Development of an Unstable Slope Management System

Appendix A

WA-RD 270.2

Final Report
December 1991

Washington State Department of Transportation
Washington State Transportation Commission
Department of Transportation in cooperation with
U.S. Department of Transportation
Federal Highway Administration

WASHINGTON STATE DEPARTMENT OF TRANSPORTATION

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
WA-RD 270.2		

4. TITLE AND SUBTITLE	5. REPORT DATE
Development of an Unstable Slope Management System	December 1991
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S)	8. PERFORMING ORGANIZATION REPORT NO.
Carlton L. Ho (1) and Sonja S. Norton (2)	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Washington State Transportation Center and Washington State University Pullman, WA 99164-2910	10. WORK UNIT NO.
	11. CONTRACT OR GRANT NO. GC-8720, T2
	13. TYPE OF REPORT AN PERIOD COVERED
12. SPONSORING AGENCY NAME AND ADDRESS Washington State Department of Transportation Olympia, WA 98504	Final Report, 7/89-12-91
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

(1) Washington State University
(2) U. S. Forest Service, Durango, CO

16. ABSTRACT

This report presents a prototype of an Unstable Slope Management System (USMS) and corresponding user's guide. The USMS is a computer program that prioritizes unstable slopes. The system is composed of two parts: a database, and priority programs. The database was developed using dBASE III Plus, Ashton-Tate. The priority programs were developed using the expert shell system CLIPS, a NASA developed language. The resulting USMS, at this point, is not an expert system; it is a management system. The USMS was developed by the aid of conversations with Washington State Department of Transportation (WSDOT) personnel. In addition, a questionnaire was sent to WSDOT personnel concerned with unstable slope maintenance. From the conversations and responses to the questionnaire, the factors concerned with site importance were identified. Also, a method to determine the total importance was proposed. The USMS identifies factors that determine the importance of a failure site. These factors pertain to the cause of instability, cost of repair, use of road, and safety to motorists. Data pertaining to these factors is collected for each failure site and stored in the database. Priority ratings are assigned by the priority rating programs to the data for each site. The priority ratings are multiplied by a weight. The sum of the products represents the total priority. The total priority is a number from 0 to 100, 100 indicates the highest importance. The total priority represents the importance of the failure site based on the factors identified in the USMS. The total priority of a failure site is independent of all other failure sites.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
landslides, slopes, management system,	No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22616.

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
None	None	327	

DOT FORM 3110.022

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

APPENDIX A

USMS.CLP

Executes CLIPS Program
(USMS.CLP)

This program controls the execution of all of the CLIPS programs. The programs are executed in the following order:

1. PROBTYPE.CLP
2. TRIMPEDE.CLP
3. PUBRISK.CLP
4. PAVEDAM.CLP
5. STRUCTUR.CLP
6. TEMPLOAD.CLP
7. FAILFREQ.CLP
8. COSTPR.CLP
9. FSIZEPR.CLP
10. EQUAKE.CLP
11. ADTROADT.CLP
12. DIRT.CLP
13. ROCK.CLP
14. ECONIMPO.CLP
15. PERMLOAD.CLP
16. GEOHAZ.CLP

```
;USMS.CLP                    Last Revision: 7-31-91
; UPDATED FOR CLIPS VER 5.0
; FROM CONTROL.CLP
;  USES BINARY LOAD COMMAND IN CLIPS
(bload "d:\\clipsfil\\probtype.bin")
(reset)
(run)
(printout t "PROBTYPE.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\trimpede.bin")
(reset)
(run)
(printout t "TRIMPEDE.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\pubrisk.bin")
(reset)
(run)
(printout t "PUBRISK.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\pavedam.bin")
(reset)
(run)
(printout t "PAVEDAM.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\structur.bin")
(reset)
(run)
(printout t "STRUCTUR.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\tempload.bin")
(reset)
(run)
(printout t "TEMPLOAD.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\failfreq.bin")
(reset)
(run)
(printout t "FAILFREQ.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\costpr.bin")
(reset)
(run)
(printout t "COSTPR.CLP has been executed" crlf)
(clear)
```

```
(bload "d:\\clipsfil\\fsizepr.bin")
(reset)
(run)
(printout t "FSIZEPR.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\equake.bin")
(reset)
(run)
(printout t "EQUAKE.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\adtroadt.bin")
(reset)
(run)
(printout t "ADTROADT.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\dirt.bin")
(reset)
(run)
(printout t "DIRT.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\rock.bin")
(reset)
(run)
(printout t "ROCK.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\econimpo.bin")
(reset)
(run)
(printout t "ECONIMPO.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\permload.bin")
(reset)
(run)
(printout t "PERMLOAD.CLP has been executed" crlf)
(clear)
(bload "d:\\clipsfil\\geohaz.bin")
(reset)
(run)
(printout t "GEOHAZ.CLP has been executed" crlf)
(clear)
```

APPENDIX B

COSTPR.CLP

Repair Cost Priority Rating Program (COSTPR.CLP)

This program determines the importance of the repair cost. The repair cost can be either the actual or an estimated value.

The ranges that were developed were determined from the Slope Inventory provided by WSDOT. This inventory lists approximate costs of repair for each site. By grouping these costs on the basis of failure type, typical cost ranges can be determined for each failure type and then these ranges were combined. This approach to the problem is considered to be adequate method with respect to the uncertainty and the wide range of the costs.

There is no inflation correction included within this program. The cost ranges therefore will remain in 1990 dollars.

It may be better to not rate the costs and use the dollar amount in the decision making because of budgeting considerations.

Figure B.1 Repair Cost Flow Chart

```
;COSTPR.CLP                    Last Revision  10-22-90
;
;            Direct Cost Priorities
;
;*****************************************************************
;        INPUT FILE= DATACOST.TXT
;        OUTPUT FILE= RATCOST.TXT
;*****************************************************************
;*****************************************************************
;                  READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Site ID, Failure Number and the actual or estimated total
; cost. The information is read by the use of a loop.
;    Definition of Variables
;        count: a number from 1 to 3 which tells the program
;            whether to assign the read data to site-id
;            (count= 1), failnum (count= 2), cost (count= 3)
;        read-file: is a flag that notifies the program that a
;            new piece of information can be read.
;        data-read: is the temporary address of the read
;            information until it can be properly identified.
;        cost: the total cost of repairing the failure
;        site-id: records the identification number for the site
;        failnum: indicates how many times the site has failed
;        ?c, ?l, ?read-file, ?data-read, ?close-files: are used
;            to bind the appropriate information to a CLIPS
;            address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "c:\\clipsfil\\datacost.txt" datacost "r")
    (open "c:\\clipsfil\\ratcost.txt" outcost "w")
    (assert (count 1))
    (assert (read-file)))

(defrule read-file                          ;Begins loop
    ?read-file <- (read-file)
    =>
    (retract ?read-file)
    (assert (data-read =(read datacost))))

(defrule read-site-id
```

```
    ?c <- (count 1)
    ?data-read <- (data-read ?site-id&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 2))
    (assert (read-file)))

(defrule read-failnum
    ?c <- (count 2)
    ?data-read <- (data-read ?failnum&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failnum ?failnum))
    (assert (count 3))
    (assert (read-file)))

(defrule read-cost
    ?c <- (count 3)
    ?data-read <- (data-read ?cost&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (cost ?cost)))

(defrule close-all-files                    ;checks if the
    ?close-files <- (data-read EOF)             ;EOF has been
    ?c <- (count ?count)
    =>                                       ;reached, if so
    (retract ?close-files ?c)               ;it closes the file
    (close))

;*****************************************************************
;*****************************************************************
;                  DETERMINATION OF COST
;    This portion of the program assigns a priority rating to the
; cost of repairing the failed site. The cost may, if available, be
; the actual total cost or it may be an estimated value.
;         INPUT DATA
;             cost: dollar value
;
;         OUTPUT DATA
;             cost-pr:
;
;         Definition of Variables
```

```
;              cost: estimated or actual
;              cost-pr: priority rating
;*****************************************************************
(defrule cost-20,000
    ?c <- ( cost ?cost)
    (test (<= ?cost 20000))
    =>
    (assert (cost-pr 1))
    (retract ?c))

(defrule cost-20,000-50,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 20000) (<= ?cost 50000)))
    =>
    (assert (cost-pr 2))
    (retract ?c))

(defrule cost-50,000-100,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 50000) (<= ?cost 100000)))
    =>
    (assert (cost-pr 3))
    (retract ?c))

(defrule cost-100,000-250,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 100000) (<= ?cost 250000)))
    =>
    (assert (cost-pr 4))
    (retract ?c))

(defrule cost-250,000-600,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 250000) (<= ?cost 600000)))
    =>
    (assert (cost-pr 5))
    (retract ?c))

(defrule cost-600,000-1,000,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 600000) (<= ?cost 1000000)))
    =>
    (assert (cost-pr 7))
    (retract ?c))
```

```
(defrule cost-1,000,000-5,000,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 1000000) (<= ?cost 5000000)))
    =>
    (assert (cost-pr 8))
    (retract ?c))

(defrule cost-5,000,000-10,000,000
    ?c <- ( cost ?cost)
    (test (&& (> ?cost 5000000) (<= ?cost 10000000)))
    =>
    (assert (cost-pr 9))
    (retract ?c))

(defrule cost-10,000,000
    ?c <- ( cost ?cost)
    (test (> ?cost 10000000))
    =>
    (assert (cost-pr 10))
    (retract ?c))


;*************************************************************
;*************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
    ?rdno <- (site-id ?site-id)
    ?fn <- (failnum ?failnum)
    ?cpr <- (cost-pr ?cost-pr)
    =>
    (fprintout outcost ?site-id " " ?failnum " " ?cost-pr crlf)
    (retract  ?cpr ?rdno ?fn)
    (assert (count 1))
    (assert (read-file)))
```

APPENDIX C

FAILFREQ.CLP

Failure Frequency *Priority Rating* Program (FAILFREQ.CLP)

In this program the time interval between the failure date and the current date is determined. The time span is used to estimate the frequency at which the site fails; this program does not actually calculate the failure frequency. Within this program all dates are converted into years by use of the equation,

$$date = year + \frac{month-1}{12} + \frac{day}{365} \qquad (1)$$

The system calculates time intervals for each site. However the *priority rating* really only applies to the most recent failure. The *priority ratings* for the earlier failures would not indicate the failure frequency of the slope. This method, however, was deemed acceptable until enough data is available to determine a more accurate method. A program that would be able to identify failure date patterns would be a more accurate approach. However, this method would require several years of data.

Figure C.1 Failure Frequency Flow Chart

```
; FAILFREQ.CLP                    Last Revision 10-22-90
;
;              FAILURE FREQUENCY
;
;*****************************************************************
;       INPUT FILE = DATAFF.TXT
;       OUTPUT FILE = RATFF.TXT

;*****************************************************************
;              READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Site ID, failure number, the current date, and the previous
; failure date. The information is read by the use of a loop.
;   Definition of Variables
;       count: a number from 1 to 8 which tells the program
;           whether to assign the read data to current-month
;           (count= 1), current-day (count= 2), current-year
;           (count= 3), site-id (count= 4), failnum (count= 5),
;           failure-month (count= 6), failure-day (count= 7),
;           failure-year (count= 8)
;       read-file: is a flag that notifies the program that a
;           new piece of information can be read.
;       data-read: is the temporary address of the read
;           information until it can be properly identified.
;       site-id: records the identification number for the site
;       failnum: indicates how many times the site has failed.
;       current-month: number representation of the current month
;       current-day: current day of the month
;       current-year: present year
;       failure-month: month that the last failure on the site
;           took place
;       failure-day: day of the month that the last failure took
;           place on
;       failure-year: year that the last failure took place
;       ?c, ?read-file, ?data-read, ?close-files: are used
;           to bind the appropriate information to a CLIPS
;           address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "c:\\clipsfil\\dataff.txt" dataff "r")
    (open "c:\\clipsfil\\date.txt" date "r")
```

```
    (open "c:\\clipsfil\\ratff.txt" outff "w")
    (assert (count 1))
    (assert (read-file1)))

(defrule read-file                          ;Begins loop
    ?read-file <- (read-file)
    =>
    (retract ?read-file)
    (assert (data-read =(read dataff))))


(defrule read-file-date                     ;Begins loop
    ?read-file1 <- (read-file1)
    =>
    (retract ?read-file1)
    (assert (data-read1 =(read date))))

(defrule read-current-date-month
    ?c <- (count 1)
    ?data-read <- (data-read1 ?current-month&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (current-month ?current-month))
    (assert (count 2))
    (assert (read-file1)))

(defrule read-current-date-day
    ?c <- (count 2)
    ?data-read <- (data-read1 ?current-day&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (current-day ?current-day))
    (assert (count 3))
    (assert (read-file1)))

(defrule read-current-date-year
    ?c <- (count 3)
    ?data-read <- (data-read1 ?current-year&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (current-year ?current-year))
    (assert (count 4))
    (assert (read-file)))
```

```
(defrule read-site-id
    ?c <- (count 4)                              ;checks if it is
    ?data-read <- (data-read ?site-id&~EOF)      ;a site-id line
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 5))
    (assert (read-file)))

(defrule read-failnum
    ?c <- (count 5)                              ;checks if it is
    ?data-read <- (data-read ?failnum&~EOF)      ;a site-id line
    =>
    (retract ?data-read ?c)
    (assert (failnum ?failnum))
    (assert (count 6))
    (assert (read-file)))

(defrule read-previous-failure-date-month
    ?c <- (count 6)
    ?data-read <- (data-read ?failure-month&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failure-month ?failure-month))
    (assert (count 7))
    (assert (read-file)))

(defrule read-previous-failure-date-day
    ?c <- (count 7)
    ?data-read <- (data-read ?failure-day&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failure-day ?failure-day))
    (assert (count 8))
    (assert (read-file)))

(defrule read-previous-failure-date-year
    ?c <- (count 8)
    ?data-read <- (data-read ?failure-year&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failure-year ?failure-year)))

(defrule close-all-files                         ;checks if the
```

```
    ?close-files <- (data-read EOF)        ;EOF has been
    ?cd <- (current-date ?current-date)
    ?c <- (count ?count)
    =>                                     ;reached, if so
    (retract ?close-files ?c ?cd)          ;it closes the file
    (close))

;*****************************************************************
;*****************************************************************
;   DETERMINE SPAN BETWEEN the PREVIOUS FAILURE and the PRESENT
;   This portion of the program calculates the current and failure
; date in terms of years. It then proceeds to  determine the time
; span between the last time the slope failed  and the present
; date.
;       INPUT DATA
;           current & failure-month: number from 1 to 12
;           current & failure-day: number from 1 to 31
;           current & failure-year: 19??
;
;       OUTPUT DATA
;           gap:
;
;       Definition of Variables
;           current-month, day, year: present date
;           failure-month, day, year: date of the last time the
;               site failed
;           gap: the time span in years between the present
;               date and the most recent failure date
;
;*****************************************************************
(defrule determine-current-date
    ?cm <- (current-month ?current-month)
    ?cd <- (current-day ?current-day)
    ?cyr <- (current-year ?current-year)
    =>
    (assert (current-date
        =(+ ?current-year (/ (- ?current-month 1) 12)
        (/ ?current-day 365))))
    (retract ?cm ?cd ?cyr))

(defrule determine-failure-date
    ?fm <- (failure-month ?failure-month)
    ?fd <- (failure-day ?failure-day)
    ?fyr <- (failure-year ?failure-year)
```

```
    =>
    (assert (fail-date
        =(+ ?failure-year (/ (- ?failure-month 1) 12)
        (/ ?failure-day 365))))
    (retract ?fm ?fd ?fyr))

(defrule determine-gap
    ?fd <- (fail-date ?fail-date)
    ?cd <- (current-date ?current-date)
    =>
    (assert (gap =(- ?current-date ?fail-date)))
    (retract ?fd))

;*****************************************************************
;*****************************************************************
;            DETERMINE FAILURE FREQUENCY
;   This portion of the program prioritizes the failure frequency
; of the site. Failure frequency is only based on the present date
; and the date of the most recent failure of the site, not on the
; average time span between failures. This method was deemed
; acceptable because there will be very little data available and
; for the first few years of the data collection this method should
; be reasonable. This method will indicate frequent failing sites
; and sparsely failing sites. When more data is available it may
; be beneficial to create programs that are able to search for
; specific patterns in failure times which could then be used to
; predict approximate future failure dates.
;       INPUT DATA
;           gap: internally determined, years
;
;       OUTPUT DATA
;           failure-frequency-pr
;
;*****************************************************************
(defrule ff->=10yr
    ?g <- (gap ?gap)
    (test (>= ?gap 10))
    =>
    (assert (failure-frequency-pr 1))
    (retract ?g))

(defrule ff-10yr>gap>=5yr
    ?g <- (gap ?gap)
    (test (&& (< ?gap 10) (>= ?gap 5)))
```

```
    =>
    (assert (failure-frequency-pr 3))
    (retract ?g))

(defrule ff-5yr>gap>=2yr
    ?g <- (gap ?gap)
    (test (&& (< ?gap 5) (>= ?gap 2)))
    =>
    (assert (failure-frequency-pr 6))
    (retract ?g))

(defrule ff-2yr>gap>=1yr
    ?g <- (gap ?gap)
    (test (&& (< ?gap 2) (>= ?gap 1)))
    =>
    (assert (failure-frequency-pr 9))
    (retract ?g))

(defrule ff-1yr>gap
    ?g <- (gap ?gap)
    (test (< ?gap 1))
    =>
    (assert (failure-frequency-pr 10))
    (retract ?g))

;*****************************************************************
;*****************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
    ?rdno <- (site-id ?site-id)
    ?fn <- (failnum ?failnum)
    ?ffpr <- (failure-frequency-pr ?failure-frequency-pr)
    =>
    (fprintout outff ?site-id " " ?failnum " "
        ?failure-frequency-pr crlf)
    (retract ?ffpr ?rdno ?fn)
    (assert (count 4))
    (assert (read-file)))
```

APPENDIX D

FSIZEPR.CLP

Failure Conditions *Priority Rating* Program (FSIZEPR.CLP)

This program accounts for the importance of the failure size, water level, and date. The failure size is the volume in cubic yards of the failure. The failure water level indicates the water content in the slope with respect to the average water content for the slope at that particular time of year. The failure date is the date that the site failed.

This program considers past failures; however it could, if the information were available, be applied to future failures.

The numeric ranges for the failure size within the program were determined quite arbitrarily because there was no data available. Therefore these ranges should be inspected and adjusted where it is necessary.

The water level within the slope should be described numerically. Table 7 shows an explanation of the numeric values. Note all comments are with respect to the average water content for that period of the year. The ranges within the program were determined with the idea that future versions of the program would be able to use the average water level of similar past failures as data for future failures. Therefore the future numbers may not be integers.

The failure date is the month and day of the date in terms of years, so that the season can be determined. The failure date is calculated, within the database, by the equation

$$failure\ date = \frac{failuremonth - 1}{12} + \frac{faiureday}{365} \qquad (2)$$

Table 7 Water Level Descriptions and Values

Description	Value
Extremely High	10
High	7
Normal	5
Low	3
Extremely Low	1

The seasons within this program were determined with the mountains in mind. However another definition of the seasons may be more practical or the use of different definitions of the seasons for different geographical regions may be better.

Figure D.1 Failure Size Flow Chart

Figure D.2 Failure Water Level Flow Chart

Figure D.3 Failure Date Flow Chart

```
; FSIZEPR.CLP                    Last Revision 10-22-90
;
;        FAILURE SIZE, WATER LEVEL, & DATE PRIORITIES
;
;*****************************************************************
;        INPUT FILE = DATAFWD.TXT
;        OUTPUT FILE = RATFWD.TXT

;*****************************************************************
;                READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Site ID, failure Number, failure size (cubic yards), water
; level within the slope, failure date. The information is read by
; the use of a loop.
;   Definition of Variables
;        count: a number from 1 to 5 which tells the program
;            whether to assign the read data to site-id
;            (count= 1), failnum (count= 2), fail-size
;            (count= 3), water-level (count= 4), fail-date
;            (count= 5)
;        read-file: is a flag that notifies the program that a
;            new piece of information can be read.
;        data-read: is the temporary address of the read
;            information until it can be properly identified.
;        site-id: records the identification number for the site
;        failnum: indicates how many times the site has failed
;        failsize: cubic yards
;        water-level: number 1 to 10
;        fail-date: month + day in terms of years
;        ?c, ?read-file, ?data-read, ?close-files: are used
;            to bind the appropriate information to a CLIPS
;            address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "c:\\clipsfil\\datafwd.txt" datafwd "r")
    (open "c:\\clipsfil\\ratfwd.txt" outfwd "w")
    (assert (count 1))
    (assert (read-file)))

(defrule read-file                    ;Begins loop
    ?read-file <- (read-file)
```

```
    =>
    (retract ?read-file)
    (assert (data-read =(read datafwd))))

(defrule read-site-id
    ?c <- (count 1)
    ?data-read <- (data-read ?site-id&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 2))
    (assert (read-file)))

(defrule read-failnum
    ?c <- (count 2)
    ?data-read <- (data-read ?failnum&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failnum ?failnum))
    (assert (count 3))
    (assert (read-file)))

(defrule read-fail-size
    ?c <- (count 3)
    ?data-read <- (data-read ?fail-size&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (fail-size ?fail-size))
    (assert (count 4))
    (assert (read-file)))

(defrule read-water-level
    ?c <- (count 4)
    ?data-read <- (data-read ?water-level&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (water-level ?water-level))
    (assert (count 5))
    (assert (read-file)))

(defrule read-fail-date
    ?c <- (count 5)
    ?data-read <- (data-read ?fail-date&~EOF)
    =>
```

```
    (retract ?data-read ?c)
    (assert (fail-date ?fail-date)))

(defrule close-all-files                    ;checks if the
    ?close-files <- (data-read EOF)         ;EOF has been
    ?c <- (count ?count)
    =>                                      ;reached, if so
    (retract ?close-files ?c)               ;it closes the file
    (close))

;*****************************************************************
;*****************************************************************
;            FAILURE SIZE
;   This portion of the program assigns a priority rating to the
; past failure size.
;        INPUT DATA
;            fail-size: cubic yards
;
;        OUTPUT DATA
;            fail-size-pr:
;
;        Definition of Variables
;            fail-size: future failure size that is in terms of
;                cubic yards
;
;*****************************************************************
(defrule fail-size-50,000
    ?fs <- (fail-size ?fail-size)
    (test (>= ?fail-size 5.00000e+4))
    =>
    (assert (fail-size-pr 10))
    (retract ?fs))

(defrule fail-size-50,000-40,000
    ?fs <- (fail-size ?fail-size)
    (test (&& (< ?fail-size 5.00000e+4)
              (>= ?fail-size 4.00000e+4)))
    =>
    (assert (fail-size-pr 9))
    (retract ?fs))

(defrule fail-size-40,000-30,000
    ?fs <- (fail-size ?fail-size)
    (test (&& (< ?fail-size 4.00000e+4)
```

```
            (>= ?fail-size 3.00000e+4)))
    =>
    (assert (fail-size-pr 7))
    (retract ?fs))

(defrule fail-size-30,000-20,000
    ?fs <- (fail-size ?fail-size)
    (test (&& (< ?fail-size 3.00000e+4)
            (>= ?fail-size 2.00000e+4)))
    =>
    (assert (fail-size-pr 6))
    (retract ?fs))

(defrule fail-size-20,000-10,000
    ?fs <- (fail-size ?fail-size)
    (test (&& (< ?fail-size 2.00000e+4)
            (>= ?fail-size 1.00000e+4)))
    =>
    (assert (fail-size-pr 5))
    (retract ?fs))

(defrule fail-size-10,000-5,000
    ?fs <- (fail-size ?fail-size)
    (test (&& (< ?fail-size 1.00000e+4)
            (>= ?fail-size 5.00000e+3)))
    =>
    (assert (fail-size-pr 3))
    (retract ?fs))

(defrule fail-size-5,000
    ?fs <- (fail-size ?fail-size)
    (test (< ?fail-size 5.00000e+3))
    =>
    (assert (fail-size-pr 2))
    (retract ?fs))

;*****************************************************************
;*****************************************************************
;            WATER LEVEL
;   This portion of the program assigns a priority rating to the
; past failure water levels.
;        INPUT DATA
;            water-level: number from 1 to 10
;
```

```
;        OUTPUT DATA
;            water-level-pr
;
;        Definition of Variables
;            water-level: severity of water level at the time of
;                failure. The original descriptions of the water
;                content of the site were made with respect to
;                the average water content at that particular
;                time of year. The original ratings are listed
;                below.
;                    Extremely high = 10
;                    High = 7
;                    Normal = 5
;                    Low = 3
;                    Extrememly Low = 1
;
;*****************************************************************
(defrule water-level-10
    ?wl <- (water-level ?water-level)
    (test (>= ?water-level 9.500000))
    =>
    (assert (water-level-pr 10))
    (retract ?wl))

(defrule water-level-9
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 9.500000)
            (>= ?water-level 8.500000)))
    =>
    (assert (water-level-pr 9))
    (retract ?wl))

(defrule water-level-8
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 8.500000)
            (>= ?water-level 7.500000)))
    =>
    (assert (water-level-pr 8))
    (retract ?wl))

(defrule water-level-7
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 7.500000)
            (>= ?water-level 6.500000)))
```

```
    =>
    (assert (water-level-pr 7))
    (retract ?wl))

(defrule water-level-6
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 6.500000)
              (>= ?water-level 5.500000)))
    =>
    (assert (water-level-pr 6))
    (retract ?wl))

(defrule water-level-5
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 5.500000)
              (>= ?water-level 4.500000)))
    =>
    (assert (water-level-pr 5))
    (retract ?wl))

(defrule water-level-4
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 4.500000)
              (>= ?water-level 3.500000)))
    =>
    (assert (water-level-pr 4))
    (retract ?wl))

(defrule water-level-3
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 3.500000)
              (>= ?water-level 2.500000)))
    =>
    (assert (water-level-pr 3))
    (retract ?wl))

(defrule water-level-2
    ?wl <- (water-level ?water-level)
    (test (&& (< ?water-level 2.500000)
              (>= ?water-level 1.500000)))
    =>
    (assert (water-level-pr 2))
    (retract ?wl))
```

```
(defrule water-level-1
    ?wl <- (water-level ?water-level)
    (test (< ?water-level 1.500000))
    =>
    (assert (water-level-pr 1))
    (retract ?wl))

;*****************************************************************
;*****************************************************************
;            FAILURE DATE
;   This section assigns a priority rating to the date past
; failure date.
;
;       INPUT DATA
;           fail-date: years, numbers < = 1.0000
;
;       OUTPUT DATA
;           fail-date-pr
;
;       Definition of Variables
;           fail-date: failure month and day in terms of years
;               summer: June 1 to Aug 31
;               fall: Sept 1 to Oct 31
;               winter: Nov 1 to Mar 31
;               spring: April 1 to May 31
;  *** NOTE: the general range for the priorities were taken from
; the questionnaire. The season range was developed to correspond
; to the seasons in the mountains.
;*****************************************************************

(defrule fail-date-.417-.668-summer
    ?fd <- (fail-date ?fail-date)
    (test (&& (> = ?fail-date 4.17000e-1)
            (< ?fail-date 6.68000e-1)))
    =>
    (assert (fail-date-pr 5))
    (retract ?fd))

(defrule fail-date-.668-.835-fall
    ?fd <- (fail-date ?fail-date)
    (test (&& (> = ?fail-date 6.68000e-1)
            (< ?fail-date 8.35000e-1)))
    =>
    (assert (fail-date-pr 7))
```

```
    (retract ?fd))

(defrule fail-date-.835-.252-winter
    ?fd <- (fail-date ?fail-date)
    (test (|| (&& (>= ?fail-date 8.35000e-1)
                  (<= ?fail-date 1.00000))
              (&& (<= ?fail-date 2.52000e-1)
                  (> ?fail-date 0))))
    =>
    (assert (fail-date-pr 10))
    (retract ?fd))

(defrule fail-date-.252-.417-spring
    ?fd <- (fail-date ?fail-date)
    (test (&& (> ?fail-date 2.52000e-1)
              (< ?fail-date 4.17000e-1)))
    =>
    (assert (fail-date-pr 7))
    (retract ?fd))

;*****************************************************************
;*****************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
    ?rdno <- (site-id ?site-id)
    ?fn <- (failnum ?failnum)
    ?fspr <- (fail-size-pr ?fail-size-pr)
    ?wlpr <- (water-level-pr ?water-level-pr)
    ?fdpr <- (fail-date-pr ?fail-date-pr)
    =>
    (fprintout outfwd ?site-id " " ?failnum " " ?fail-size-pr " "
        ?water-level-pr " " ?fail-date-pr crlf)
    (retract ?fspr ?rdno ?wlpr ?fdpr ?fn)
    (assert (count 1))
    (assert (read-file)))
```

APPENDIX E

PUBRISK.CLP

Public Risk Potential *Priority Rating* Program (PUBRISK.CLP)

This program was developed in order to try to account for the possibility of a catastophic public risk resulting from the effects of the failure. The valid responses to the lawsuit query are very limited as they exist now. In future versions a more varied response list would be more accurate. The valid responses are:

1. **personal injury or property damage-high**: indicates that there has been a personal injury or property damage for a past failure at this site or that there is a potential personal injury or property damage for a future failure at this site.
2. **personal injury or property damage-low**: indicates that there is a low possibility for personal injuries or property damage.
3. **toxins**: indicates that there is a high probability of the release of toxins to the environment as a result of a failure.
4. **property**: indicates that a failure will or could result in damage to an economically vital structure or to an essential public facility.
5. **NONE**: indicates that there is no likelihood of public risk.

These definitions were developed by a lay person and should be revised.

Figure E.1 Lawsuit Potential Flow Chart

```
; PUBRISK.CLP                    Last Revision 11-5-91
;
;              PUBLIC RISK POTENTIAL
;
;*****************************************************************
;       INPUT FILE = DATAPUB.TXT
;       OUTPUT FILE = RATPUB.TXT

;*****************************************************************
;              READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Road ID and Damage Type information. The information is
; read by the use of a loop.
;   Definition of Variables
;       count: a number from 1 to 3 which tells the program
;           whether to assign the read data to site-id
;           (count= 1), failnum (count= 2), damage-type
;           (count= 3)
;       read-file: is a flag that notifies the program that a
;           new piece of information can be read.
;       data-read: is the temporary address of the read
;           information until it can be properly identified.
;       site-id: records the identification number for the site
;       failnum: indicates how many times the site has failed.
;       damage-type: damage that may create public risk.
;       pubrisk-pr: priority rating
;       ?c, ?read-file, ?data-read, ?close-files: are used
;           to bind the appropriate information to a CLIPS
;           address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "d:\\clipsfil\\datapub.txt" datapub "r")
    (open "d:\\clipsfil\\ratpub.txt" outpub "w")
    (assert (count 1))
    (assert (read-file)))

(defrule read-file                          ;Begins loop
    ?read-file <- (read-file)
    =>
    (retract ?read-file)
    (assert (data-read =(read datapub))))
```

```
(defrule read-site-id
    ?c <- (count 1)
    ?data-read <- (data-read ?site-id&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 2))
    (assert (read-file)))


(defrule read-failnum
    ?c <- (count 2)
    ?data-read <- (data-read ?failnum&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failnum ?failnum))
    (assert (count 3))
    (assert (read-file)))

(defrule read-damage-type
    ?c <- (count 3)
    ?data-read <- (data-read ?damage-type&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (damage-type ?damage-type)))

(defrule close-all-files                    ;checks if the
    ?close-files <- (data-read EOF)         ;EOF has been
    ?c <- (count ?count)
    =>                                      ;reached, if so
    (retract ?close-files ?c)               ;it closes the file
    (close))

;*****************************************************************
;*****************************************************************
;            DETERMINATION OF PUBLIC RISK POTENTIAL
;   This portion of the program assigns suitable priorities to
; different types of problems that may create public risk.
;        INPUT DATA
;            damage-type: High-personal, Low-personal,
;                Toxins, Property, None
;        OUTPUT DATA
;            pubrisk-pr
;
```

```
;        Definition of Variables
;            damage-type: type of damage that may create public risk.
;                High-Personal: high possibility of public risk.
;                Low-Personal: low possibility of public risk.
;                Toxins: contains Toxins on or near the site
;                    that could be released during a failure.
;                Property: large, expensive, and vital public
;                    property (i.e. water treatment plant).
;            pubrisk-pr: priority rating.
;*****************************************************************
(defrule damage_None
    ?dt <- (damage-type ?damage-type)
    (test (eq ?damage-type None))
    =>
    (assert (pubrisk-pr 0))
    (retract ?dt))

(defrule damage_High-personal
    ?dt <- (damage-type ?damage-type)
    (test (eq ?damage-type High-personal))
    =>
    (assert (pubrisk-pr 10))
    (retract ?dt))

(defrule damage_Low-personal
    ?dt <- (damage-type ?damage-type)
    (test (eq ?damage-type Low-personal))
    =>
    (assert (pubrisk-pr 7))
    (retract ?dt))

(defrule damage_toxins
    ?dt <- (damage-type ?damage-type)
    (test (eq ?damage-type Toxins))
    =>
    (assert (pubrisk-pr 10))
    (retract ?dt))

(defrule damage_property
    ?dt <- (damage-type ?damage-type)
    (test (eq ?damage-type Property))
    =>
    (assert (pubrisk-pr 10))
    (retract ?dt))
```

```
;!! decision has been made
;*****************************************************************
;*****************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
    ?rdno <- (site-id ?site-id)
    ?fn <- (failnum ?failnum)
    ?pubpr <- (pubrisk-pr ?pubrisk-pr)
    =>
    (fprintout outpub ?site-id " "  ?failnum " " ?pubrisk-pr
                    crlf)
    (retract ?pubpr ?rdno ?fn)
    (assert (count 1))
    (assert (read-file)))
```

APPENDIX F

PAVEDAM.CLP

Pavement Damage *Priority Rating* Program
(PAVEDAM.CLP)

This program accounts for the severity of the damage to the road surface due to a failure type of erosion, settlement, or wave-action. Only these failure types were considered because of there speed of progression. It is assumed that these failure types occur slowly enough to allow for noticeable degradation of the road surface. This degradation must occur primarily from the failure process and not normal wear. The other failure types are considered to happen too rapidly and therefore all repairs would need to be completed at one time.

This program requires two input *factors*, problem type and pavement damage. Refer to Appendix G for the proper responses to the problem type. The valid responses for the pavement damage are:

1. **severe**: indicates that the pavement deterioration is potentially hazardous to motorists and that immediate repair is necessary.
2. **moderate**: indicates that the deterioration of the pavement causes motorists to slow down, but the damage itself is not hazardous. This indicates that repairs will have to be made soon.
3. **low**: indicates that the pavement deterioration is noticeable to drivers but is not dangerous at legal speeds, just an irritant.
4. **NONE**: indicates that there is no damage to the road.

In future versions of the program additional and more specific responses would be helpful.

Figure F.1 Pavement Damage Flow Chart

```
; PAVEDAM.CLP                    Last Revision 10-22-90
;
;                PAVEMENT DAMAGE
;
;*****************************************************************
;        INPUT FILE = DATAPAVE.TXT
;        OUTPUT FILE = RATPAVE.TXT

;*****************************************************************
;                READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Site ID, Failure Number, Problem Type, and Pavement Damage
; information. The information is read by the use of a loop.
;    Definition of Variables
;        count: a number from 1 to 4 which tells the program
;            whether to assign the read data to site-id
;            (count= 1), failnum (count= 2), problem-type
;            (count= 3), or pavement-damage (count=4)
;        read-file: is a flag that notifies the program that a
;            new piece of information can be read.
;        data-read: is the temporary address of the read
;            information until it can be properly identified.
;        problem-type: type of failure
;        site-id: records the identification number for the site
;        failnum: indicates how many times the site has failed
;        pavement-damge: indicates if the pavement has been
;            damaged due to the failure
;        ?c, ?read-file, ?data-read, ?close-files: are used
;            to bind the appropriate information to a CLIPS
;            address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "c:\\clipsfil\\datapave.txt" datapave "r")
    (open "c:\\clipsfil\\ratpave.txt" outpave "w")
    (assert (count 1))
    (assert (read-file)))

(defrule read-file                    ;Begins loop
    ?read-file <- (read-file)
    =>
    (retract ?read-file)
```

```
    (assert (data-read =(read datapave))))     ;reads the data

(defrule read-site-id
    ?c <- (count 1)
    ?data-read <- (data-read ?site-id&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 2))
    (assert (read-file)))

(defrule read-failnum
    ?c <- (count 2)
    ?data-read <- (data-read ?failnum&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (failnum ?failnum))
    (assert (count 3))
    (assert (read-file)))

(defrule read-problem-type
    ?c <- (count 3)
    ?data-read <- (data-read ?problem-type&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (problem-type ?problem-type))
    (assert (count 4))
    (assert (read-file)))

(defrule read-pavement-damage
    ?c <- (count 4)
    ?data-read <- (data-read ?pavement-damage&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (pavement-damage ?pavement-damage)))

(defrule close-all-files                    ;checks if the
    ?close-files <- (data-read EOF)         ;EOF has been
    ?c <- (count ?count)
    =>                                      ;reached, if so
    (retract ?close-files ?c)               ;it closes the file
    (close))

;*************************************************************
```

```
;*****************************************************************
;        DETERMINATION OF PAVEMENT DAMAGE SEVERITY
;    This portion of the program determines if the failure has
; damaged the pavement and determines the extent of the damage.
; Pavement damage as a direct result of the failure is considered
; to only occur for the problem types of Erosion, Settlement and
; Wave-Action.
;        INPUT DATA
;            problem-type: Rockfall, Fast-Landslide, Erosion,
;                Fast-Debris-Flow, Slow-Landslide, Settlement,
;                Slow-Debris-Flow, Wave-Action, Piping
;            pavement-damage: severe, moderate, low, NONE
;        OUTPUT DATA
;            pavement-damage-pr
;
;        Definition of Variables
;            problem-type: type of failure problem
;            pavement-damage: indicates the extent of the damage
;                severe: dangerous (holes, drop offs,...)
;                moderate: causes drivers to slow down
;                low: noticeable to drivers but not a danger
;            pavement-damage-pr: pavement damage priority rating
;*****************************************************************
(defrule severe-pavement-damage
    ?pt <- (problem-type ?problem-type)
    ?pv <- (pavement-damage ?pavement-damage)
    (test (eq ?pavement-damage severe))
    =>
    (if (||(eq ?problem-type Erosion)
            (eq ?problem-type Wave-Action)
            (eq ?problem-type Settlement))
        then (assert (pavement-damage-pr 10))
        else (assert (pavement-damage-pr 0)))
    (retract ?pt ?pv))

(defrule moderate-pavement-damage
    ?pt <- (problem-type ?problem-type)
    ?pv <- (pavement-damage ?pavement-damage)
    (test (eq ?pavement-damage moderate))
    =>
    (if (||(eq ?problem-type Erosion)
            (eq ?problem-type Wave-Action)
            (eq ?problem-type Settlement))
        then (assert (pavement-damage-pr 7))
```

```
        else (assert (pavement-damage-pr 0)))
    (retract ?pt ?pv))

(defrule low-pavement-damage
    ?pt <- (problem-type ?problem-type)
    ?pv <- (pavement-damage ?pavement-damage)
    (test (eq ?pavement-damage low))
    =>
    (if (||(eq ?problem-type Erosion)
            (eq ?problem-type Wave-Action)
            (eq ?problem-type Settlement))
        then (assert (pavement-damage-pr 3))
        else (assert (pavement-damage-pr 0)))
    (retract ?pt ?pv))

(defrule no-pavement-damage
    ?pt <- (problem-type ?problem-type)
    ?pv <- (pavement-damage ?pavement-damage)
    (test (eq ?pavement-damage NONE))
    =>
    (assert (pavement-damage-pr 0))
    (retract ?pt ?pv))

;!! decision has been made
;*****************************************************************
;*****************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
    ?rdno <- (site-id ?site-id)
    ?fn <- (failnum ?failnum)
    ?pdpr <- (pavement-damage-pr ?pavement-damage-pr)
    =>
    (fprintout outpave ?site-id " " ?failnum " "
        ?pavement-damage-pr crlf)
    (retract ?pdpr ?rdno ?fn)
    (assert (count 1))
    (assert (read-file)))
```

```
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load NONE))
    =>
    (assert (perm-dynamic-pr 0))
    (retract ?dl ?dll ?dlt))

(defrule dynamic-load-on-site-new
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load dynamic))
    (test (eq ?dynamic-load-location on-site))
    (test (eq ?dynamic-load-time NEW))
    =>
    (assert (perm-dynamic-pr 9))
    (retract ?dl ?dll ?dlt))

(defrule dynamic-load-on-near-site-new
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load dynamic))
    (test (eq ?dynamic-load-location on-near-site))
    (test (eq ?dynamic-load-time NEW))
    =>
    (assert (perm-dynamic-pr 10))
    (retract ?dl ?dll ?dlt))

(defrule dynamic-load-near-site-new
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load dynamic))
    (test (eq ?dynamic-load-location near-site))
    (test (eq ?dynamic-load-time NEW))
    =>
    (assert (perm-dynamic-pr 8))
    (retract ?dl ?dll ?dlt))

(defrule dynamic-load-on-site-old
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
```

```
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load dynamic))
    (test (eq ?dynamic-load-location on-site))
    (test (eq ?dynamic-load-time OLD))
    =>
    (assert (perm-dynamic-pr 6))
    (retract ?dl ?dll ?dlt))

(defrule dynamic-load-on-near-site-old
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load dynamic))
    (test (eq ?dynamic-load-location on-near-site))
    (test (eq ?dynamic-load-time OLD))
    =>
    (assert (perm-dynamic-pr 7))
    (retract ?dl ?dll ?dlt))

(defrule dynamic-load-near-site-old
    ?dl <- (dynamic-load ?dynamic-load)
    ?dll <- (dynamic-load-location ?dynamic-load-location)
    ?dlt <- (dynamic-load-time ?dynamic-load-time)
    (test (eq ?dynamic-load dynamic))
    (test (eq ?dynamic-load-location near-site))
    (test (eq ?dynamic-load-time OLD))
    =>
    (assert (perm-dynamic-pr 5))
    (retract ?dl ?dll ?dlt))

;!! decision has been made

;*****************************************************************
;*****************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
    ?rdno <- (site-id ?site-id)
    ?slpr <- (perm-static-pr ?perm-static-pr)
    ?dlpr <- (perm-dynamic-pr ?perm-dynamic-pr)
    =>
    (fprintout outprml ?site-id " "  ?perm-static-pr " "
```

```
    ?perm-dynamic-pr crlf)
(retract ?slpr ?dlpr ?rdno)
(assert (count 1))
(assert (read-file)))
```

APPENDIX P

ROCK.CLP

Rock Classification *Priority Rating* Program (ROCK.CLP)

This program determines the priority ratings for the presence, orientation, and structure type of the rock that composes the failure site. This program determines three priority ratings: joint-layer, loose, and intact priority ratings.

The first portion of the program considers the jointing and layering of the rock structure. If the rock is composed of different types of rock, the rock types are assumed to have approximately the same compressive strength. The type of jointing that can be specified are:

1. **NONE**: the rock is not jointed or rock is not present on the site.
2. **horizontal**: the joints within the rock are horizontal.
3. **down-slope-dip**: the joints slope towards the road and dip in the same general direction as the slope.
4. **cross-slope-dip**: the joints slope in the opposite direction that the slope dips and away from the road.
5. **vertical**: the joints within the rock are vertical.
6. **down-slope-block**: there are two jointing systems, down-slope-dip and cross-slope-dip, which result in the formation of blocks that dip towards the road.
7. **horizontal-block**: there are two jointing systems, vertical and horizontal, which result in the formation of blocks within the rock.

The types of layering that can be specified are:

1. **NONE**: the rock is not jointed or rock is not present on the site.
2. **horizontal**: the joints within the rock are horizontal.
3. **down-slope-dip**: the joints slope towards the road and dip in the same general direction as the slope.
4. **cross-slope-dip**: the joints slope in the opposite direction that the slope dips and away from the rock

The second portion of the program determines what type of loose rock is present on the site. The proper responses to the variable, loose, are:

1. **NONE**: there is no loose rock present.
2. **gravel**: there is an appreciable quantity of loose gravel present on the site.
3. **boulders**: there are boulders present on the site.

The final portion of the program deals with whether or not the rock is homogeneous, one rock type, and intact, absence of discontinuities. The proper responses to the intact variable are:

1. **NONE**: the rock is not homogeneous and/or intact.
2. **Yes**: the rock is homogeneous and intact.

To improve this program, it could be useful to account for the different compressive strengths of the different types of rock that may compose the site. Also the

priority ratings within this program will probably need to be adjusted.

Figure P.1 Rock Flow Chart

Figure P.2 Rock Jointing and Layering (1) Flow Chart

Figure P.3 Rock Jointing and Layering (2) Flow Chart

Figure P.4 Rock Jointing and Layering (3) Flow Chart

```
; ROCK.CLP                    Last Revision  11-12-90
;
;                GEOLOGY: ROCK
;
;*****************************************************************
;        INPUT FILE = DATAROCK.TXT
;        OUTPUT FILE = RATROCK.TXT

;*****************************************************************
;                READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Site ID, Presence of loose Rock, Presence of homogeneous,
; intact or jointed rock, and the layering of the rock. The
; information is read by the use of a loop.
;    Definition of Variables
;        count: a number from 1 to 5 which tells the program
;            whether to assign the read data to site-id
;            (count= 1), loose (count= 2), homogeneous-intact
;            (count=3), joints (count= 4), layers (count= 5)
;        loop: is a counter that keeps track of how many priority
;            ratings have been determined.  When the appropriate
;            number has been determined, information for a new
;            site is read and the loop begins again.
;        read-file: is a flag that notifies the program that a
;            new piece of information can be read.
;        data-read: is the temporary address of the read
;            information until it can be properly identified.
;        site-id: records the identification number for the site
;        loose: indicates the presence of loose rock on the site.
;        homogeneous-intact: indicates if the rock is solid and
;            consists of one type of rock.
;        joints: indicates the type of jointing in the rock.
;        layers: indicates the presence of layers of rock and
;            their orientation.
;        ?c, ?l, ?read-file, ?data-read, ?close-files: are used
;            to bind the appropriate information to a CLIPS
;            address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "c:\\clipsfil\\datarock.txt" datar "r")
    (open "c:\\clipsfil\\ratrock.txt" outr "w")
```

```
    (assert (count 1))
    (assert (loop 0))
    (assert (read-file)))

(defrule read-file                          ;Begins loop
    (loop 0)
    ?read-file <- (read-file)
    =>
    (retract ?read-file)
    (assert (data-read =(read datar))))          ;reads the data

(defrule read-site-id
    ?c <- (count 1)
    ?data-read <- (data-read ?site-id&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 2))
    (assert (read-file)))

(defrule read-loose "Presence_of_loose_rock"
    ?c <- (count 2)
    ?data-read <- (data-read ?loose&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (loose ?loose))
    (assert (count 3))
    (assert (read-file)))

(defrule read-homogeneous-intact
    ?c <- (count 3)
    ?data-read <- (data-read ?homogeneous-intact&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (homogeneous-intact ?homogeneous-intact))
    (assert (count 4))
    (assert (read-file)))

(defrule read-joints
    ?c <- (count 4)
    ?data-read <- (data-read ?joints&~EOF)
    =>
    (retract ?data-read ?c)
    (assert (joints ?joints))
```

```
    (assert (count 5))
    (assert (read-file)))

(defrule read-layers
    ?c <- (count 5)
    ?data-read <- (data-read ?layers& ~EOF)
    =>
    (retract ?data-read ?c)
    (assert (layers ?layers)))

(defrule close-all-files                    ;checks if the
    ?close-files <- (data-read EOF)             ;EOF has been
    ?c <- (count ?count)
    ?l <- (loop ?loop)
    =>                                       ;reached, if so
    (retract ?close-files ?c ?l)            ;it closes the file
    (close))


;********************* ROCK LAYERING & JOINTING **********************
;    This portion of the program deals with the different
; combinations of rock layering and jointing. This program only
; deals with layers of rock which have approximately the same
; compressive strength.
;    INPUT DATA
;        layers: NONE, horizontal, down-slope-dip ( \\\\|_ ),
;            cross-slope-dip ( ////|_ )
;        loops
;        joints: NONE, horizontal, down-slope-dip ( \\\\|_ ),
;            cross-slope-dip ( ////|_ ), vertical,
;                    down-slope-block, horizontal-block
;
;    OUTPUT DATA
;        joint-layer-pr: priority rating for the particular
;            layering and jointing present in the rock
;
;    Definition of Variables
;        layers: indicates whether or not the rock is layered and
;            specifies in which direction the layers slope. The
;            layers can slope towards the road (down-slope-dip),
;            away from the road (cross-slope-dip) or not slope at
;            all (horizontal)
;        joints: indicates whether or not the rock is jointed and
;            specifies what kind of jointing is present
```

```
;          ?j, ?lay, ?l: retraction addresses
;
;*****************************************************************
; The section below considers unlayered & unjointed sites
(defrule no-layering-no-joints
     ?lay <- (layers ?layers)
     ?j <- (joints ?joints)
     ?l <- (loop ?loop)
     (test (eq ?layers NONE))
     (test (eq ?joints NONE))
     =>
     (assert (joint-layer-pr 0))
     (retract ?l ?lay ?j)
     (assert (loop =(+ ?loop 1))))

; This section considers only layered rock, no joints
(defrule down-slope-dip-layers
     ?lay <- (layers ?layers)
     ?l <- (loop ?loop)
     ?j <- (joints ?joints)
     (test (eq ?layers down-slope-dip))
     (test (eq ?joints NONE))
     =>
     (assert (joint-layer-pr 9))
     (retract ?l ?lay ?j)
     (assert (loop =(+ ?loop 1))))

(defrule cross-slope-dip-layers
     ?lay <- (layers ?layers)
     ?l <- (loop ?loop)
     ?j <- (joints ?joints)
     (test (eq ?layers cross-slope-dip))
     (test (eq ?joints NONE))
     =>
     (assert (joint-layer-pr 1))
     (retract ?l ?lay ?j)
     (assert (loop =(+ ?loop 1))))

(defrule horizontal-layers
     ?lay <- (layers ?layers)
     ?l <- (loop ?loop)
     ?j <- (joints ?joints)
     (test (eq ?layers horizontal))
     (test (eq ?joints NONE))
```

```
    =>
    (assert (joint-layer-pr 2))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))


; The section below considers layered rock & horizontal joints
(defrule no-layering-horizontal-joints
    ?lay <- (layers ?layers)
    ?j <- (joints ?joints)
    ?l <- (loop ?loop)
    (test (eq ?layers NONE))
    (test (eq ?joints horizontal))
    =>
    (assert (joint-layer-pr 5))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

(defrule down-slope-dip-layers-horizontal-joints
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
    ?j <- (joints ?joints)
    (test (eq ?layers down-slope-dip))
    (test (eq ?joints horizontal))
    =>
    (assert (joint-layer-pr 10))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

(defrule cross-slope-dip-layers-horizontal-joints
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
    ?j <- (joints ?joints)
    (test (eq ?layers cross-slope-dip))
    (test (eq ?joints horizontal))
    =>
    (assert (joint-layer-pr 8))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

(defrule horizontal-layers-horizontal-joints
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
    ?j <- (joints ?joints)
```

```
    (test (eq ?layers horizontal))
    (test (eq ?joints horizontal))
    =>
    (assert (joint-layer-pr 5))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

; The section below considers layered rock & vertical joints
(defrule no-layering-vertical-joints
    ?lay <- (layers ?layers)
    ?j <- (joints ?joints)
    ?l <- (loop ?loop)
    (test (eq ?layers NONE))
    (test (eq ?joints vertical))
    =>
    (assert (joint-layer-pr 9))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

(defrule down-slope-dip-layers-vertical-joints
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
    ?j <- (joints ?joints)
    (test (eq ?layers down-slope-dip))
    (test (eq ?joints vertical))
    =>
    (assert (joint-layer-pr 10))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

(defrule cross-slope-dip-layers-vertical-joints
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
    ?j <- (joints ?joints)
    (test (eq ?layers cross-slope-dip))
    (test (eq ?joints vertical))
    =>
    (assert (joint-layer-pr 8))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

(defrule horizontal-layers-vertical-joints
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
```

```
    ?lay <- (layers ?layers)
    ?l <- (loop ?loop)
    ?j <- (joints ?joints)
    (test (eq ?layers horizontal))
    (test (eq ?joints cross-slope-dip))
    =>
    (assert (joint-layer-pr 7))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

; The section below considers unlayered rock & down sloping blocks
; or wedges joints
(defrule no-layering-down-slope-block-joints
    ?lay <- (layers ?layers)
    ?j <- (joints ?joints)
    ?l <- (loop ?loop)
    (test (eq ?layers NONE))
    (test (eq ?joints down-slope-block))
    =>
    (assert (joint-layer-pr 10))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

; This section considers unlayered rock & horizontal blocks or
; wedges joints
(defrule no-layering-horizontal-block-joints
    ?lay <- (layers ?layers)
    ?j <- (joints ?joints)
    ?l <- (loop ?loop)
    (test (eq ?layers NONE))
    (test (eq ?joints horizontal-block))
    =>
    (assert (joint-layer-pr 6))
    (retract ?l ?lay ?j)
    (assert (loop =(+ ?loop 1))))

;!! decision has been made loop= 1

;********************** LOOSE ROCK ******************************
;   This portion of the program identifies whether or not loose
; rock is present on the site. The loose rock may be specified as
; gravel or boulders. The site may consist only of loose rock or it
; may have intact (or near intact) rock below the loose rock.
;
```

```
;    INPUT VARIABLES
;         loose: NONE, gravel, boulders
;         loop
;
;    OUTPUT VARIABLES
;         loose-pr
;
;    DEFINITION OF VARIABLES
;         loose: indicates what type of loose rock exists on the
;              site
;
;*****************************************************************
(defrule no-loose-rock
     ?loos <- (loose ?loose)
     ?l <- (loop ?loop)
     (test (eq ?loose NONE))
     =>
     (assert (loose-pr 0))
     (retract ?l ?loos )
     (assert (loop =(+ ?loop 1))))

(defrule loose-gravel
     ?loos <- (loose ?loose)
     ?l <- (loop ?loop)
     (test (eq ?loose gravel))
     =>
     (assert (loose-pr 7))
     (retract ?l ?loos )
     (assert (loop =(+ ?loop 1))))

(defrule loose-boulders
     ?loos <- (loose ?loose)
     ?l <- (loop ?loop)
     (test (eq ?loose boulders))
     =>
     (assert (loose-pr 5))
     (retract ?l ?loos )
     (assert (loop =(+ ?loop 1))))

;!! decision has been made loop= 2

;******************** HOMOGENEOUS & INTACT ROCK ********************
;    This section determines if the site is composed of solid rock
; that is of one type.
```

```
;
;    INPUT VARIABLES
;         homogeneous-intact: NONE, Yes
;         loop
;
;    OUTPUT VARIABLES
;         intact-pr
;
;    DEFINITION OF VARIABLES
;         homogeneous-intact: identifies homogeneous, solid rock
;         ?hi, ?l: retraction addresses
;
;*****************************************************************
(defrule no-homogeneous-intact-rock
     ?hi <- (homogeneous-intact ?homogeneous-intact)
     ?l <- (loop ?loop)
     (test (eq ?homogeneous-intact NONE))
     =>
     (assert (intact-pr 0))
     (retract ?l ?hi )
     (assert (loop =(+ ?loop 1))))

(defrule homogeneous-intact-rock
     ?hi <- (homogeneous-intact ?homogeneous-intact)
     ?l <- (loop ?loop)
     (test (eq ?homogeneous-intact Yes))
     =>
     (assert (intact-pr 1))
     (retract ?l ?hi )
     (assert (loop =(+ ?loop 1))))

;!! decision has been made, loop= 3

;*****************************************************************
;*****************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
; the next slope.
(defrule continue-read
     ?l <- (loop ?loop)
     (test (|| (= ?loop 3) (> ?loop 3)))
     ?rdno <- (site-id ?site-id)
     ?jlpr <- (joint-layer-pr ?joint-layer-pr)
```

```
?lpr <- (loose-pr ?loose-pr)
?ipr <- (intact-pr ?intact-pr)
=>
(fprintout outr ?site-id " " ?joint-layer-pr " "
    ?loose-pr " " ?intact-pr crlf)
(retract ?l ?rdno ?jlpr ?lpr ?ipr)
(assert (count 1))
(assert (loop 0))
(assert (read-file)))
```

APPENDIX Q

TRIMPEDE.CLP

Traffic Impedance *Priority Rating* Program (TRIMPEDE.CLP)

This program considers the degree of traffic impedance resulting from the failure, and if paved detours are used. There are two factors considered in this program, traffic impedance and detours. The valid responses to the traffic impedance factor are:

1. **Road-Closed**: the entire road is closed to traffic.
2. **one-way-traffic**: only one direction of traffic can pass at one time.
3. **three-quarter-half**: this response can only be used if one-way-traffic is not applicable. It indicates that 3/4 or less of the road's width is affected.
4. **half-shoulder**: this response can only be used if one-way-traffic is not applicable. It indicates that up to 1/2 of the road's width is affected.
5. **shoulder**: only the shoulder of the road was affected.

The proper responses to the detour factor are:

1. **Yes:** paved detours are possible
2. **NONE:** paved detours are not possible.

The possibility of paved detours is considered only for the cases of Road-Closed and one-way-traffic. In all other cases, at least two lanes of traffic would be open, which is assumed to be adequate.

Figure Q.1 Traffic Impedance Flow Chart

```
; TRIMPEDE.CLP                    Last Revision 10-22-90
;
;                 TRAFFIC IMPEDANCE
;
;*****************************************************************
;        INPUT FILE = DATATI.TXT
;        OUTPUT FILE = RATTI.TXT

;*****************************************************************
;                 READS TEXT FILE
; This portion of the program reads the text file and extracts
; the Road ID, Road Impedance, and Detour Availability information.
; The information is read by the use of a loop.
;    Definition of Variables
;        count: a number from 1 to 4 which tells the program
;            whether to assign the read data to road-id
;            (count= 1), failnum (count= 2), road-impede
;            (count= 3), or detours (count=4)
;        read-file: is a flag that notifies the program that a
;            new piece of information can be read.
;        data-read: is the temporary address of the read
;            information until it can be properly identified.
;        site-id: records the identification number for the site
;        failnum: indicates how many times this site has failed
;        road-impede: indicates how severely the road is impeded
;        detours: indicates if detours are available
;        ?c, ?l, ?read-file, ?data-read, ?close-files: are used
;            to bind the appropriate information to a CLIPS
;            address so that the assigned facts can be retracted.
;*****************************************************************

(defrule open-file
    (initial-fact)
    =>
    (open "c:\\clipsfil\\datati.txt" datati "r")
    (open "c:\\clipsfil\\ratti.txt" outti "w")
    (assert (count 1))
    (assert (read-file)))

(defrule read-file                          ;Begins loop
    ?read-file <- (read-file)
    =>
    (retract ?read-file)
    (assert (data-read =(read datati))))          ;reads the data
```

```
(defrule read-site-id
    ?c <- (count 1)
    ?data-read <- (data-read ?site-id& ~ EOF)
    =>
    (retract ?data-read ?c)
    (assert (site-id ?site-id))
    (assert (count 2))
    (assert (read-file)))


(defrule read-failnum
    ?c <- (count 2)
    ?data-read <- (data-read ?failnum& ~ EOF)
    =>
    (retract ?data-read ?c)
    (assert (failnum ?failnum))
    (assert (count 3))
    (assert (read-file)))

(defrule read-road-impede
    ?c <- (count 3)
    ?data-read <- (data-read ?road-impede& ~ EOF)
    =>
    (retract ?data-read ?c)
    (assert (road-impede ?road-impede))
    (assert (read-file))
    (assert (count 4)))

(defrule read-road-detours
    ?c <- (count 4)
    ?data-read <- (data-read ?detours& ~ EOF)
    =>
    (retract ?data-read ?c)
    (assert (detours ?detours)))

(defrule close-all-files                    ;checks if the
    ?close-files <- (data-read EOF)         ;EOF has been
    ?c <- (count ?count)
    =>                                      ;reached, if so
    (retract ?close-files ?c)               ;it closes the file
    (close))

;*****************************************************************
;*****************************************************************
```

```
;            TRAFFIC IMPEDANCE SEVERITY
;   This portion of the program identifies how severely a road is
; impeded and if detour routes are available and then assigns a
; priority rating.
;   INPUT DATA
;       road-impede: NONE, Road-Closed, one-way-traffic,
;           three-quarter-half, half-shoulder, shoulder
;       detours: Yes, NONE
;
;   OUTPUT DATA
;       road-impede-pr:
;
;   Definition of Variables
;       road-impede: indicates if and to what extent the road is
;           impeded
;               Road-Closed: failure has closed road to traffic
;               one-way-traffic: traffic can only progress in
;                   one direction at a time
;               three-quarter-half: 3/4 to 1/2 of the road is
;                   closed (this description can only be used
;                   when two way traffic is available
;               half-shoulder: 1/2 to the shoulder is closed
;               shoulder: only the shoulder is affected
;       detours: indicates if paved detours are possible
;       road-impede-pr: road impedance priority rating
;
;*****************************************************************
(defrule Road-Closed
    ?ri <- (road-impede ?road-impede)
    ?d <- (detours ?detours)
    (test (eq ?road-impede Road-Closed))
    =>
    (if (eq ?detours Yes)
        then (assert (road-impede-pr 9)))
    (if (eq ?detours NONE)
        then (assert (road-impede-pr 10)))
    (retract ?ri ?d))

(defrule One-Way-Traffic
    ?ri <- (road-impede ?road-impede)
    ?d <- (detours ?detours)
    (test (eq ?road-impede one-way-traffic))
    =>
    (if (eq ?detours Yes)
```

```
        then (assert (road-impede-pr 7)))
    (if (eq ?detours NONE)
        then (assert (road-impede-pr 8)))
    (retract ?ri ?d))

(defrule three-quarter-half
    ?ri <- (road-impede ?road-impede)
    ?d <- (detours ?detours)
    (test (eq ?road-impede three-quarter-half))
    =>
    (assert (road-impede-pr 8))
    (retract ?ri ?d))

(defrule half-to-shoulder
    ?ri <- (road-impede ?road-impede)
    ?d <- (detours ?detours)
    (test (eq ?road-impede half-shoulder))
    =>
    (assert (road-impede-pr 5))
    (retract ?ri ?d))

(defrule shoulder
    ?ri <- (road-impede ?road-impede)
    ?d <- (detours ?detours)
    (test (eq ?road-impede shoulder))
    =>
    (assert (road-impede-pr 2))
    (retract ?ri ?d))

(defrule No-impedance
    ?ri <- (road-impede ?road-impede)
    ?d <- (detours ?detours)
    (test (eq ?road-impede NONE))
    =>
    (assert (road-impede-pr 0))
    (retract ?ri ?d))

; decision has been

;*************************************************************
;*************************************************************
; This rule checks to see if all of the priority ratings have
; been determined for one particular site.  If they have been
; determined it will begin the loop again and read data from
```

```
; the next slope.
(defrule continue-read
    ?road-no <- (site-id ?site-id)
    ?fn <- (failnum ?failnum)
    ?impedepr <- (road-impede-pr ?road-impede-pr)
    =>
    (fprintout outti ?site-id " " ?failnum " " ?road-impede-pr
                    crlf)
    (retract ?impedepr ?road-no ?fn)
    (assert (count 1))
    (assert (read-file)))
```

APPENDIX R

MAINMENU.PRG

Main Control Program for the Database (MAINMENU.PRG)

This program allows access to every dBASE program. The program displays the Figure DD.1. The user can then branch to different menus and programs. See the MAIN MENU section IN Appendix DD for further information on the functions of each choice in the menu.

This program follows the same general format as all of the menu programs.

```
** MAINMENU.PRG***
SET TALK OFF
SET ECHO OFF
STORE " " TO choice
USE date
CLEAR
?"Please Enter the Current Date."
@ 2,10 SAY "Current Month (1 to 12): "
@ 2,35 GET nowmonth
@ 4,10 SAY "Current Day (1 to 31): "
@ 4,35 GET nowday
@ 6,10 SAY "Current Year (19?? or 20??):"
@ 6,40 GET nowyear
READ
DO WHILE .t.
  CLEAR
  ?
  ?"  **************************************************"
  ?"                      MAIN MENU"
  ?"  **************************************************"
  ?
  ?"          1       User Instructions"
  ?
  ?"          2       Working with the Temporary Databases"
  ?
  ?"          3       Working with the Permanent Databases"
  ?
  ?"          4       Add an entire new site"
  ?
  ?"          5       Show all records"
  ?
  ?"          6       Create Input Files for the CLIPS programs"
  ?
  ?"          7       List/Print the Priority Ratings"
  ?
  ?"          Q       Quit"
  ?
  ?
  WAIT "  Enter Task Code " TO choice
  DO CASE
    CASE choice="1"
      Do userinst
    CASE choice="2"
      DO tempmenu
```

```
    CASE choice="3"
      DO permmenu
    CASE choice="4"
      DO addall
    CASE choice="5"
      DO showall
    CASE choice="6"
      DO input
    CASE choice="7"
      DO outmenu
    CASE UPPER(choice)="Q"
     CLOSE ALL
     RETURN
    OTHERWISE
      LOOP
  ENDCASE
ENDDO
```

APPENDIX S

INPUT.PRG

Creates CLIPS Input Files
INPUT.PRG

This program converts the database files to DOS text files to be used as the input files for the CLIPS programs.

```
**INPUT.PRG**
*    This program creates the text files used for the CLIPS
* programs.
SET TALK OFF
SET ECHO OFF
* Create the COST input file, DATACOST
USE cost
INDEX ON siteid + failnum TO costsort
COPY TO costpr FIELDS siteid, failnum, totalcost
USE costpr
COPY TO c:\clipsfil\datacost.txt TYPE DELIMITED WITH BLANK
?"DATACOST.TXT CREATED"
*
* Create the FSIZEPR input file, DATAFWD
USE failcond
INDEX ON siteid + failnum TO condsort
COPY TO fsizepr FIELDS siteid, failnum, volume, waterlevel, faildate
USE fsizepr
COPY TO c:\clipsfil\datafwd.txt TYPE DELIMITED WITH BLANK
?"DATAFWD CREATED"
*
* Create the PROBTYPE input file, DATAPROB
USE failcond
INDEX ON siteid + failnum TO condsort
COPY TO probtype FIELDS siteid, failnum, probtype
USE probtype
COPY TO c:\clipsfil\dataprob.txt TYPE DELIMITED WITH BLANK
?"DATAPROB CREATED"
*
* Create the TRIMPEDE input file, DATATI
USE damage
INDEX ON siteid + failnum TO damsort
COPY TO trimpede FIELDS siteid, failnum, roadimpede, detours
USE trimpede
COPY TO c:\clipsfil\datati.txt TYPE DELIMITED WITH BLANK
?"DATATI CREATED"
*
* Create the STRUCTUR input file, DATASTRU
USE damage
INDEX ON siteid + failnum TO damsort
COPY TO struct FIELDS siteid, failnum, structype, strucdamag
USE struct
COPY TO c:\clipsfil\datastru.txt TYPE DELIMITED WITH BLANK
?" DATASTRU CREATED"
```

```
*
* Create the LAWSUIT input file, DATALAW
USE damage
INDEX ON siteid + failnum TO damsort
COPY TO lawsuit FIELDS siteid, failnum, damagetype
USE lawsuit
COPY TO c:\clipsfil\datalaw.txt TYPE DELIMITED WITH BLANK
?"DATALAW CREATED"
*
* Create the PAVEDAM input file, DATAPAVE
USE damage
INDEX ON siteid + failnum TO damsort
COPY TO pavedam FIELDS siteid, failnum, probtype, pavedamage
USE pavedam
COPY TO c:\clipsfil\datapave.txt TYPE DELIMITED WITH BLANK
?"DATAPAVE CREATED"
*
* Create the TEMPLOAD input file, DATATMPL
USE tempload
INDEX ON siteid + failnum TO tmplsort
COPY TO c:\clipsfil\datatmpl.txt TYPE DELIMITED WITH BLANK
?"DATATMPL CREATED"
*
* Create the FAILFREQ input file, DATAFF
USE failcond
INDEX ON siteid + failnum TO condsort
COPY TO failfreq FIELDS siteid, failnum, l
failmonth, failday, failyear
USE failfreq
COPY TO c:\clipsfil\dataff.txt TYPE DELIMITED WITH BLANK
?"DATAFF CREATED"
*
* Create FAILFREQ input file, DATE
USE date
COPY TO c:\clipsfil\date.txt TYPE DELIMITED WITH BLANK
?"DATE CREATED"
*
* Create the EQUAKE input file, DATAEQ
USE geology
INDEX ON siteid TO geosort
COPY TO equake FIELDS siteid, accelcoeff
USE equake
COPY TO c:\clipsfil\dataeq.txt TYPE DELIMITED WITH BLANK
?"DATAEQ CREATED"
```

```
*
* Create the ADTROADT input file, DATAATRT
USE identity
INDEX ON siteid TO idensort
COPY TO adtroadt FIELDS siteid, roadtype, adt
USE adtroadt
COPY TO c:\clipsfil\dataatrt.txt TYPE DELIMITED WITH BLANK
?"DATAATRT CREATED"
*
* Create the DIRT input file, DATADIRT
USE geology
INDEX ON siteid TO geosort
COPY TO dirt FIELDS siteid, composed, firstsoil, secondsoil, soillayers, topsoil,
lowsoil, spt
USE dirt
COPY TO c:\clipsfil\datadirt.txt TYPE DELIMITED WITH BLANK
?"DATADIRT CREATED"
*
* Create the ROCK input file, DATAROCK
USE geology
INDEX ON siteid TO geosort
COPY TO rock FIELDS siteid, loose, homogintac, joints, ì
rocklayers
USE rock
COPY TO c:\clipsfil\datarock.txt TYPE DELIMITED WITH BLANK
?"DATAROCK CREATED"
*
* Create the ECONIMPO input file, DATAECON
USE identity
INDEX ON siteid TO idensort
COPY TO econimpo FIELDS siteid, accesstype, population
USE econimpo
COPY TO c:\clipsfil\dataecon.txt TYPE DELIMITED WITH BLANK
?"DATAECON CREATED"
*
* Create the PERMLOAD input file, DATAPRML
USE permload
INDEX ON siteid TO prmlsort
COPY TO c:\clipsfil\dataprml.txt TYPE DELIMITED WITH BLANK
?"DATAPRML created"
*
* Create the GEOHAZ input file, DATAGHAZ
USE geology
INDEX ON siteid TO geosort
```

```
COPY TO geohaz FIELDS siteid, geohazard
USE geohaz
COPY TO c:\clipsfil\dataghaz.txt TYPE DELIMITED WITH BLANK
? "DATAGHAZ CREATED"
RETURN
```

APPENDIX T

DELALLPR.PRG

Program to Delete All Output Data
(DELALLPR.PRG)

This program deletes all data from the output databases. The output databases contain the *priority ratings* determined from the CLIPS programs. There is an output database for each CLIPS output file.

```
**DELALLPR.PRG**
* This program deletes the information in the output databases.
SET TALK OFF
SET ECHO OFF
USE ratprob
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratti
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratpave
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratstru
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE rattmpl
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratff
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratcost
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
```

```
   PACK
ENDDO
USE ratfwd
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE rateq
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratatrt
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratdirt
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratrock
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratecon
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
USE ratprml
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
```

```
USE ratghaz
DO WHILE .NOT. EOF()
   * Delete the site
   DELETE
   PACK
ENDDO
?" The files have been deleted"
WAIT
CLOSE ALL
RETURN
```

APPENDIX U

OUTPUT.PRG

Creates Output Databases (OUTPUT.PRG)

This program creates output databases from the CLIPS output files. The created databases contain the *priority ratings* determined from CLIPS. It creates an individual database file for each CLIPS output file.

```
**OUTPUT.PRG**
* Creates the priority databases
SET TALK OFF
SET ECHO OFF
USE ratprob
APPEND FROM c:\clipsfil\ratprob.txt TYPE DELIMITED WITH BLANK
?"ratprob imported"
USE ratti
APPEND FROM c:\clipsfil\ratti.txt TYPE DELIMITED WITH BLANK
?"ratti imported"
USE ratlaw
APPEND FROM c:\clipsfil\ratlaw.txt TYPE DELIMITED WITH BLANK
?"ratlaw imported"
USE ratpave
APPEND FROM c:\clipsfil\ratpave.txt TYPE DELIMITED WITH BLANK
?"ratpave imported"
USE ratstru
APPEND FROM c:\clipsfil\ratstru.txt TYPE DELIMITED WITH BLANK
?"ratstru imported"
USE rattmpl
APPEND FROM c:\clipsfil\rattmpl.txt TYPE DELIMITED WITH BLANK
?"rattmpl imported"
USE ratff
APPEND FROM c:\clipsfil\ratff.txt TYPE DELIMITED WITH BLANK
?"ratff imported"
USE ratcost
APPEND FROM c:\clipsfil\ratcost.txt TYPE DELIMITED WITH BLANK
?"ratcost imported"
USE ratfwd
APPEND FROM c:\clipsfil\ratfwd.txt TYPE DELIMITED WITH BLANK
?"ratfwd imported"
USE rateq
APPEND FROM c:\clipsfil\rateq.txt TYPE DELIMITED WITH BLANK
?"rateq imported"
USE ratatrt
APPEND FROM c:\clipsfil\ratatrt.txt TYPE DELIMITED WITH BLANK
?"ratatrt imported"
USE ratdirt
APPEND FROM c:\clipsfil\ratdirt.txt TYPE DELIMITED WITH BLANK
?"ratdirt imported"
USE ratrock
APPEND FROM c:\clipsfil\ratrock.txt TYPE DELIMITED WITH BLANK
?"ratrock imported"
USE ratecon
```

```
APPEND FROM c:\clipsfil\ratecon.txt TYPE DELIMITED WITH BLANK
?"ratecon imported"
USE ratprml
APPEND FROM c:\clipsfil\ratprml.txt TYPE DELIMITED WITH BLANK
?"ratprml imported"
USE ratghaz
APPEND FROM c:\clipsfil\ratghaz.txt TYPE DELIMITED WITH BLANK
?"ratghaz imported"
CLOSE ALL
SET PRINT OFF
RETURN
```

APPENDIX V

WEIGHT.PRG

Calculates the *Total Priority Rating* (WEIGHT.PRG)

This program calculates the *Total Priority Rating*. It begins by applying the proper weights to each factor in the *temporary* and *permanent* output databases, see Appendices W and X. The weighted *temporary* factors are added and the sum is stored in the Failure Condition database. The weighted *permanent* factors are added and the sum is stored in the Identity. These two sums are then added divided by 19, the total number of factors. The final number can theoretically range from 0 to 100, where 100 indicates an extremely important site.

Executing this program will cause the weighted *permanent* and *temporary* factors and their respective sums to be printed. The total priorities for each site are then printed.

```
**WEIGHT.PRG**
*
SET TALK OFF
SET ECHO OFF
DO tempwt
DO permwt
SET PRINT ON
?
?
?
?"              TOTAL PRIORITY RATING"
?
? " SITE ID         Failure No.          TOTAL PR"
?"*****************************************************"
SELECT 1
USE identity
SELECT 2
USE failcond
DO WHILE .NOT. EOF()
   SELECT 2
   sid=siteid
   nf=failnum
   tf=twt
   SELECT 1
   GO TOP
   LOCATE FOR siteid=sid
   pf=pwt
   totalpr=(pf + tf)/19
   ? siteid, SPACE(3), nf, SPACE(10), totalpr
   SELECT 2
   SKIP
ENDDO
SET PRINT OFF
WAIT
CLOSE ALL
RETURN
```

APPENDIX W

TEMPWT.PRG

Program to Calculate *Temporary* Weighted *Factors* (TEMPWT.PRG)

This program multiplies the *temporary priority ratings* by an appropriate weight. The program then sums these products. The total for each site is then stored in the Failure Condition database. The total for each site is also printed.

The weight values for each *factor* are listed in Table 8.

Table 8 Weights for the *Temporary factors*

factors	weights
1. problem type	6
2. traffic impedance	5
3. lawsuit potential	1
4. pavement damage	3
5. structure type & damage	5
6. temp. static & dynamic load	3
7. failure frequency	8
8. repair cost	15
9. failure size	2
10. failure water level	10
11. failure date	1
	**********
WEIGHT TOTAL	**59**

```
**TEMPWT.PRG**
*
SET TALK OFF
SET ECHO OFF
SELECT 1
USE ratprob
SELECT 2
USE ratti
SELECT 3
USE ratlaw
SELECT 4
USE ratpave
SELECT 5
USE ratstru
SELECT 6
USE rattmpl
SELECT 7
USE ratff
SELECT 8
USE ratcost
SELECT 9
USE ratfwd
SELECT 10
USE failcond
? " Make Sure that PRINTER is ON"
WAIT
SET PRINT ON
?
?
?"SITE ID      Failure No.        Sum of Temp. PR"
?"**********************************************************"
DO WHILE .NOT. EOF()
   SELECT 1
   sid=siteid
   nf=failnum
   probwt= probtypepr * 6
   SELECT 2
   GO TOP
   LOCATE FOR siteid=sid .AND. failnum=nf
   impedewt= impedepr * 5
   SELECT 3
   GO TOP
   LOCATE FOR siteid=sid .AND. failnum=nf
   lawwt= lawpr * 1
```

```
    SELECT 4
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    pavedamwt= pavedampr * 3
    SELECT 5
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    strucwt= (strucpr + strdampr)/2 * 5
    SELECT 6
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    tmplwt= (tstaticpr + tdynapr)/2 * 3
    SELECT 7
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    ffwt= failfreqpr * 8
    SELECT 8
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    costwt= costpr * 15
    SELECT 9
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    sizewt= sizepr * 2
    waterwt= waterpr * 10
    datewt= datepr * 1
    twt1= probwt + impedewt + lawwt + pavedamwt + strucwt + tmplwt
    twt2= ffwt + costwt + sizewt + waterwt + datewt
    ttwt= twt1 + twt2
    ? siteid, SPACE(6), failnum, SPACE (10), ttwt
    SELECT 10
    GO TOP
    LOCATE FOR siteid=sid .AND. failnum=nf
    REPLACE twt WITH ttwt
    SELECT 1
    SKIP
ENDDO
SET PRINT OFF
WAIT
CLOSE ALL
RETURN
```

APPENDIX X

PERMWT.PRG

Program to Calculate *Permanent* Weighted *Factors*
(PERMWT.PRG)

This program multiplies the *permanent priority rating* by an appropriate weight. The program then sums these products. The total for each site is then stored in the Identity database. The total for each site is printed.

The weight values for each *permanent factor* are listed below.

Table 9 Weights for the Permanent Factors

factors	weights
1. ADT	12
2. road type	10
3. seismic	3
4. soil type & layers	4
5. rock joint-layers & loose & intact	4
6. economic	3
7. *perm.* static & dynamic load	3
8. geographical hazard	2
	**********
PERMANENT WEIGHT TOTAL	**41**

```
**WEIGHT.PRG**
*
SET TALK OFF
SET ECHO OFF
SELECT 1
USE ratatrt
SELECT 2
USE rateq
SELECT 3
USE ratdirt
SELECT 4
USE ratrock
SELECT 5
USE ratecon
SELECT 6
USE ratprml
SELECT 7
USE ratghaz
SELECT 8
USE identity
SET PRINT ON
?
?
?
?" SITE ID              Sum of Perm. PR"
?"*************************************"
DO WHILE .NOT. EOF()
   SELECT 1
   sid=siteid
   adtwt= adtpr * 12
   roadtypewt= roadtypepr * 10
   SELECT 2
   GO TOP
   LOCATE FOR siteid=sid
   seismicwt= seismicpr * 3
   SELECT 3
   GO TOP
   LOCATE FOR siteid=sid
   soilwt= (soiltypepr + soillaypr)/2 * 4
   SELECT 4
   GO TOP
   LOCATE FOR siteid=sid
   rockwt= (jointlaypr + loosepr + intactpr)/3 * 4
   SELECT 5
```

```
    GO TOP
    LOCATE FOR siteid=sid
    econwt= economicpr * 3
    SELECT 6
    GO TOP
    LOCATE FOR siteid=sid
    pldwt= (pstaticpr + pdynapr)/2 * 3
    SELECT 7
    GO TOP
    LOCATE FOR siteid=sid
    ghazwt= geohazpr * 2
    pwt1= adtwt + seismicwt + roadtypewt + soilwt + rockwt
    pwt2= econwt + pldwt + ghazwt
    ppwt= pwt1 + pwt2
    ? siteid, SPACE(10), ppwt
    SELECT 8
    GO TOP
    LOCATE FOR siteid=sid
    REPLACE pwt WITH ppwt
    SELECT 1
    SKIP
ENDDO
SET PRINT OFF
WAIT
CLOSE ALL
RETURN
```

APPENDIX Y

ADDCOST.PRG

Program to Append the Cost Database (ADDCOST.PRG)

This program appends a record to the Cost database. It gives the user the option to view Cost Instructions (Figure DD.14), which list the valid responses to each *factor*. The data input screen that appears when this program is executed is shown in Figure DD.15. This program follows the same general format that all of the append programs follow.

This program calculates the labor costs, equipment costs, repaving costs, and earthwork costs and enters this data into the database.

```
**ADDCOST.PRG**
*    This program allows the user to enter cost information for
* a new failure.
SET TALK OFF
SET ECHO OFF
CLEAR
USE cost
STORE " " TO b
? "Do you wish to see an explanation of the variables and a list"
? "of valid responses?"
WAIT "     Please answer Y or N " TO b
IF UPPER(b)="Y"
   DO costinst
ENDIF
SET FORMAT TO cost.fmt
APPEND BLANK
READ
newlabor= manhours * manrate
REPLACE laborcost WITH newlabor
newequip= equiphours * equiprate
REPLACE equipcost WITH newequip
newearth= earthwork * earthrate
REPLACE earthcost WITH newearth
newpave= repavehour * paverate
REPLACE pavecost WITH newpave
RETURN
```

APPENDIX Z

EDITCOST.PRG

Program to Edit Information in the Cost Database (EDITCOST.PRG)

This program edits existing data within the Cost Database. It gives the user the option to view the Cost Instructions (Figure DD.14), which lists the valid responses to each *factor*. The program then asks for the Site ID and the failure number of the failure site that is to be edited. It then displays Figure DD.15 with the appropriate information included. This program follows the same general format that all of the edit programs follow.

This program calculates the labor costs, equipment costs, repaving costs, and earthwork costs and enters this data into the Cost database.

```
**EDITCOST.PRG**
*   This program edits information already present in the COST
* database.
SET TALK OFF
SET ECHO OFF
USE cost
STORE " " TO a
? "Do you wish to see an explanation of the variables and a list"
? "of valid responses?"
WAIT "        Please answer Y or N " TO a
IF UPPER(a)="Y"
   DO costinst
ENDIF
DO findsite
CLEAR
SET FORMAT TO cost.fmt
READ
newlabor= manhours * manrate
REPLACE laborcost WITH newlabor
newequip= equiphours * equiprate
REPLACE equipcost WITH newequip
newearth= earthwork * earthrate
REPLACE earthcost WITH newearth
newpave= repavehour * paverate
REPLACE pavecost WITH newpave
RETURN
```

APPENDIX AA

DELCOST.PRG

Program to Delete a Record from the Cost Database (DELCOST.PRG)

This program deletes information for a specific site from the Cost Database. The program asks the user for the Site ID and the failure number of the site to be deleted. This program is an example of a deletion program. All other delete programs follow the same format.

```
**DELCOST.PRG**
* This program deletes the cost information for a certain site.
SET TALK OFF
SET ECHO OFF
USE cost
*  Ask user for the appropriate site and then find the site.
DO findsite
IF .NOT. sitefound .AND. failnumfound
   * No such site found
   RETURN
ELSE
   * Delete the site
   DELETE
   PACK
   @10,1 SAY "The site cost information has been deleted."
   WAIT
ENDIF
RETURN
```

APPENDIX BB

SHOWCOST.PRG

Program to Display Records in the Cost Database (SHOWCOST.PRG)

This program displays all of the information stored in the Cost Database. The program gives the user the option to print this information.

This program follows the same general format as all of the show programs.

```
**SHOWCOST.PRG**
* This program displays a site in the COST Database. It also
* allows the user to print this information.
SET TALK OFF
SET ECHO OFF
USE cost
CLEAR
STORE " " TO a
? "Do you wish to print this information?"
WAIT "       Please enter Y or N " TO a
IF UPPER(a)="Y"
   SET PRINT ON
ENDIF
SET FORMAT TO cost.fmt
DISPLAY ALL siteid, failnum, laborcost, equipcost, earthcost,
pavecost, totalcost OFF
WAIT
SET PRINT OFF
RETURN
```

APPENDIX CC

QUESTIONNAIRE

Questionnaire

The questionnaire was sent to various WSDOT Personnel. The following pages are a copy of what they received.

DIRECTIONS FOR QUESTIONNAIRE

Throughout the questionnaire I ask for you to rate an item from 1 to 10. Ten represents the most important, severe, or extreme case. One represents the least important, severe, or extreme case. Any item within an area can have any number. For example if you feel landslides, rockfall, and flows are all extremely important assign them all the value of 10 or which ever value you feel is appropriate. You do not always have to assign values of 1 or 10, just the value you feel is adequate.

If you feel an item is unnecessary please put a "UN" in the appropriate blank. If you feel an item is not applicable please put a "NA" in the appropriate blank.

Please, feel free to add any additional items to a category or make additional comments.

DEFINITIONS

Some of the following defintions have been taken from Landslides Analysis and Control Special Report 176 Chapter 2 "Slope Movement Types and Processes" by David J. Varnes.

TYPES OF MOVEMENT

Slides: movements that consist of shear strains and displacements along one or several surfaces that are visible or may be inferred.

Debris s Flows: composed of unconsolidated materials that can be wet or dry, with fast or slow movements. A velocity distribution of a flow resembles that of a viscous fluid.

Lateral Spreads: flows that occur laterally instead of down slope.

Fill or Subgrade Settlement: considered to be compaction of the fill material.

Degradation of Fill: settlement of the fill due to the actual decomposition of the fill material.

TYPES OF DISTRESS

Settlement: considered to be the vertical displacement between two adjacent sections of road.

Cracking: any type (longitudinal, transverse, or alligator); specify the length, width,

and depth of the crack.

Undulations: caused by freeze-thaw action or by the shrink-swell potential of the soil and can be described as a solitary heave or a dip or a series of heaves and dips (waves); specify the total vertical displacement, crest to trough or original road surface to the crest or trough, of the displacement.

SEASONS

Spring: March 1 through May 31

Summer: June 1 through August 31

Fall: September 1 through November 30

Winter: December 1 through Feburary 28 (29)

Feburary 13, 1990

NAME _____
ACTUAL JOB TITLE _____
DISTRICT _____
DATE _____

Unstable Slope Management Questionnaire

I. FACTORS CONCERNING PRIORITIZATION OF SLOPES

TYPES OF SLOPE MOVEMENTS

(Please refer to the definitions section for further clarification)

Rate the failure types according to COST, where 10 is the most costly and 1 is the least costly.

ASSUME that the slope movements in each case below will have the same mass and volume, be the same proximity to a certain road, and have no buildings in their area.

Type of Slope Movements	Overall Cost	Repair Cost	COMMENTS:
1. Slides	_____	_____	
2. Rockfall	_____	_____	
3. Debris Flows	_____	_____	
4. Lateral Spreads	_____	_____	
5. Fill Degradation & Settlement	_____	_____	
6. Others? (Specify)	_____	_____	

Indicate the relative site ACCESSIBILITY for each type of failure, where 10 is the most accessible and 1 is the least accessible.

ASSUME that the slope movements in each case below will have the same mass and volume, be the same proximity to a certain road, and have no buildings in their area.

Type of Slope Movements	Equipment Accessibility	COMMENTS:
1. Slides	_____	
2. Rockfall	_____	
3. Debris Flows	_____	
4. Lateral Spreads	_____	
5. Fill Degradation & Settlement	_____	
6. Others? (Specify)	_____	

Indicate the relative amount of time required to repair each type of failure, where 10 is the most time required and 1 is the least time required.

ASSUME that the slope movements in each case below will have the same mass and volume, be the same proximity to a certain road, and have no buildings in their area.

Type of Slope Movements	Repair Time	COMMENTS:
1. Slides	_____	
2. Rockfall	_____	
3. Debris Flows	_____	
4. Lateral Spreads	_____	
5. Fill Degradation & Settlement	_____	
6. Others? (Specify)	_____	

Indicate how difficult each type of failure is to repair, where 10 is the most difficult and 1 is the least difficult.

ASSUME that the slope movements in each case below will have the same mass and volume, be the same proximity to a certain road, and have no buildings in their area.

Type of Slope Movements	Repair Difficulty	COMMENTS:
1. Slides	_____	
2. Rockfall	_____	
3. Debris Flows	_____	
4. Lateral Spreads	_____	
5. Fill Degradation & Settlement	_____	
6. Others? (Specify)	_____	

Indicate which type of failure is the most difficult to contract work for, where 10 is the most difficult and 1 is the least difficult.

ASSUME that the slope movements in each case below will have the same mass and volume, be the same proximity to a certain road, and have no buildings in their area.

Type of Slope Movements	Contracting	COMMENTS:
1. Slides	_____	
2. Rockfall	_____	
3. Debris Flows	_____	
4. Lateral Spreads	_____	
5. Fill Degradation & Settlement	_____	
6. Others? (Specify)	_____	

Indicate which type of failure is the most important when deciding in which order slopes should be repaired, where 10 is the most important and 1 is the least important.

ASSUME that the slope movements in each case below will have the same mass and volume, be the same proximity to a certain road, and have no buildings in their area.

Type of Slope Movements	Programming	COMMENTS:
1. Slides	_____	
2. Rockfall	_____	
3. Debris Flows	_____	
4. Lateral Spreads	_____	
5. Fill Degradation & Settlement	_____	
6. Others? (Specify)	_____	

TYPE OF STRUCTURES INVOLVED

Indicate the importance of each type of structure if it were in danger of being severely damaged or destroyed. (10 - most important; 1 - insignificant)

1. Railroad bridges_____
2. Automobile bridges

	Road Capacity (average daily traffic: thousands/day)			
	>30	10 to 30	5 to 10	<5
Interstates	___	___	___	___
Multi-lane arterials	___	___	___	___
2-lane primary hwy	___	___	___	___
Gravel roads	___	___	___	___
Frontage road	___	___	___	___

3. Pedestrian bridges_____
4. Homes _____
5. Storage Buildings _____
6. Industrial or Commercial Buildings _____
7. Roads _____

	Road Capacity (average daily traffic: thousands/day)			
	>30	10 to 30	5 to 10	<5
Interstates	___	___	___	___
Multi-lane arterials	___	___	___	___
2-lane primary hwy	___	___	___	___
Gravel roads	___	___	___	___
Frontage road	___	___	___	___

8. Drainage Structures _____
9. Utility Infra-Structure
   (power lines, sewers, water lines,...) _____
10. Others? _____

If there were a chance that the slopes involved could affect any storage tanks or other structures containing hazardous wastes, fuels, explosives, or other pollutants, how would you rate these structures in comparison with the structures previously listed? _____

DISTRESS OF ROAD

Please indicate the dimensions of each of the distress types that would cause the road to be unsafe, failed, impaired and uncomfortable to drivers. See the definitions section for further clarification.

Magnitude (actual displacement) of Distress that will cause the road to be:

Type of Distress	Unsafe	Failed	Impaired	Uncomfortable to Drivers
1. Settlement	_____	_____	_____	_____
2. Cracking - length	_____	_____	_____	_____
- width	_____	_____	_____	_____
- depth	_____	_____	_____	_____
3. Undulations	_____	_____	_____	_____
4. Road Covered by Debris	_____	_____	_____	_____
5. Others? (Specify)				

1. Does misalignment of a road ever occur due to slope movements? (yes or no)

If so is it ever a serious problem in itself? (Are the misalignments ever large enough to be a hazard to motorists?) Explain and indicate the dimensions.

2. When evaluating the quality and life of a road, how much importance do you assign ride quality? (very important,..., not important) Explain.

To what degree do you consider these types of distress to be unsafe? (10 - very unsafe; 1 - safe)

_____ 1. Settlement
_____ 2. Cracking
_____ 3. Misalignment
_____ 4. Debris on road
_____ 5. Undulations
_____ 6. Others? (Specify)

MOBILIZATION & SAFETY OF WORK FORCE

(Please refer to the definitions section for further clarification on seasons)

For the next sections 10 indicates the most important case and 1 indicates the least difficult case.

GENERAL SLOPE FAILURE

How difficult is it to mobilize a work force during the specific seasons to repair a general slope failure? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	___	___	___	___	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	___	___	___	___	
3. Large - 4 or more pieces of equip. crew of 7 or more	___	___	___	___	
4. Emergency Contract	___	___	___	___	

How safe is each situation for the maintenance crews? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	___	___	___	___	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	___	___	___	___	
3. Large - 4 or more pieces of equip. crew of 7 or more	___	___	___	___	
4. Emergency Contract	___	___	___	___	

Time of Day: MORNING (7am to 12pm)

How difficult is it to mobilize a work force during the specific seasons in the morning? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	___	___	___	___	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	___	___	___	___	
3. Large - 4 or more pieces of equip. crew of 7 or more	___	___	___	___	
4. Emergency Contract	___	___	___	___	

How safe is each situation for the maintenance crews? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	___	___	___	___	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	___	___	___	___	
3. Large - 4 or more pieces of equip. crew of 7 or more	___	___	___	___	
4. Emergency Contract	___	___	___	___	

Time of Day: AFTERNOON (12pm to 6pm)

How difficult is it to mobilize a work force during the specific seasons in the afternoon? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	____	____	____	____	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	____	____	____	____	
3. Large - 4 or more pieces of equip. crew of 7 or more	____	____	____	____	
4. Emergency Contract	____	____	____	____	

How safe is each situation for the maintenance crews? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	____	____	____	____	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	____	____	____	____	
3. Large - 4 or more pieces of equip. crew of 7 or more	____	____	____	____	
4. Emergency Contract	____	____	____	____	

Time of Day: NIGHT TIME (6pm to 7am)

How difficult is it to mobilize a work force during the specific seasons at Night? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	____	____	____	____	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	____	____	____	____	
3. Large - 4 or more pieces of equip. crew of 7 or more	____	____	____	____	
4. Emergency Contract	____	____	____	____	

How safe is each situation for the maintenance crews? (1 to 10)

Size of Slope Repair Job	Seasons: Spring	Summer	Fall	Winter	COMMENTS:
1. Small - 1 piece of equip. crew of 1-3 people	____	____	____	____	
2. Medium - 2 to 3 pieces of equip. crew of 3-8 people	____	____	____	____	
3. Large - 4 or more pieces of equip. crew of 7 or more	____	____	____	____	
4. Emergency Contract	____	____	____	____	

1. At what point are emergency contracts issued? Explain.

2. Does season affect the cost of repair? (ie. Spring more expensive...) Explain.

3. Does the time of day have any significance on how easily a work force can be mobilized? (Yes or No) Explain.

4. Will a failure that must be immediately repaired cause significant problems if it fails during off hours, holidays, weekends? (Yes or No) Explain.

Will costs go up significantly? (Yes or No) Explain.

Will the use of the equipment cost more? (Yes or No) Explain.

Are workers more difficult to find? (Yes or No) Explain.

5. Does the fact that a slope is wooded or is grassy make a significant impact upon the ease or difficulty of clean up or the cost of clean up? (Yes or No) Explain.

6. Are most slopes fairly accessible (easy to get equipment to where it is needed)? (Yes or No) Explain.

7. When considering repairing a slope, how important is the accessibility of the site?

8. If machinery is unable to access the slope, will the slope be repaired? (Yes or No) Explain.

9. If the slope must be repaired without using machinery (i.e. manual labor) how much more expensive will it be?

COSTS

Below are some factors that can contribute to the costs that slope movements may incur. Please indicate the relative importance of these factors and add any additional factors you feel are necessary.

(Indicate 1 to 10, where 10 represents the most important and 1 represents the least important factor.)

DIRECT COSTS

____ 1. Distress to road and actual incurred repair costs (including any stabilization or improvement steps)
____ 2. Size of failure
____ 3. Debris removal
____ 4. Damage to structures
____ 5. Whether or not work must be done in off hours (overtime or the incidence occurs during off hours)
____ 6. Rehabilitation measures (drainage system, rock bolts,...)
____ 7. Consultation fees
____ 8. Equipment Costs
____ 9. Others (Specify)

INDIRECT COSTS

____ 1. Cost of traffic delay
____ 2. Cost of extra mileage due to use of alternative route
____ 3. Hindrance to local business (denies or hinders access leads to discouraging business)
____ 4. Excess wear on vehicles due to rougher roads
____ 5. Are there vehicle restrictions (size, weight) on alternate routes that could lead to expense?
____ 6. Deterioration of alternate route due to increase in traffic
____ 7. Lawsuits filed due to loss or damage to life or property
____ 8. Economic impact on the area
____ 9. Environmental impact on the area
____ 10. Others? (Specify)

How important is the cost? Does cost importance vary with the amount? (ie. importance goes up with cost, importance goes down with cost,...)

Listed below are several categories of cost. Please indicate the relative importance of each category.

(10 - very important; 1 - least important)

____ 1. Maintenance Costs
____ 2. Emergency Repair Costs
____ 3. Construction Costs
____ 4. Labor Costs
____ 5. Equipment Costs
____ 6. Material Costs

How is the cost determined?

COMMENTS:

DECISION FACTORS

In this section there are several different scenarios that consider different factors, for each scenario indicate the significance of each factor in the decision making process.
(10 - very important; 1 - insignificant)

For Scenarios 1, 2, and 3 assume that a land slide covers one direction of traffic of the interstate on Snowqualamie Pass during the winter. The slide causes direct damage to a major hotel near a ski area.

Scenario No. 1

If the factors below were involved in determining the priority of a slope please rate each factor according to its importance in deciding which slopes are more critical.

____ 1. Possibility of fatality or injury
____ 2. Property damage (except to road itself)
____ 3. Cost
____ 4. Type of failure
____ 5. Type of distress to road
____ 6. Type of road
____ 7. Alternate Routes (availability & convenience)
____ 8. Volume of traffic
____ 9. Frequency of failure
____ 10. Size of failure
____ 11. Fluctuation of ground water table
____ 12. Earthquake effects
____ 13. Flooding
____ 14. Road sliding into a body of water
____ 15. Road sliding off a cliff
____ 16. Others? (Specify)

COMMENTS:

Scenario No. 2

____ 1. Property damage (except to road itself)
____ 2. Cost
____ 3. Type of failure
____ 4. Type of distress to road
____ 5. Type of road
____ 6. Alternate Routes (availability & convenience)
____ 7. Volume of traffic
____ 8. Frequency of failure
____ 9. Size of failure
____ 10. Fluctuation of ground water table
____ 11. Earthquake effects

COMMENTS:

Scenario No. 3

____ 1. Property damage (except to road itself)
____ 2. Cost
____ 3. Type of failure
____ 4. Type of distress to road
____ 5. Type of road
____ 6. Volume of traffic
____ 7. Frequency of failure
____ 8. Size of failure

COMMENTS:

For Scenarios 4 and 5 assume that a land slide covers the entire width of a portion of the interstate along the Columbian River Gorge during the winter. Assume the affected area lies somewhere between The Dalles and Portland.

Scenario No. 4

If the factors below were involved in determining the priority of a slope please rate each factor according to its importance in deciding which slopes are more critical.

____ 1. Possibility of fatality or injury
____ 2. Property damage (except to road itself)
____ 3. Cost
____ 4. Type of failure
____ 5. Type of distress to road
____ 6. Type of road
____ 7. Alternate Routes (availability & convenience)
____ 8. Volume of traffic
____ 9. Frequency of failure
____ 10. Size of failure
____ 11. Fluctuation of ground water table
____ 12. Earthquake effects
____ 13. Flooding
____ 14. Road sliding into a body of water
____ 15. Road sliding off a cliff
____ 16. Others? (Specify)

COMMENTS:

Scenario No. 5

____ 1. Property damage (except to road itself)
____ 2. Cost
____ 3. Type of failure
____ 4. Type of distress to road
____ 5. Type of road
____ 6. Volume of traffic
____ 7. Frequency of failure
____ 8. Size of failure

COMMENTS:

MISC. QUESTIONS

1. Should this system be concerned only with slopes that fail regularly (Yes or No)? Explain.

2. If it was possible to incorporate into the system, would it be useful to have the system indicate which slopes should be improved and how? (put drainage in, fencing, retaining walls, move road,...)

Or should it be limited to evaluating the slopes and stating whether or not they should be improved?

Or should it just evaluate the slope?

3. What is the current practice when a slope fails near an interstate highway so that it affects the road's performance? Please list in chronological order the steps involved.

4. What is the current practice when a slope fails near a multi-lane primary highway so that it affects the road's performance? Please list in chronological order the steps involved.

5. What is the current practice when a slope fails near a 2-lane primary highway so that it affects the road's performance? Please list in chronological order the steps involved.

6. What is the current practice when a slope fails near a gravel road so that it affects the road's performance? Please list in chronological order the steps involved.

7. What is the current practice when a slope fails near a frontage road so that it affects the road's performance? Please list in chronological order the steps involved.

II. TYPES OF REPORTS GENERATED BY & REQUIREMENTS OF USMS

INFORMATION INCLUDED IN REPORTS

Listed below are several categories of information that the user can choose to have included in the report. Please indicate if any other categories should be included or if any categories are unnecessary.

NA - Not Acceptable
UN - Unnecessary
X - Acceptable

____ 1. Prioritization Scores
____ 2. Slope Identification Information_
____ 3. Cost Information
____ 4. Induced Distress (actual and possible?)
____ 5. Other?

1. How do you foresee using the USMS in your daily work?

2. The database for the USMS will have to be updated periodically so that accurate prioritizations can be made; who should be responsible for updating the database.

3. How would the database be best updated? (ie. send out updated discs, have a central system,...)

4. How frequently should the database be updated?

ADDITIONAL COMMENTS:

APPENDIX DD

USER GUIDE

User Guide

This section is intended to instruct the user on how to use the USMS. The User Guide provides information on how to install the USMS, execute the USMS, and identify problems within the USMS. The User Guide begins with a command summary for the USMS. It then provides installation instructions. The User Guide then explains the menus and screens within the dBASE portion of the system. After this section, the user is instructed as how to execute the CLIPS programs. The last section within the User Guide addresses any problems that may occur during the use of the USMS.

Within the User Guide are examples of the screens that are displayed during the use of the USMS. These examples contain example fictional data. The fictional data is meant to clarify the use of the USMS.

BASIC OPERATION STEPS FOR THE USMS

1. To enter dBASE and the Main dBASE Program, enter the **C:\DBASE** directory and then enter the following commands in that order:

   DBASE

   ESC

   DO MAINMENU

2. To enter data for a failure site, enter the number **4** (see MAIN MENU section) while in the Main Menu. The program will then allow the user

to enter data for a failure site to each database.

3. To create input files for the CLIPS programs, from the Main Menu enter the number **6** (see MAIN MENU section).

4. To exit dBASE type

   QUIT

5. To execute the CLIPS programs, enter the directory **C:\CLIPS** and then enter the following commands in that order:

   CLIPS

   (batch "c:\\clipsfil\\control.clp")

   (exit)

   The lower case commands must be entered in lower case.

6. To calculate the *total priority rating*, enter dBASE and the Main Menu. Then enter the number **7**. The Output Menu will then appear (see the OUTPUT MENU section). From the Output Menu enter the following commands in that order:

   D (Delete old data to import new data)

   F (Import the data from CLIPS files)

   3 (Calculate & Print the TOTAL Priority Rating).

INSTALLATION OF THE USMS

As it exists now, the USMS must be installed onto a hard disk. Before the CLIPS

programs can be copied onto the hard disk a subdirectory named **C:\CLIPSFIL** must be created. It is to this directory that the CLIPS programs must be copied. The specific subdirectory of **C:\CLIPSFIL** must exist because the read statements within the CLIPS programs call the input files from this directory and store the output information to this directory. The database must also access this subdirectory. If you wish to change the name of the directory you will have to change the drive specifications within each CLIPS program and within the database programs INPUT.PRG and OUTPUT.PRG.

The CLIPS software system and the dBASE software system have installation programs which assist the user in copying the programs to the hard disk. It is recommended that the CLIPS system be copied to a directory named **C:\CLIPS** and that the dBASE III Plus system be copied to a directory named **C:\DBASE**.

Once both systems have been installed, the USMS programs can be copied to the hard disk. Copy the CLIPS programs, located on the CLIPS programs floppy disk, to the **C:\CLIPSFIL** directory and copy the database programs and files, located on database disks 1 and 2, to the **C:\dBASE** directory.

ENTERING THE dBASE PORTION OF THE USMS

After the systems and programs have been installed, the USMS can be entered. The dBASE system must be entered before the USMS can be executed. dBASE automatically enters the user into *The Assistant*. In order to execute the main database program, the *Dot prompt* must be present. The *Dot prompt* can be accessed by pressing

Esc, while in *The Assistant*. Once the *Dot prompt* is present type

DO MAINMENU

which will cause the program named MAINMENU.PRG (Appendix R) to execute. The program then prompts the user for the current date. Once the date has been entered, the Main Menu (Fig. DD.1) will appear on the screen.

Main Menu

The Main Menu allows access to all programs and databases that are concerned with the database system of the USMS.

1 User Instructions

User Instructions can be seen if the number 1 is entered, from the Main Menu. The User Instructions are a brief description of and guide to the USMS (see Figs. DD.2 & DD.3). After viewing the instructions the system returns to the Main Menu (Fig. DD.1).

2 Working with the Temporary Databases

This option allows the user to edit, append, display, and delete the information for a failure site in the *temporary* databases. The Temporary Data Menu (Fig. DD.4) will then appear. See Temporary Data Menu section for further information.

****

MAIN MENU

****

1	User Instructions
2	Working with the Temporary Databases
3	Working with the Permanent Databases
4	Add an entire new site
5	Show all records
6	Create Input Files for the CLIPS programs
7	List/Print the Priority Ratings
Q	Quit

Enter Task Code

Figure DD.1 Main Menu Screen

Welcome to the wonderful world of the Unstable Slope Management System.The intent of this system is to try to provide the user with a systematic approach to the managementof slopes that are in need of repair and maintenance.

The system is composed of two software systems, dbase III+ and CLIPS.As you have already found, the system begins with the DBASE programs. Theseprograms allow you to enter and manipulate data that is to be used in the CLIPS programs. The CLIPS programs assign various priority ratings to the site so that different sites can be compared in a methodical manner.

INPUT DATA

The data for this system are divided into two different categories: Temporary and Permanent. Permanent data are information that will not, for all practical purposes, change over time (i.e. geologic information). Temporary data are information that will or may vary each time the site fails.

The first step in the system is to input the data into the appropriate databases. After the data are entered the CLIPS input files are created. DBASEcan then be exited and the CLIPS programs can be executed. After the CLIPS programs have been executed and the priorities have been determined DBASE must be reentered. The weighted priorities and total priority can then be determined for each site.

The following is an explanation of what each task choice leads to.

2. WORKING WITH THE TEMPORARY DATABASES: allows the opportunity to add, delete, edit, or display information in the following databases:

a. DAMAGE: concerned with Pavement Damage, Road Impedance, Availability of Paved Detours, Type of and Damage to a Structure near or on site, and the Lawsuit Potential.

b. FAILURE CONDITIONS: concerned with Problem Type,Water Level, Failure Date, and Failure Dimensions.

c. TEMPORARY LOADS: concerned with the presence and location of temporary static and dynamic loads.

d. REPAIR COSTS: concerned with labor costs, equipment costs, earthwork costs, repavement costs, and the Total Cost ofrepairing a site.

(continued)

Figure DD.2 USMS User Instructions

3. WORKING WITH THE PERMANENT DATABASES: allows the opportunity to add, delete, edit, or display information in the following databases:

a. IDENTITY: concerned with Road Number, Mile Post, Side of Road, District, County, Road Type, ADT, Access Type, and Population.

b. GEOLOGY: concerned with Soil Composition, Primary Soil Constituent, Secondary Soil Constituent, Orientation of Soil Layers, USCS of Top Layer, Classification of Lower Layer, Type of Loose Rock, Intact Rock, Rock Joints, Rock Layers, acceleration coefficient, and geographical hazards.

c. PERMANENT LOADS: concerned with the presence, location, and application time of static and dynamic loads.

4. ADD AN ENTIRE NEW SITE: accesses all databases in order to add one site.

5. CREATE INPUT FILES FOR THE CLIPS PROGRAMS: updates the databases concerned with the input data for the programs and converts the DBASE files to DOS files so that CLIPS can read the data. After this is done the CLIPS programs may be run.

6. LIST/PRINT THE PRIORITY RATINGS: after the CLIPS programs run, the priorty ratings may be listed or printed. The weighting factors can then be applied so that the total priority is determined. After the CLIPS programs have run the text files are converted into dbase files that are then placed into the appropriate databases.
Weight: calculates the total priority
Outtemp: shows temporary priority ratings with with print options.
Outperm: displays the permanent priority ratings with print options.

Figure DD.3 USMS User Instructions (continuation)

****
TEMPORARY DATA MENU
****
(Applies to the Cost, Damage, Failure Condition, and
Temporary Load Databases)

A	Append ALL Temporary Information Databases
D	Delete a Record from ALL Temporary Databases
E	Edit a Record in ALL Temporary Databases
1	Manipulate the Repair Cost Database
2	Manipulate the Damage Database
3	Manipulate the Failure Conditions Database
4	Manipulate the Temporary Load Database
R	Return to the MAIN MENU

Enter your choice

Figure DD.4 Temporary Menu Screen

There are four databases that are considered *temporary*. They are the Damage, the Failure Conditions, the Repair Cost, and the Temporary Load databases.

3 Working with the Permanent Databases

From the Main Menu the user can edit, add, display, or delete information in the *permanent* databases by entering the number 3. The Permanent Data Menu (Fig. DD.5) will then appear. See *Permanent* Data Menu section for further information on each database.

4 Add an entire new site

From the Main Menu an entire new failure site can be added to all of the databases by entering the number 4. A new site to the Cost, Damage, Failure Conditions, Temporary Load, Identity, Geology, and Permanent Load databases, in that order, can then be added. See the *TEMPORARY* DATABASE TYPE MENU and the *PERMANENT* DATABASE TYPE MENU sections for definitions of and proper responses to the *factors*.

5 Show all records

The user can display all of the information that is stored in the databases for a particular site by entering the number 5, while in the Main Menu. The program will then display all of the information in each database and pause after

****

PERMANENT DATA MENU

****

(Applies to the Geology, Identity, Permanent Load Databases)

A Append ALL Permanaent Information Databases
D Delete a Record from ALL Permanent Databases
E Edit a Record in ALL Permanent Databases
1 Manipulate the Identity Database
2 Manipulate the Geology Database
3 Manipulate the Permanent Load Database
R Return to the MAIN MENU

Enter your choice

Figure DD.5 Permanent Data Menu

each database. See the COST MENU section 4.4.7 for further information.

6 Creates Input Files for the CLIPS programs

Once all of the data have been entered and all necessary corrections have been made, the CLIPS input text files need to be created. By entering the number 6, the program, INPUT.PRG (see Appendix S) will begin creating several files. For each file it creates it will ask the user if it should overwrite several files. These files are the previously existing input files; if they are no longer necessary or are already saved on a floppy disk, answer Y. If the user answers N it will not overwrite the existing file. If this is the case, all programs will terminate and the user will be returned to the *Dot prompt*. The files cannot be renamed because, as the USMS exists now, the CLIPS programs will be unable to recognize the newly named files. The programs will instead read the previously existing files.

Once the files have been created, the priority ratings can be calculated. In order to this you must exit dBASE, type

QUIT,

and enter CLIPS.

7 List/Print the Priority Ratings

This option allows the user to display or print the *temporary* or *permanent priority ratings*, or calculate and print the *total priority rating*. When the number 7 is entered, the Output Menu will be displayed (Fig. DD.6). See the OUTPUT

****

OUTPUT (PRIORITY RATINGS) MENU

****

(Applies to the Temporary Priority Ratings and Permanent Priority Ratings Database)

D	Delete old data to import new data
F	Import the data from CLIPS files
1	Display/Print Temporary Priority Ratings
2	Display/Print Permanent Priority Ratings
3	Calculate & Print the TOTAL Priority Rating
R	Return to the MAIN MENU

Enter your choice

Figure DD.6 Output (Priority Ratings) Menu

MENU section for further information.

Q Quit

This command can be executed from the Main Menu and terminates the MAINMENU.PRG program. It returns the user to the Dot prompt mode of dBASE.

Temporary Data Menu

From the Temporary Data Menu each of these databases can be entered separately or all at once. The first three options, A, D, and E, allow entrance to all *temporary* databases.

A Append ALL Temporary Information Databases

By entering "A" the new information for a failure site can be added to each of the *temporary* databases. New data can be added to the Cost, Damage, Failure Conditions, and Temporary Load databases, in that order. See the COST MENU section for information on appending a database.

D Delete a Record from ALL Temporary Databases

By entering "D", the information for a failure site can be deleted from each of the *temporary* databases. As each database is entered, the user will be

asked to enter the Site ID and the failure number of the appropriate site.

E Edit a Record in ALL Temporary Databases

By entering "E" all information in the *temporary* databases can be edited for a particular failure site. The databases Cost, Damage, Failure Conditions, and Temporary Load will be accessed in the order shown. See the COST MENU section for information on editing information in a database.

Temporary Database Type Menu

The *Temporary* Database Type Menu directs the user to a *temporary* database (Cost, Damage, Failure Conditions, Temporary Load)

1 Manipulate the Repair Cost Database

If the number 1 is entered, the Cost Menu (Fig. DD.7) will be displayed. From this menu data for a failure site can be added to, deleted from, or edited in the Cost database. The data that are included within this database are the amount of man, equipment, earthwork, and repavement hours required to repair the affected road. It also includes the corresponding cost rates (cost per hour) and the total cost of repairing the road. The labor costs, equipment costs, earthwork costs, and repavement costs are calculated automatically within the ADDCOST.PRG (Appendix

****

COST MENU

****

Task Code	Task
A	Add a Record to the Cost Database
D	Delete a Record from the Cost Database
E	Edit a Record in the Cost Database
S	Display and/or print all records
R	Return to MAINMENU

Enter your choice

Figure DD.7 Cost Menu

Y) and EDITCOST.PRG (Appendix Z) programs. The information within this database is used by the CLIPS program COSTPR.CLP. See Appendix B for definitions of and proper responses to these *factors*.

2 Manipulate the Damage Database

If the number 2 is entered, the Damage Menu (Fig. DD.8) will be displayed. The data that can be entered in the Damage database are the problem type, pavement damage, road impedance, availability of paved detours, structure type, structure damage, and lawsuit damage type. This information is the input data for the CLIPS programs PAVEDAM.CLP, TRIMPEDE.CLP, STRUCTUR.CLP, and LAWSUIT.CLP. See Appendices F, Q, H, and E for definitions of and appropriate replies to these variables.

3 Manipulate the Failure Conditions Database

If the number 3 is entered, the Failure Conditions Menu (Fig. DD.9) will be displayed. The data that are contained in the Failure Conditions database are the problem type, the water level, the failure month, day, and year, and the length, depth, width, and volume of the failure. The failure date is calculated automatically. The volume is not automatically calculated because the volume of the slide may not be defined as the product of the width, depth, and length of the failure. This

****

DAMAGE MENU

****

Task Code	Task
A	Add a Record to the Damage Database
D	Delete a Record from the Damage Database
E	Edit a Record in the Damage Database
S	Display and/or print all records
R	Return to the MAIN MENU

Enter your choice

Figure DD.8 Damage Menu

****

FAILURE CONDITIONS MENU

****

Task Code	Task
A	Add a Record to the Failure Conditions Database
D	Delete a Record from the Failure Conditions Database
E	Edit a Record in the Failure Conditons Database
S	Display and/or print all records
R	Return to MAIN MENU

Enter your choice

Figure DD.9 Failure Conditions Menu

information is the input data for the CLIPS programs FAILFREQ.CLP, FSIZEPR.CLP, and PROBTYPE.CLP. See Appendices C, D, and G for definitions of and valid replies to these *factors*.

4 Manipulate the Temporary Load Database

If the number 4 is entered, the Temporary Load Menu (Fig. DD.10) will be displayed. The data that are stored in the Temporary Load database pertain to whether or not there are static and/or dynamic loads on or near the site. This information is used in the CLIPS program TEMPLOAD.CLP. See appendix I for definitions of and valid responses to these *factors*.

Permanent Data Menu

There are three databases that are considered to be *permanent* databases. These are the Identity, the Geology, and the Permanent Load database. From the Permanent Data Menu each of these databases can be entered separately or all at once. The information in these databases can be appended, edited, or deleted.

A Append ALL Permanent Information Databases

If the letter A is entered, a new failure site can be added to each of the *permanent* databases. The new information will be appended to the

****

TEMPORARY LOAD MENU

****

Task Code	Task
A	Add a Record to the Temporary Load Database
D	Delete a Record from the Temporary Load Database
E	Edit a Record in the Temporary Load Database
S	Display and/or print all records
R	Return to the MAIN MENU

Enter your choice

Figure DD.10 Temporary Load Menu

databases in the following order: Identity, Geology, Permanent Load. See the COST MENU section for further information on appending a database.

D Delete a Record from ALL Permanent Databases

If the letter D is entered, the information for a failure site can be deleted from all *permanent* databases. As each database is entered, the user will be asked to enter the Site ID of the appropriate site.

E Edit a Record in ALL Permanent Databases

If the letter E is entered, information for a specific site can be edited. The databases will be accessed in the following order: Identity, Geology, Permanent Load

Permanent Database Type Menu

The *Permanent* Database Type Menu will allow the user to select a *permanent* database type (Identity, Geology, Permanent Load).

1 Manipulate the Identity Database

If the number 1 is entered, the Identity Menu (Fig. DD.11) will be displayed. The data that are stored in the Identity database are the Road Number, Mile Post, Side of Road, District Number, County Name, Road

```
**********************************************************
                    IDENTITY MENU
**********************************************************

Task Code          Task

    A              Add a Record to the Identity Database

    D              Delete a Record from the Identity Database

    E              Edit a Record in the Identity Database

    S              Display and/or print all records

    R              Return to MAIN MENU

Enter your choice
```

Figure DD.11 Identity Menu

Type, ADT, Access Importance, and Area Population. This information, except for the specific site identity factors, are used in the CLIPS programs ADTROADT.CLP and ECONIMPO.CLP, see Appendices J and L. The identity factors are not used in any CLIPS programs, for definitions of these *factors* and proper responses see Fig. DD.12 (instructions in dbase).

2 Manipulate the Geology Database

If the number 2 is entered, the Geology Menu, (Fig. DD.13) will be displayed. The data that are stored in the Geology database pertain to several categories, soil, rock, seismicity, and geography. The *factors* in the soil category are the composition, primary constituent, secondary constituent, orientation of layers, top layer soil type, under layer soil type, and SPT. This information is used by the CLIPS program DIRT.CLP; see Appendix K for *factor* definitions. The *factors* in the rock category are the loose rock type, intact, joint orientation, and layer orientation. This information is used by the CLIPS program ROCK.CLP; see Appendix P for further information. The information included in the seismic and geography categories are the acceleration coefficient and the geographical hazard, respectively. This information is used in the CLIPS programs EQUAKE.CLP and GEOHAZ.CLP; see Appendices M and N for further information.

USER INSTRUCTIONS FOR THE IDENTITY DATABASE

Site ID: failure identification number used by the USMS to identify specific sites.
Site ID = xyyzzz
where x is the WSDOT district number
y is a single or double digit road number
z is the failure number (3 digits)
Failure Number: failure history, 1 = first failure
Road Number: State Highway Road Number
Mile Post: the mile post where the failure took place
Side of Road: direction, relative to the road, that the failure took place
(E, W, N, S)
District: WSDOT district number
County: county name
Road Type: Type of road
Interstate: federal highway
Multi-lane: more than 3 lanes
two-lane-primary:
Frontage:
Gravel:
ADT: Average Daily Traffic

ECONOMIC IMPORTANCE OF THE ROAD:
Access Type: pertains to the economic value of the road
sole-access: the road is the only access to a certain area
long-detours: if the road is closed, the area can still be accessed, but the alternate route adds many more miles and more time
no-detours-required: if the road is closed the area either economically insignificant or the detour route is about the same length and time.
Population: population of the economic area affected.

Figure DD.12 Identity Database User Instructions

****

GEOLOGY MENU

****

Task Code	Task
A	Add a Record to the Geology Database
D	Delete a Record from the Geology Database
E	Edit a Record in the Geology Database
S	Display and/or print all records
R	Return to MAIN MENU

Enter your choice

Figure DD.13 Geology Menu

3 Manipulate the Permanent Load Database

If the number 3 is entered, the Permanent Load Menu, (Fig. DD.14) will be displayed. The data that are stored in the Permanent Load Database pertains to the existence or absence of static and dynamic loads, their locations relative to the site, and their application times. This information is the input information for the CLIPS program PERMLOAD.CLP; see Appendix O for definitions of and proper responses to the *factors*.

Output Menu

The Output Menu enables the user to delete old output data, import data from CLIPS files, display and print *priority ratings*, and calculate and print *total priority ratings*.

D Delete old data to import new data

If the user wishes to import the CLIPS output text files into dBASE, the existing information in the output databases should first be deleted. This can be accomplished by entering the letter D, which executes the DELALLPR.PRG (see Appendix T) program. If the old data are not deleted, the new data from the CLIPS programs will be appended

****

PERMANENT LOAD MENU

****

Task Code	Task
A	Add a Record to the Permanent Load Database
D	Delete a Record from the Permanent Load Database
E	Edit a Record in the Permanent Load Database
S	Display and/or print all records
R	Return to the MAIN MENU

Enter your choice

Figure DD.14 Permanent Load Menu

to the existing databases. If the old and new output data are the same, the information will be duplicated.

F Import the data from CLIPS files

To import the CLIPS output data, enter the letter F. This executes the program OUTPUT.PRG (see Appendix U), which will import the text files into dBASE.

1 Display/Print Temporary Priority Ratings

2 Display/Print Permanent Priority Ratings

By entering the numbers 1 or 2, either the *temporary priority ratings* or the *permanent priority ratings* will be displayed.

3 Calculate & Print the TOTAL Priority Rating

To calculate the *total priority rating* enter the number 3. This will execute the WEIGHT.PRG (see Appendix V) program. The program will then begin multiplying the *temporary priority ratings* with the appropriate weights and then displays the results. This same procedure is repeated for the *permanent priority ratings*. It will then display and print the *total priority rating*. The *total priority rating* is the summation of these weighted priority ratings divided by 19, the number of *factors*. The *total priority rating* is a number that can theoretically range from 0 to 100, 100

being the highest priority. See Fig. DD.15 for an output example.

Cost Menu

The Cost Menu allows the user to edit, delete, edit, and display information within the Cost Database.

A Add a Record to the Cost Database

E Edit a Record in the Cost Database

Example of Appending or Editing a Database:

If the user chooses to edit or add a failure site in any of the databases the same series of events will occur. Therefore the process will be explained by using the Cost database as an example. To edit or add a record to the cost database enter the letter E or A, while in the Cost Menu. The next prompt will ask

Do you wish to see an explanation of the variables and a list of valid responses?

Please answer Y or N

If the user answers Y, the information contained in Fig. DD.16, Cost instructions, will appear on the screen and the program will pause before it continues. If the user answers N, the program will continue immediately. It is at this point that the edit and append program will differ. Unlike the append program, the edit

SITE ID	Failure No.	Sum of Temp. PR
11100	1	311.50

SITE ID	Sum of Perm. PR
11100	376.33

TOTAL PRIORITY RATING

SITE ID	Failure No.	TOTAL PR
11100	1	36.20

Figure DD.15 Example Output

USER INSTRUCTIONS FOR THE COST DATABASE

Man Hours: total amount of labor hours required to repair the site.
Equipment Hours: total amount of hours that heavy equipment was used to repair the site.
Earthwork Hours: total amount of hours required to complete all earth removal or replacement.
Repavement Hours: total amount of hours that were spent repaving the road.
Cost/hr: current rate for that particular service ($/hr)

Figure DD.16 Cost Database User Instructions

programs will ask for the Site ID number of the site the user wishes to edit. The user can then enter the number of the desired site. If a *temporary* database is being edited the user will be asked to enter the failure number of the site. The program will then display Fig. DD.17, Cost Input Format Screen. For the append program there will be no responses to the *factors*. For the edit program the responses that were previously entered for the specific site will appear. See Appendix Y for an example of an append program. See Appendix Z for an example of an edit program.

This same process is repeated for the tasks of edit and append for both, *temporary* and *permanent*, database types. The following is a list of the appropriate screens that will be displayed for the remaining databases. The valid response instructions and data input screens for each of the databases are, respectively, as follows: Failure Conditions database Figs. DD.18 and DD.19, Damage database Figs. DD.20 and DD.21, Temporary Load database Figs. DD.22 and DD.23, Identity database Figs. DD.24 and DD.25, Geology database Figs. DD.26 through DD.28, and Permanent Load database Figs. DD.29 and DD.30.

Example of Deleting a Failure Site from a Database:

D Delete a Record from the Cost Database

The procedure for the deletion of a failure site in a database is the

SITE ID 11100	Failure Number 1

Man Hours 125	cost/hr 40.00	Labor Cost 5000
Equipment Hours 90	cost/hr 200.00	Equipment Cost 18000
Earthwork Hours 60	cost/hr 200.00	Earthwork Cost 12000
Repavement Hours 30	cost/hr 500.00	Repavement Cost 15000

TOTAL COST $ 50000

Repair Proceedure: memo Press CONTROL-PGDN to enter a note,
CONTROL-PGUP to return.

Figure DD.17 Cost Input Format Screen

USER INSTRUCTIONS FOR THE FAILURE CONDITIONS DATABASE

Problem Type (failure mechanism): Rockfall, Fast-Landslide, Slow-Landslide, Fast-Debris-Flow, Slow-Debris-Flow, Settlement, Piping, Erosion, Wave-Action

Water Level: water level wihtin the slope with respect to what is normal for that particular time of year.

- Extremely High = 10
- High = 7
- Normal = 5
- Low = 3
- Extremely Low = 1

Failure Date: month: integer 1 to 12
day: integer 1 to 31
year: 19?? or 20??

Failure Dimensions: yards

Figure DD.18 Failure Conditions User Instructions

CONDITIONS of the SITE at FAILURE

SITE ID **11100** Failure Number **1**

Problem Type **Rockfall**

Water Level **10**

Failure Date: Month **3** Day **1** Year **1988**
Month+Day, yrs **0.17** (automatically calculated)

Failure Dimensions:
Length, yd **100**
Width, yd **35** Volume, cuyd **7000**
Depth, yd **2**

Figure DD.19 Failure Conditions Input Format Screen

SITE ID	11100	Failure Number 1

Severity of pavement damage due to failure mechanism:
Problem Type **Rockfall**
Pavement Damage **NONE**

Availability of Paved Detours and the severity of the road impedance:
Road Impedance **one-way-traffic** Paved Detours Available? **Yes**

Presence of Structure and damage inflicted due to failure:
Structure Type **NONE**
Structure Damage **NONE**
Comments: memo

Lawsuit Possibility Information:
Damage Type **fatality high**
Lawsuit Note: memo

Figure DD.20 Damage Input Format Screen

USER INSTRUCTIONS FOR THE DAMAGE DATABASE

Problem Type: Rockfall, Fast-Landslide, Slow-Landslide, Fast-Debris-Flow, Slow-Debris-Flow, Settlement, Piping, Erosion, Wave-Action

Pavement Damage: the corresponding priority rating can only be determined for the case of Erosion or Settlement.

- severe: dangerous [holes, drop offs,...]
- moderate: causes traffic to slow down
- low: noticeable to riders but not a danger

Road Impedance: indicates how much of the road is covered by debris from the failure.

- NONE: road is not covered at all
- Road-Closed: entire road covered or damaged
- one-way-traffic: only one direction of traffic can pass at a time
- three-quarter-half: used only if one-way-traffic does not apply. More than 1/2 and less than 3/4 of the road is covered.
- half-shoulder: used only if one-way-traffic does not apply. More than the shoulder and less than 1/2 of the road is covered.
- shoulder: only the shoulder is affected

Paved Detours available: Yes, NONE

Structure Type: structures that would be affected if the site failed.

- toxins: any structure that contains toxic/hazardous materials
- commercial-bldg: commercial buildings
- home: private residences
- RR: railroad tracks, in use
- bridge: highway or railroad bridges
- utilities: gas, water, sewer, phone

Structure Damage: amount of damage the structure sustained

- demolished: totally destroyed
- not-usable: still intact but unstable and disfunctional [liability concern?]
- unusable-3months: out of service for > 3 months
- unusable-1-3months: out of service for 1 to 3
- unusable-1month: out of service for < 1 month
- restrict-use: functional with load or use restrictions
- differential-settlement:

Lawsuit damage: fatality-high, fatality-low, toxins, property

Figure DD.21 Damage Database User Instructions

USER INSTRUCTIONS FOR THE TEMPORARY LOAD DATABASE

Static Load: static, NONE
Static Load Location:
NONE: no static load
near-site: static load located near the site
on-site: static load located on the site
on-near-site: static load located on and near the site
Dynamic Load: dynamic, NONE
Dynamic Load Location:
NONE: no dynamic load
near-site: dynamic load located near the site
on-site: dynamic load located on the site
on-near-site: dynamic load located on and near the site

Figure DD.22 Temporary Load Database User Instructions

PRESENCE OF TEMPORARY LOADS

SITE ID **11100** Failure Number **1**

Presence of a Static Load and location wrt the failure site:
Static Load **static**
Static Load Location **near-site**

Presence of Dynamic Loading and location wrt the failure site:
Dynamic Load **NONE**
Dynamic Load Location **NONE**

Figure DD.23 Temporary Load Input Format Screen

USER INSTRUCTIONS FOR THE IDENTITY DATABASE

Site ID: failure identification number used by the USMS to identify specific sites.
Site ID = xyyzzz
where x is the WSDOT district number
y is a single or double digit road number
z is the failure number (3 digits)
Failure Number: failure history, 1= first failure
Road Number: State Highway Road Number
Mile Post: the mile post where the failure took place
Side of Road: direction, relative to the road, that the failure took place
(E, W, N, S)
District: WSDOT district number
County: county name
Road Type: Type of road
Interstate: federal highway
Multi-lane: more than 3 lanes
two-lane-primary:
Frontage:
Gravel:
ADT: Average Daily Traffic

ECONOMIC IMPORTANCE OF THE ROAD:
Access Type: pertains to the economic value of the road
sole-access: the road is the only access to a certain area
long-detours: if the road is closed, the area can still be accessed, but the alternate route adds many more miles and more time
no-detours-required: if the road is closed the area either economically insignificant or the detour route is about the same length and time.
Population: population of the economic area affected.

Figure DD.24 Identity Database User Instructions

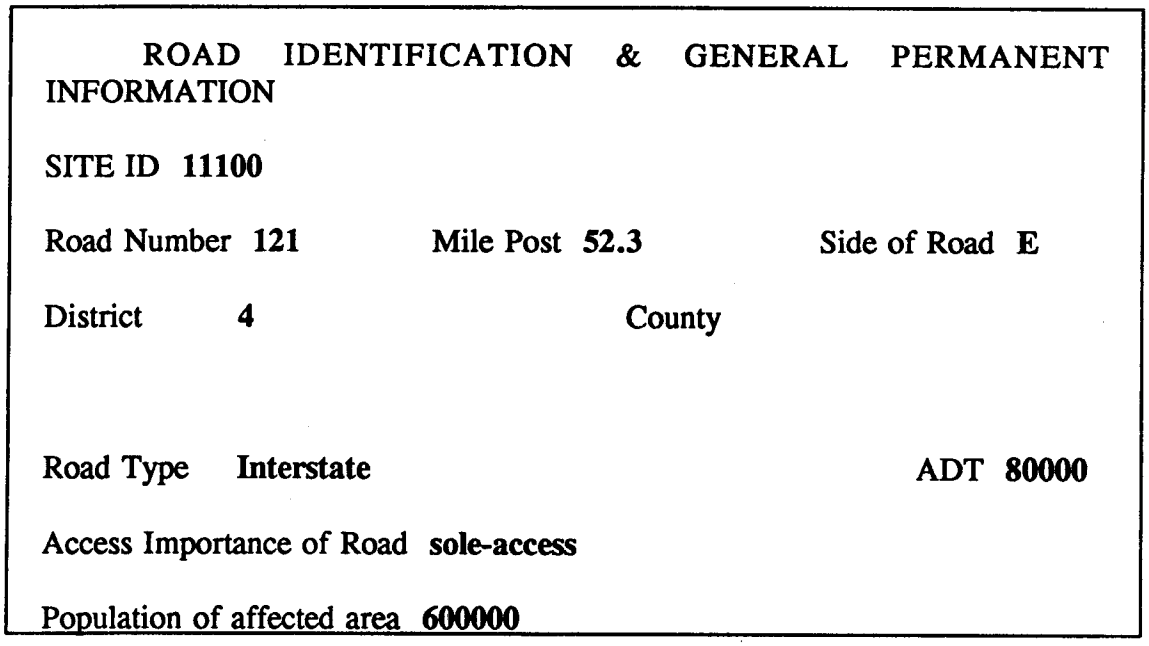

Figure DD.25 Identity Input Format Screen

USER INSTRUCTIONS FOR THE GEOLOGY DATABASE

SOIL PROPERTIES:

Overall Soil Composition:

similar: composed of 90% or more of one classification.

cohesive: 50-90% of classification is cohesive.

cohesionless: 50-90% of classification is cohesionless.

NONE:

Primary Soil Constituent: USCS classification type that the is mainly composed of.

Secondary Soil Constituent: NONE if the composition is similar otherwise use USCS classification.

Orientation of Soil Layers:

NONE: no layers

down-slope-dip: layers dip in same direction as slope dips, dips toward road.

cross-slope-dip: layers dip in opposite direction as the slope, away from the road.

horizontal:

USCS of Top Layer: USCS of top layer of soil in a two layered system. NONE if the soil is not layered.

Classification of Under Layer:

NONE: if not layered

CL, ROCK, GW, GP, GM, SW, SP, SM

ROCK PROPERTIES:

Type of Loose Rock: indicates the presence or absence of free rocks on the site.

NONE, gravel, boulders

Rock Intact?: indicates the if the rock is solid and of one type.

Yes, NONE

Orientation of Rock Joints:

NONE: if the rock is not jointed or is intact

horizontal:

vertical:

down-slope-dip: the joints dip towards the road

cross-slope-dip: the joint dip away from the road

Orientation of Rock Layers:

NONE: if the rock is not layered or is intact

horizontal:

down-slope-dip: layers dip towards the road

cross-slope-dip: layers dip away from the road

Figure DD.26 Geology Database User Instructions

(continuation of Geology Instructions)

SEISMIC INFORMATION
acceleration coefficient: from WSDOT seismicity map
GEOGRAPHICAL HAZARDS
geographical hazards:
NONE
water-body-small-1-side: water body lies on one side of road
water-body-small-2-sides:water body on both sides of road
water-body-large-1-side:
water-body-large-2-sides:
cliff-1-side: cliff on one side of road
blind-curve-2-directions:
blind-curve-1-way:

Figure DD.27 Geology Instructions (continuation)

GEOLOGICAL and GEOGRAPHICAL INFORMATION SITE ID **11100**

SOIL CLASSIFICATION

Soil Type and General Composition:
Overall Soil Composition **similar**
Primary Soil Constituent **SP** Secondary Soil Constiuent **NONE**

Type of Layering Present in Soil Structure:
Orientation of the Soil Layers **down-slope-dip**
USCS of Top Layer **SP** Classification of Under Layer **ROCK**

(a)

ROCK CLASSIFICATION

Type of Loose Rock **NONE**

Presence or absence of layering and jointing in the rock:
Is the rock intact and of one type of rock? **NONE**
Orientation of Rock Joints **down-slope-dip**
Orientation of Rock Layers **down-slope-dip**

SEISMIC INFORMATION:
Acceleration Coefficient **0.25**

(b)

Figure DD.28 (a) Geology Input Format Screen 1; (b) Geology Input Format Screen 2

USER INSTRUCTIONS FOR THE PERMANENT LOAD DATABASE

Static Load: static, NONE
Static Load Location:
  NONE: no static load
  near-site: static load located near the site
  on-site: static load located on the site
  on-near-site: static load located on and near the site
Static Application Time:
  NEW: maximum settlement has not occurred yet
  OLD: maximum settlement has occurred
Dynamic Load: dynamic, NONE
Dynamic Load Location:
  NONE: no dynamic load
  near-site: dynamic load located near the site
  on-site: dynamic load located on the site
  on-near-site: dynamic load located on and near the site
Dynamic Application Time:
  NEW: maximum settlement has not occurred yet
  OLD: maximum settlement has occurred

Figure DD.29 Permanent Load Database User Instructions

PERMANENT LOADS ON/NEAR SITE

SITE ID **11100**

Presence, Location, and Time of Application of a Static Load:

Static Load **static**

Location wrt site **on-near-site**

Application Time **NEW**

Presence, Location, and Time of Application of a DYNAMIC Load:

Dynamic Load **dynamic**

Location wrt site **on-near-site**

Application Time **NEW**

Figure DD.30 Permanent Load Input Format Screen

basically the same for all databases. To delete a specific failure site from a *temporary* database the Site ID and the failure number must be entered. To delete a specific failure site from a *permanent* database only the Site ID must be entered. See Appendix AA for an example of deletion program.

Example of Displaying a the Information in a Database:

S Display and/or print all records

This option in a database menu, for example the Cost Menu, will display all records contained in the database. The user can print this information as well. See Appendix BB for an example of this type of program.

EXECUTING CLIPS PROGRAMS

Once the DOS text files have been created from the database files, the CLIPS programs can be executed. To enter the CLIPS system, enter the **C:\CLIPS** directory and then type

CLIPS.

You are now in the CLIPS system. To run the program that controls the execution of all of the CLIPS programs, CONTROL.CLP (see Appendix A), type the command

(batch "c:\\clipsfil\\control.clp").

This must be typed in exactly as it is seen here, parenthesis included, lower case. Once

these programs have been executed you may exit CLIPS and reenter dBASE. To exit CLIPS type

(exit),

all in lower case.

PROBLEMS?

Following are a list of possible problems that could occur while using the USMS.

Missing *Priority Ratings*

If a *priority rating* is missing it is probably because the input data were misspelled or it was an invalid response. If this is the case, the CLIPS program cannot recognize the response and therefore it cannot fire any rules. This causes the program to prematurely terminate. This means that only the *priority ratings* that were calculated before the invalid data were encountered will appear in the output file. The input files are indexed on the site number and then the failure number of the site. Therefore, the site that follows the last *priority rating* in the output file is probably where the problem lies.

If you suspect a program to have terminated prematurely, you may wish to view the output files in order to discover the problem. To view a specific output file of a CLIPS program after executing the control program, exit CLIPS. You may view the output file by using DOS commands, type or print, or you may enter a text editor. See

the appropriate Appendix for the CLIPS program to get the name of the output file.

Records of Zeros

If the user chooses to append a failure site but no data are entered, zeros will be entered into the database for the responses. To delete this record of zeros, the user must enter the "delete a file" option for the appropriate database. The program will then ask the user for the Site ID; enter zero. If you are in a *temporary* database the program will ask you for the failure number; enter zero. The program then deletes the record from the database.