

SHRP-LTPP/UWP-92-615

**Data Readability and Completeness
FWDSCAN Version 1.30
Program Background and
User's Guide**

PCS/Law



**Strategic Highway Research Program
National Research Council
Washington, DC 1992**

SHRP-LTPP/UWP-92-615

Contract P-001

Program Manager: *Neil F. Hawks*

Project Manager: *Cheryl Richter*

Production Editor: *Francine Burgess*

Program Area Secretary: *Cindy Baker, Francine Burgess*

November 1992

key words:

falling weight deflectometer

FWDCHECK

FWDFSCAN

long-term pavement performance

Strategic Highway Research program

National Academy of Sciences

2101 Constitution Avenue N.W.

Washington, DC 20418

(202) 334-3774

This paper represents the views of the author(s) only, and is not necessarily reflective of the views of the National Research Council, the views of SHRP, or SHRP's sponsor. The results reported here are not necessarily in agreement with the results of other SHRP research activities. They are reported to stimulate review and discussion within the research community.

Acknowledgments

The research described herein was supported by the Strategic Highway Research Program (SHRP). SHRP is a unit of the National Research Council that was authorized by section 128 of the Surface Transportation and Uniform Relocation Assistance Act of 1987.

Contents

| | |
|----------------------------|-----|
| Acknowledgments | iii |
| List of Figures | vii |
| List of Tables | ix |
| Abstract | xi |
| Executive Summary | 2 |
| Purpose | 2 |
| Background | 2 |
| Program Description | 3 |
| Program Usage | 10 |
| Interactive Mode | 10 |
| Command Line Mode | 11 |
| Other Considerations | 13 |
| Technical Assistance | 13 |
| Appendix | 15 |

List of Figures

| | | |
|---|---|---|
| 1 | An Example of a Block of Header Information | 4 |
| 2 | Excerpt of a Data Block, Including the Beginning of a History Block | 6 |
| 3 | Excerpt of an Example Output File Containing Errors | 8 |
| 4 | Example Output File Containing No Errors | 9 |

List of Tables

| | | |
|---|--------------------------------|----|
| 1 | Summary of Function Keys | 12 |
|---|--------------------------------|----|

Abstract

Nondestructive deflection testing using falling weight deflectometers is one element of the monitoring effort currently underway by the Strategic Highway Research Program (SHRP) for the Long-Term Pavement Performance (LTPP) study. Because accurate data is key to the success of the LTPP study, SHRP has implemented a number of measures to ensure the quality of deflection data. They include equipment comparison and calibration, standardized field testing procedures and field data checks, and quality assurance software.

Equipment calibration and field data checks built into the FWD data acquisition software are the first line of defense against involved deflection data. The second line of defense is a computer program, called FWDSCAN, which verifies the integrity, completeness, and compliance with the established test pattern of the field data after it is delivered to the SHRP regional office. For the final stage in the quality assurance process, a computer program called FWDCHECK has been developed to analyze deflection data for test section homogeneity, the degree to which test pit data is representative of the section, the presence of data outliers within the section, and overall reasonableness from a structural capacity viewpoint.

This report focuses on the FWDSCAN program. In the first part of the report, a detailed description of the program is given. A detailed description of the program usage is given in the next portion of the report. Finally, a complete printout of the computer source code is included in the last part of the report.

Executive Summary

PURPOSE

The purpose of this report is to describe the use of the first FWD Quality Assurance product for use by the SHRP Regional Coordination Offices (RCO). Program **FWDESCAN** is intended to check FWD data files for completeness and readability, and generates an output file summarizing the results of the checking process.

BACKGROUND

One of the most important data items that will be collected during the monitoring phase of the SHRP LTPP study is the deflection response of GPS and SPS pavement test sections under an applied load. In order to measure this response, SHRP is utilizing a non-destructive testing device called the Falling Weight Deflectometer (FWD). Each SHRP Regional Coordination Office (RCO) contractor is responsible for storing, maintaining and operating one FWD unit and the towing vehicle supplied by SHRP for the FWD deflection data collection.

In order to provide a uniform and standardized field deflection measurement procedure for SHRP-FWD units within each of the four operating SHRP RCO's, a SHRP publication titled "SHRP-LTPP Manual for FWD Testing and Operational Field Guidelines" was released for use in January 1989. Part of the field data collection scheme is a computer software system for test set-up and data collection and storage. While the main purpose of the SHRP FWD program is the automated data collection process, five separate computerized data checks are built in to the data collection software to alert the FWD operator to potential data problems. They are:

- Roll-Off - an electrical check of the deflection sensors to verify that the signal attenuates with time.
- Decreasing Deflections - a check to verify that deflections are lower at increasing distances from the load.
- Out of Range - a check to verify that deflections are less than the maximum deflection that the sensor is capable of recording accurately.
- Load Variation - a check that the load for a particular drop is within a specified tolerance of the average load for that drop height at that location.
- Deflection Variation - a check that the normalized deflection for a given sensor for a particular drop is within a specified tolerance of the average normalized deflection for that sensor for that drop height at that location.

After completion of each test section and before leaving the site, backup copies of the deflection data are made in order to safeguard the information collected in the field.

One of these copies, along with the printed hard copy produced by the data collection software, is mailed to the SHRP RCO where the data must be reconstituted into files as they originally existed in the field and then verified.

All data received by the SHRP RCO must be checked to insure that it has been restored to its original form and that all data is present, complete and in a readable form. Unlike the data collection software, however, the objective of these checks is not to eliminate data but rather to flag potential errors or problems, and if possible, correct them before the information is processed further.

To accomplish this, a microcomputer program called **FWDSCAN** has been developed by the P-001 Technical Advisory Staff. This program automatically checks the data for completeness and readability, and generates an output file summarizing the results of the checking process. A description of **FWDSCAN** and its usage are presented in the remainder of this document.

PROGRAM DESCRIPTION

Each FWD data file can be thought of as being comprised of two primary parts. The first part consists of 36 lines of header information, an example of which is contained in Figure 1. The second part of the data file, known as the data block, consists of the loads, deflections, temperatures and stations collected during a given pass.

Verifying the contents of the header information consists of two distinct parts. The first part involves the comparison of the data items in the file to either lists of possible values or specified values; e.g., that the load plate radius is the correct value. The second part of the verification procedure consists of checking that the remaining items contain reasonable values for what they represent; e.g., that calibration factors (Figure 1, lines 10 to 20, third data item) are between .98 and 1.02 for all deflectors.

The data block of the FWD data file consists of a repeating series of lines defining the testing at each station. For each station, the following data should be present:

1. A line beginning with an "S" containing station identification, lane specification and other information that occurs once at each station (Figure 2, line 1)
2. Twelve or sixteen lines (depends on the pavement type being tested) of load and deflection data (Figure 2, lines 2 through 13)
3. Operator comments (optional)

4. 151 or 301 line blocks, the first line of which begins with a "B", containing load and deflection time history data for a single drop height at this station (Figure 2, lines 14 to 44, for example), repeated up to 3 or 4 times at each station, depending on the pavement type being tested.

Accordingly, verification of the data block contents consists of scanning each line of the remainder of the FWD data file to ensure its readability and completeness. In addition, to insure the reasonableness of the data, the verification procedure also entails a comparison of the data with valid data ranges; i.e., lane specification codes (Figure 2, line 1, J0) must be from the list specified in the FWD manual (e.g., F0, J2, C4, etc.), deflection values must range from 0.1 to 80.0 mils, loads must range from 500 to 32,000 lbf, and air and surface temperatures must range from 0° to 160° F.

A list of the specific data items that are checked during the scanning process is provided below. Some are checks for the presence of specific data items, some other checks are for data within an acceptable range of numbers, and finally, some of these checks are simply a report of what was present in the data file.

I. Miscellaneous checks

- A. Report scan start time and date
- B. Verify line length for each line
- C. Report number of skipped records
- D. Report total number of records processed
- E. Report scan end time and date

II. Checks of header information

- A. Determine units for data collection
- B. Report total expected number of records
- C. Determine data collection date
- D. Verify use of edition 10 of Dynatest software
- E. Determine number of active deflectors
- F. Determine deflector range
- G. Determine FWD serial number
- H. Determine deflection filtering mode
- I. Determine load cell gain
- J. Determine deflector gains
- K. Determine operator's name

Figure 2: Excerpt of a data block, including the beginning of a history block

```

1  S -50J0 -3 4 I60713 26 38 Heights -----
2  554 92 83 78 73 67 56 35 8800 3.63 3.27 3.05 2.89 2.63 2.19 1.36
3  553 92 84 79 74 69 58 38 8784 3.63 3.31 3.09 2.93 2.72 2.28 1.49
4  552 90 82 78 73 68 57 37 8776 3.54 3.22 3.05 2.89 2.68 2.23 1.45
5  554 91 84 79 74 68 58 38 8808 3.58 3.31 3.09 2.93 2.68 2.28 1.49
6  775 130 119 114 106 99 83 56 12320 5.13 4.69 4.48 4.19 3.89 3.26 2.22
7  774 132 120 116 107 100 83 56 12304 5.21 4.73 4.56 4.23 3.93 3.26 2.22
8  772 131 119 115 106 99 83 57 12272 5.17 4.69 4.52 4.19 3.89 3.26 2.26
9  772 131 119 115 106 99 83 57 12272 5.17 4.69 4.52 4.19 3.89 3.26 2.26
10 1108 187 170 162 153 141 119 81 17608 7.38 6.70 6.39 6.03 5.57 4.68 3.20
11 1107 186 170 162 153 141 120 81 17592 7.33 6.70 6.39 6.03 5.57 4.73 3.20
12 1102 187 170 161 152 141 119 81 17512 7.38 6.70 6.35 5.99 5.57 4.68 3.20
13 1102 187 170 162 153 141 119 81 17520 7.38 6.70 6.39 6.03 5.57 4.68 3.20
14 B150 1 rd1 rd2S -50-----
15 3 -1 -1 -1 -1 -1 -1 -1 4 0 0 0 0 0 0 0 0
16 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1
18 0 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
19 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
20 0 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
21 0 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
22 0 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
23 0 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
24 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
25 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
26 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
27 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
28 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
29 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
30 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
31 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
34 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
35 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -1 -1 -1
36 5 -1 -1 -1 -1 -1 -1 -1 12 -1 -1 -1 -1 -1 -1 -1
37 24 -1 -1 -1 -1 -1 -1 -1 36 -1 -1 -1 -1 -1 -1 -1
38 47 -1 -1 -1 -1 -1 -1 -1 59 -1 0 -1 -1 -1 -1 -1
39 74 -1 0 -1 0 -1 -1 -1 94 0 0 0 0 -1 -1 -1
40 119 1 1 0 0 0 -1 -1 148 2 2 1 1 0 0 -1
41 184 3 2 2 1 1 0 -1 225 4 3 2 2 1 0 -1
42 268 6 4 3 3 2 1 -1 311 8 6 5 4 2 1 -1
43 352 12 9 6 5 3 2 -1 389 14 11 9 6 4 2 -1
44 426 16 13 10 9 7 2 0 456 18 14 12 11 9 3 0

```

III. Checks at each station

A. Peak data block

1. Verify use of a valid lane specification
2. Verify that stations are increasing (subject to applicability at particular points in the test sequence)
3. Verify that stations are within the 500 foot limits of the test section (warning only, does not require remedial action)
4. Determine number of peak records recorded
5. Verify temperatures are within acceptable range
6. Verify deflections are within deflector limits
7. Verify that deflections decrease at increasing distance from the load plate (warning only, does not require remedial action)
8. Verify load is within acceptable range
9. Check for comments
10. Check for unidentifiable data
11. Check for new subsection id
12. Check for unexpected end of file

B. History data block

1. Determine number of history records recorded
2. Verify deflections are within deflector limits
3. Verify load is within acceptable range
4. Check for comments
5. Check for unidentifiable data
6. Check for new subsection id
7. Check for unexpected end of file

All results of these scanning procedures are written to an output file as a permanent record of this process having been performed. Erroneous or inconsistent data items are noted by lines beginning with an asterisk. These lines fully describe which data element is incorrect or inconsistent. Some errors in the data file generate more than 1 warning message, and sometimes they effect whether subsequent data is interpreted correctly. Figure 3, an excerpt from one of these files, contains 2 warnings, only the first of which is truly an error. In this case, the unrecognized lane specification at station 219 causes the lane specification at station 231 to appear to be out of the normal sequence of testing. Figure 4 contains the entire contents of a scan of an FWD data file that had no errors.

Figure 3: Excerpt of an example output file containing errors

150 history records successfully read in block 2 at station 207
150 history records successfully read in block 3 at station 207
* Undefined lane specification (J3)) at station 219
12 peak records successfully read at station 219
150 history records successfully read in block 1 at station 219
150 history records successfully read in block 2 at station 219
150 history records successfully read in block 3 at station 219
* Test sequence locations not in expected order at station 231
Testing J2 (RIGID) at station 231
12 peak records successfully read at station 231
150 history records successfully read in block 1 at station 231
150 history records successfully read in block 2 at station 231
150 history records successfully read in block 3 at station 231
Testing J3 (RIGID) at station 238
12 peak records successfully read at station 238
150 history records successfully read in block 1 at station 238

Figure 4: Example output file containing NO errors

FWDSCAN started at 16:32:53 on 07-12-1989
English units data collection
969 total records in this file
Data collected on 890309
7 active deflectors
Deflectors have 2000 micron range
FWD Serial Number - 8002-059
Deflections filtered by 120Hz filter
Load cell relative gain = .999
Deflector 1 relative gain = .992
Deflector 2 relative gain = .992
Deflector 3 relative gain = .996
Deflector 4 relative gain = .989
Deflector 5 relative gain = .995
Deflector 6 relative gain = .991
Deflector 7 relative gain = .991
Deflector 8 relative gain = 1
Deflector 9 relative gain = 1
Deflector 10 relative gain = 1
Data collected by OPERATOR NAME
Testing J0 (RIGID) at station -50
12 peak records successfully read at station -50
150 history records successfully read in block 1 at station -50
150 history records successfully read in block 2 at station -50
150 history records successfully read in block 3 at station -50
Testing J0 (RIGID) at station 565
12 peak records successfully read at station 565
150 history records successfully read in block 1 at station 565
150 history records successfully read in block 2 at station 565
150 history records successfully read in block 3 at station 565
FWDSCAN ended at 16:33:02 on 07-12-1989

In addition to the data checks being performed, an additional data file (with an extension of .PKS) is created containing only header information and peak deflection data. The purpose of this file is to improve the speed at which the remaining Quality Assurance programs can perform their tasks.

PROGRAM USAGE

Significant effort has been made to make FWDSCAN simple and straightforward to use at the RCO. There are 2 ways in which to use FWDSCAN. The first method would be to use FWDSCAN in its interactive mode where files to scan are either picked from a list or entered by name directly. The second method would be to specify the name of the file to be scanned on the MS-DOS command line, either one at a time, or multiple files one after the other by use of advanced MS-DOS features. All of these methods are described below.

Interactive Mode

Using FWDSCAN in its interactive mode consists of only 2 screens that the user must be familiar with. The first of these screens is the title screen, where no input is required. The second, and most important screen, contains all of the fields that make the program function.

- Field 1: Path to data files - the path to the desired FWD data files may be entered in this field by typing the full path (assumes default drive if no drive is specified) or by pressing <Return> for the current directory. The path does NOT require a backslash as the LAST character. If an error is detected when attempting to change to an invalid or nonexistent directory, an error message will appear on the screen.
- Field 2: Show a list of files - a yes/no question that allows the user to select the file to be scanned from the list of FWD data files (denoted with the file spec "*.FWD" only) in the specified directory. If the response is YES, then the user is placed in the directory list and cursor keys are used to highlight a file in the list that can be selected by pressing <Return>. <PgUp> and <PgDn> can also be used to move backwards or forwards one page at a time, where such a quantity of files exists. <Esc> allows the user to exit the file list WITHOUT selecting one of the files.

Field 3: Data file name - if a file was picked from the list of files in the specified directory, that file's name appears in this field and the file will be scanned when <PgDn> is pressed. If the field is blank, enter a valid MS-DOS filename, and press <PgDn> to scan the file. If the file does not exist, an error message will appear on the screen.

The user should also be familiar with the various FUNCTION KEYS that can be used during the use of interactive mode. These keys are listed in Table 1 along with a brief explanation of their function. Note that the various FUNCTION KEYS available to the user appear at the bottom of the screen.

Command Line Mode

Command line mode of FWDSCAN allows the user to specify the name of the file to be scanned directly on the command line. For example,

```
C:\> FWDSCAN D:\PATH\FILENAME.EXT
```

will scan the file 'filename.ext' located on drive d: in directory \path without user intervention. With this mode, it is possible to perform these scans from within a batch file so that many data files may be scanned overnight or over the course of the day. For example, the following command line will scan all data files that match the file specification in the current directory.

```
C:\DATA\SHRP> FOR %C IN (*.FWD) DO \PROGS\SHRP\FWDSCAN %C
```

Similarly, a batch file of the following form would allow multiple directories to be scanned without an operator being present.

```
FOR %%C IN (\DATA\FWD\DIR1\*.FWD) DO FWDSCAN %%C  
FOR %%C IN (\DATA\FWD\DIR2\*.FWD) DO FWDSCAN %%C  
etc.
```

This example batch file assumes that it is being run from the directory containing the FWDSCAN program and it could easily be modified to allow it to be run from another directory entirely. The user should be aware that the output files will be located in the same directory as the FWD data files.

Please also note the differences between the direct command line example and the batch file example. While differing uses of paths are shown, the key difference between the

Table 1
SUMMARY OF FUNCTION KEYS

| <u>KEY(S)</u> | <u>FUNCTION</u> |
|----------------------------------|---|
| <F10> | EXIT - the <F10> key is used to exit the program. |
| <Esc> | ESCAPE - returns the user to field 2 from the file list without selecting a file. |
| <PgDn>,<PgUp> | PAGE DOWN - used to EXECUTE the scan for the file that has been picked, in other words, to continue the execution of the program. PAGE DOWN or PAGE UP - used in the directory window if more than 20 files are present, to move from one page of the list to the next/previous page, where applicable. |
| <↑>,<↓> | ARROW KEYS - these keys allow the user to move from one field to another on the data entry screen, as well as to move from file to file in the directory window. Also, when more than one page of files are available in the directory window, pressing <DOWN> on the last row of the window places the cursor on the first row of the next page of the list. Pressing <UP> when on the top line of the second or subsequent page of the list of files will move the cursor to the bottom line of the previous page in the list. |
| <Home>,<End> | HOME or END - these keys allow the user to quickly move to the first or last field within the data entry screen, as well as the first or last file in the current page of the directory window. |
| <Space Bar> | SPACE BAR - the <Space Bar> key is used to exit the various warnings or errors that appear at the bottom of the data entry screen. |
| <CR>,<Enter> | CARRIAGE RETURN or ENTER - used to accept a data input value once it has been entered. |

two examples is that batch files use '%C' and direct commands use '%C'. Please refer to your MS-DOS manual for a complete explanation of this difference.

OTHER CONSIDERATIONS

Depending on the results of the verification procedure, a number of scenarios are possible for the case of "bad data". For all scenarios, the first two remedial steps are the same. The first step would consist of redoing the restore procedure and repeating the verification. The purpose of repeating the restore step is that the restore step itself may have been the cause of the data corruption, and repeating it MAY eliminate the problem.

If that fails to remedy the problem, the second step would be to request that the FWD operator transmit another copy of the data to the SHRP RCO. If that also fails to remedy the problem, then additional steps must be taken, depending on the exact nature of the problem with the data. Specific resolution of problems will be made, on a case-by-case basis, by the RCO in cooperation with the SHRP staff in Washington, D.C.

With successful completion of the scanning procedure for a test section, backup diskettes containing multiple copies of the FWD data for this section may be released for re-use by the RCO. The scanned data files should then be backed up by whatever means are normally used for all LTPP data files in the RCO. In addition, the FWD data is now ready to be tested by the remaining Quality Assurance programs.

TECHNICAL ASSISTANCE

If further technical assistance is required in the usage of this program, please contact PCS/Law Engineering, Beltsville, Maryland at 301-604-5105.

REVISION NOTES

Version 1.30 - June 14, 1990

- Added support for version 20 of the Dynatest data collection software.
- Altered some of the 'error' messages which are actually warnings so that the text begins with an '!' instead of a '*'. These include the situations of 'not decreasing deflections', 'load outside expected range', and 'peak deflection out of range'.

- Took care of the occasional problem of stray characters placed by the data collection software in the joint/crack opening field. Program should no longer crash if random characters are encountered there.

Version 1.20 - August 8, 1989

- Added a number of checks which have come to light during development of additional QA programs, see III.A.2., III.A.3., and III.A.7. above.
- Added generation of a 'peaks' data file, for use with other QA programs.
- Altered file list window to show ONLY '.FWD' data files, to conform with the "SHRP Manual for FWD Testing" file naming convention.

Version 1.10 - July 28, 1989

- Added command line mode, for use directly and with batch files.
- Added the name of the scanned FWD file to the *.out file.

Version 1.00 - July 13, 1989

- Initial release.

Appendix

FWDFCAN 1.30 PROGRAM LISTING

```

DECLARE SUB ParseCommandLine (NumArgs%, Args$(), MaxArgs%)
DECLARE SUB SkippedRecs (Total%)
DECLARE SUB DisplayCopyright ()
DECLARE SUB CheckHeader (InitNumPeaks%, InitNumMHBLOCKS%, ExitCode%)
DECLARE SUB ReadPeaks ()
DECLARE SUB ReadNextLine (DataType%, LineLength%)
DECLARE SUB ReadHistory ()
DECLARE SUB ParseHistoryBlockHeader (NumRecsInBlock%, HistStation%)
DECLARE SUB ParsePeaksBlockHeader (TestStation%)
DECLARE SUB GetFileName (FPath$, File$, Ext%)
DECLARE SUB NormalColor ()
DECLARE SUB HiliteColor ()
DECLARE SUB TitleColor ()
DECLARE SUB GetString (NRow%, NCol%, FldLen%, Entry$, Justify$, PopupFlag%, HelpNum%, LogDesc$, Update$,
ExitCode%)
DECLARE SUB PopupError ()
DECLARE SUB ScreenBorder ()
DECLARE SUB DisplayFileNames (NumMatches%, ShowFiles$, FPath$, File$, Ext$, ExitCode%, HelpNum%)
'INCLUDE: 'declare.inc'
'INCLUDE: 'cmnblank.inc'

```

```

COMMON SHARED /scan1/ LineCounter&, LineData$, NumHistBlocks%, HistoryBlock%
COMMON SHARED /scan2/ MMaxDefl, EMaxDefl, NumDeflRecs%, FileWidth%, TotalRecs%
COMMON SHARED /scan3/ NumStationsTested%

```

```

CONST True% = -1, False% = 0, MaxArgs% = 1
REDIM Args$(MaxArgs%)

```

```

CALL ParseCommandLine(NumArgs%, Args$(), MaxArgs%)
GP.Monitor% = Monitor%
CALL DisplayCopyright
FPath$ = ""
DO

```

```

IF NumArgs% = 0 THEN
CALL GetFileName(FPath$, File$, Ext%)
Source$ = FPath$ + File$ + Ext$
Target$ = FPath$ + File$
ELSE
Source$ = Args$(1)
IF NOT Exist%(Source$) THEN END
SP% = INSTR(Args$(1), ".")
IF SP% <> 0 THEN
File$ = LEFT$(Args$(1), SP% - 1)
Ext$ = LTRIMS(RTRIMS(RIGHT$(Args$(1), LEN(Args$(1)) - (SP% - 1))))
ELSE
File$ = LTRIMS(RTRIMS(Args$(1)))
Ext$ = ""
END IF
Target$ = File$
END IF

```

```

OPEN Source$ FOR INPUT AS #1
IF UCASE$(Ext$) = ".OUT" THEN
Ext2$ = ".TXT"
ELSE
Ext2$ = ".OUT"
END IF
Ext3$ = ".PKS"
Target2$ = Target$ + Ext2$
Target3$ = Target$ + Ext3$
OPEN Target2$ FOR OUTPUT AS #2
OPEN Target3$ FOR OUTPUT AS #3
CLS
LineCounter& = 0
HistoryBlock% = 0
PRINT #2, "FWOSCAN (v1.20) of file: "; Source$
PRINT #2, "Started at "; TIME$; " on "; DATE$
LS% = LEN(Source$)
Msg$ = "Reading FWD data from file: "

```

```

IF LSX > 51 THEN
  C1X = 41 - LSX \ 2
  LOCATE 9, 27: PRINT Msg$
  LOCATE 10, C1X: PRINT Source$
ELSE
  C1X = 41 - (LSX + 28) \ 2
  LOCATE 10, C1X: PRINT Msg$; Source$
END IF
LOCATE 12, 32: PRINT "Reading line #"
NumStationsTested% = 0
DO
  IF LineCounter% < 37 THEN
    CALL CheckHeader(NumDeflRecs%, NumHistBlocks%, ExitCode%)
    IF ExitCode% = 9999 THEN
      ExitCode% = 0
      EXIT DO
    END IF
  ELSE
    CALL ReadNextLine(DataType%, LineLength%)
    SELECT CASE DataType%
      CASE 1 'peak deflection data block
        HistoryBlock% = 0
        CALL SkippedRecs(NumSkipped%)
        CALL ReadPeaks
      CASE 2 'sensor history block data
        CALL SkippedRecs(NumSkipped%)
        CALL ReadHistory
      CASE 3 'comments in data file
        CALL SkippedRecs(NumSkipped%)
        PRINT #2, "Comment -"; LTRIMS(RTRIMS(LineData$))
      CASE 4 'unknown data
        NumSkipped% = NumSkipped% + 1
      CASE 5
        CALL SkippedRecs(NumSkipped%)
        PRINT #2, "** Subsection ID "; LTRIMS(RTRIMS(LineData$))
      CASE -1
        EXIT DO
      CASE 0 'starts with ????'
        PRINT #2, "** Data of unknown type on line"; LineCounter%
        NumSkipped% = NumSkipped% + 1
      CASE ELSE
        NumSkipped% = NumSkipped% + 1
    END SELECT
  END IF
LOOP
IF TotalRecs% <> LineCounter% - 1 THEN
  PRINT #2, "** Total lines read ("; LineCounter%; ") NOT equal to expected number of lines ("; TotalRecs%;
  "...)"
END IF
PRINT #2, "FWDSCAN ended at "; TIME$; " on "; DATE$
CLOSE
IF NumArgs% = 1 THEN
  COLOR 7, 0, 0
  CLS
  PRINT "Thank you for using FWDCAN. Have a nice day!"
  EXIT DO
END IF
LOOP
END

SUB CheckHeader (InitNumPeaks%, InitNumWHBlocks%, ExitCode%) STATIC
CALL ReadNextLine(DataType%, LineLength%)
PRINT #3, LineData$
LOCATE 12, 46: PRINT LineCounter%
SELECT CASE LineCounter%
  CASE 1
    FileWidth% = VAL(MID$(LineData$, 2, 4))
    IF FileWidth% = 32 THEN

```

```

    PRINT #2, "Metric units data collection"
ELSE
    PRINT #2, "English units data collection"
END IF
TotalRecs& = VAL(MID$(LineData$, 8, 6))
PRINT #2, TotalRecs&; "total records in this file"
FileDate$ = MID$(LineData$, 14, 6)
PRINT #2, "Data collected on "; FileDate$
SFileName$ = MID$(LineData$, 20, 8)
NumHeaderLines% = VAL(MID$(LineData$, 28, 2))
IDS = MID$(LineData$, 30, 1)
Edition% = VAL(MID$(LineData$, 31, 2))
IF Edition% <> 10 AND Edition% <> 20 THEN
    PRINT #2, "** If FWD data, not collected with edition 10 or 20 of Dynatest's software..."
    ExitCode% = 9999
    EXIT SUB
END IF
CASE 2
NumDeflectors% = VAL(LEFT$(LineData$, 1))
PRINT #2, NumDeflectors%; "active deflectors"
Range% = VAL(MID$(LineData$, 2, 1))
SELECT CASE Range%
    CASE 0
        MMaxDefl = 2000
        EMaxDefl = 2000 / 25.4
        PRINT #2, "Deflectors have 2000 micron range"
    CASE 1
        MMaxDefl = 2540
        EMaxDefl = 100
        PRINT #2, "Deflectors have 100 mil range"
    CASE ELSE
        PRINT #2, "Deflectors have an unknown range"
END SELECT
TLockType% = VAL(MID$(LineData$, 3, 1))
HCtrlType% = VAL(MID$(LineData$, 4, 1))
DMIType% = VAL(MID$(LineData$, 5, 1))
PossFiltTypes% = VAL(MID$(LineData$, 6, 1))
StorOpts% = VAL(MID$(LineData$, 7, 1))
TempsBlanked% = VAL(MID$(LineData$, 8, 1))
FWDSN$ = MID$(LineData$, 9, 8)
PRINT #2, "FWD Serial Number - "; FWDSN$
DMICalib& = VAL(MID$(LineData$, 17, 8))
FilterMode% = VAL(MID$(LineData$, 25, 1))
SELECT CASE FilterMode%
    CASE 0
        PRINT #2, "Deflections are unfiltered"
    CASE 1
        PRINT #2, "Deflections filtered by 120Hz filter"
    CASE 2
        PRINT #2, "Deflections filtered by 60Hz filter"
    CASE 5
        PRINT #2, "Deflections filtered by 120Hz filter w. history"
    CASE 6
        PRINT #2, "Deflections filtered by 60Hz filter w. history"
    CASE ELSE
        PRINT #2, "Deflections filtered by an UNKNOWN method"
END SELECT
PrinterType% = VAL(MID$(LineData$, 26, 1))
PrintStyle% = VAL(MID$(LineData$, 27, 1))
PaperLeng% = VAL(MID$(LineData$, 28, 4))
LinesPerInch% = VAL(MID$(LineData$, 32, 1))
CASE 3
MLPR = VAL(MID$(LineData$, 1, 4))
IF MLPR <> 150 THEN PRINT #2, "** Metric load plate size (; MLPR; ") not correct"
MRadial1 = VAL(MID$(LineData$, 5, 4))
MRadial2 = VAL(MID$(LineData$, 9, 4))
MRadial3 = VAL(MID$(LineData$, 13, 4))
MRadial4 = VAL(MID$(LineData$, 17, 4))

```

```

MRadial5 = VAL(MID$(LineData$, 21, 4))
MRadial6 = VAL(MID$(LineData$, 25, 4))
MRadial7 = VAL(MID$(LineData$, 29, 4))
ELPR = VAL(MID$(LineData$, 33, 6))
IF ELPR <> 5.9 THEN PRINT #2, "** English load plate size ("; ELPR; ") not correct"
ERadial1 = VAL(MID$(LineData$, 39, 6))
ERadial2 = VAL(MID$(LineData$, 45, 6))
ERadial3 = VAL(MID$(LineData$, 51, 6))
ERadial4 = VAL(MID$(LineData$, 57, 6))
ERadial5 = VAL(MID$(LineData$, 63, 6))
ERadial6 = VAL(MID$(LineData$, 69, 6))
ERadial7 = VAL(MID$(LineData$, 75, 6))
CASE 4
Wdrive$ = LEFT$(LineData$, 2)
Wdirectory$ = LTRIMS(RTRIMS(MID$(LineData$, 3, 26)))
Wext$ = MID$(LineData$, 29, 4)
CASE 5
CASE 6
CASE 7
CASE 8
CASE 9
CASE 10
Sensor$ = LEFT$(LineData$, 3)
SerialNum = VAL(MID$(LineData$, 4, 5))
RelGain = VAL(MID$(LineData$, 10, 5))
IF RelGain < .98 OR RelGain > 1.02 THEN
PRINT #2, "** Load cell relative gain ("; RelGain; ") out of range"
ELSE
PRINT #2, "Load cell relative gain ="; RelGain
END IF
InitGain = VAL(MID$(LineData$, 16, 5))
CASE 11 TO 20
Sensor$ = LEFT$(LineData$, 3)
SerialNum = VAL(MID$(LineData$, 4, 5))
RelGain = VAL(MID$(LineData$, 10, 5))
IF RelGain < .98 OR RelGain > 1.02 THEN
PRINT #2, "** Deflector"; LineCounter& - 10; "relative gain ("; RelGain; ") out of range"
ELSE
PRINT #2, "Deflector"; LineCounter& - 10; "relative gain ="; RelGain
END IF
InitGain = VAL(MID$(LineData$, 16, 5))
CASE 21
Operator$ = LTRIMS(RTRIMS(LineData$))
PRINT #2, "Data collected by "; Operator$
CASE 22
TestUnits% = VAL(MID$(LineData$, 1, 1))
TestTemp% = VAL(MID$(LineData$, 2, 1))
StnReq% = VAL(MID$(LineData$, 3, 1))
TestChks% = VAL(MID$(LineData$, 4, 1))
RejPrmpt% = VAL(MID$(LineData$, 5, 1))
StatType% = VAL(MID$(LineData$, 6, 1))
TmpReq% = VAL(MID$(LineData$, 7, 1))
CndReq% = VAL(MID$(LineData$, 8, 1))
CASE 23
CASE 24
CASE 25
CASE 26
CASE 27
CASE 28
CASE 29
CASE 30
Posit% = INSTR(LineData$, ".")
ActiveDrops% = Posit% - 1
CASE 31
CASE 32
CheckText$ = LEFT$(LineData$, ActiveDrops%)
InitNumPlots% = InCount2%(CheckText$, "***")
CASE 33

```

```

    CheckText$ = LEFT$(LineData$, ActiveDrops%)
    InitNumPeaks% = InCount2%(CheckText$, "***")
CASE 34
    CheckText$ = LEFT$(LineData$, ActiveDrops%)
    InitNumLHBlocks% = InCount2%(CheckText$, "***")
CASE 35
    CheckText$ = LEFT$(LineData$, ActiveDrops%)
    InitNumWHBlocks% = InCount2%(CheckText$, "***")
CASE 36
    'setup name
END SELECT
END SUB

SUB DisplayCopyright STATIC
SCREEN 0: WIDTH 80: CLS
PRINT
PRINT "
PRINT "
PRINT "
PRINT "
PRINT "
LOCATE 10, 28: PRINT "FWD Data Scanning Software"
LOCATE 12, 35: PRINT "Version 1.30"
LOCATE 14, 20: PRINT "
For EXCLUSIVE Use by the"
LOCATE 15, 20: PRINT "Strategic Highway Research Program (SHRP)"
LOCATE 16, 20: PRINT "and its contractors and sub-contractors."
LOCATE 22, 8: PRINT "Support material Copyright (c) 1989,1990 PCS/Law Engineering Inc."
LOCATE 23, 10: PRINT "Additional material Copyright (c) 1988,1989 Crescent Software"
SLEEP 5
CALL ClearBuf
END SUB

SUB GetFileName (FPath$, File$, Ext$) STATIC
STATIC ZPS
WindowType% = 1: CLS
IF ZPS = "" THEN ZPS = "N"
WFile$ = File$
IF Ext$ <> "" THEN
    WFile$ = WFile$ + Ext$
END IF
CALL ScreenBorder
CALL TitleColor
Title$ = " FWD Data File Selection "
TL% = LEN(Title$)
Col% = ((80 - TL%) / 2) + 1
LOCATE 2, Col%: PRINT Title$
CALL NormalColor
LOCATE 7, 7: PRINT "Directory path for data file: ";
LOCATE 10, 7: PRINT "Do you want a list of data files for this path (Y/N) "
LOCATE 13, 7: PRINT "Deflection Data File Name: "
CALL HiliteColor
LOCATE 7, 37: PRINT FPath$
LOCATE 10, 60: PRINT ZPS
LOCATE 13, 34: PRINT WFile$
CALL NormalColor
LOCATE 25, 4
PRINT " F10:Quit "; CHR$(24); CHR$(25);
PRINT " Home End PgDn";
Item% = 1
MaxItem% = 3
DO
    SELECT CASE Item%
    CASE 1
        OldPath$ = FPath$
        CALL GetString(7, 37, 32, FPath$, "L", 0, 0, "", "", ExitCode%)
        FPath$ = LTRIM$(RTRIM$(UCASE$(FPath$)))
        CurrDrive$ = CHR$(GetDrive%)
        CurrDir$ = GetDir$(CurrDrive$)
        CurrPath$ = CurrDrive$ + ":" + CurrDir$

```

```

IF FPath$ <> "" THEN
IF MIDS(FPath$, 2, 1) = ":" THEN
  ChkDrive$ = LEFT$(FPath$, 1)
  IF NOT GoodDrive$(ChkDrive$) THEN 'check if valid drive
    REDIM PUText$(1)
    PUText$(1) = "Drive " + ChkDrive$ + " is not a valid choice... Please try another path."
    CALL PopupError
    ExitCode% = 0
    FPath$ = OldPath$
  ELSE 'drive OK, check dir
    IF RIGHT$(FPath$, 1) = "\" THEN
      FPath$ = LEFT$(FPath$, LEN(FPath$) - 1)
    END IF
    IF RIGHT$(FPath$, 1) = ":" THEN
      FPath$ = FPath$ + "\"
    END IF
    CALL CDir(FPath$, ErrFlag%)
    IF NOT ErrFlag% THEN 'path OK
      CALL CDir(CurrPath$, ErrFlag%) ' switch back to curr dir
    ELSE 'path not OK
      REDIM PUText$(2)
      PUText$(1) = "Error occurred switching to " + FPath$
      PUText$(2) = "May not be a valid path... Please try again."
      CALL PopupError
      ExitCode% = 0
      FPath$ = OldPath$
    END IF
  END IF
ELSE 'no drive letter in specified path
  IF RIGHT$(FPath$, 1) = "\" THEN
    FPath$ = LEFT$(FPath$, LEN(FPath$) - 1)
  END IF
  CALL CDir(FPath$, ErrFlag%)
  IF NOT ErrFlag% THEN 'path OK
    CALL CDir(CurrPath$, ErrFlag%) ' switch back to curr dir
  ELSE 'path not OK
    REDIM PUText$(2)
    PUText$(1) = "Error occurred switching to " + FPath$
    PUText$(2) = "May not be a valid path... Please try again."
    CALL PopupError
    ExitCode% = 0
    FPath$ = OldPath$
  END IF
END IF
IF FPath$ <> "" AND RIGHT$(FPath$, 1) <> "\" THEN FPath$ = FPath$ + "\"
LOCATE 7, 37: PRINT FPath$
CASE 2
DO
  OldZPS = ZPS
  CALL GetString(10, 60, 1, ZPS, "L", 0, 0, "", "", ExitCode%)
  ZPS = UCASE$(ZPS)
  SELECT CASE ZPS
  CASE "Y"
    ShowFiles$ = FPath$ + "*.FWD"
    NumMatches% = FCount$(ShowFiles$)
    IF NumMatches% > 0 THEN
      CALL DisplayFileNames(NumMatches%, ShowFiles$, FPath$, File$, Ext$, ExitCode%, 0)
      WFiles$ = File$ + Ext$
    ELSE
      REDIM PUText$(1)
      PUText$(1) = "No files found matching " + ShowFiles$
      CALL PopupError
      ZPS = "N"
    END IF
  CASE "N"
    'go on
  CASE ELSE

```

```

        REDIM PUText$(1)
        PUText$(1) = "Please choose a Y or N only... try again!"
        CALL PopupError
        ExitCode% = 0
    END SELECT
    IF ExitCode% <> 0 THEN EXIT DO
LOOP
CASE 3
DO
    OldWFile$ = WFile$
    CALL GetString(13, 34, 12, WFile$, "L", 0, 0, "", "", ExitCode%)
    WFile$ = LTRIMS(RTRIMS(UCASE$(WFile$)))
    LF = LEN(WFile$)
    FOR VW = 1 TO LF
        chk = ASC(MID$(WFile$, VW, 1))
        IF chk = 32 THEN
            REDIM PUText$(1)
            PUText$(1) = "SPACES ARE NOT ALLOWED IN FILE NAMES"
            CALL PopupError
            WFile$ = OldWFile$
            ExitCode% = 0
            EXIT FOR
        END IF
    NEXT VW
    IF ExitCode% <> 0 THEN
        SPX = INSTR(WFile$, ".")
        IF SPX <> 0 THEN
            File$ = LEFT$(WFile$, SPX - 1)
            Ext$ = LTRIMS(RTRIMS(RIGHT$(WFile$, LEN(WFile$) - (SPX - 1))))
        ELSE
            File$ = LTRIMS(RTRIMS(LEFT$(WFile$, 8)))
            Ext$ = ""
        END IF
    END IF
    EXIT DO
END IF
LOOP
END SELECT
SELECT CASE ExitCode%                                'determine next action
CASE 71                                             'home
    Item% = 1
CASE 79                                             'end
    Item% = MaxItem%
CASE 15, 75, 72                                     'Shift-Tab, left arrow, up arrow
    Item% = Item% - 1
CASE 9, 13, 77, 80                                 'Tab, CR, right arrow, down arrow
    Item% = Item% + 1
CASE 81                                             'PgDn
    IF File$ = "" THEN
        REDIM PUText$(1)
        PUText$(1) = "A file name must be entered... please try again!"
        CALL PopupError
        Item% = 3
    ELSE
        ChkName$ = FPath$ + File$ + Ext$
        IF NOT Exist$(ChkName$) THEN
            REDIM PUText$(1)
            PUText$(1) = "File not found... Please try again."
            CALL PopupError
            File$ = ""
            Ext$ = ""
            ExitCode% = 0
            Item% = 3
        ELSE
            ExitCode% = 1
        END IF
    END IF
CASE 68                                             'F10: quit

```

```

    COLOR 7, 0, 0
    CLS
    PRINT "Thank you for using FWDSKAN. Have a nice day!"
    END
    CASE ELSE
    ' do nothing
    END SELECT
    IF Item% < 1 THEN Item% = 1
    IF Item% > MaxItem% THEN Item% = MaxItem%
    LOOP
END SUB

SUB ParseHistoryBlockHeader (NumRecsInBlock%, HistStation$) STATIC
'process start of history data block
NumRecsInBlock% = VAL(MID$(LineData$, 2, 4))
BlockType% = VAL(MID$(LineData$, 6, 3))
HistStation$ = LTRIMS(RTRIMS(MID$(LineData$, 17, 8)))
END SUB

SUB ParsePeaksBlockHeader (TestStation$) STATIC
'process start of sensor data block
TestStation$ = LTRIMS(RTRIMS(MID$(LineData$, 2, 8)))
LaneSpec$ = UCASE$(LTRIMS(RTRIMS(MID$(LineData$, 10, 4))))
SpecOK% = True%
PointOK% = True%
PvmtType$ = LEFT$(LaneSpec$, 1)
PointLocation% = VAL(RIGHT$(LaneSpec$, 1))
CurrStation = VAL(TestStation$)
IF NumStationsTested% > 1 AND (OldPointLocation% = PointLocation%) AND (OldStation >= CurrStation) THEN
    PRINT #2, "** Station NOT increasing from"; OldStation; "to"; CurrStation
END IF
IF (CurrStation < 0 OR CurrStation > 500) AND PointLocation% <> 0 THEN
    PRINT #2, "** Station is outside section boundaries and is NOT a test pit location"
END IF
OldStation = CurrStation
SELECT CASE PvmtType$
    CASE "F"
        IF LEN(LaneSpec$) <> 2 THEN
            SpecOK% = False%
        ELSE
            SectionType$ = "FLEX"
            SELECT CASE PointLocation%
                CASE 0, 1, 3
                    IF NumStationsTested% <> 1 AND PointLocation% <> OldPointLocation% THEN
                        IF PointLocation% <> 1 OR OldPointLocation% <> 0 THEN
                            PointOK% = False%
                        END IF
                    END IF
                CASE ELSE
                    SpecOK% = False%
            END SELECT
        END IF
    CASE "C", "J"
        IF LEN(LaneSpec$) <> 2 THEN
            SpecOK% = False%
        ELSE
            IF PvmtType$ = "C" THEN
                SectionType$ = "CRCP"
            ELSE
                SectionType$ = "RIGID"
            END IF
            SELECT CASE PointLocation%
                CASE 0
                    IF NumStationsTested% <> 1 AND OldPointLocation% <> 0 THEN
                        PointOK% = False%
                    END IF
                CASE 1
                    IF NumStationsTested% <> 1 AND PointLocation% <> OldPointLocation% THEN

```

```

        IF PointLocation% <> 1 OR OldPointLocation% <> 0 THEN
            PointOK% = False%
        END IF
    END IF
CASE 2
    IF NumStationsTested% <> 1 THEN
        IF OldPointLocation% <> 3 THEN
            PointOK% = False%
        END IF
    END IF
CASE 3
    IF OldPointLocation% <> 2 THEN
        PointOK% = False%
    END IF
CASE 4
    IF NumStationsTested% <> 1 THEN
        IF OldPointLocation% <> 5 THEN
            PointOK% = False%
        END IF
    END IF
CASE 5
    IF OldPointLocation% <> 4 THEN
        PointOK% = False%
    END IF
CASE ELSE
    SpecOK% = False%
END SELECT
END IF
CASE ELSE
    SpecOK% = False%
END SELECT
IF SpecOK% THEN
    IF OldSectionType$ <> SectionType$ AND OldSectionType$ <> "" THEN
        PRINT #2, "** Found change in pavement type from "; OldSectionType$; " to "; SectionType$; " at station
"; TestStations$
    END IF
    IF NOT PointOK% THEN
        PRINT #2, "** Test sequence locations not in expected order at station "; TestStations$
    END IF
    PRINT #2, "Testing "; LaneSpec$; " ("; SectionType$; ") at station "; TestStations$
    OldSectionType$ = SectionType$
    OldPointLocation% = PointLocation%
ELSE
    IF SectionType$ = "" THEN
        PRINT #2, "** No lane specification at station "; TestStations$
    ELSE
        PRINT #2, "** Undefined lane specification ("; LaneSpec$; ") at station "; TestStations$
    END IF
END IF
MASphTemp = VAL(MID$(LineData$, 14, 5))
IF MASphTemp < -18 OR MASphTemp > 72 THEN
    PRINT #2, "** Metric asphalt temperature outside normal range at station "; TestStations$
END IF
JntOpening$ = MID$(LineData$, 19, 2) 'to work around occasional stray characters
ItsOK% = True%
FOR aa% = 1 TO 2
    abc$ = MID$(JntOpening$, 1, aa%)
    IF NOT INSTR(1, "0123456789", abc$) THEN
        ItsOK% = False%
    EXIT FOR
END IF
NEXT aa%
IF ItsOK% THEN JntOpen% = VAL(JntOpening$)
MSurfTemp = VAL(MID$(LineData$, 21, 3))
IF MSurfTemp < -18 OR MSurfTemp > 72 THEN
    PRINT #2, "** Metric surface temperature outside normal range at station "; TestStations$
END IF
MAirTemp = VAL(MID$(LineData$, 24, 3))

```

```

IF MAirTemp < -18 OR MAirTemp > 72 THEN
  PRINT #2, "** Metric air temperature outside normal range at station "; TestStation$
END IF
StatDir$ = MID$(LineData$, 27, 1)
StatMode% = VAL(MID$(LineData$, 28, 1))
TimeOfDay$ = MID$(LineData$, 29, 4)
EAsphTemp = VAL(MID$(LineData$, 33, 4))
IF EAsphTemp < 0 OR EAsphTemp > 160 THEN
  PRINT #2, "** English asphalt temperature outside normal range at station "; TestStation$
END IF
ESurfTemp = VAL(MID$(LineData$, 37, 4))
IF ESurfTemp < 0 OR ESurfTemp > 160 THEN
  PRINT #2, "** English surface temperature outside normal range at station "; TestStation$
END IF
EAirTemp = VAL(MID$(LineData$, 41, 4))
IF EAirTemp < 0 OR EAirTemp > 160 THEN
  PRINT #2, "** English air temperature outside normal range at station "; TestStation$
END IF
END SUB

SUB ReadHistory
  HistoryBlock% = HistoryBlock% + 1
  CALL ParseHistoryBlockHeader(BlockRecs%, HistStation$)
  NumHistRecsRead% = 0
  FOR J% = 1 TO BlockRecs%
    CALL ReadNextLine(DataType%, LineLength%)
    LOCATE 12, 46: PRINT LineCounter&
    SELECT CASE DataType%
      CASE -1
        'encountered end of file
        PRINT #2, "** Found end of file when trying to read history"; HistoryBlock%; ", record"; J%; "at station
"; HistStation$
        EXIT FOR
      CASE 0, 4
        'unknown data in line #xxxx
        PRINT #2, "** Found unknown data when trying to read history"; HistoryBlock%; ", record"; J%; "at station
"; HistStation$
        EXIT FOR
      CASE 1
        'process new start of sensor data block
        PRINT #2, NumHistRecsRead%; "history records successfully read in block"; HistoryBlock%; "at station
"; HistStation$
        PRINT #2, "** Found start of sensor data block when trying to read history"; HistoryBlock%; ", record";
J%; "at station "; HistStation$
        CALL ReadPeaks
        EXIT SUB
      CASE 2
        'process new start of history block
        PRINT #2, NumHistRecsRead%; "history records successfully read in block"; HistoryBlock%; "at station
"; HistStation$
        PRINT #2, "** Found start of new history block when trying to read history"; HistoryBlock%; ", record";
J%; "at station "; HistStation$
        CALL ReadHistory
        EXIT SUB
      CASE 3
        'found a comment
        PRINT #2, "** Found comment when trying to read history"; HistoryBlock%; ", record"; J%; "at station ";
HistStation$
        PRINT #2, "Comment -"; LTRIM$(RTRIM$(LineData$))
        EXIT FOR
      CASE 5
        'found subsection id
        PRINT #2, "** Found next subsection ID when trying to read history"; HistoryBlock%; ", record"; J%; "at
station "; HistStation$
        EXIT FOR
      CASE ELSE
        'process data normally
        FOR LX = 1 TO 2
          HTestLoad = VAL(MID$(LineData$, 1, 4))
          FOR K% = 1 TO 7
            Posit% = K% * 4 + 1
            HDefl = VAL(MID$(LineData$, Posit%, 4))
            IF HDefl > MMaxDefl THEN
              PRINT #2, "** Deflection out of range in history"; HistoryBlock%; "at station "; HistStation$
            END IF
          
```

```

        NEXT KX
        NEXT LX
        NumHistRecsRead% = NumHistRecsRead% + 1
    END SELECT
    NEXT JX
    PRINT #2, NumHistRecsRead%; "history records successfully read in block"; HistoryBlock%; "at station ";
    HistStation$
END SUB

```

```

SUB ReadNextLine (DataType%, LineLength%) STATIC
    STATIC OldDataType%
    IF NOT EOF(1) THEN
        LINE INPUT #1, LineData$
        LineLength% = LEN(LineData$)
        IF FileWidth% <> LineLength% AND LineCounter% <> 0 THEN
            PRINT #2, "** Line"; LineCounter%; "is"; LineLength%; "characters long, should be"; FileWidth%;
            "characters..."
        END IF
        DataType$ = LEFT$(LineData$, 1)
        DataType% = INSTR("SB'E*- 1234567890", DataType$)
        OldDataType% = DataType%
        LineCounter% = LineCounter% + 1
        IF DataType% = 4 THEN
            IF UCASE$(LEFT$(LineData$, 3)) = "EOF" THEN
                DataType% = -1
            END IF
        END IF
    ELSE
        DataType% = -1
        'end of file occurred
    END IF
END SUB

```

```

SUB ReadPeaks
    NumStationsTested% = NumStationsTested% + 1
    CALL ParsePeaksBlockHeader(TestStation$)
    PRINT #3, LineData$
    NumPeaksRead% = 0
    FOR I% = 1 TO NumDeflRecs%
        CALL ReadNextLine(DataType%, LineLength%)
        LOCATE 12, 46: PRINT LineCounter%
        SELECT CASE DataType%
            CASE -1
                'end of file encountered
                PRINT #2, "** Found end of file when trying to read drop"; I%; "at station "; TestStation$
                EXIT FOR
            CASE 0, 4
                'unknown data in line #xxxx
                PRINT #2, "** Found unknown data when trying to read drop"; I%; "at station "; TestStation$
                EXIT FOR
            CASE 1
                PRINT #2, NumPeaksRead%; "peak records successfully read at station "; TestStation$
                PRINT #2, "** Found start of sensor data block when trying to read drop"; I%; "at station "; TestStation$
                CALL ReadPeaks
                EXIT SUB
            CASE 2
                'start of history block
                PRINT #2, NumPeaksRead%; "peak records successfully read at station "; TestStation$
                PRINT #2, "** Found start of history block when trying to read drop"; I%; "at station "; TestStation$
                HistoryBlock% = 0
                CALL ReadHistory
                EXIT SUB
            CASE 3
                'found a comment
                PRINT #2, "** Found comment when trying to read drop"; I%; "at station "; TestStation$
                PRINT #2, "Comment -"; LTRIM$(RTRIM$(LineData$))
                EXIT FOR
            CASE 5
                'found subsection id
                PRINT #2, "** Found next subsection ID when trying to read drop"; I%; "at station "; TestStation$
                EXIT FOR
            CASE ELSE
                'normal processing
                PRINT #3, LineData$
                MTestLoad = VAL(MID$(LineData$, 1, 4))
        END SELECT
    NEXT I%
END SUB

```

```

IF MTestLoad < 31.5 OR MTestLoad > 2013.8 THEN
  PRINT #2, "! Metric load value ("; MTestLoad; ") outside expected range"
END IF
FOR J% = 1 TO 7
  Posit% = J% * 4 + 1
  MDefl = VAL(MIDS(LineData$, Posit%, 4))
  IF MDefl > MMaxDefl THEN
    PRINT #2, "! Metric peak deflection out of range in drop"; NumPeaksRead%; "at station ";
TestStations$
  END IF
  SELECT CASE J%
    CASE 1
      LastDefl = MDefl
    CASE ELSE
      IF LastDefl < MDefl THEN
        PRINT #2, "! Metric deflection NOT decreasing at sensor"; J%; "in drop"; I%; "at station ";
TestStations$
      END IF
      LastDefl = MDefl
  END SELECT
NEXT J%
IF FileWidth% > 32 THEN
  ETestLoad = VAL(MIDS(LineData$, 33, 6))
  IF ETestLoad < 500 OR ETestLoad > 32000 THEN
    PRINT #2, "! English load value ("; ETestLoad; ") outside expected range"
  END IF
  FOR J% = 1 TO 7
    Posit% = J% * 6 + 33
    EDefl = VAL(MIDS(LineData$, Posit%, 6))
    IF EDefl > EMaxDefl THEN
      PRINT #2, "! English peak deflection out of range in drop"; NumPeaksRead%; "at station ";
TestStations$
    END IF
    SELECT CASE J%
      CASE 1
        LastDefl = EDefl
      CASE ELSE
        IF LastDefl < EDefl THEN
          PRINT #2, "! English deflection NOT decreasing at sensor"; J%; "in drop"; I%; "at station ";
TestStations$
        END IF
        LastDefl = EDefl
    END SELECT
  NEXT J%
  END IF
  NumPeaksRead% = NumPeaksRead% + 1
END SELECT
NEXT I%
PRINT #2, NumPeaksRead%; "peak records successfully read at station "; TestStations$
END SUB

SUB SkippedRecs (Total%) STATIC
  IF Total% > 0 THEN
    PRINT #2, "** Skipped"; Total%; "records of undetermined types"
  END IF
  Total% = 0
END SUB

```

