".006"
65 C$(7)=".007"
70 C$(8)=".008"
100 INPUT "PLEASE ENTER THE EPROM LEVEL (LOW/HIGH) ††",C9$
110 C10$="B:"+C9$+".DAT"
120 OPEN "I",#1,C10$
121 FOR K=1 TO 8
130    C11$=C9$+C$(K)
140    OPEN "I",#2,C11$
150    FOR N=1 TO 63
160       INPUT #1,X1$
167       REM PRINT X1$
170       X2$=INPUT$(50,#2)
175       NN=1
180       FOR J=1 TO 16
190          X3$(J)=MID$(X2$,NN,2)
200          NN=NN+3
210       NEXT J
220       FOR J=1 TO 16
230          X4$=X4$+X3$(J)
240       NEXT J
245       REM PRINT X1$
250       IF (X4$=X1$)=0  THEN 300
252       X4$=""
255    NEXT N
256    GOTO 260
260    PRINT "PASS",K
261    CLOSE #2
262 NEXT K
263 CLOSE #1
264 END
300 PRINT "FAIL"
310 END
```

Figure 6-2 EPROM Check-Out Program Listing

U.S. Department
sportation

irch and
al Programs
nistration

Square
ge, Massachusetts 02142

Official Business
Penalty for Private Use $300

Postage and Fees Paid
Research and Special
Programs Administration
DOT 513