HSD-TR-89-010

NOISE AND SONIC BOOM IMPACT TECHNOLOGY

Initial Development of an Assessment System for Aircraft Noise (ASAN): Software Listing

Volume IV of IV Volumes

Sanford Fidell
Nicolaas Reddingius
Michael Harris
B. Andrew Kugler

BBN Systems & Technologies Corporation
21120 Vanowen Street
Canoga Park, CA 91303

June 1989

Final Report for Period February 1987 - October 1988

Approved for public release; distribution is unlimited.

19980925 000

Noise and Sonic Boom Impact Technology Program
Human Systems Division
Air Force Systems Command
Brooks Air Force Base, TX 78235-5000

DTIC QUALITY INSPECTED 1

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise as in any manner construed, as licensing the holder, or any other person or corporation; or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this report and it is releasable to the National Technical Information Service (NTIS), where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

ROBERT C. KULL, JR, Capt, USAF
NSBIT Program Manager

FOR THE COMMANDER

MICHAEL G. MACNAUGHTON, COL, USAF
Deputy Commander Development & Acquisition

Please do not request copies of this report from the Human Systems Division. Copies may be obtained from DTIC. Address your request for additional copies to:

Defense Technical Information Center
Cameron Station
Alexandria VA 22301-6145

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify HSD/SORT, Brooks AFB TX 78235-5000, to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Project 04515, Report 6624, NSBIT Task Order 0003	5. MONITORING ORGANIZATION REPORT NUMBER(S) HSD-TR-89-010 Vol. IV
6a. NAME OF PERFORMING ORGANIZATION BBN Systems & Technologies Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Advanced Development Program Office HSD/YA-NSBIT
6c. ADDRESS (City, State, and ZIP Code) 21120 Vanowen Street Canoga Park, CA 91303	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6573
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Advanced Development Program Office 8b. OFFICE SYMBOL (If applicable) HSD/YA-NSBIT	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-86-C-0530
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB OH 45433-6573	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 63723F; PROJECT NO. 3037; TASK NO. 02; WORK UNIT ACCESSION NO. 01

11. TITLE (Include Security Classification)
Initial Development of an Assessment System for Aircraft Noise (ASAN), Volume IV: Source Code Listings

12. PERSONAL AUTHOR(S)
Fidell, Sanford; Reddingius, Nicolaas; Harris, Michael, and Kugler, B. Andrew

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM 2/12/87 TO 7/31/89		276

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	source code computer program ASAN
12	07		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This is the fourth volume of a volume report summarizing the development and current contrents of a preliminary prototype version of an Assessment System for Aircraft Noise (ASAN). ASAN is a computer-based system intended to assist members of the United States Air Force (USAF) environmental planning community in addressing noise-related issues in developing environmental impact analysis documents, in compliance with USAF and other regulations, especially the National Environmental Policy Act (NEPA) of 1969.

This volume contains technical appendices and listings of the source code for the preliminary prototype version of ASAN.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert C. Kull, Captain USAF	22b. TELEPHONE (Include Area Code) (513) 255-3384 22c. OFFICE SYMBOL HSD/YA-NSBIT

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Table of Contents

APPENDIX A. SCREEN DESCRIPTION FILES 1

A.1 Screen Description File for Introductory Portions of ASAN 1
A.2 Screen Description File for MTR-related Operations 24
A.3 Screen Description File for Report-Related Portions of ASAN 102
A.4 Screen Description File for Graphic Portion of ASAN 106

APPENDIX B. PROGRAM LISTINGS 117

B.1 C Language Source Code 252

Appendix A
SCREEN DESCRIPTION FILES

This appendix contains the screen description files from which the U parser produces C code that can be compiled and linked with other object modules to produce an executable image. These files describe the appearance of the user interaction screens, and also determine what action the program takes when users elect specific actions.

A.1 Screen Description File for Introductory Portions of ASAN

```
-----------------------------------------------------------------------------
-INCLUDE STATEMENT FOR THE ASAN typedef DEFINITIONS NEEDED TO
-        COMPILE USUBS.C WITHOUT INCURRING THE WRATH OF THE COMPILER
-----------------------------------------------------------------------------

INCLUDE ASANTYPE.H
INCLUDE ASAN.H
-------------------------------------------------------------

-----------------------------------------------------------------------------
-DECLARATIONS FOR INTRODUCTORY SCREEN
-----------------------------------------------------------------------------

TEXTBLOCK introtxt2 {
   num_rows     8
   num_columns  76
   filename     txtblk/introtxt.txt
   }

WINDOW  introwindow1 {
    num_rows     10
    num_columns  78
    textblock    introtxt2, 1, 1
 }

VARIABLE animals  {
   type    INTEGER
   format  %4d
   }
   DATUM animals {
      num_rows    1
      num_columns 30
      variable    animals, 0, 1
      pickable    NO
      trailer     "animals remain in list"
      }

VARIABLE qual_entries {
   type    INTEGER
   format  %5d
   }
   DATUM qualif {
      num_rows    1
      num_columns 32
      variable    qual_entries, 0, 10
      pickable    NO
      leader      "Thus far"
      trailer     "entries qualify"
      }
   DATUM numofcit {
      num_rows    1
      num_columns 33
      variable    qual_entries, 0 , 0
      trailer     "citations meeting criteria"
```

```
        pickable    NO
        }

 VARIABLE currcit {
    type   INTEGER
    format %5d
    }
    DATUM currcit {
       num_rows    1
       num_columns 31
       variable    currcit, 0 , 22
       leader      "Display shows number"
       trailer     "of"
       pickable    NO
       }

 VARIABLE when {
    foundin "VARCHAR datap.arr"
    type   STRING
    format %4s
    }
    DATUM when {
       num_rows    1
       num_columns 16
       variable    when, 0 , 11
       leader      "Published"
       pickable    NO
       }

 TEXTLINE  cancschtxt  {"CANCEL SEARCH"}

 BUTTON    cancsch {
    num_rows    1
    num_columns 17
    textline    cancschtxt,  0,2
    helpfile    help/cnclsrch.hlp
    }


 TEXTLINE  contactxt  {"Interrogate point-of-contact database"}

 BUTTON  contacts {
     num_rows    1
     num_columns 40
     textline    contactxt, 0 ,2
     helpfile    help/contacts.hlp
  }

 TEXTLINE  humantxt  {"Interrogate human affects citation index"}

 BUTTON  human {
     num_rows    1
     num_columns 50
     textline    humantxt, 0 ,2
     helpfile    help/humcit.hlp
  }

 TEXTLINE  animaltxt  {"Interrogate animal affects citation index"}

 BUTTON  animal {
     num_rows    1
     num_columns 50
     textline    animaltxt, 0 ,2
     helpfile    help/anmlcit.hlp
  }

 TEXTLINE  structxt  {"Interrogate structural affects citation index"}

 BUTTON  struct {
     num_rows    1
     num_columns 50
     textline    structxt, 0 ,2
     helpfile    help/strctcit.hlp
  }

 TEXTLINE  sonictxt {"Interrogate noise & sonic boom modeling citation index"}

 BUTTON  sonic {
     num_rows    1
     num_columns 60
     textline    sonictxt, 0 ,2
```

```
    helpfile    help/sonic.hlp
  }

 TEXTLINE  legislatxt {"Interrogate legislative database"}

 BUTTON  legislat {
    num_rows    1
    num_columns 40
    textline    legislatxt, 0 ,2
    helpfile    help/legislat.hlp
  }


 TEXTLINE  selectdbtxt  {
 "You can now interrogate these ASAN databases:"}


 WINDOW  inquiry  {
    num_rows      10
    num_columns   78
    line          0,0,  0,77
    textline      selectdbtxt,  1,15
    button        contacts,     3,10, "CALL qsetup",
                                      "NEW_SCREEN contactscreen"
    button        human,        4,10, "CALL qsetup",
                                      "NEW_SCREEN hcitlscrn"
    button        animal,       5,10, "CALL qsetup",
                                      "NEW_SCREEN acitlscrn"
    button        struct,       6,10, "CALL qsetup",
                                      "NEW_SCREEN scitlscrn"
    button        sonic,        7,10, "CALL qsetup",
                                      "NEW_SCREEN mcitlscrn"
    button        legislat,     8,10, "NEW_SCREEN legislatscreen"
       }

 SCREEN  dbinq {
   mainscreen  YES
   border      YES
   window      inquiry,      10,1
-   window      majoraction4, 19,1
   window      introwindow1, 1, 1
   }

-----------------------------------------------------------------------------
-                  Declarations for POINT OF CONTACT DATABASE SCREEN
-----------------------------------------------------------------------------

 TEXTLINE  stsearchtxt  {"START SEARCH"}

 BUTTON  stsearch {
   num_rows    1
   num_columns 15
   textline    stsearchtxt, 0 ,2
   helpfile    help/stsearch.hlp
   }


 TEXTLINE  caff814txt  {"AFF-814 attributes"}

 BUTTON  caff814 {
   num_rows    1
   num_columns 20
   textline    caff814txt, 0 ,2
   helpfile    help/caff814.hlp
   }

 TEXTLINE  ctribaltxt  {"Tribal"}

 BUTTON  ctribal {
   num_rows    1
   num_columns 9
   textline    ctribaltxt, 0 ,2
   helpfile    help/ctribal.hlp
   }

 TEXTLINE  cmilitarytxt  {"Military"}

 BUTTON  cmilitary {
   num_rows    1
   num_columns 11
   textline    cmilitarytxt, 0 ,2
   helpfile    help/cmilitar.hlp
```

```
    }

 TEXTLINE  cfederaltxt  ("Federal")

 BUTTON  cfederal {
    num_rows     1
    num_columns 10
    textline     cfederaltxt, 0 ,2
    helpfile     help/cfederal.hlp
    }

 TEXTLINE  cstatetxt  ("State")

 BUTTON  cstate {
    num_rows     1
    num_columns 8
    textline     cstatetxt, 0 ,2
    helpfile     help/cstate.hlp
    }

 TEXTLINE  ccountytxt  ("County")

 BUTTON  ccounty {
    num_rows     1
    num_columns 9
    textline     ccountytxt, 0 ,2
    helpfile     help/ccounty.hlp
    }

 TEXTLINE  ccitytxt ("City")

 BUTTON  ccity {
    num_rows     1
    num_columns 7
    textline     ccitytxt, 0 ,2
    helpfile     help/ccity.hlp
  }

 VARIABLE address  {
      type    STRING
      format      %-25s
   }

 DATUM  address {
      num_rows    1
      num_columns     77
      variable    address, 0 ,28
      leader      "City and/or state address: "
      helpfile    help/address.hlp
   }
 VARIABLE contnam  {
      type    STRING
      format      %-15s
   }

 DATUM   contnam {
      num_rows    1
      num_columns     77
      variable    contnam, 0 ,28
      leader      "Point of contact last name:"
      helpfile    help/contnam.hlp
   }

 VARIABLE affselec {
    type    STRING
    format      %-15s
    }
    DATUM affselec {
      num_rows     1
      num_columns 50
      variable     affselec, 0 , 32
      leader       "Affiliation currently selected:"
      pickable     NO
   }

 TEXTLINE  agentxt  ("Agency:")

 TEXTLINE searchtxt  ("Search by:")

 WINDOW    contactsearch   {
        num_rows       17
        num_columns    78
```

```
        line          0,0,  0,77
        textline      searchtxt,  1, 1
        datum         contnum,    3, 1
        datum         address,    4, 1
        textline      agentxt,    5, 1
        button        cnity,      5,10,  "CALL setaff 0"
        button        ccounty,    5,18,  "CALL setaff 1"
        button        cstate,     5,27,  "CALL setaff 2"
        button        cfederal,   5,37,  "CALL setaff 3"
        button        cmilitary,  5,48,  "CALL setaff 4"
        button        ctribal,    5,60,  "CALL setaff 5"
        datum         affselec,   6, 1
        button        caff814,    9, 1,  "NEW_SCREEN contatsearch"
        button        stsearch,  16, 1,  "CALL psrch"
        button        cancsch,   16,45,  "CALL qsetup",
                                         "NEW_SCREEN dbinq"
      }


SCREEN  contactscreen {
    title       "POINT OF CONTACT DATABASE"
    window      contactsearch,    2,1
-   window      majoraction4,    19,1
    border      YES
 }

-----------------------------------------------------------------------
-           Point of contact attribute search screen
-----------------------------------------------------------------------

TEXTLINE  minoratxt  ("Minor attributes")

TEXTLINE  scrolltxt  ("Scroll window for more alternatives")

TEXTBLOCK  minorat  {
     filename       txtblk/minor.txt
     num_rows       10
     num_columns    35
     border         YES
     }

WINDOW  minorat   {
     num_rows       10
     num_columns    35
     textline       minoratxt,  1,0
     textblock      minorat,    2,0
     textline       scrolltxt,  12,0
     }

TEXTBLOCK  majorat  {
     filename   txtblk/major.txt
     num_rows   10
     num_columns    35
     border     YES
     }


VARIABLE minorat  {
     type   STRING
     format     %-45s
  }

DATUM   minorat {
     num_rows   1
     num_columns    77
     variable   minorat, 0 ,22
     leader     "Type minor attribute:"
     helpfile   help/minor.hlp
  }


VARIABLE majorat  {
     type   STRING
     format     %-15s
  }

DATUM   majorat {
     num_rows   1
     num_columns    77
     variable   majorat, 0 ,22
     leader     "Type major attribute:"
     helpfile   help/major.hlp
```

```
}

TEXTLINE  majoratxt  {"Major attributes"}

WINDOW    contatsearch   {
        num_rows       17
        num_columns    78
        line           0,0,  0,77
        datum          majorat,      1, 1  ~, "ADD_WINDOW minorat  5  41"
        datum          minorat,      2, 1  ~, "CALL dummy"
        textline       majoratxt,    4, 1
        textline       scrolltxt,   15, 1
        textblock      majorat,      5, 1
        textblock      minorat,      5,41
        button         stsearch,    16, 1,~  "REMOVE_WINDOW"
                                             "CALL pcsrch"
        button         cancsch,     16,45,  "CALL qsetup",
                                             "NEW_SCREEN dbinq"

      }


SCREEN  contatsearch {
    title        "POINT OF CONTACT ATTRIBUTE SEARCH"
    window       contatsearch,     2,1
~    window       majoraction4,    19,1
    border       YES
 }


-----------------------------------------------------------------------
~           Point of contact display screen
~
~*****NER:  all of these poc variables need to be declared in a header file
~
-----------------------------------------------------------------------

VARIABLE F_NAME {
   foundin  "VARCHAR f_name.arr"
   type                STRING
   format              %-10s
}

DATUM contactname1 {
   num_rows            1
   num_columns         25
   variable            F_NAME, 0, 15
   leader              "Contact Name:"
   pickable            NO
}

VARIABLE L_NAME {
   foundin  "VARCHAR l_name.arr"
   type                STRING
   format              %-15s
}

DATUM contactname2 {
   num_rows            1
   num_columns         30
   variable            L_NAME, 0, 1
   pickable            NO
}

VARIABLE CONTITLE { ~TITLE may be a reserved word
   foundin  "VARCHAR contitle.arr"
   type                STRING
   format              %-45s
}

DATUM contitle {
   num_rows            1
   num_columns         70
   variable            CONTITLE, 0, 9
   leader              "Title:"
   pickable            NO
}


VARIABLE OFFICE {
   foundin  "VARCHAR office.arr"
   type                STRING
```

```
    format              %-45s
}

DATUM office {
    num_rows            1
    num_columns         70
    variable            OFFICE, 0, 15
    leader              "Office:"
    pickable            NO
}

VARIABLE AGENCY_DEPT {
    foundin  "VARCHAR agency_dept.arr"
    type                STRING
    format              %-45s
}

DATUM agency {
    num_rows            1
    num_columns         70
    variable            AGENCY_DEPT, 0, 10
    leader              "Agency:"
    pickable            NO
}

VARIABLE ST_ADD_DIV {   ~***only 25 characters for a street address?
    foundin  "VARCHAR st_add_div.arr"
    type                STRING
    format              %-25s
}

DATUM st_add_div {
    num_rows            1
    num_columns         50
    variable            ST_ADD_DIV, 0, 17
    leader              "Street Address:"
    pickable            NO
}

VARIABLE PO_BOX {
    foundin  "VARCHAR po_box.arr"
    type                STRING
    format              %-10s
}

DATUM po_box {
    num_rows            1
    num_columns         40
    variable            PO_BOX, 0, 18
    leader              "Post Office Box:"
    pickable            NO
}

VARIABLE MISC_ADD {
    foundin  "VARCHAR misc_add.arr"
    type                STRING
    format              %-10s
}

DATUM misc_add {
    num_rows            1
    num_columns         20
    variable            MISC_ADD, 0,2
    pickable            NO
}

VARIABLE CITY_BASE {
    foundin  "VARCHAR city_base.arr"
    type                STRING
    format              %-25s
}

DATUM city_base {
    num_rows            1
    num_columns         30
    variable            CITY_BASE, 0, 1
    pickable            NO
}

VARIABLE  STATE {
    foundin  "VARCHAR state.arr"
    type                STRING
```

```
    format              %-2s
  }

  DATUM state {
     num_rows           1
     num_columns        5
     variable           STATE, 0, 1
     pickable           NO
  }

  VARIABLE ZIPCODE {
     foundin  "VARCHAR zipcode.arr"
     type               STRING
     format             %-9s
  }

  DATUM zipcode {
     num_rows           1
     num_columns        10
     variable           ZIPCODE, 0, 1
     pickable           NO
  }

  VARIABLE MAIL_CODE {
     foundin  "VARCHAR mail_code.arr"
     type               STRING
     format             %-10s
  }

  DATUM mail_code {
     num_rows           1
     num_columns        40
     variable           MAIL_CODE, 0, 12
     leader             "Mail Code:"
     pickable           NO
  }

  VARIABLE PHONE {
     foundin  "VARCHAR phone.arr"
     type               STRING
     format             %-20s
  }

  DATUM phone {
     num_rows           1
     num_columns        40
     variable           PHONE, 0, 20
     leader             "Telephone Number:"
     pickable           NO
  }

  VARIABLE AFFILIATIO {
     foundin  "VARCHAR affiliatio.arr"
     type               STRING
     format             %-8s
  }

  DATUM affiliatio {
     num_rows           1
     num_columns        30
     variable           AFFILIATIO, 0, 17
     leader             "Affiliation:"
     pickable           NO
  }

  VARIABLE MAJOR_ATTRIB {
     foundin  "VARCHAR major_attrib.arr"
     type               STRING
     format             %-16s
  }

  DATUM major_attribute {
     num_rows           1
     num_columns        35
     variable           MAJOR_ATTRIB, 0, 19
     leader             "Major Attribute:"
     pickable           NO
  }

  VARIABLE MINOR_ATTRIBUTE {
     foundin  "VARCHAR minor_attribute.arr"
     type               STRING
```

```
    format              %-45s
  }

  DATUM minor_attribute {
    num_rows            1
    num_columns         70
    variable            MINOR_ATTRIBUTE, 0, 19
    leader              "Minor Attribute:"
    pickable            NO
  }

  VARIABLE AREA {
    foundin  "VARCHAR area.arr"
    type                STRING
    format              %-15s
  }

  DATUM area {
    num_rows            1
    num_columns         30
    variable            AREA, 0, 7
~   leader              "AREA:"   ~(whatever that means)
    pickable            NO
  }

  VARIABLE SCOPE {
    foundin  "VARCHAR scope.arr"
    type                STRING
    format              %-10s
  }

  DATUM scope {
    num_rows            1
    num_columns         30
    variable            SCOPE, 0, 8
~   leader              "Scope:"  ~(whatever that means)
    pickable            NO
  }

-----------------------------------------------------------------------------

  WINDOW  poostuff  {
    num_rows        17
    num_columns     78
    line            0, 0, 0,77
    datum           contactname1,       1, 2
    datum           contactname2,       1,40
    datum           contitle,           2, 2
    datum           office,             3, 2
    datum           agency,             4, 2
    datum           st_add_div,         5, 2
    datum           po_box,             6, 2
    datum           misc_add,           6,25
    datum           city_base,          7, 2
    datum           state,              7,25
    datum           zipcode,            7,35
    datum           mail_code,          8, 2
    datum           phone,              9, 2
    datum           affiliatio,         9,41
    datum           major_attribute,   10, 2
    datum           minor_attribute,   11, 2
    datum           area,              12, 2
    datum           scope,             12,40
  }

TEXTLINE nextaddr {"Show next address"}

BUTTON  nextaddr {
   num_rows     1
   num_columns 25
   textline     nextaddr,  0, 2
   helpfile     help/nohelp.hlp
   }


WINDOW  getoffscreen  {
    num_rows        4
    num_columns     78
    line            0, 0, 0,77
    button          nextaddr,  2, 1, "CALL nxtpoc"
~    button          stsearch,  2, 1,  "CALL dummy"
    button          cancsch,   2,45,  "CALL qsetup",
```

```
                                        "MEN_SCREEN dbinq"
  }

 SCREEN  poodisplay {
      title          "DISPLAY POINT OF CONTACT INFORMATION"
      window          poostuff,        2,1
      window          getoffscreen,   18,1
      border          YES
      }

 ---------------------------------------------------------------------------
 -          Human effects citation database screen
 ---------------------------------------------------------------------------

 VARIABLE titlefrag {
    foundin "VARCHAR titlefrag.arr"
    type      STRING
    format    %-60s
    }
    DATUM titlefrag {
       num_rows     1
       num_columns 65
       variable    titlefrag, 0 ,0
       helpfile    help/titlfrag.hlp
       }

 TEXTLINE keywrdtxt {"Keyword categories"}

 BUTTON  keywrd  {
    num_rows      1
    num_columns  25
    textline     keywrdtxt,  0, 2
    helpfile     help/keyword.hlp
    }

 VARIABLE dates2  {
    type      INTEGER
    format    %4d
    }
    DATUM   dates2 {
       num_rows      1
       num_columns  15
       variable     dates2, 0 ,4
       leader       "and"
       trailer      "(year)"
       helpfile     help/date.hlp
       }

 VARIABLE dates1  {
    type      INTEGER
    format    %4d
    }
    DATUM   dates1 {
       num_rows      1
       num_columns  27
       variable     dates1, 0 ,14
       leader       "Date between:"
       trailer      "(year)"
       helpfile     help/date.hlp
       }

 VARIABLE authornam  {
    foundin "VARCHAR authornam.arr"
    type      STRING
    format    %-40s
    }
    DATUM   author {
       num_rows      1
       num_columns  77
       variable     authornam, 0 ,20
       leader       "Author's last name:"
       helpfile     help/author.hlp
       }

 VARIABLE citautnam1 {                          - Note: The string lengths
    foundin "VARCHAR40 authorlist[0].arr"       - are to get by the parser.
    type      STRING                            - The program will figure out
    format    %-30s                             - whether to allow 30 or 40
    }                                           -columns on the display.
    DATUM citautnam1 {
       num_rows      1
       num_columns  40
```

```
        variable      citautnam1, 0, 9
        leader        "Author:"
        pickable      NO
        }
    VARIABLE citautnam2 {
       foundin "VARCHAR40 authorlist[1].arr"
       type      STRING
       format    %-30s
       }
       DATUM citautnam2 {
          num_rows      1
          num_columns   40
          variable      citautnam2, 0, 9
          pickable      NO
          }

    VARIABLE citautnam3 {
       foundin "VARCHAR40 authorlist[2].arr"
       type      STRING
       format    %-30s
       }
       DATUM citautnam3 {
          num_rows      1
          num_columns   40
          variable      citautnam3, 0, 9
          pickable      NO
          }

    VARIABLE citautnam4 {
       foundin "VARCHAR40 authorlist[3].arr"
       type      STRING
       format    %-30s
       }
       DATUM citautnam4 {
          num_rows      1
          num_columns   40
          variable      citautnam4, 0, 9
          pickable      NO
          }

    VARIABLE citautnam5 {
       foundin "VARCHAR40 authorlist[4].arr"
       type      STRING
       format    %-30s
       }
       DATUM citautnam5 {
          num_rows      1
          num_columns   40
          variable      citautnam5, 0, 9
          pickable      NO
          }

    VARIABLE citautnam6 {
       foundin "VARCHAR40 authorlist[5].arr"
       type      STRING
       format    %-20s
       }
       DATUM citautnam6 {
          num_rows      1
          num_columns   23
          variable      citautnam6, 0, 0
          pickable      NO
          }

    VARIABLE citautnam7 {
       foundin "VARCHAR40 authorlist[6].arr"
       type      STRING
       format    %-20s
       }
       DATUM citautnam7 {
          num_rows      1
          num_columns   23
          variable      citautnam7, 0, 0
          pickable      NO
          }

    VARIABLE citautnam8 {
       foundin "VARCHAR40 authorlist[7].arr"
       type      STRING
       format    %-20s
       }
       DATUM citautnam8 {
```

```
        num_rows      1
        num_columns   23
        variable      citautnum8, 0, 0
        pickable      NO
        }

 VARIABLE citautnum9 {
    foundin "VARCHAR40 authorlist[8].arr"
    type      STRING
    format    %-20s
    }
    DATUM citautnum9 {
       num_rows      1
       num_columns   23
       variable      citautnum9, 0, 0
       pickable      NO
       }

 VARIABLE citautnum10 {
    foundin "VARCHAR40 authorlist[9].arr"
    type      STRING
    format    %-20s
    }
    DATUM citautnum10 {
       num_rows      1
       num_columns   23
       variable      citautnum10, 0, 0
       pickable      NO
       }


 TEXTLINE entitltxt  {"Title:"}

 WINDOW    hcitlscrn   {
    num_rows      16
    num_columns   78
    line          0, 0, 0,77
    textline      searchtxt,  1, 1
    datum         qualif,     1,45
    datum         author,     3, 1, "CALL VCAPITAL &authornum",
                                    "NEWVALS"
    textline      entitltxt,  4, 1
    datum         titlefrag,  4, 9, "CALL VCAPITAL &titlefrag",
                                    "NEWVALS"
    datum         dates1,     8, 1
    datum         dates2,     8,27
    button        keywrd,    10, 1, "CALL hsrch010"
    button        stsearch,  15, 1, "CALL hsrch001"
    button        cancsch,   15,45, "CALL qsetup",
                                    "NEW_SCREEN dbinq"
    }

 SCREEN  hcitlscrn {
    title       "HUMAN EFFECTS CITATION SEARCH"
    window      hcitlscrn, 2,1
-    window      majoractions3,   18,1
    border      YES
    }
-----------------------------------------------------------------------------
-         Declarations for animal effects citation database screen
-----------------------------------------------------------------------------


WINDOW    acitlscrn   {
    num_rows      16
    num_columns   78
    line          0,0,  0,77
    textline      searchtxt,  1, 1
    datum         qualif,     1,45
    datum         author,     3, 1, "CALL VCAPITAL &authornum",
                                    "NEWVALS"
    textline      entitltxt,  4, 1
    datum         titlefrag,  4, 9, "CALL VCAPITAL &titlefrag",
                                    "NEWVALS"
    datum         dates1,     8, 1
    datum         dates2,     8,27
    button        keywrd,    10, 1, "CALL asrch010"
    button        stsearch,  15, 1, "CALL asrch001"
    button        cancsch,   15,45, "CALL qsetup",
                                    "NEW_SCREEN dbinq"
    }
```

```
SCREEN  acitlscrn {
     title       "ANIMAL EFFECTS CITATION SEARCH"
     window      acitlscrn, 2,1
~     window      majoraction3,    18,1
     border      YES
     }

-----------------------------------------------------------------------
~          Declaration for structural effects citation database screen
-----------------------------------------------------------------------

WINDOW    scitlscrn   {
     num_rows       16
     num_columns    78
     line           0,0,  0,77
     textline       searchtxt,  1, 1
     datum          qualif,     1,45
     datum          author,     3, 1, "CALL VCAPITAL &authornam",
                                      "NEWVALS"
     textline       entitltxt,  4, 1
     datum          titlefrag,  4, 9, "CALL VCAPITAL &titlefrag",
                                      "NEWVALS"
     datum          dates1,     8, 1
     datum          dates2,     8,27
     button         keywrd,    10, 1, ~"CALL ssrch010"
                                      "CALL dummy"
     button         stsearch,  15, 1, "CALL ssrch001"
     button         cancsch,   15,45, "CALL qsetup",
                                      "NEW_SCREEN dbinq"
     }

SCREEN  scitlscrn {
     title       "STRUCTURAL EFFECTS CITATION SEARCH" ~nr 2/4 Corrected
     window      scitlscrn, 2,1
~     window      majoraction3,    18,1
     border      YES
     }

-----------------------------------------------------------------------
~   Declaration for noise and sonic boom modeling effects citation database
-----------------------------------------------------------------------

TEXTLINE  notimplement  {"NOT IMPLEMENTED IN PROTOTYPE VERSION OF ASAN"}

WINDOW    mcitlscrn   {
   num_rows      16
   num_columns   78
   line          0, 0, 0, 77
   textline      searchtxt,      1, 1
   datum         qualif,         1,45
   datum         author,         3, 1, "CALL VCAPITAL &authornam",
                                       "NEWVALS"
   textline      entitltxt,      4, 1
   datum         titlefrag,      4, 9, "CALL VCAPITAL &titlefrag",
                                       "NEWVALS"
   datum         dates1,         8, 1
   datum         dates2,         8,27
   button        keywrd,        10, 1, ~"CALL msrch010"
                                       "CALL dummy"
   button        stsearch,      15, 1, "CALL msrch001"
   button        cancsch,       15,45, "CALL qsetup",
                                       "NEW_SCREEN dbinq"
   }

SCREEN  mcitlscrn {
     title       "NOISE AND SONIC BOOM MODELING EFFECTS SEARCH"
     window      mcitlscrn, 2,1
~     window      majoraction3,    18,1
     border      YES
     }

-----------------------------------------------------------------------
          ~   Declaration for LEGISLATIVE DATABASE SCREEN
-----------------------------------------------------------------------

WINDOW    legislatscreen   {
     num_rows       16
```

```
      num_columns   78
      line          0,0,  0,77
      textline      searchtxt,  1, 1
      datum         qualif,     1,30
      datum         author,     3, 1,  "CALL VCAPITAL &authornam",
                                       "NEWVALS"
      textline      emtitltxt,  4, 1
      datum         titlefrag,  4, 9,  "CALL VCAPITAL &titlefrag",
                                       "NEWVALS"
      datum         dates1,     8, 1
      datum         dates2,     8,27
      button        keywrd,    10, 1,  ~"NEW_SCREEN lkeywrdschscreen"
                                       "CALL dummy"
-     textline      notimplement,   5,15
      button        stsearch, 15,1,  "CALL dummy"
      button        cancsch,  15,45, "CALL qsetup",
                                     "NEW_SCREEN dbinq"
      }

SCREEN  legislatscreen {
      title       "LEGISLATIVE CITATION DATABASE SEARCH"
      window      legislatscreen, 2,1
-     window      majoraction3,   18,1
      border      YES
      }


-------------------------------------------------------------------------------
-        Declaration for Animal effects keyword category search screen
-------------------------------------------------------------------------------

VARIABLE study2 {
   type    STRING
   format  %-20s
   }
   DATUM   study2 {
      num_rows      1
      num_columns   37
      variable      study2, 0 ,14
      leader        "Study type 2:"
      helpfile      help/study2.hlp
      }
   DATUM   study2d {
      num_rows      1
      num_columns   37
      variable      study2, 0 ,14
      leader        "Study type 2:"
      pickable      NO
      }

VARIABLE study1 {
   type    STRING
   format  %-20s
   }
   DATUM   study1 {
      num_rows      1
      num_columns   37
      variable      study1, 0 ,14
      leader        "Study type 1:"
      helpfile      help/study1.hlp
      }
   DATUM   study1d {
      num_rows      1
      num_columns   37
      variable      study1, 0 ,14
      leader        "Study type 1:"
      pickable      NO
      }

VARIABLE expdesc {
   foundin "VARCHAR expdesc.arr"
   type     STRING
   format   %-20s
   }
   DATUM expdesc {
      num_rows      1
      num_columns   40
      variable      expdesc, 0 ,19
      leader        "Experimental type:"
      helpfile      help/methodol.hlp
      }
   DATUM expdescd {
```

```
        num_rows        1
        num_columns     40
        variable        expdesc, 0 ,19
        leader          "Experimental type:"
        pickable        NO
        }

    VARIABLE noistype  {
       foundin "VARCHAR noistype.arr"
       type      STRING
       format    %-20s
       }
       DATUM   noistype {
          num_rows        1
          num_columns     37
          variable        noistype, 0 ,12
          leader          "Noise type:"
          helpfile        help/nsetype.hlp
          }
       DATUM   noistyped {
          num_rows        1
          num_columns     37
          variable        noistype, 0 ,12
          leader          "Noise type:"
          pickable        NO
          }

    VARIABLE species4  {
       foundin "VARCHAR40 species4.arr"
       type      STRING
       format    %-40s
       }
       DATUM species4  {
          num_rows        1
          num_columns     65
          variable        species4, 0 ,16
          leader          "Species type 4:"
          helpfile        help/species.hlp
          }
       DATUM species4d {
          num_rows        1
          num_columns     65
          variable        species4, 0 ,16
          leader          "Species type 4:"
          pickable        NO
          }

    VARIABLE species3  {
       foundin "VARCHAR40 species3.arr"
       type      STRING
       format    %-40s
       }
       DATUM species3  {
          num_rows        1
          num_columns     65
          variable        species3, 0 ,16
          leader          "Species type 3:"
          helpfile        help/species.hlp
          }
       DATUM species3d {
          num_rows        1
          num_columns     65
          variable        species3, 0 ,16
          leader          "Species type 3:"
          pickable        NO
          }

    VARIABLE species2  {
       foundin "VARCHAR40 species2.arr"
       type      STRING
       format    %-40s
       }
       DATUM   species2 {
          num_rows        1
          num_columns     65
          variable        species2, 0 ,16
          leader          "Species type 2:"
          helpfile        help/species.hlp
          }
       DATUM species2d {
          num_rows        1
          num_columns     65
```

```
        variable        species2, 0 ,16
        leader          "Species type 2:"
        pickable        NO
        }

 VARIABLE species1  {
    foundin "VARCHAR40 species1.arr"
    type     STRING
    format   4-40s
    }
    DATUM   species1 {
        num_rows        1
        num_columns     65
        variable        species1, 0 ,16
        leader          "Species type 1:"
        helpfile        help/species.hlp
        }
    DATUM species1d {
        num_rows        1
        num_columns     65
        variable        species1, 0 ,16
        leader          "Species type 1:"
        pickable        NO
        }

 TEXTBLOCK  animal  {
      num_rows        8
      num_columns     35
      filename        txtblk/animal.txt
      border          YES
      }

 TEXTBLOCK  var_animal  {
      num_rows        8
      num_columns     45
      filename        txtblk/varanimal.txt
      border          YES
      }

 WINDOW spec2ntry {
    num_rows      1
    num_columns   50
    datum         species2,   0, 1, "CALL VCAPITAL &species2",
                                    "NEWVALS",
                                    "CALL queryal2"
    }
 WINDOW spec3ntry {
    num_rows      1
    num_columns   50
    datum         species3,   0, 1, "CALL VCAPITAL &species3",
                                    "NEWVALS",
                                    "CALL queryal3"
    }
 WINDOW spec4ntry {
    num_rows      1
    num_columns   50
    datum         species4,   0, 1, "CALL VCAPITAL &species4",
                                    "NEWVALS",
                                    "CALL queryal4"
    }



WINDOW    akeywrdsearch   {
    num_rows      19
    num_columns   78
    line          0,0,  0,77
    datum         qualif,     1, 1
    datum         animals,    1,40
    datum         species1,   2, 1, "CALL VCAPITAL &species1",
                                    "NEWVALS",
                                    "CALL queryal1"
    datum         species2d,  3, 1
    datum         species3d,  4, 1
    datum         species4d,  5, 1
    datum         aoistype,   6, 1
    datum         expdescd,   6,38
```

```
    datum         study1d,     7, 1
    datum         study2d,     7,38
    textblock     animal,      8,23
    textline      scrolltxt, 16,23
    button        stsearch,  18, 1, "CALL asrch002"
    button        cancsch,   18,55, "CALL qsetup",
                                    "NEW_SCREEN dbinq"
    helpfile      help/zohelp.hlp
    }

 SCREEN  akeysrch {
      title       "ANIMAL EFFECTS KEYWORD SEARCH"
      window      akeywrdsearch, 2,1
      border      YES
      }
 WINDOW    akeyalt       {    - alternate used for pop-up textblock
    num_rows      19
    num_columns   78
    line          0,0,  0,77
    datum         qualif,      1, 1
    datum         animals,     1,40
    datum         species1d,   2, 1
    datum         species2d,   3, 1
    datum         species3d,   4, 1
    datum         species4d,   5, 1
    datum         noistype,    6, 1, "CALL VCAPITAL &noistype",
                                     "NEWVALS",
                                     "CALL dummy"
    datum         expdesc,     6,38, "CALL VCAPITAL &expdesc",
                                     "NEWVALS",
                                     "CALL dummy"
    datum         study1,      7, 1, "CALL dummy"
    datum         study2,      7,38, "CALL dummy"
    textblock     var_animal, 8,18
    textline      scrolltxt, 16,23
    button        stsearch,  18, 1, "CALL asrch002"
    button        cancsch,   18,55, "CALL qsetup",
                                     "NEW_SCREEN dbinq"
    helpfile      help/zohelp.hlp
    }


 SCREEN  akeyalt {
      title       "ANIMAL EFFECTS KEYWORD SEARCH"
      window      akeyalt, 2,1
      border      YES
      }

-----------------------------------------------------------------------------
-        Declaration for Human effects keyword category search screen
-----------------------------------------------------------------------------

TEXTBLOCK  exprtype  {
      num_rows       10
      num_columns    35
      filename       txtblk/methodol.txt
      border         YES
      }

WINDOW  exprtype   {
      num_rows       10
      num_columns    35
      textblock      exprtype,    0,0
      }

TEXTBLOCK  noistype  {
      num_rows       10
      num_columns    35
      filename       txtblk/nsetype.txt
      border         YES
      }

WINDOW  noistype   {
      num_rows       10
      num_columns    35
      textblock      noistype,    0,0
      }

TEXTBLOCK  hefdesc  {
      num_rows       10
      num_columns    35
      filename       txtblk/humimp.txt
```

```
    border          YES
    }

VARIABLE desctype  {
   foundin "VARCHAR desctype.arr"
   type     STRING
   format   %-45s
   }
   DATUM   desctype {
      num_rows        1
      num_columns     77
      variable        desctype, 0 ,24
      leader          "Effect descriptor type:"
      helpfile        help/humimp.hlp
      }
   DATUM   structeff {
      num_rows        1
      num_columns     77
      variable        desctype, 0 ,22
      leader          "Structural effect on:"
      helpfile        help/strimpac.hlp
      }



WINDOW    hkeywrdsearch   {
   num_rows      19
   num_columns   78
   line          0, 0, 0,77
   datum         qualif,      1,30
   datum         desctype,    2, 1, "CALL VCAPITAL &desctype",
                                    "NEWVALS",
                                    "CALL hsrch011"
   textblock     hefdesc,     6,23
   datum         noistype,    3, 1, "CALL VCAPITAL &noistype",
                                    "NEWVALS",
                                    "CALL hsrch012"
   datum         expdesc,     4, 1, "CALL VCAPITAL &expdesc",
                                    "NEWVALS",
                                    "CALL hsrch013"
   textline      scrolltxt, 16,23
   button        stsearch,   18, 1, "CALL hsrch002"
   button        cancsch,    18,45, "CALL qsetup",
                                    "NEW_SCREEN dbinq"
    }

SCREEN  hkeysrch {
     title       "HUMAN EFFECTS KEYWORD SEARCH"
     window      hkeywrdsearch, 2,1
-     window      majoraction3,   18,1
     border      YES
     }

-----------------------------------------------------------------------------
-     declaration for Structural effects keyword category search screen
-----------------------------------------------------------------------------

TEXTBLOCK  sefdesc  {
     num_rows        10
     num_columns     35
     filename        txtblk/strimpac.txt
     border          YES
     }

WINDOW    skeywrdsearch   {
   num_rows      18
   num_columns   78
   line          0, 0, 0,77
   datum         qualif,      1,30
   datum         structeff,   1, 1, "CALL VCAPITAL &desctype",
                                    "NEWVALS",
                                    "ADD_WINDOW noistype 7 24"
   textblock     sefdesc,     5,23
   datum         noistype,    2, 1, "REMOVE_WINDOW",
                                    "CALL VCAPITAL &noistype",
                                    "NEWVALS",
                                    "ADD_WINDOW exprtype  7  24",
```

```
                                        "CALL dummy"
   datum        expdesc,      3, 1, "CALL VCAPITAL &expdesc",
                                        "RENVALS",
                                        "CALL dummy"
   textline     scrolltxt,  15,23
   button       stsearch,   17, 1, "CALL dummy"
   button       cancsch,    17,45, "CALL qsetup",
                                        "NEW_SCREEN dbinq"
   }

SCREEN  skeysrch {
     title      "STRUCTURAL EFFECTS KEYWORD SEARCH"
     window     skeywrdsearch, 2,1
-     window     majorsection3,   18,1
     border     YES
     }


-----------------------------------------------------------------------------
-   Declaration for Noise & Sonic-boom effects keyword category search screen
-----------------------------------------------------------------------------

TEXTBLOCK  sbdesc  {
     num_rows       10
     num_columns    35
     filename       txtblk/sbimpac.txt
     border         YES
     }

-  VARIABLE noiseff  {
-       type           STRING
-       format         %-45s
-    }

-  DATUM   noiseff {
-       num_rows       1
-       num_columns    77
-       variable       noiseff, 0 ,32
-       leader         "Noise and sonic boom effect on:"
-       helpfile       help/nohelp.hlp
-    }


WINDOW    sbkeywrdsearch   {
     num_rows       18
     num_columns    78
     line          0,0, 0,77
-     datum         noiseff,      1,1,   "CALL dummy"
-    datum          qualif,        1,30  - ar 2/5 Added
-     textblock     sbdesc,        5,23
-     datum         noistype,      2,1,  "CALL dummy"
-     datum         expdesc,      3,1,  "CALL dummy"
     textline      notimplement,            5,20
-     textline      scrolltxt,   15,23
      button       stsearch,    17, 1, "CALL dummy"
      button       cancsch,     17,45, "CALL qsetup",
                                        "NEW_SCREEN dbinq"
     }

SCREEN  nkeysrch {
     title      "NOISE AND SONIC-BOOM EFFECTS KEYWORD SEARCH"
     window     sbkeywrdsearch, 2,1
-     window     majorsection3,   18,1
     border     YES
     }


-----------------------------------------------------------------------------
-   Declaration for LEGISLATIVE effects keyword category search screen
-----------------------------------------------------------------------------

-WINDOW    lkeywrdsearch   {
-     num_rows       18
-     num_columns    78
-     line          0,0, 0,77
-     datum         noiseff,      1,1,   "CALL dummy"
-    datum          qualif,        1,30  - ar 2/5 Added
-     datum         noistype,      2,1,  "CALL dummy"
-     datum         expdesc,      3,1,  "CALL dummy"
-     textblock     sbdesc,        5,23
-     textline      andy,            5,20
-     textline      scrolltxt,   15,23
```

```
-     button       stsearch,   17,1
-     button       cancsch,    17,45, "CALL qsetup",
-                                     "NEW_SCREEN dbinq"
-     }

-SCREEN  lkeywrdschscreen {
-     title      "LEGISLATIVE EFFECTS SEARCH BY KEYWORD"
-     window     lkeywrdsearch, 2,1
-     window     majorsection3,   18,1
-     border     YES
-     }


-----------------------------------------------------------------------------
-            CITATION DISPLAY SCREEN
-----------------------------------------------------------------------------

TEXTLINE  shwreviewtxt  ("SHOW REVIEW")

BUTTON    shwreview  {
     num_rows   1
     num_columns    15
     textline   shwreviewtxt,  0,2
     helpfile   help/citation.hlp
     }

TEXTLINE  dbinquirtxt  ("DATABASE INQUIRY SCREEN")

BUTTON    dbinquir {
     num_rows   1
     num_columns    25
     textline   dbinquirtxt,  0, 2
     helpfile   help/citation.hlp
     }

TEXTLINE  altseltxt  ("Alternative selections you can now make:")


TEXTLINE  rescopetxt  ("RESCOPE SEARCH")

BUTTON    rescope {
     num_rows   1
     num_columns    27
     textline   rescopetxt,  0, 2
     helpfile   help/rescope.hlp
     }


TEXTBLOCK  slpintf  {
     filename   blrplt/slpintf.bpl
     num_rows   16
     num_columns    76
     border     YES
     }

WINDOW  shwnxtcit   {
          num_rows      17
          num_columns   78
       textblock    slpintf,  1,1
     }

TEXTLINE  shwnxtcitxt  ("Show next citation")

BUTTON    shwnxtcit {
   num_rows    1
   num_columns 22
   textline    shwnxtcitxt,  0, 2
   helpfile    help/nohelp.hlp
   }

TEXTLINE  shwcritrvtxt  ("Show critical review (if any)")

BUTTON    shwcritrv {
     num_rows   1
     num_columns    35
     textline   shwcritrvtxt,  0, 2
     helpfile   help/nohelp.hlp
     }

TEXTLINE  shwabstxt  ("Show abstract (if any)")

BUTTON    shwabst {
```

```
        num_rows      1
        num_columns   24
        textline      shwabstxt,  0, 2
        helpfile      help/nohelp.hlp
        }

  TEXTLINE  prnthiscitxt  {"Print this citation"}

  BUTTON    prnthiscit {
        num_rows      1
        num_columns   24
        textline       prnthiscitxt,  0, 2
        helpfile    help/nohelp.hlp
        }

  TEXTLINE  prncitxt  {"PRINT ALL CITATIONS"}

  BUTTON    prncit {
        num_rows    1
        num_columns     27
        textline    prncitxt,  0, 2
        helpfile    help/nohelp.hlp
        }

  WINDOW  citdispaction  {
     num_rows       2
     num_columns    78
     button         shwnxtcit,  0, 1, "CALL shwnxtcit"
     button         shwabst,    0,40, "CALL dsplabst"
     button         shwcritrv,  1, 1, "CALL dsplcrit"
     button         prnthiscit, 1,40, "CALL dummy"
     }



  VARIABLE shwdesc1  {
     foundin "VARCHAR60 emtdesc[0].arr"
     type      STRING
     format    %-60s
    }
     DATUM shwdesc1 {
        num_rows      1
        num_columns   65
        variable      shwdesc1, 0 ,1
        pickable      NO
        }

  VARIABLE shwdesc2  {
     foundin "VARCHAR60 emtdesc[1].arr"
     type      STRING
     format    %-60s
     }
     DATUM shwdesc2 {
        num_rows      1
        num_columns   65
        variable      shwdesc2, 0 ,1
        pickable      NO
        }

  VARIABLE shwdesc3  {
     foundin "VARCHAR60 emtdesc[2].arr"
     type      STRING
     format    %-60s
     }
     DATUM shwdesc3 {
        num_rows      1
        num_columns   65
        variable      shwdesc3, 0 ,1
        pickable      NO
        }

  VARIABLE shwdesc4  {
     foundin "VARCHAR60 emtdesc[3].arr"
     type      STRING
     format    %-60s
     }
     DATUM   shwdesc4 {
        num_rows      1
        num_columns   65
```

```
        variable     shwdesc4, 0 ,1
        pickable     NO
        }


    VARIABLE suitable  {
       foundin "VARCHAR suitable.arr"
       type     STRING
       format  %1s
       }
       DATUM suitable {
          num_rows     1
          num_columns 23
          variable     suitable,  0, 21
          leader       "Suitability rating:"
          pickable     NO
          }

    VARIABLE entrynum  {
       foundin "VARCHAR enumb.arr"
       type     STRING
       format   %-6s
       }
       DATUM   entrynum {
          num_rows     1
          num_columns  30
          variable     entrynum,  0, 23
          leader       "ARAW Citation number:"
          pickable     NO
          }



    TEXTLINE shwtitxt  {"Title:"}

    WINDOW  citdisplay  {
       num_rows      17
       num_columns   78
       datum          entrynum,    0, 1
       datum          suitable,    0,32
       datum          when,        0,60
       datum          citautnam1,  2, 1
       datum          citautnam2,  3, 1
       datum          citautnam3,  4, 1
       datum          citautnam4,  5, 1
       datum          citautnam5,  6, 1
       datum          citautnam6,  2,42
       datum          citautnam7,  3,42
       datum          citautnam8,  4,42
       datum          citautnam9,  5,42
       datum          citautnam10, 6,42
       textline       shwtitxt,    8, 2
       datum          shwdesc1,    8, 9
       datum          shwdesc2,    9, 9
       datum          shwdesc3,   10, 9
       datum          shwdesc4,   11, 9
       button          shwartcit,  14, 1, "ADD_WINDOW shwartcit  4 1"  **** ??????
       button         prncit,     16, 1, "CALL dummy"
       button         rescope,    16,28, "CALL mclistc",
                                         "CALL rescsrch"
       button         cancsch,    16,60, "CALL mclistc",
                                         "CALL qsetup",
                                         "NEW_SCREEN dbinq"
       }

    VARIABLE selcrit  {
       type    STRING
       format     %-15s
       }

    DATUM selcrit {
       num_rows     1
       num_columns  70
       variable     selcrit, 0 , 22
       leader       "Selection Criterion: "
       pickable     NO
       }
```

```
WINDOW  citdisphead  {
   num_rows      3
   num_columns   78
   datum         selcrit,   0, 1
   datum         currcit,   1, 1
   datum         numofcit,  1,32
   line          2, 0, 2, 77
   }

SCREEN  citdspl {
   title   "DETAILED DISPLAY OF RETRIEVED CITATIONS"
   window  citdisphead,   2,1
   window  citdisplay,    5,1
   border  YES
   }

-----------------------------------------------------------------------------
-            ABSTRACT/CRITICAL REVIEW DISPLAY SCREEN
-----------------------------------------------------------------------------

TEXTLINE  nulltext {"        "}

WINDOW  shwartabs  {
   num_rows      1
   num_columns   15
   textline      nulltext, 0,2
   }

TEXTLINE  shwcritrv  {"Show another review (if any)"}

BUTTON    shwartrev {
     num_rows      1
     num_columns   24
     textline      shwcritrv,  0, 2
     helpfile      help/nohelp.hlp
     }

WINDOW  shwartrev  {
   num_rows      1
   num_columns   31
   button        shwartrev, 0, 1,  "REMOVE_WINDOW",
                                   "CALL dsplcrit"
   }


TEXTLINE  absdone  {"Done viewing this text"}

BUTTON    absdone {
     num_rows      1
     num_columns   29
     textline      absdone,  0, 2
     helpfile      help/nohelp.hlp
     }

TEXTBLOCK memotext {
   num_rows      12
   num_columns   74
   filename      txtblk/memotxt.txt
   }

WINDOW  absdisplay  {
   num_rows      17
   num_columns   77
   textline      shwtitxt,    1,  2
   datum         shwdesc1,    1,  9
   textblock     memotext,    3,  2
   button        absdone,     16,40,  "REMOVE_WINDOW",
                                      "NEW_SCREEN citdspl",
                                      "ADD_WINDOW citdispaction 19 1"
   }

SCREEN  showabs {
   title   "DISPLAY ABSTRACTS AND CRITICAL REVIEWS"
   window  citdisphead,   2,1
   window  absdisplay,    5,1
   border  YES
   }

-END OF ASARCIT.SDF
```

A.2 Screen Description File for MTR-related Operations

```
  ----------------------------------------------------------------------
-INCLUDE STATEMENT FOR THE ASAN typedef DEFINITIONS NEEDED TO
-        COMPILE USUBS.C WITHOUT INCURRING THE WRATH OF THE COMPILER
-----------------------------------------------------------------------

INCLUDE ASANTYPE.H
INCLUDE ASAN.H

-----------------------------------------------------------------------
-  DECLARATIONS IN NON-LEXICAL OBJECT SEQUENCE ................
-----------------------------------------------------------------------

- Multiple Choice Space
-----------------------

VARIABLE  muldat00  {
   foundin "VARCHAR30 displmult[0].arr"
   type     STRING
   format   %-30s
   }

DATUM  muldat00  {
   num_rows    1
   num_columns 32
   variable    muldat00, 0, 2
   pickable    NO
   }

VARIABLE  muldat01 {
   foundin "VARCHAR30 displmult[1].arr"
   type     STRING
   format   %-30s
   }

DATUM  muldat01  {
   num_rows    1
   num_columns 32
   variable    muldat01, 0, 2
   pickable    NO
   }

VARIABLE  muldat02  {
   foundin "VARCHAR30 displmult[2].arr"
   type     STRING
   format   %-30s
   }

DATUM  muldat02 {
   num_rows    2
   num_columns 32
   variable    muldat02, 0, 2
   pickable    NO
   }

VARIABLE  muldat03  {
   foundin "VARCHAR30 displmult[3].arr"
   type     STRING
   format   %-30s
   }

DATUM  muldat03  {
   num_rows    1
   num_columns 32
   variable    muldat03, 0, 2
   pickable    NO
   }

VARIABLE  muldat04  {
   foundin  "VARCHAR30 displmult[4].arr"
   type     STRING
   format    %-30s
   }

DATUM  muldat04  {
   num_rows    1
   num_columns 32
```

```
    variable    muldat04, 0, 2
    pickable    NO
    }

 VARIABLE muldat05  {
    foundin  "VARCHAR30 displmult[5].arr"
    type     STRING
    format   %-30s
    }

 DATUM  muldat05  {
    num_rows    1
    num_columns 32
    variable    muldat05, 0, 2
    pickable    NO
    }

 VARIABLE muldat06  {
    foundin "VARCHAR30 displmult[6].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat06 {
    num_rows    1
    num_columns 32
    variable    muldat06, 0, 2
    pickable    NO
    }

 VARIABLE muldat07  {
    foundin "VARCHAR30 displmult[7].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat07 {
    num_rows    1
    num_columns 32
    variable    muldat07, 0, 2
    pickable    NO
    }

 VARIABLE muldat08  {
    foundin "VARCHAR30 displmult[8].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat08 {
    num_rows    1
    num_columns 32
    variable    muldat08, 0, 2
    pickable    NO
    }

 VARIABLE muldat09  {
    foundin "VARCHAR30 displmult[9].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat09 {
    num_rows    1
    num_columns 32
    variable    muldat09, 0, 2
    pickable    NO
    }

 VARIABLE muldat10  {
    foundin "VARCHAR30 displmult[10].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat10 {
    num_rows    1
    num_columns 32
    variable    muldat10, 0, 2
    pickable    NO
    }
```

```
VARIABLE muldat11  {
   foundin "VARCHAR30 dsplmult[11].arr"
   type     STRING
   format   %-30s
   }

DATUM   muldat11 {
   num_rows    1
   num_columns 32
   variable    muldat11, 0, 2
   pickable    NO
   }

VARIABLE muldat12  {
   foundin "VARCHAR30 dsplmult[12].arr"
   type     STRING
   format   %-30s
   }

DATUM   muldat12  {
   num_rows    1
   num_columns 32
   variable    muldat12, 0, 2
   pickable    NO
   }

VARIABLE muldat13  {
   foundin "VARCHAR30 dsplmult[13].arr"
   type     STRING
   format   %-30s
   }

DATUM   muldat13    {
   num_rows    1
   num_columns 32
   variable    muldat13, 0, 2
   pickable    NO
   }

VARIABLE muldat14  {
   foundin "VARCHAR30 dsplmult[14].arr"
   type     STRING
   format   %-30s
   }

DATUM   muldat14 {
   num_rows   1
   num_columns    32
   variable   muldat14, 0, 2
   pickable   NO
   }

VARIABLE muldat15  {
   foundin "VARCHAR30 dsplmult[15].arr"
   type    STRING
   format  %-30s
   }

DATUM   muldat15 {
   num_rows    1
   num_columns 32
   variable    muldat15, 0, 2
   pickable    NO
   }

VARIABLE muldat16  {
   foundin "VARCHAR30 dsplmult[16].arr"
   type    STRING
   format  %-30s
   }

DATUM   muldat16 {
   num_rows    1
   num_columns 32
   variable    muldat16, 0, 2
   pickable    NO
   }

VARIABLE muldat17  {
   foundin "VARCHAR30 dsplmult[17].arr"
   type    STRING
   format  %-30s
```

```
    }

 DATUM   muldat17 {
    num_rows     1
    num_columns 32
    variable     muldat17, 0, 2
    pickable     NO
    }

 VARIABLE muldat18  {
    foundin "VARCHAR30 dsplmult[18].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat18 {
    num_rows     1
    num_columns 32
    variable     muldat18, 0, 2
    pickable     NO
    }

 VARIABLE muldat19  {
    foundin "VARCHAR30 dsplmult[19].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat19 {
    num_rows     1
    num_columns 32
    variable     muldat19, 0, 2
    pickable     NO
    }

 - Planner attributes
 --------------------

 VARIABLE planrnam {
    foundin "VARCHAR planrnam.arr"
    type      STRING
    format    %-30s
 }

 DATUM planrnam {
    num_rows     1
    num_columns 50
    variable     planrnam, 0, 20
    leader       "Your name, please:"
    helpfile     help/username.hlp
 }

 VARIABLE password {
    type     STRING
    format   %-10s
 }

 DATUM password {
    num_rows     1
    num_columns 55
    variable     password, 0 , 23
    leader       "Please enter password:"
    helpfile     help/password.hlp
 }

 - Other General Storage for Verification, Tests, Etc.
 - ----------------------------------------------------

 VARIABLE  newstrnm {
      foundin        "VARCHAR a2bv.arr"
      type           STRING
      format         %-30s
 }

 - Noise Source Attributes
 - -------------------------

 VARIABLE  srcid {
    foundin "VARCHAR srcid.arr"
    type      STRING
    format    %-9s
    }
```

```
DATUM   mtrnam {
   num_rows       1
   num_columns    60
   variable       srcid, 0, 21
   leader         "Name of current MTR:"
   pickable       NO
   }

VARIABLE  srcdesc {
   foundin "VARCHAR srcdesc.arr"
   type      STRING
   format    4-54s
   }

DATUM   srcds {
      num_rows       1
      num_columns    76
      variable       srcdesc, 0, 22
      leader         "Description:          "
      helpfile       help/mtrdesc.hlp
      }

DATUM   mtrdesc {
      num_rows       1
      num_columns    70
      variable       srcdesc, 0, 7
      leader         "Note: "
      pickable       NO
      }

VARIABLE  datepubl   {
     foundin  "VARCHAR srcpdate.arr"
     type      STRING
     format    4-12s
}

VARIABLE  schedule   {
   foundin  "VARCHAR srcsched.arr"
   type      STRING
   format    4-50s
   }

DATUM   schedule {
      num_rows     1
      num_columns 73
      variable     schedule, 0, 22
      leader       "Scheduling activity: "
      helpfile     help/schdmtr.hlp
      }

VARIABLE  srcorig   {
   foundin   "VARCHAR srcorig.arr"
   type      STRING
   format    4-50s
   }

DATUM   origmtr {
   num_rows  1
   num_columns   73
   variable      srcorig, 0, 22
   leader        "Originating activity:"
   helpfile      help/origmtr.hlp
   }

~ MTR Attributes
~ ---------------

VARIABLE  curartcc {                          ~ CURRENT Set
   foundin "VARCHAR curartcc.arr"
   type      STRING
   format    43s
   }

DATUM curartcc {
      num_rows       1
      num_columns    27
      variable       curartcc,  0,16
      leader         "ARTCC:          "
      helpfile       help/curartcc.hlp
      }

VARIABLE  curwidright {
```

```
    type   INTEGER
    format %2d
    }

 DATUM   curwidright {
       num_rows       1
       num_columns    25
       variable       curwidright,  0,16
       leader         "Width (right): "
       helpfile       help/curwidrt.hlp
       }

 VARIABLE  curwidleft {
    type   INTEGER
    format %2d
    }

 DATUM   curwidleft {
       num_rows       1
       num_columns    25
       variable       curwidleft,  0,16
       leader         "Width (left):  "
       helpfile       help/curwidlf.hlp
      }

 VARIABLE  curhighalt {
    foundin   "ALTSPEC curhighalt.spec"
    type       STRING
    format     %9s
    }

 DATUM   curhighalt {
       num_rows       1
       num_columns    27
       variable       curhighalt, 0,16
       leader         "High altitude: "
       helpfile       help/curhialt.hlp
       }

 VARIABLE  curlowalt {
      foundin   "ALTSPEC curlowalt.spec"
      type       STRING
      format     %9s
}

 VARIABLE  preartcc {                          - PREVIOUS Set
    foundin "VARCHAR preartcc.arr"
    type      STRING
    format    %3s
    }

    DATUM preartcc {
       num_rows       1
       num_columns    8
       variable       preartcc,  0,4
       pickable       NO
       }

VARIABLE  prewidright {
   type   INTEGER
   format %2d
   }

 DATUM  prewidright {
      num_rows       1
      num_columns    8
      variable       prewidright,  0,4
      pickable       NO
      }

VARIABLE  prewidleft {
   type           INTEGER
   format         %2d
   }

 DATUM  prewidleft {
      num_rows       1
      num_columns    8
      variable       prewidleft,  0,4
      pickable       NO
      }
```

```
VARIABLE  prehighalt {
   foundin "ALTSPEC prehighalt.spec"
   type     STRING
   format   49s
   }

DATUM  prehighalt {
      num_rows       1
      num_columns    10
      variable       prehighalt, 0,1
      pickable       NO
      }

VARIABLE  prelowalt {
     foundin    "ALTSPEC prelowalt.spec"
     type        STRING
     format      49s
}

- Coordinates
- ------------

VARIABLE emtlat  {                              - CURRENT or ENTER Set
     foundin     "COORDINATE emt.lat"
     type        STRING
     format      4-13s
}

VARIABLE emtlong  {
     foundin     "COORDINATE emt.lon"
     type        STRING
     format      4-13s
}

VARIABLE shwlat  {                              - PREVIOUS or SHOW Set
     foundin     "COORDINATE show.lat"
     type        STRING
     format      4-13s
}

VARIABLE shwlong  {
     foundin     "COORDINATE show.lon"
     type        STRING
     format      4-13s
}

- Navigation Point Properties
- ----------------------------

VARIABLE  curfixtyp {                           - CURRENT or ENTER Set
     foundin    "VARCHAR curfixtyp.arr"
     type        STRING
     format      412s
}

VARIABLE  curfixdist {
     type           INTEGER
     format         43d
}

VARIABLE  curfixrad {
     type           INTEGER
     format         43d
}

VARIABLE  curfixid {
     foundin     "VARCHAR curfixid.arr"
     type           STRING
     format         45s
}

VARIABLE  curnavpt {
     foundin     "VARCHAR curnavpt.arr"
     type           STRING
     format         43s
}

VARIABLE  prefixtyp {                           - PREVIOUS or DISPLAY Set
     foundin    "VARCHAR prefixtyp.arr"
     type        STRING
     format      412s
}
```

```
VARIABLE  prefixdist {
     type          INTEGER
     format        %3d
}

VARIABLE  prefixrad {
     type          INTEGER
     format        %3d
}

VARIABLE  prefixid {
     foundin     "VARCHAR prefixid.arr"
     type          STRING
     format        %5s
}

VARIABLE  premavpt {
     foundin     "VARCHAR premavpt.arr"
     type          STRING
     format        %3s
}

- Aircraft Parameters
- --------------------

VARIABLE  acname  {
   foundin  "VARCHAR ac_name.arr"
   type     STRING
   format   %-12s
   }

DATUM   acname {
      num_rows  1
      num_columns  25
      variable     acname, 0, 10
      leader       "Aircraft:"
      pickable     NO
      }

VARIABLE  acnamel  {
   foundin "VARCHAR tid.arr"
   type     STRING
   format   %-12s
   }

DATUM   acnamel {
      num_rows  1
      num_columns 35
      variable    acnamel, 0, 16
      leader      "Aircraft name:"
      helpfile    help/mtraircr.hlp
  }

- Operations Specification
- -------------------------

VARIABLE   day  {                          - GENERIC or JANUARY
     foundin   "OPERATIONS ops[0].day"
     type      INTEGER
     format        %4d
}

VARIABLE   janday  {
     foundin   "OPERATIONS ops[0].day"
     type      INTEGER
     format        %4d
}

VARIABLE   night  {
     foundin   "OPERATIONS ops[0].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   jannite  {
     foundin   "OPERATIONS ops[0].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   febday  {
     foundin   "OPERATIONS ops[1].day"
```

```
        type        INTEGER
        format        %4d
    }

    VARIABLE   febnite  {
        foundin   "OPERATIONS ops[1].nite"
        type      INTEGER
        format        %4d
    }

    VARIABLE   marday  {
        foundin   "OPERATIONS ops[2].day"
        type      INTEGER
        format        %4d
    }

    VARIABLE   marnite  {
        foundin   "OPERATIONS ops[2].nite"
        type      INTEGER
        format        %4d
    }

    VARIABLE   aprday  {
        foundin   "OPERATIONS ops[3].day"
        type      INTEGER
        format        %4d
    }

    VARIABLE   apnite  {
        foundin   "OPERATIONS ops[3].nite"
        type      INTEGER
        format        %4d
    }

    VARIABLE   mayday  {
        foundin   "OPERATIONS ops[4].day"
        type      INTEGER
        format        %4d
    }

    VARIABLE   maynite  {
        foundin   "OPERATIONS ops[4].nite"
        type      INTEGER
        format        %4d
    }

    VARIABLE   junday  {
        foundin   "OPERATIONS ops[5].day"
        type      INTEGER
        format        %4d
    }

    VARIABLE   junnite  {
        foundin   "OPERATIONS ops[5].nite"
        type      INTEGER
        format        %4d
    }

    VARIABLE   julday  {
        foundin   "OPERATIONS ops[6].day"
        type      INTEGER
        format        %4d
    }

    VARIABLE   julnite  {
        foundin   "OPERATIONS ops[6].nite"
        type      INTEGER
        format        %4d
    }

    VARIABLE   augday  {
        foundin   "OPERATIONS ops[7].day"
        type      INTEGER
        format        %4d
    }

    VARIABLE   augnite  {
        foundin   "OPERATIONS ops[7].nite"
        type      INTEGER
        format        %4d
    }
```

```
VARIABLE   sepday  {
     foundin   "OPERATIONS ops[8].day"
     type      INTEGER
     format        %4d
}

VARIABLE   sepnite  {
     foundin   "OPERATIONS ops[8].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   octday  {
     foundin   "OPERATIONS ops[9].day"
     type      INTEGER
     format        %4d
}

VARIABLE   octnite  {
     foundin   "OPERATIONS ops[9].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   novday  {
     foundin   "OPERATIONS ops[10].day"
     type      INTEGER
     format        %4d
}

VARIABLE   novnite  {
     foundin   "OPERATIONS ops[10].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   decday  {
     foundin   "OPERATIONS ops[11].day"
     type      INTEGER
     format        %4d
}

VARIABLE   decnite  {
     foundin   "OPERATIONS ops[11].nite"
     type      INTEGER
     format        %4d
}

- Missions
- ---------

VARIABLE   misname  {
   foundin "VARCHAR misslabl.arr"
   type    STRING
   format  %-7s
   }

DATUM  misnamel {
      num_rows     1
      num_columns  24
      variable     misname, 0, 26
      leader       "Name of current mission: "
      pickable     NO
      }

VARIABLE   newmisname  {
      foundin      "VARCHAR cid.arr"
      type         STRING
      format       %-7s
      }

DATUM  misname  {
      num_rows     1
      num_columns  24
      variable     newmisname, 0, 14
      leader       "Mission name:"
      helpfile     help/missname.hlp
     }

TEXTLINE select  {
   "Right now you can type ? for help, <CTRL> C to quit, or move the cursor")
TEXTLINE canvass   ("CONDUCT AN ENVIRONMENTAL ASSESSMENT" )
```

```
TEXTLINE housekp  {"PERFORM DATABASE HOUSEKEEPING" }
TEXTLINE introtxt {"VIEW GENERAL INFORMATION ABOUT THIS PROGRAM"}

TEXTLINE selacmistxt   {"Select aircraft and mission for MTR"}
TEXTLINE modcurmtrtxt {"Modify current MTR"}
TEXTLINE chgmtrtxt     {"Select another MTR"}

TEXTLINE nextnavptxt  {"Enter next navigation point"}
TEXTLINE curprotxt
{ "CURRENT      PREVIOUS                    CURRENT      PREVIOUS"}
TEXTLINE curprotxt3
{ "-------      --------                    -------      ---------"}
TEXTLINE cancelmtrtxt {"Cancel this MTR data entry"}
TEXTLINE savemtrtxt   {"Save this MTR"}

TEXTLINE showmoremtrtxt {"Show more MTR names (if any)"}
TEXTLINE recallmtrtxt   {"Recall one of the following MTRs:"}
TEXTLINE stnewmtrtxt    {"Start new MTR "}
TEXTLINE definemtrtxt  {"Enter route waypoints"}

TEXTLINE actxt  {"Aircraft:"}

TEXTLINE opseasontxt {"Operations are seasonal"}
TEXTLINE opthruyrtxt {"Operations are even throughout year"}
TEXTLINE instritxt   {"Please enter day and night operations by month"}
TEXTLINE dayaite     {" DAY  NIGHT  DAY  NIGHT  DAY  NIGHT  DAY  NIGHT"}
TEXTLINE instr2txt   {"Please enter daytime and night operations per month"}
TEXTLINE cancmistxt  {"Abandon this mission"}
TEXTLINE entmisdbtxt {"Enter this mission into database"}
TEXTLINE stnewmistxt  {"Start new mission"}

TEXTBLOCK  introtxt  {
    num_rows     14
    num_columns  76
    border       YES
    filename     txtblk/intro.txt
 }

BUTTON  assessment   {
   num_rows     1
   num_columns  38
   textline     envass, 0, 2
   helpfile     help/assess.hlp
   }

BUTTON  chgmtr {
   num_rows       1
   num_columns    35
   textline       chgmtrtxt,  0, 2
   helpfile       help/selmtr.hlp
   }

BUTTON  housekeeping {
   num_rows     1
   num_columns  33
   textline     housekp, 0, 2
   helpfile     help/housekp.hlp
   }

BUTTON  introtxt {
   num_rows     1
   num_columns  56
   textline     introtxt, 0, 2
   helpfile     help/introhel.hlp
   }

BUTTON  modcurmtr {
   num_rows    1
   num_columns 35
   textline    modcurmtrtxt,  0, 2
   helpfile    help/nohelp.hlp
      - no help here cuz this button doesn't do anything
   }

WINDOW password {
    num_rows     1
    num_columns  60
    datum        password, 0, 5
   }

TEXTLINE titltxt
{"Developed for Noise and Sonic Boom Impact Technology Program"}
```

```
TEXTLINE tit2txt {"under U.S. Air Force Contract F33615-86-C-0530"}

TEXTLINE tit3txt {"by BBN Laboratories, Inc."}

TEXTLINE tit4txt  {"February, 1988"}

TEXTLINE tit5txt
{"Unreleased demonstration of Prototype Version...Not for General Use"}

TEXTLINE diafo {"Done viewing general information on ASAN"}

BUTTON diafo {
    num_rows    1
    num_columns 76
    textline    diafo,  0, 2
}

WINDOW  introduction  {
    num_rows     15
    num_columns  78
    textblock    introtxt,   0, 1
    button       diafo, 14, 1, "REMOVE_WINDOW"
 }

WINDOW  introwindow1 {
    num_rows     21
    num_columns  78
    line         1, 0, 1, 77
    title        "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAN)"
    textline     tit1txt, 2, 9
    textline     tit2txt, 3, 16
    textline     tit3txt, 4, 27
    textline     tit4txt,  5, 32
    textline     tit5txt,  7, 6
    line         8,0, 8,77
    datum        plasram, 12,  1,
                 "CALL VCAPITAL &plasram",
                 "ADD_WINDOW password 15 3",
                 "UPDATE_DATUM password",
                 "REMOVE_WINDOW",
                 "CALL pwcheck"
}

WINDOW  introwindow2  {
    num_rows     9
    num_columns  78
    line         5, 0, 5, 77
    textline     select,        6, 3
    button       introtxt,      7,15, "ADD_WINDOW introduction 3 1"
    button       assessment,    8, 2, "CALL peprobst"
    button       housekeeping, 8,44, "CALL pedbhsek"
}

SCREEN  firstscreen {
   titlescreen  YES
   border       YES
   title        "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAN)"
   window       introwindow1,  1,1
   window       introwindow2, 13,1
   }

BUTTON   opseason {
   num_rows     3
   num_columns 41
   textline     opseasontxt,  1, 3
   border       YES
   helpfile     help/vrblops.hlp
   }

BUTTON   opsteady {
   num_rows     3
   num_columns 41
   textline     opthruyrtxt,  1, 3
   border       YES
   helpfile     help/evenops.hlp
   }

BUTTON mulbut00  {
   num_rows     1
   num_columns 1
   helpfile     help/mulbut.hlp
   }
```

```
BUTTON mulbut01 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut02 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut03 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut04 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut05 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut06 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut07 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut08 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut09 {
   num_rows    1
   num_columns 1
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut10 {
   num_rows    1
   num_columns 2
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut11 {
   num_rows    1
   num_columns 2
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut12 {
   num_rows    1
   num_columns 2
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut13 {
   num_rows    1
   num_columns 2
   helpfile    help/mulbut.hlp
   }

BUTTON mulbut14 {
   num_rows    1
   num_columns 2
   helpfile    help/mulbut.hlp
```

```
    }

 BUTTON  mulbut15  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
    }

 BUTTON  mulbut16  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
   }

 BUTTON  mulbut17  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
   }

 BUTTON  mulbut18  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
   }

 BUTTON  mulbut19  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
   }

--------------------------------------------------------------------------------
-         Declarations for MTR DATA ENTRY SCREEN
--------------------------------------------------------------------------------

 BUTTON    stnewmis {
      num_rows        1
      num_columns     40
      textline        stnewmistxt, 0 ,2
      helpfile        help/stnewmis.hlp
     }

 BUTTON   selacmis {
      num_rows        1
      num_columns     40
      textline        selacmistxt,  0, 2
      helpfile        help/selacm.hlp
      }

TEXTLINE  specnwmistxt   {"Specify new mission"}

 BUTTON   specnwmis {
      num_rows        1
      num_columns     30
      textline        specnwmistxt,  0, 2
      helpfile        help/newmiss.hlp
      }

WINDOW  mtrdataentry   {
   num_rows     8
   num_columns 78
   button       chgmtr,     0, 2, "CALL pschgmtr"
   button       selacmis,   2, 2, "CALL pmtrmis"
   button       modcurmtr, 4, 2, "CALL dummy"
   button       specnwmis, 6, 2, "ADD_WINDOW newmisnm  13 3",
                                 "UPDATE_DATUM misname",
                                 "CALL WHERENAMI Screen Window Datum Button",
                                 "CALL strcpy oldscreen Screen",
                                 "CALL psnwmisn cid.arr"

   }

WINDOW  mtrnamcom   {
   num_rows       3
   num_columns    78
   line           2,0, 2,77
   datum          mtrnam,  0, 2
   datum          mtrdesc, 1, 2
   }

WINDOW  mtrnamcoml   {
```

```
    num_rows       4
    num_columns    78
    line           3,0, 3,77
    datum          mtrnam,  0, 2
    datum          mtrdesc, 1, 2
    datum          misnamel,  2, 2
    }

-----------------------------------------------------------------------------
~         Declarations for DEFINE/MODIFY SCREEN   (defmodmtrscreen)
-----------------------------------------------------------------------------

DATUM   prelowalt {
     num_rows       1
     num_columns    10
     variable       prelowalt, 0,1
     pickable       NO
  }

DATUM   curlowalt {
     num_rows       1
     num_columns    29
     variable       curlowalt,   0, 16
     leader         "Low altitude:  "
     helpfile       help/curloalt.hlp
  }

DATUM   prefixtype {
     num_rows       1
     num_columns    13
     variable       prefixtyp,  0,0
     pickable       NO
  }

DATUM   curfixtype {
     num_rows       1
     num_columns    24
     variable       curfixtyp,  0,11
     leader         "Fix type: "
     helpfile       help/curfixtp.hlp
  }

-----------------------------------------------------------------------------
~~          Declarations for entering coordinates
-----------------------------------------------------------------------------

DATUM   entlong {
     num_rows   1
     num_columns    25
     variable   entlong, 0 ,11
     leader     "Longitude:"
     helpfile   help/markmap.hlp
  }

DATUM   entlat {
     num_rows   1
     num_columns    25
     variable   entlat, 0 ,11
     leader     "Latitude: "
     helpfile   help/markmap.hlp
  }

-----------------------------------------------------------------------------
~          Declarations for show coordinates
-----------------------------------------------------------------------------

DATUM   shwlat {
     num_rows   1
     num_columns    14
     variable   shwlat, 0 ,0
     pickable   NO
     helpfile   help/combmap.hlp
  }

DATUM   shwlong {
     num_rows   1
     num_columns    14
     variable   shwlong, 0 ,0
     pickable   NO
     helpfile   help/combmap.hlp
  }
```

```
DATUM   prefixdist {
     num_rows       1
     num_columns    8
     variable       prefixdist,  0,4
     pickable       NO
  }

DATUM   curfixdist {
     num_rows       1
     num_columns    26
     variable       curfixdist,  0,16
     leader         "Fix distance: "
     helpfile       help/curfixdi.hlp
  }

DATUM   prefixrad {
     num_rows       1
     num_columns    8
     variable       prefixrad,  0,4
     pickable       NO
  }

DATUM   curfixrad {
     num_rows       1
     num_columns    26
     variable       curfixrad,  0,16
     leader         "Fix radial:   "
     helpfile       help/curfixra.hlp
  }

DATUM   prefixid {
     num_rows       1
     num_columns    10
     variable       prefixid, 0,4
     pickable       NO
  }

DATUM   curfixid {
     num_rows  1
     num_columns    26
     variable       curfixid, 0,16
     leader         "Fix ID:       "
     helpfile       help/curfixid.hlp
  }

DATUM   prenavpt {
     num_rows       1
     num_columns    10
     variable       prenavpt, 0,4
     pickable       NO
}

DATUM   curnavpt {
     num_rows       1
     num_columns    24
     variable       curnavpt,   0, 16
     leader         "Nav. Point:   "
     helpfile       help/curnavpt.hlp
  }

DATUM   newmtrnm  {
     num_rows       1
     num_columns    40
     variable        newmtrnm,   0,7
     leader         "Name:"
     helpfile       help/mtrmtrnm.hlp
  }

BUTTON   nextnavpt {
     num_rows   1
     num_columns    35
     textline   nextnavptxt,  0, 2
     helpfile   help/nxtnav.hlp
     }
```

```
TEXTLINE  retmtrentxt    {"Continue without selecting MTR"}
TEXTLINE  retmtrentxt1   {"Continue without creating mission"}
TEXTLINE  retmtrentxt2   {"Save mission in database"}
TEXTLINE  retnomis       {"Continue without selecting mission"}
```

```
BUTTON retmtrmt {
   num_rows     1
   num_columns  40
   textline     retmtrmtxt, 0 ,2
   helpfile     help/nomsmtr.hlp
   }

BUTTON retmtrmt1 {
   num_rows     1
   num_columns  40
   textline     retmtrmtxt1, 0 , 2
   helpfile     help/nomiss.hlp
   }

BUTTON retmtrmt2 {
   num_rows     1
   num_columns 30
   textline     retmtrmtxt2, 0 , 2
   helpfile     help/savemiss.hlp
   }

BUTTON  retnomis {
   num_rows     1
   num_columns 40
   textline     retnomis, 0 , 2
   helpfile     help/nomsmis.hlp
   }
```

```
WINDOW  defnodmtr   {
      num_rows       15
      num_columns    78
      textline       curpretxt,     0, 16
      textline       curpretxt3,    1, 16
      datum          curnavpt,      2,  1, "CALL VCAPITAL &curnavpt",
                                            "NEWVALS"
      datum          prenavpt,      2, 26
      datum          curfixid,      3,  1, "CALL VCAPITAL &curfixid",
                                            "NEWVALS"
      datum          prefixid,      3, 26
      datum          curfixrad,     4,  1
      datum          prefixrad,     4, 26
      datum          curfixdist,    5,  1
      datum          prefixdist,    5, 26
      datum          emtlat,        2, 38,  "CALL lat2dec &emt",
                                            "NEWVALS"
      datum          shwlat,        2, 64
      datum          emtlong,       3, 38,  "CALL lon2dec &emt",
                                            "NEWVALS"
      datum          shwlong,       3, 64
      datum          curfixtype,    4, 38,  "CALL VCAPITAL &curfixtyp",
                                            "NEWVALS"
      datum          prefixtype,    4, 64
      datum          curlowalt,     7,  2,  "CALL alt2dec &curlowalt",
                                            "NEWVALS"
      datum          prelowalt,     7, 26
      datum          curhighalt,    8,  2,  "CALL alt2dec &curhighalt",
                                            "NEWVALS"
      datum          prehighalt,    8, 26
      datum          curwidleft,    9,  2
      datum          prewidleft,    9, 27
      datum          curwidright,  10, 2
      datum          prewidright,  10, 27
      datum          curartcc,     11,  2,  "CALL VCAPITAL &curartcc",
                                            "NEWVALS"
      datum          preartcc,     11, 26
      button         nextnavpt,    13,  2,  "CALL nxtmtrpt"
      }

BUTTON    cancelmtr {
      num_rows    1
      num_columns     35
      textline    cancelmtrtxt,  0, 2
      helpfile    help/canclmtr.hlp
      }

BUTTON    savemtr {
      num_rows    1
      num_columns     20
      textline    savemtrtxt,  0, 2
      helpfile    help/savemtr.hlp
```

```
     }

WINDOW  mtraction   {
      num_rows      2
      num_columns   78
      line          0,0, 0,77
      button        savemtr,     1, 2, "CALL savemtr"
      button        cancelmtr,   1,40, "CALL cancmtr"
     }

SCREEN  defmodmtrscreen {
     title      "DEFINE/MODIFY MTR"
     window     mtrnamroom,  2, 1
     window     defmodmtr,   6, 1
     window     mtraction, 20, 1
     border     YES
     }

-----------------------------------------------------------------------------
-                 Declarations for SELECT ANOTHER MTR SCREEN
-----------------------------------------------------------------------------

BUTTON    showmoremtr {
     num_rows   1
     num_columns    37
     textline   showmoremtrtxt, 0 ,2
     helpfile   help/shmrmtrs.hlp
  }

BUTTON    stnewmtr {
     num_rows   1
     num_columns    40
     textline   stnewmtrtxt, 0 ,2
     helpfile   help/startmtr.hlp
    }

WINDOW    newmtrnm {
   num_rows      1
   num_columns   40
   datum         newmtrnm,  0, 2
   }

WINDOW    strecalmtr {
   num_rows      4
   num_columns   78
   button        stnewmtr, 1, 2,
                 "ADD_WINDOW newmtrnm 6  3",
                 "UPDATE_DATUM newmtrnm",
                 "CALL pnewmtrn a2bv.arr"
   button        showmoremtr,  1, 41,
                 "CALL mabunch"
   textline      recallmtrtxt, 3, 2
   }

WINDOW  curdat  {
    num_rows     10
    num_columns  32
    datum  muldat00, 0, 0
    datum  muldat01, 1, 0
    datum  muldat02, 2, 0
    datum  muldat03, 3, 0
    datum  muldat04, 4, 0
    datum  muldat05, 5, 0
    datum  muldat06, 6, 0
    datum  muldat07, 7, 0
    datum  muldat08, 8, 0
    datum  muldat09, 9, 0
}

WINDOW    curdat1  {
    num_rows     10
    num_columns  32
    datum  muldat10, 0, 0
    datum  muldat11, 1, 0
    datum  muldat12, 2, 0
    datum  muldat13, 3, 0
    datum  muldat14, 4, 0
    datum  muldat15, 5, 0
    datum  muldat16, 6, 0
    datum  muldat17, 7, 0
    datum  muldat18, 8, 0
    datum  muldat19, 9, 0
```

```
}

WINDOW  curmtrbut  {
    num_rows       10
    num_columns    2
    button  mulbut00, 0, 0, "CALL MTRcoan displmult[0].arr"
    button  mulbut01, 1, 0, "CALL MTRcoan displmult[1].arr"
    button  mulbut02, 2, 0, "CALL MTRcoan displmult[2].arr"
    button  mulbut03, 3, 0, "CALL MTRcoan displmult[3].arr"
    button  mulbut04, 4, 0, "CALL MTRcoan displmult[4].arr"
    button  mulbut05, 5, 0, "CALL MTRcoan displmult[5].arr"
    button  mulbut06, 6, 0, "CALL MTRcoan displmult[6].arr"
    button  mulbut07, 7, 0, "CALL MTRcoan displmult[7].arr"
    button  mulbut08, 8, 0, "CALL MTRcoan displmult[8].arr"
    button  mulbut09, 9, 0, "CALL MTRcoan displmult[9].arr"
}

WINDOW    curmtrbut1  {
    num_rows       12
    num_columns    2
    button  mulbut10,   0, 0, "CALL MTRcoan displmult[10].arr"
    button  mulbut11,   1, 0, "CALL MTRcoan displmult[11].arr"
    button  mulbut12,   2, 0, "CALL MTRcoan displmult[12].arr"
    button  mulbut13,   3, 0, "CALL MTRcoan displmult[13].arr"
    button  mulbut14,   4, 0, "CALL MTRcoan displmult[14].arr"
    button  mulbut15,   5, 0, "CALL MTRcoan displmult[15].arr"
    button  mulbut16,   6, 0, "CALL MTRcoan displmult[16].arr"
    button  mulbut17,   7, 0, "CALL MTRcoan displmult[17].arr"
    button  mulbut18,   8, 0, "CALL MTRcoan displmult[18].arr"
    button  mulbut19,   9, 0, "CALL MTRcoan displmult[19].arr"
    }

WINDOW retmtrent {
    num_rows    1
    num_columns 45
    button      retmtrent, 0, 1, "CALL mslistc",
                                 "CALL pmntrent"
    }

SCREEN  chgcurmtrscreen {
    title   "SELECT ANOTHER MTR"
    window  mtrnamcom,  2, 1
    window  strecalmtr, 5, 1
    window  curmtrbut,  9, 3
    window  curmtrbut1, 9,42
    window  curdat,  9, 5
    window  curdat1, 9,44
    window  retmtrent, 20, 2
    border  YES
    }

-----------------------------------------------------------------------
-                    Declarations for  new  mtr name
-----------------------------------------------------------------------

DATUM  datepubl {
    num_rows  1
    num_columns   50
    variable      datepubl, 0, 22
    leader        "Date of publication: "
    helpfile  help/datepubl.hlp
}

BUTTON  definemtr {
    num_rows   1
    num_columns    40
    textline   definemtrtxt, 0 ,2
    helpfile   help/getmap.hlp
}

WINDOW  defnewmtr   {
    num_rows       9
    num_columns    77
    datum          origmtr,     0, 1
    datum          schedule,    2, 1
    datum          srcds,       4, 1
    datum          datepubl,    6, 1
    button         definemtr,   8, 1,        "CALL emtmtrpt"
}

SCREEN  mtrdefinescreen  {
    title      "MTR DEFINITION"
```

```
      window     mtrnumcom,        3,1
      window     defaumtr,         7,2
      window     mtraction,       20,1
      border     YES
      }

-----------------------------------------------------------------------
-                 Declarations for  MISSION REQUIREMENTS WINDOW
-----------------------------------------------------------------------

VARIABLE   ac_in_form  {
     type          INTEGER
     format         %2d
     lowlimit       1
     uplimit        16
     default        2
  }

VARIABLE   pwrstunt  {
     foundin       "VARCHAR pr_pwr_m.arr"
     type          STRING
     format        %-6s
  }

DATUM   pwrstunt {
   num_rows       1
   num_columns    10
   variable       pwrstunt, 0, 2
   leader         "["
   trailer        "]"
   pickable       NO
   }

VARIABLE   mistype  {
     type          STRING
     format        %-1s
  }

DATUM   mistype1  {
     num_rows       1
     num_columns    18
     variable       mistype, 0, 14
     leader         "Mission type:"
     helpfile       help/mistype.hlp
  }

DATUM   mistype {
     num_rows  1
     num_columns    18
     variable       mistype, 0, 14
     leader         "Mission type:"
     pickable       NO
  }

VARIABLE  prealtlev   {
     foundin       "ALTSPEC prelowalt.spec"
     type          STRING
     format        %9s
  }

DATUM   prealtlev {
     num_rows       1
     num_columns    10
     variable       prealtlev, 0, 0
     pickable       NO
  }

VARIABLE  curaltlev   {
     foundin       "ALTSPEC curlowalt.spec"
     type          STRING
     format        %9s
  }

DATUM   curaltlev {
     num_rows       1
     num_columns    21
     variable       curaltlev, 0, 11
     leader         "Alt:       "
     helpfile       help/altmtr.hlp
  }

VARIABLE  ac_pre_pwr   {
```

```
        type          DOUBLE
        format        %12.31f
        lowlimit      0.0
        uplimit       4000.0
        default       100.0
    }

  DATUM   prepwrset {
        num_rows      1
        num_columns   12
        variable      ac_pre_pwr, 0, 0
        pickable      NO
    }

  VARIABLE  ac_cur_pwr   {
        type          DOUBLE
        format        %10.31f
        lowlimit      0.0
        uplimit       4000.0
        default       100.0
    }

  DATUM   curpwrset {
        num_rows      1
        num_columns   21
        variable      ac_cur_pwr, 0, 11
        leader        "Power:    "
        helpfile      help/pwrspec.hlp
    }

  VARIABLE  ac_pre_spd   {
        type          INTEGER
        format        %3d
    }

  DATUM   prespeed {
        num_rows      1
        num_columns   9
        variable      ac_pre_spd, 0, 0
        trailer       "KTS"
        pickable      NO
    }

  VARIABLE  ac_cur_spd   {
        type          INTEGER
        format        %3d
        lowlimit      100
        uplimit       600
        default       450
    }

  DATUM   curspeed {
        num_rows      1
        num_columns   21
        variable      ac_cur_spd, 0, 11
        leader        "Speed:    "
        trailer       "KTS"
        helpfile      help/mtrspeed.hlp
    }

  DATUM   preavpt1 {
        num_rows      1
        num_columns   3
        variable      preavpt, 0,0
        pickable      NO
    }

  DATUM   curavpt1 {
        num_rows      1
        num_columns   21
        variable      curavpt,   0, 11
        leader        "Wavpoint:"
        helpfile      help/navname.hlp
    }

  TEXTLINE  canclparatxt  {"Cancel data entry for this mission"}

  BUTTON    canclpara {
        num_rows      1
        num_columns   35
        textline      canclparatxt,  0, 2
        helpfile      help/canclmtr.hlp
```

```
    }

TEXTLINE  saveparatxt  ("Save this mission's data")

BUTTON   savepara {
     num_rows        1
     num_columns     30
     textline        saveparatxt,  0, 2
     helpfile        help/savemiss.hlp
     }

TEXTLINE  operdatatxt  ("Ready to enter operational data")

BUTTON   operdata {
     num_rows        1
     num_columns     33
     textline        operdatatxt,  0, 2
     helpfile        help/entrops.hlp
     }

TEXTBLOCK  mtr  {
    num_rows      10
    num_columns   38
    filename      txtblk/mtr.txt
 }

TEXTLINE  mtrlabeltxt  ("NAV     FIX        FIX TYPE      WIDTH")

TEXTLINE  mtrtxt  ("REFERENCE INFORMATION FOR DATA ENTRY")

TEXTLINE  curpretxt1  ("CURRENT     PREVIOUS")

TEXTLINE  curpretxt2  ("-------     ---------")

WINDOW opentry {
   num_rows          10
   num_columns       78
   button opsteady,   2, 19, "CALL psaddops  1"
   button opseason,   7, 19, "CALL psaddops 12"
}

WINDOW  fltpara  {
      num_rows       17
      num_columns    78
      datum          mistype,      0, 2
      datum          acname,       1, 2
      textline       mtrtxt,       0,40
      textline       mtrlabeltxt, 2,38
      textline       curpretxt1,  3,14
      textline       curpretxt2,  4,14
      datum          curnavpt1,    5, 1, "CALL VCAPITAL &curnavpt",
                                           "NEWVALS"
      datum          prenavpt1,    5,25
      datum          curspeed,     7, 1
      datum          prespeed,     7,25
      datum          curpwrset,    9, 2
      datum          prepwrset,    9,22
      datum          pwrstunt,     10, 15
      datum          curaltlev,    12, 2, "CALL alt2dec &curlowalt",
                                           "NEWVALS"
      datum          prealtlev,    12,23
      box            1, 37, 14, 77
      line           3, 38,  3, 76
      line           15, 0, 15, 77
      button         nextnavpt,  14, 2, "CALL nxtmispt"
      button         operdata,   16, 2, "NEW_SCREEN  opentry"
      button         cancelpara, 16,40, "CALL cancmis"
      textblock      mtr,          4,38
    }

SCREEN  mtrflt  {
    title       "FLIGHT PARAMETER ENTRY"
    window      mtrnamcom,      2,1
    window      fltpara,        5,1
    border      YES
    }

SCREEN  opentry  {
    title       "FLIGHT OPERATION DATA ENTRY FOR MTR"
    window      mtrnamcom,      3,1
    window      opentry,        6,1
    border      YES
```

```
    }

-----------------------------------------------------------------------
-              Declarations for seasonal MTR and REQUIREMENTS
-----------------------------------------------------------------------

DATUM   decnite {
     num_rows  1
     num_columns   6
     variable      decnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   decday {
     num_rows  1
     num_columns   10
     variable      decday, 0, 5
     leader        "DEC:"
     helpfile      help/mnthday.hlp
  }

DATUM   novnite {
     num_rows  1
     num_columns   6
     variable      novnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   novday {
     num_rows  1
     num_columns   10
     variable      novday, 0, 5
     leader        "NOV:"
     helpfile      help/mnthday.hlp
  }

DATUM   octnite {
     num_rows  1
     num_columns   6
     variable      octnite, 0, 0
     helpfile      help/mnthnite.hlp
  }
DATUM   octday {
     num_rows  1
     num_columns   10
     variable      octday, 0, 5
     leader        "OCT:"
     helpfile      help/mnthday.hlp
  }

DATUM   sepnite {
     num_rows  1
     num_columns   6
     variable      sepnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   sepday {
     num_rows  1
     num_columns   10
     variable      sepday, 0, 5
     leader        "SEP:"
     helpfile      help/mnthday.hlp
  }

DATUM   augnite {
     num_rows  1
     num_columns   6
     variable      augnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   augday {
     num_rows  1
     num_columns   10
     variable      augday, 0, 5
     leader        "AUG:"
     helpfile      help/mnthday.hlp
  }

DATUM   julnite {
     num_rows  1
```

```
        num_columns    6
        variable       julnite, 0, 0
        helpfile       help/mnthnite.hlp
    }

  DATUM   julday {
        num_rows  1
        num_columns    10
        variable       julday, 0, 5
        leader         "JUL:"
        helpfile       help/mnthday.hlp
    }

  DATUM   junnite {
        num_rows  1
        num_columns    6
        variable       junnite, 0, 0
        helpfile       help/mnthnite.hlp
    }

  DATUM   junday {
        num_rows  1
        num_columns    10
        variable       junday, 0, 5
        leader         "JUN:"
        helpfile       help/mnthday.hlp
    }

  DATUM   maynite {
        num_rows  1
        num_columns    6
        variable       maynite, 0, 0
        helpfile       help/mnthnite.hlp
    }

  DATUM   mayday {
        num_rows  1
        num_columns    10
        variable       mayday, 0, 5
        leader         "MAY:"
        helpfile       help/mnthday.hlp
    }

  DATUM   aprnite {
        num_rows  1
        num_columns    6
        variable       aprnite, 0, 0
        helpfile       help/mnthnite.hlp
    }
  DATUM   aprday {
        num_rows  1
        num_columns    10
        variable       aprday, 0, 5
        leader         "APR:"
        helpfile       help/mnthday.hlp
    }

  DATUM   marnite {
        num_rows  1
        num_columns    6
        variable       marnite, 0, 0
        helpfile       help/mnthnite.hlp
    }

  DATUM   marday {
        num_rows  1
        num_columns    10
        variable       marday, 0, 5
        leader         "MAR:"
        helpfile       help/mnthday.hlp
    }

  DATUM   febnite {
        num_rows  1
        num_columns    6
        variable       febnite, 0, 0
        helpfile       help/mnthnite.hlp
    }

  DATUM   febday {
        num_rows  1
        num_columns    10
```

```
      variable      febday, 0, 5
      leader        "FEB:"
      helpfile      help/mmthday.hlp
   }

 DATUM   jannite {
      num_rows  1
      num_columns   6
      variable      jannite, 0, 0
      helpfile      help/mmthnite.hlp
   }

 DATUM   janday {
      num_rows  1
      num_columns   10
      variable      janday, 0, 5
      leader        "JAN:"
      helpfile      help/mmthday.hlp
   }

 WINDOW   month  {
      num_rows   14
      num_columns    77
      textline   instritxt,  1, 1
      textline   daynite,    3, 0
      datum  janday,     4, 1
      datum  jannite,    4,12
      datum  febday,     5, 1
      datum  febnite,    5,12
      datum  marday,     6, 1
      datum  marnite,    6,12
      datum  aprday,     4,20
      datum  aprnite,    4,31
      datum  mayday,     5,20
      datum  maynite,    5,31
      datum  junday,     6,20
      datum  junnite,    6,31
      datum  julday,     4,39
      datum  julnite,    4,50
      datum  augday,     5,39
      datum  augnite,    5,50
      datum  sepday,     6,39
      datum  sepnite,    6,50
      datum  octday,     4,58
      datum  octnite,    4,69
      datum  novday,     5,58
      datum  novnite,    5,69
      datum  decday,     6,58
      datum  decnite,    6,69
      button savepara,  13, 2, "CALL savemis"
      button canclpara, 13,40, "CALL cancmis"
      }
```

```
-      Declarations for new SELECT AIRCRAFT AND MISSION FOR MTR
```

```
TEXTLINE  shwmoremistxt  {"Show more mission names (if any)"}

BUTTON   shwmoremis {
     num_rows       1
     num_columns    38
     textline       shwmoremistxt,  0, 2
     helpfile       help/shmrmiss.hlp
     }

TEXTLINE  recalmis  {"Recall one of the following missions"}

WINDOW  newmisnm {
   num_rows      1
   num_columns   40
   datum         misname,  0, 2
   }

WINDOW  curmiss  {
    num_rows     16
    num_columns  78
    button  stnewmis,    1, 2, "ADD_WINDOW newmisnm  7 3",
                               "UPDATE_DATUM misname",
                               "CALL psnwmisn cid.arr"
    button  shwmoremis,  1, 40, "CALL ssbunch"
    textline   recalmis, 3, 2
```

```
    datum  muldat00,     4, 4
    datum  muldat01,     5, 4
    datum  muldat02,     6, 4
    datum  muldat03,     7, 4
    datum  muldat04,     8, 4
    datum  muldat05,     9, 4
    datum  muldat06,     10, 4
    datum  muldat07,     11, 4
    datum  muldat08,     12, 4
    datum  muldat09,     13, 4
    datum  muldat10,     4,42
    datum  muldat11,     5,42
    datum  muldat12,     6,42
    datum  muldat13,     7,42
    datum  muldat14,     8,42
    datum  muldat15,     9,42
    datum  muldat16,     10,42
    datum  muldat17,     11,42
    datum  muldat18,     12,42
    datum  muldat19,     13,42
    button  mulbut00,    4, 2, "CALL MIScoan dsplmult[0].arr"
    button  mulbut01,    5, 2, "CALL MIScoan dsplmult[1].arr"
    button  mulbut02,    6, 2, "CALL MIScoan dsplmult[2].arr"
    button  mulbut03,    7, 2, "CALL MIScoan dsplmult[3].arr"
    button  mulbut04,    8, 2, "CALL MIScoan dsplmult[4].arr"
    button  mulbut05,    9, 2, "CALL MIScoan dsplmult[5].arr"
    button  mulbut06,    10,2, "CALL MIScoan dsplmult[6].arr"
    button  mulbut07,   11, 2, "CALL MIScoan dsplmult[7].arr"
    button  mulbut08,   12, 2, "CALL MIScoan dsplmult[8].arr"
    button  mulbut09,   13, 2, "CALL MIScoan dsplmult[9].arr"
    button  mulbut10,    4,40, "CALL MIScoan dsplmult[10].arr"
    button  mulbut11,    5,40, "CALL MIScoan dsplmult[11].arr"
    button  mulbut12,    6,40, "CALL MIScoan dsplmult[12].arr"
    button  mulbut13,    7,40, "CALL MIScoan dsplmult[13].arr"
    button  mulbut14,    8,40, "CALL MIScoan dsplmult[14].arr"
    button  mulbut15,    9,40, "CALL MIScoan dsplmult[15].arr"
    button  mulbut16,    10,40, "CALL MIScoan dsplmult[16].arr"
    button  mulbut17,   11,40, "CALL MIScoan dsplmult[17].arr"
    button  mulbut18,   12,40, "CALL MIScoan dsplmult[18].arr"
    button  mulbut19,   13,40, "CALL MIScoan dsplmult[19].arr"
    button  retncmis,   15, 2, "CALL sslistc",
                               "CALL NEW_SCREEN oldscreen"
    }

TEXTLINE  selmisstxt  {"Select mission"}

BUTTON   selmiss {
    num_rows        1
    num_columns     38
    textline        selmisstxt,  0, 2
    helpfile        help/selmiss.hlp
    }

TEXTLINE  fltparinftxt  {"Specify flight parameter information"}

BUTTON   fltparinf {
    num_rows        1
    num_columns     37
    textline        fltparinftxt,  0, 2
    helpfile        help/fltparam.hlp
    }

WINDOW    selacmiss  {
    num_rows            15
    num_columns         77
    datum  acnamel,     1,1, "CALL vfyacmtr tid.arr"
    button selmiss,     3,2, "CALL pschgmis"
    button fltparinf,   5,2, "CALL psmtrflt"
    button retmtreat1, 13,2, "CALL cancmis"
    }

SCREEN  spemtrmis  {
    title      "SELECT AIRCRAFT AND MISSION FOR MTR"
    window     mtrnamcoml, 3,1
    window     selacmiss,  7,1
    border     YES
    }

SCREEN  chgmis  {
    title      "SELECT AIRCRAFT AND MISSION FOR MTR"
    window     mtrnamcoml, 2,1
    window     curmiss,    6,1
```

```
    border     YES
    }

-----------------------------------------------------------------------
~        Declarations for Day-Night window
-----------------------------------------------------------------------

DATUM   night {
    num_rows       1
    num_columns    16
    variable       night, 0, 8
    leader         "Night:"
    helpfile       help/niteops.hlp
  }

DATUM   day {
    num_rows       1
    num_columns    16
    variable       day, 0, 6
    leader         "Day:"
    helpfile       help/dayops.hlp
  }

WINDOW   daynite  {
    num_rows     12
    num_columns 77
    datum     acname,      1, 1
    textline instr2txt,   3, 1
    datum     day,         5, 2
    datum     night,       5,15
    button    savepara,   10, 2, "CALL expmops",
                                 "CALL savemis"
    button    canclpara, 10,40, "CALL cancmis"
    border    YES
    }

-----------------------------------------------------------------------
~        Declarations for mission specification window
-----------------------------------------------------------------------

DATUM   sortie {
    num_rows       1
    num_columns    37
    variable       ac_in_form, 0, 35
    leader         "Number of aircraft in formation:  "
    helpfile       help/numform.hlp
  }

VARIABLE  misdesc {
   foundin "VARCHAR misdesc.arr"        ~this description is for mission
   type     STRING
   format   %-60s
   }

DATUM   misdesc1 {
   num_rows       1
   num_columns    76
   variable       misdesc, 0, 7
   leader         "Descr:"
   pickable       NO
   }

DATUM   misdesc {
   num_rows       1
   num_columns    76
   variable       misdesc, 0, 14
   leader         "Description:"
   helpfile       help/misdesc.hlp
   }

WINDOW  mtrnamcom3   {
   num_rows       3
   num_columns    78
   line           2,0, 2,77
   datum          misname1,  0, 2
   datum          misdesc1, 1, 2
   }

WINDOW   misspec  {
   num_rows              16
   num_columns           77
   datum   misdesc,      1,1
```

```
    datum   mistype1,     3,2
    datum   sortie,       5,2
    button  retmtrent1, 14,2,  "CALL NEW_SCREEN oldscreen"
    button  retmtrent2, 14,47, "CALL entermis"
  }

SCREEN  misspec  {
     title       "MISSION SPECIFICATION FOR AN MTR"
     window      mtrnamcom3, 3,1
     window      misspec,    6,1
     border      YES
     }

~01-14-88  st   will try to fix this file so it wil be only for assessments
~01-14-88  st   changes will be commented with date
~01-16-88  st   change title on first screen, change lat. long. format lon
~               map control
~01-17-88  st   con't to fix sdf file  commented with 1-17-88

~------------------------------------------------------------------------
~ DECLARATIONS FOR SCREEN HEADER
~------------------------------------------------------------------------

VARIABLE assename {
    foundin  "ASANHEADR ASSESSMENT.name"
    type     STRING
    format   %-30s
 }

DATUM  assename {
    num_rows     1
    num_columns  50
    variable     assename, 0 , 18
    leader       "Assessment name: "
    pickable     NO
 }

DATUM  assename1 {
    num_rows     1
    num_columns  75
    variable     assename, 0 , 30
    leader       "Name of current assessment: "
    pickable     NO
 }

VARIABLE comment {
    foundin  "ASANHEADR ASSESSMENT.desc"
    type     STRING
    format   %-66s
 }

DATUM  comment {
    num_rows     1
    num_columns  76
    variable     comment, 0 , 10
    leader       "Comment: "
    pickable     NO
 }

VARIABLE  curmapnam {
    type   STRING
    format %-30s
 }

DATUM  curmapnam {
    num_rows    1
    num_columns 77
    variable    curmapnam, 0 , 19
    leader      "Current map name: "
    pickable    NO
 }

WINDOW  lin  {
    num_rows     1
    num_columns  78
    line         0, 0, 0,77
 }

WINDOW  assnamcom  {
    num_rows      3
    num_columns   78
    datum         assename1,  0,1
```

```
    datum          comment,  1, 2
    line           2, 0, 2,77
 }

WINDOW  assnamcom1  {
    num_rows      4
    num_columns   78
    datum         assename1, 0, 0
    datum         comment, 1, 1
    datum         curmapnam, 2, 1
    line          3, 0, 3,77
 }

-----------------------------------------------------------------------
-    Declarations for buttons used in majoraction footer
-----------------------------------------------------------------------

TEXTLINE probstatxt { "REVIEW CURRENT ASSESSMENT STATUS" }

BUTTON  probstat  {
    num_rows      1
    num_columns   35
    textline      probstatxt,  0, 2
    helpfile      help/probstat.hlp
 }

TEXTLINE probstatxt2 { "REVIEW ASSESSMENT STATUS" }

BUTTON  probstat2  {
    num_rows      1
    num_columns   26
    textline      probstatxt2,  0, 2
    helpfile      help/revstat.hlp
 }

TEXTLINE probdef {"ADD INFORMATION TO CURRENT ASSESSMENT" }

BUTTON  probdef  {
    num_rows      1
    num_columns   39
    textline      probdef,  0, 2
    helpfile      help/probdef.hlp
 }

TEXTLINE probdef1 {"ADD TO ASSESSMENT DEFINITION" }

BUTTON  probdef1  {
    num_rows      1
    num_columns   31
    textline      probdef1,  0, 2
    helpfile      help/probdef.hlp
 }

TEXTLINE analysis { "ANALYZE DATA" }

BUTTON  analysis  {
    num_rows      1
    num_columns   15
    textline      analysis,  0, 2
    helpfile      help/datanal.hlp
 }

TEXTLINE reportgen { "MAKE A REPORT" }

BUTTON  reportgen  {
    num_rows      1
    num_columns   16
    textline      reportgen,  0, 2
    helpfile      help/reportgen.hlp
 }

TEXTLINE vwchklistxt { "VIEW CHECKLIST FOR CURRENT ASSESSMENT" }

BUTTON  vwchklist  {
    num_rows      1
    num_columns   42
    textline      vwchklistxt,  0, 2
    helpfile      help/viewlist.hlp
 }

-----------------------------------------------------------------------
-        Declarations for majoraction footers
```

```
---------------------------------------------------------------------------

WINDOW  majoraction  {          ~header for problem status screen
    num_rows       4
    num_columns    78
    line           0, 0, 0,77
    title          "Alternative actions you can now take:"
    button         probdaf,      2, 2, "NEW_SCREEN probdafscreen"
    button         analysis,     2,56, "NEW_SCREEN analysiscreen"
    button         vwchklist,    3, 2, "NEW_SCREEN viewchecklistscreen"
    button         reportgen,    3,56, "CALL dummy2"
 }

WINDOW  majoraction1  {         ~header for problem definition screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       analysis,     2, 2, "NEW_SCREEN analysiscreen"
    button       probstat2,    2,48, "CALL psprobst"
    button       vwchklist,    3, 2, "NEW_SCREEN viewchecklistscreen"
    button       reportgen,    3,48, "CALL dummy2"
 }

WINDOW  majoraction2  {         ~header for data analysis screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       probdaf1,     2, 2, "NEW_SCREEN probdafscreen"
    button       reportgen,    2,47, "CALL dummy2"
    button       vwchklist,    3, 2, "NEW_SCREEN viewchecklistscreen"
    button       probstat2,    3,47, "CALL psprobst"
 }

WINDOW  majoraction3  {         ~header for report generation screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       vwchklist,    2, 2, "NEW_SCREEN viewchecklistscreen"
    button       probstat2,    2,47, "CALL psprobst"
    button       probdaf1,     3, 2, "NEW_SCREEN probdafscreen"
    button       analysis,     3,47, "NEW_SCREEN analysiscreen"
 }

WINDOW  majoraction4  {         ~header for view checklist screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       probstat2,    2, 2, "CALL psprobst"
    button       reportgen,    2,47, "CALL dummy2"
    button       probdaf1,     3, 2, "NEW_SCREEN probdafscreen"
    button       analysis,     3,47, "NEW_SCREEN analysiscreen"
 }


TEXTLINE loadglobal ("LOAD LOCAL DATA FOR GENERAL ACCESS")

BUTTON  loadglobal   {
   num_rows      1
   num_columns   39
   textline      loadglobal, 0, 2
   helpfile      help/loadglob.hlp
   }

WINDOW  majoraction5  {         ~footer for database housekeeping screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       loadglobal,  2, 2, "CALL dummy"
    button       assessment,  3, 2, "CALL psprobst"
 }

SCREEN  mtrntry {
   title     "MTR DATA ENTRY"
   window    mtrnmcom,      3,1
   window    mtrdateentry,  7,1
   window    majoraction3, 18,1
   border    YES
```

```
    }

 ------------------------------------------------------------------------
 -              Declarations for problem status screen
 ------------------------------------------------------------------------

 VARIABLE  startdate {
     foundin "VARCHAR startdate.arr"
     type    STRING
     format  %-21s
  }

 DATUM  startdate {
     num_rows    1
     num_columns 40
     variable    startdate, 0 ,15
     leader      "Date started: "
     pickable    NO
  }

 VARIABLE  lastdate {
     foundin  "VARCHAR lastdate.arr"
     type     STRING
     format   %-21s
  }

 DATUM  lastdate {
     num_rows     1
     num_columns  55
     variable     lastdate, 0 , 28
     leader       "Date of last modification: "
     pickable     NO
  }

 VARIABLE  planrlast {
     foundin  "VARCHAR planrlst.arr"
     type     STRING
     format   %-30s
  }

 DATUM  planrlast {
     num_rows     1
     num_columns 55
     variable    planrlast, 0 ,19
     leader      "Last modified by: "
     pickable    NO
  }

 WINDOW  probstat {
     num_rows     5
     num_columns  78
     datum        startdate,  0, 2
     datum        lastdate,   2, 2
     datum        planrlast,  4, 2
  }

 TEXTLINE  chgasstxt  {"Work on a different assessment"}

 BUTTON  chgass  {                  -1-16-88
     num_rows     1                 -1-16-88
     num_columns  70                -1-16-88
     textline     chgasstxt,  0, 2    -1-16-88
     helpfile     help/probdef.hlp    -1-16-88
  }


 WINDOW  chgassess  {           -change to different assessment
     num_rows    1
     num_columns 78
     button      chgass, 0, 2, "CALL pschgass"
  }

 ------------------------------------------------------------------------
 -         Declarations for ENVIRONMENTAL ASSESSMENT DEFINITION SCREEN
 ------------------------------------------------------------------------

 TEXTLINE  modmoatxt  {"Work with MOA information (number or type of aircraft, missions, etc.)"}

 BUTTON  modmoa {
     num_rows   1
     num_columns    76
     textline   modmoatxt, 0 ,2
```

```
    helpfile   help/moawork.hlp
 }

TEXTLINE  modmtrtxt  {"Work with MTR information (number or type of aircraft, missions, etc.)"}

BUTTON  modmtr {
    num_rows    1
    num_columns 76
    textline    modmtrtxt, 0 ,2
    helpfile    help/mtrwork.hlp
 }

TEXTLINE  modmaptxt  {"Work with map information (designate land uses, update maps)"}

BUTTON  modmap  {
    num_rows    1
    num_columns 76
    textline    modmaptxt, 0 ,2
    helpfile    help/mapwork.hlp
 }

TEXTLINE  selectxt  {"Actions you can now take to add information to this assessment:"}

WINDOW  selaction  {
    num_rows    7
    num_columns 77
    textline    selectxt, 0, 1
    button      modmap,   2, 1, "CALL dummy2"
    button      modmtr,   4, 1, "CALL psmtreat"
    button      modmoa,   6, 1, "CALL dummy"
 }

SCREEN  probdefscreen {
    title   "ENVIRONMENTAL ASSESSMENT DEFINITION"
    window  assnamcom,  3,1
    window  selaction,  7, 2
    window  majoraction1, 18 , 1
    border  YES
 }

-----------------------------------------------------------------------------
-        Declaration for entering coordinates window
-----------------------------------------------------------------------------

TEXTLINE  entupplf
  {"Enter upper-left corner coordinates of area of current interest"}

TEXTLINE  entlowrt
  {"Enter lower right corner coordinates of area of current interest"}

DATUM   shwlat2 {
     num_rows   1
     num_columns    25
     variable   shwlat, 0 ,11
     leader     "Latitude: "
     helpfile   help/combmap.hlp
  }

DATUM   shwlong2 {
     num_rows   1
     num_columns    25
     variable   shwlong, 0 ,12
     leader     "Longitude: "
     helpfile   help/combmap.hlp
  }

WINDOW  entcoor  {
    num_rows      10
    num_columns   78
    textline     entupplf,  2, 2
    datum        entlat,    3, 1,   "CALL lat2dec &ent","NEWVALS"
    datum        entlong,   3, 38,  "CALL lon2dec &ent","NEWVALS"
    textline     entlowrt,  5, 2
    datum        shwlat2,   6, 1,   "CALL lat2dec &show","NEWVALS"
    datum        shwlong2,  6, 38,  "CALL lon2dec &show","NEWVALS"
 }

-----------------------------------------------------------------------------
-        Declarations for DATA ANALYSIS SCREEN
-----------------------------------------------------------------------------

TEXTLINE  geodatinqtxt {"Make geodata inquiries on map screen"}
```

```
BUTTON  geodatinq {
    num_rows    1
    num_columns 76
    textline    geodatinqtxt, 0 ,2
    helpfile    help/geoinq.hlp
 }

TEXTLINE  compnoiseftxt  {"Compare noise effects"}  -1-17-88

BUTTON  compnoisef  {                       -1-17-88
    num_rows    1                           -1-17-88
    num_columns 76                          -1-17-88
    textline    compnoiseftxt, 0 ,2             -1-17-88
    helpfile    help/compnois.hlp               -2-06-88
 }                              -1-17-88

TEXTLINE  calcnoiseftxt  {"Calculate noise effects in specified area"}

BUTTON  calcnoisef  {
    num_rows    1
    num_columns 76
    textline    calcnoiseftxt, 0 ,2
    helpfile    help/effcalc.hlp
 }

TEXTLINE  calcnoiseaptxt  {"Calculate noise exposure in specified area"}

BUTTON  calcnoiseap  {
    num_rows    1
    num_columns 76
    textline    calcnoiseaptxt, 0 ,2
    helpfile    help/expcalc.hlp
 }

TEXTLINE  calcqklooktxt  {"Calculate quicklook (point) exposure estimate"}

BUTTON  calcqklook {
    num_rows    1
    num_columns 50
    textline    calcqklooktxt, 0 ,2
    helpfile    help/qklook.hlp
 }

TEXTLINE  entcoortxt  {"Enter coordinates from keyboard"}

BUTTON  entcoor {
    num_rows    1
    num_columns 40
    textline    entcoortxt, 0 ,2
    helpfile    help/kbdntry.hlp
 }

TEXTLINE  usemaptxt  {"Use map screen"}

BUTTON  usemap {
    num_rows    1
    num_columns 30 textline    usemaptxt, 0 ,2
    helpfile    help/usemap.hlp
 }

TEXTLINE  selectanstxt  {"Actions you can now take to analyze environmental assessment data:"}

TEXTLINE  defgeoarestxt  {"Specify a geographic area of interest:"}

WINDOW  datanaction  {
    num_rows      11
    num_columns   78
    textline      selectanstxt,  0, 2
    button        calcqklook,    2, 2, "CALL dummy"
    button        calcnoiseap,   3, 2, "CALL dummy"
    button        calcnoisef,    4, 2, "CALL dummy"
    button        compnoisef,    5, 2, "CALL dummy"
    button        geodatinq,     6, 2, "CALL dummy"
    textline      defgeoarestxt, 8, 2
    button        usemap,        9, 2, "CALL dummy2"
    button        entcoor,       10,2, "ADD_WINDOW entcoor  5 1"
 }

SCREEN  analysiscreen {
    title   "DATA ANALYSIS"
    window  assnmnoom,  2,1
    window  datanaction,  6,1
```

```
    window  majoractions2, 18 , 1
    border  YES
  }

TEXTLINE   showmoretxt {"Show more assessments (if any)"}

BUTTON    showmore {
     num_rows   1
     num_columns    37
     textline   showmoretxt, 0 ,2
     helpfile   help/getassmnt.hlp
  }

TEXTLINE  recallasstxt   {"Recall one of the following assessments:"}

TEXTLINE  stnewasstxt   {"Start a new assessment"}

BUTTON    stnewass {
     num_rows       1
     num_columns    28
     textline       stnewasstxt, 0 ,2
     helpfile       help/newassmnt.hlp
     }

VARIABLE newasnam  {
     foundin    "VARCHAR n2bv.arr"
     type       STRING
     format     %-30s
  }

DATUM   newasnam {
     num_rows       1
     num_columns    40
     variable       newasnam,   0 , 7
     leader         "Name:"
     helpfile       help/stnewass.hlp
  }

WINDOW  newasnam  {
     num_rows       1
     num_columns    40
     datum          newasnam,  0,1
     }

WINDOW  stnewass  {
     num_rows       1
     num_columns    78
     button         stnewass,  0, 2, "ADD_WINDOW newasnam, 6  3",
                            "UPDATE_DATUM newasnam",
                            "CALL psnewasen n2bv.arr"
     button         showmore,  0, 41,    "CALL ubunch"
     }

WINDOW  recalass  {
     num_rows       1
     num_columns    78
     textline       recallasstxt,  0, 2
     }

WINDOW    curassbut  {
      num_rows      10
      num_columns   2
      button   mulbut00,  0, 0,    "CALL ASANcomm displmult[0].arr"
      button   mulbut01,  1, 0,    "CALL ASANcomm displmult[1].arr"
      button   mulbut02,  2, 0,    "CALL ASANcomm displmult[2].arr"
      button   mulbut03,  3, 0,    "CALL ASANcomm displmult[3].arr"
      button   mulbut04,  4, 0,    "CALL ASANcomm displmult[4].arr"
      button   mulbut05,  5, 0,    "CALL ASANcomm displmult[5].arr"
      button   mulbut06,  6, 0,    "CALL ASANcomm displmult[6].arr"
      button   mulbut07,  7, 0,    "CALL ASANcomm displmult[7].arr"
      button   mulbut08,  8, 0,    "CALL ASANcomm displmult[8].arr"
      button   mulbut09,  9, 0,    "CALL ASANcomm displmult[9].arr"
    }

WINDOW    curassbut1  {
      num_rows      10
      num_columns   2
      button   mulbut10,  0, 0,    "CALL ASANcomm displmult[10].arr"
      button   mulbut11,  1, 0,    "CALL ASANcomm displmult[11].arr"
      button   mulbut12,  2, 0,    "CALL ASANcomm displmult[12].arr"
      button   mulbut13,  3, 0,    "CALL ASANcomm displmult[13].arr"
      button   mulbut14,  4, 0,    "CALL ASANcomm displmult[14].arr"
```

```
        button   mulbut15,  5, 0,    "CALL ASANcoan dsplmult[15].arr"
        button   mulbut16,  6, 0,    "CALL ASANcoan dsplmult[16].arr"
        button   mulbut17,  7, 0,    "CALL ASANcoan dsplmult[17].arr"
        button   mulbut18,  8, 0,    "CALL ASANcoan dsplmult[18].arr"
        button   mulbut19,  9, 0,    "CALL ASANcoan dsplmult[19].arr"
      }

TEXTLINE  retnoass  {"Continue without selecting assessment"}

BUTTON retassent {
   num_rows     1
   num_columns  40
   textline     retnoass, 0 ,2
   helpfile     help/noassess.hlp
   }

WINDOW  retwoass {
   num_rows     1
   num_columns  76
   button       retassent, 0,1, "CALL ulistc", "CALL psprobst"
   }

SCREEN  chgcurasscreen {
     title      "SELECT ANOTHER ASSESSMENT"
     window     assnmcom,  2,1
     window     stnewass,   6,1
     window     recalass,   8,1
     window     curassbut,  9,3
     window     curassbut1, 9,42
     window     curdat,     9,5
     window     curdat1,    9,44
     window     retwoass,   20,2
     border     YES
     }

-----------------------------------------------------------------------
-        Declarations for NEW ASSESSMENT DEFINTION
-----------------------------------------------------------------------

VARIABLE entdesc1  {
     foundin    "VARCHAR60 entdesc[0].arr"
     type    STRING
     format     %-60s
  }

DATUM   entdesc1 {
     num_rows   1
     num_columns    75
     variable   entdesc1, 0 ,0
     helpfile   help/entdesc.hlp
  }

VARIABLE entdesc2  {
     foundin    "VARCHAR60 entdesc[1].arr"
     type       STRING
     format     %-60s
  }

DATUM   entdesc2 {
     num_rows   1
     num_columns    75
     variable   entdesc2, 0 ,0
     helpfile   help/entdesc.hlp
  }

VARIABLE entdesc3  {
     foundin    "VARCHAR60 entdesc[2].arr"
     type       STRING
     format     %-60s
  }

DATUM   entdesc3 {
     num_rows   1
     num_columns    75
     variable   entdesc3, 0 ,0
     helpfile   help/entdesc.hlp
  }

VARIABLE entdesc4  {
     foundin    "VARCHAR60 entdesc[3].arr"
     type       STRING
     format     %-60s
```

```
  }

DATUM   entdesc4 {
     num_rows   1
     num_columns    75
     variable   entdesc4, 0 ,0
     helpfile   help/entdesc.hlp
  }

TEXTLINE entdesctxt  ("Please enter a brief description for this assessment")

DATUM   newassnam {
     num_rows   1
     num_columns    77
     variable   newasnam, 0 ,35
     leader     "Name of new assessment definition:"
     pickable   NO
  }

WINDOW  newasdesc  {
     num_rows    9
     num_columns 70
     datum      newassnam,  1,3
     textline   entdesctxt, 3,4
     datum      entdesc1,   4,4
     datum      entdesc2,   5,4
     datum      entdesc3,   6,4
     datum      entdesc4,   7,4
     border     YES
     }

-----------------------------------------------------------------------
-          Declarations for database housekeeping screen
-----------------------------------------------------------------------

TEXTLINE  updateinfotxt  ("Update information")

BUTTON    updateinfo {
     num_rows   1
     num_columns    50
     textline   updateinfotxt, 0 ,2
     helpfile   help/updatein.hlp
     }

TEXTLINE  asstabletxt2  ("Print list of all columns in an assessment's tables")

BUTTON    asstable2 {
     num_rows   1
     num_columns    55
     textline   asstabletxt2, 0 ,2
     helpfile   help/asstable.hlp
     }

TEXTLINE  asstabletxt1  ("Print list of an assessment's tables")

BUTTON    asstable1 {
     num_rows   1
     num_columns    50
     textline   asstabletxt1, 0 ,2
     helpfile   help/asstable.hlp
     }

TEXTLINE  assesstxt  ("Print list of all assessments")

BUTTON    assess {
     num_rows   1
     num_columns    50
     textline   assesstxt, 0 ,2
     helpfile   help/achelp.hlp
     }

TEXTLINE  dbdatetxt  ("Print all database dates")

BUTTON    dbdate {
     num_rows   1
     num_columns    50
     textline   dbdatetxt, 0 ,2
     helpfile   help/dbdate.hlp
     }

TEXTLINE  dbhskpgtxt  ("WARNING: Actions you take on this screen affect ASAM'S
                        permanent databases!")
```

```
WINDOW    dbhsekpgaction   {
          num_rows       13
          num_columns    78
          line           0,0,  0,77
          textline       dbhsekpgtxt,   2,2
          button         dbdate,       4,2,  "CALL dummy"
          button         assess,       6,2,  "CALL prntabc"
          button         asstable1,    8,2,  "CALL SUpropt 1"
          button         asstable2,   10,2,  "CALL SUpropt 2"
          button         updateinfo,  12,2,  "CALL dummy"
     }

SCREEN  dbhsekpgscreen {
     title      "DATABASE HOUSEKEEPING"
     window     dbhsekpgaction, 2,1
     window     majoraction5,   18,1
     border     YES
     }


WINDOW  quickassbut  {
       num_rows      10
       num_columns   2
       button   mulbut00,  0, 0,    "CALL SUprint dsplmult[0].arr"
       button   mulbut01,  1, 0,    "CALL SUprint dsplmult[1].arr"
       button   mulbut02,  2, 0,    "CALL SUprint dsplmult[2].arr"
       button   mulbut03,  3, 0,    "CALL SUprint dsplmult[3].arr"
       button   mulbut04,  4, 0,    "CALL SUprint dsplmult[4].arr"
       button   mulbut05,  5, 0,    "CALL SUprint dsplmult[5].arr"
       button   mulbut06,  6, 0,    "CALL SUprint dsplmult[6].arr"
       button   mulbut07,  7, 0,    "CALL SUprint dsplmult[7].arr"
       button   mulbut08,  8, 0,    "CALL SUprint dsplmult[8].arr"
       button   mulbut09,  9, 0,    "CALL SUprint dsplmult[9].arr"
     }

WINDOW    quickassbut1  {
       num_rows      10
       num_columns   2
       button   mulbut10,  0, 0,    "CALL SUprint dsplmult[10].arr"
       button   mulbut11,  1, 0,    "CALL SUprint dsplmult[11].arr"
       button   mulbut12,  2, 0,    "CALL SUprint dsplmult[12].arr"
       button   mulbut13,  3, 0,    "CALL SUprint dsplmult[13].arr"
       button   mulbut14,  4, 0,    "CALL SUprint dsplmult[14].arr"
       button   mulbut15,  5, 0,    "CALL SUprint dsplmult[15].arr"
       button   mulbut16,  6, 0,    "CALL SUprint dsplmult[16].arr"
       button   mulbut17,  7, 0,    "CALL SUprint dsplmult[17].arr"
       button   mulbut18,  8, 0,    "CALL SUprint dsplmult[18].arr"
       button   mulbut19,  9, 0,    "CALL SUprint dsplmult[19].arr"
     }

TEXTLINE  suselect  {"Print list of SUPERUSER's tables"}
TEXTLINE  hqselect  {"Print list of HEADQUARTERS' tables"}

BUTTON    suselect {
     num_rows   1
     num_columns    50
     textline   suselect, 0 ,2
     helpfile   help/nohelp.hlp
     }
BUTTON    hqselect {
     num_rows   1
     num_columns    50
     textline   hqselect, 0 ,2
     helpfile   help/nohelp.hlp
     }

WINDOW  retwoprn {
   num_rows     1
   num_columns  76
   button       retassent, 0,1, "CALL ulistc", "NEW_SCREEN dbhsekpgscreen"
   }



WINDOW  hqsuselect {
   num_rows     3
   num_columns  76
   button       hqselect, 0,1, "CALL SUpropt 3"
   button       suselect, 2,1, "CALL SUpropt 4"
   }

SCREEN  slcasscreen {
```

```
     title       "SELECT ASSESSMENT FOR DATABASE PRINTOUT"
     window      hqsuselect,   4,1
     window      recalass,     8,1
     window      quickassbut,  9,3
     window      quickassbut1, 9,42
     window      curdat,       9,5
     window      curdat1,      9,44
     window      retwoprn,    20,2
     border      YES
     }


--------------------------------------------------------------------------
-        Declarations for view CHECK-LIST SCREEN
--------------------------------------------------------------------------

TEXTLINE  otherchklst  ("View another checklist")

BUTTON    otherchklst {
     num_rows        1
     num_columns     25
     textline        otherchklst,  0, 2
     helpfile        help/othrchk.hlp
     }

TEXTBLOCK  foasi2  {
     filename        txtblk/foasi2.txt
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW    foasi2   {
     num_rows        13
     num_columns     78
     textblock        foasi2,   0,1
     button          otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  foasi1  {
     filename        txtblk/foasi1.txt
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW    foasi1   {
     num_rows        13
     num_columns     78
     textblock        foasi1,   0,1
     button          otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  ostax3  {
     filename        txtblk/ostax3.txt
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW    ostax3   {
     num_rows        13
     num_columns     78
     textblock        ostax3,   0,1
     button          otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  ostax2  {
     filename        txtblk/ostax2.txt
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW    ostax2   {
     num_rows        13
```

```
        num_columns     78
        textblock        catex2,   0,1
        button          otherchklst, 12,2,  "REMOVE_WINDOW"
        }

  TEXTBLOCK  catex1  {
        filename        txtblk/catex1.txt
        num_rows        10
        num_columns     76
        border          YES
        }

  WINDOW    catex1   {
        num_rows        13
        num_columns     78
        textblock        catex1,   0,1
        button          otherchklst, 12,2,  "REMOVE_WINDOW"
        }

  TEXTLINE  fonsi2txt  {
  "Show documentation necessary for finding of no significant impact")

  BUTTON    fonsi2 {
        num_rows   1
        num_columns    72
        textline   fonsi2txt,  0, 2
        helpfile   help/fonsi2.hlp
        }

  TEXTLINE  fonsi1txt  {
  "Show NEPA bases for finding of no significant impact (FONSI)")

  BUTTON    fonsi1 {
        num_rows   1
        num_columns    65
        textline   fonsi1txt,  0, 2
        helpfile   help/fonsi1.hlp
        }

  TEXTLINE  catex3txt  {"Show documentation needed for categorical exclusions")

  BUTTON    catex3 {
        num_rows   1
        num_columns    55
        textline   catex3txt,  0, 2
        helpfile   help/catex3.hlp
        }

  TEXTLINE  catex2txt  {
  "Show examples of proposed actions qualifying for categorical exclusions")

  BUTTON    catex2 {
        num_rows   1
        num_columns    74
        textline   catex2txt,  0, 2
        helpfile   help/catex2.hlp
        }

  TEXTLINE  catex1txt  {"Show NEPA bases for categorical exclusions (CATEX)")

  BUTTON    catex1 {
        num_rows   1
        num_columns    55
        textline   catex1txt,  0, 2
        helpfile   help/catex1.hlp
        }

  WINDOW    vwchklist   {
        num_rows       11
        num_columns    78
        line      0,0 ,  0,77
        button    catex1,        2,2, "ADD_WINDOW catex1  3  1"
        button    catex2,        4,2, "ADD_WINDOW catex2  3  1"
        button    catex3,        6,2, "ADD_WINDOW catex3  3  1"
        button    fonsi1,        8,2, "ADD_WINDOW fonsi1  3  1"
        button    fonsi2,        10,2,      "ADD_WINDOW fonsi2  3  1"
        }

  SCREEN  viewchecklistscreen {
        title      "VIEW CHECKLIST"
        window     vwchklist,  2,1
        window     majoractions4,   18,1
```

```
      border     YES
      }

 ---------------------------------------------------------------------------

 SCREEN  probstatscreen {
    title       "ENVIRONMENTAL ASSESSMENT STATUS"
    mainscreen YES
    border      YES
    window      assessroom,      3,1
    window      probstat,        7,1
    window      chgassess,      16,1
    window      majoraction,    18,1
    }

 -END OF ASAN.SDF




 -INCLUDE STATEMENT FOR THE ASAN typedef DEFINITIONS NEEDED TO
 -        COMPILE USUBS.C WITHOUT INCURRING THE WRATH OF THE COMPILER
 ---------------------------------------------------------------------------

 INCLUDE ASANTYPE.H
 INCLUDE ASAN.H

 ---------------------------------------------------------------------------
 -  DECLARATIONS IN NON-LEXICAL OBJECT SEQUENCE ................
 ---------------------------------------------------------------------------

 - Multiple Choice Space
 -----------------------

 VARIABLE  muldat00  {
    foundin "VARCHAR30 deplmult[0].arr"
    type      STRING
    format    %-30s
    }

 DATUM  muldat00  {
    num_rows     1
    num_columns 32
    variable     muldat00, 0, 2
    pickable     NO
    }

 VARIABLE  muldat01 {
    foundin "VARCHAR30 deplmult[1].arr"
    type      STRING
    format    %-30s
    }

 DATUM  muldat01  {
    num_rows     1
    num_columns 32
    variable     muldat01, 0, 2
    pickable     NO
    }

 VARIABLE  muldat02  {
    foundin "VARCHAR30 deplmult[2].arr"
    type      STRING
    format    %-30s
    }

 DATUM  muldat02 {
    num_rows     2
    num_columns 32
    variable     muldat02, 0, 2
    pickable     NO
    }

 VARIABLE  muldat03  {
    foundin "VARCHAR30 deplmult[3].arr"
    type      STRING
    format    %-30s
    }

 DATUM  muldat03  {
```

```
    num_rows     1
    num_columns 32
    variable     muldat03, 0, 2
    pickable     NO
    }

  VARIABLE  muldat04  {
    foundin  "VARCHAR30 displmult[4].arr"
    type      STRING
    format    %-30s
    }

  DATUM  muldat04  {
    num_rows     1
    num_columns 32
    variable     muldat04, 0, 2
    pickable     NO
    }

  VARIABLE muldat05  {
    foundin  "VARCHAR30 displmult[5].arr"
    type      STRING
    format    %-30s
    }

  DATUM  muldat05  {
    num_rows     1
    num_columns 32
    variable     muldat05, 0, 2
    pickable     NO
    }

  VARIABLE muldat06  {
    foundin "VARCHAR30 displmult[6].arr"
    type      STRING
    format    %-30s
    }

  DATUM   muldat06 {
    num_rows     1
    num_columns 32
    variable     muldat06, 0, 2
    pickable     NO
    }

  VARIABLE muldat07  {
    foundin "VARCHAR30 displmult[7].arr"
    type      STRING
    format    %-30s
    }

  DATUM   muldat07 {
    num_rows     1
    num_columns 32
    variable     muldat07, 0, 2
    pickable     NO
    }

  VARIABLE muldat08  {
    foundin "VARCHAR30 displmult[8].arr"
    type      STRING
    format    %-30s
    }

  DATUM   muldat08 {
    num_rows     1
    num_columns 32
    variable     muldat08, 0, 2
    pickable     NO
    }

  VARIABLE muldat09  {
    foundin "VARCHAR30 displmult[9].arr"
    type      STRING
    format    %-30s
    }

  DATUM   muldat09 {
    num_rows     1
    num_columns 32
    variable     muldat09, 0, 2
    pickable     NO
```

```
    }

 VARIABLE muldat10  {
    foundin "VARCHAR30 deplmult[10].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat10 {
    num_rows     1
    num_columns 32
    variable     muldat10, 0, 2
    pickable     NO
    }

 VARIABLE muldat11  {
    foundin "VARCHAR30 deplmult[11].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat11 {
    num_rows     1
    num_columns 32
    variable     muldat11, 0, 2
    pickable     NO
    }

 VARIABLE muldat12  {
    foundin "VARCHAR30 deplmult[12].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat12  {
    num_rows     1
    num_columns 32
    variable     muldat12, 0, 2
    pickable     NO
    }

 VARIABLE muldat13  {
    foundin "VARCHAR30 deplmult[13].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat13    {
    num_rows     1
    num_columns 32
    variable     muldat13, 0, 2
    pickable     NO
    }

 VARIABLE muldat14  {
    foundin "VARCHAR30 deplmult[14].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat14 {
    num_rows    1
    num_columns    32
    variable   muldat14, 0, 2
    pickable   NO
    }

 VARIABLE muldat15  {
    foundin "VARCHAR30 deplmult[15].arr"
    type     STRING
    format  %-30s
    }

 DATUM   muldat15 {
    num_rows     1
    num_columns 32
    variable     muldat15, 0, 2
    pickable     NO
    }

 VARIABLE muldat16  {
    foundin "VARCHAR30 deplmult[16].arr"
```

```
    type     STRING
    format   %-30s
    }

 DATUM   muldat16 {
    num_rows     1
    num_columns 32
    variable     muldat16, 0, 2
    pickable     NO
    }

 VARIABLE muldat17  {
    foundin "VARCHAR30 displmult[17].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat17 {
    num_rows     1
    num_columns 32
    variable     muldat17, 0, 2
    pickable     NO
    }

 VARIABLE muldat18  {
    foundin "VARCHAR30 displmult[18].arr"
    type      STRING
    format    %-30s
    }

 DATUM   muldat18 {
    num_rows     1
    num_columns 32
    variable     muldat18, 0, 2
    pickable     NO
    }

 VARIABLE muldat19  {
    foundin "VARCHAR30 displmult[19].arr"
    type     STRING
    format   %-30s
    }

 DATUM   muldat19 {
    num_rows     1
    num_columns 32
    variable     muldat19, 0, 2
    pickable     NO
    }

- Planner attributes
--------------------

VARIABLE planrnam {
   foundin "VARCHAR planrnam.arr"
   type      STRING
   format    %-30s
}

DATUM planrnam {
   num_rows     1
   num_columns 50
   variable     planrnam, 0, 20
   leader       "Your name, please:"
   helpfile     help/username.hlp
}

VARIABLE password {
   type     STRING
   format   %-10s
}

DATUM password {
   num_rows     1
   num_columns 55
   variable     password, 0 , 23
   leader       "Please enter password:"
   helpfile     help/password.hlp
}

- Other General Storage for Verification, Tests, Etc.
- ----------------------------------------------------
```

```
VARIABLE  nowmtrnm {
     foundin        "VARCHAR n2bv.arr"
     type           STRING
     format         %-30s
}

- Noise Source Attributes
- ------------------------

VARIABLE  srcid {
   foundin "VARCHAR srcid.arr"
   type      STRING
   format    %-9s
   }

DATUM   mtrnam {
   num_rows       1
   num_columns    60
   variable       srcid, 0, 21
   leader         "Name of current MTR:"
   pickable       NO
   }

VARIABLE  srodesc {
   foundin "VARCHAR srodesc.arr"
   type      STRING
   format    %-54s
   }

DATUM   srods {
      num_rows      1
      num_columns   76
      variable      srodesc, 0, 22
      leader        "Description:          "
      helpfile      help/mtrdesc.hlp
      }

DATUM   mtrdesc {
      num_rows       1
      num_columns    70
      variable       srodesc, 0, 7
      leader         "Note: "
      pickable       NO
      }

VARIABLE  datepubl   {
     foundin   "VARCHAR srcpdate.arr"
     type       STRING
     format     %-12s
}

VARIABLE  schedule   {
   foundin   "VARCHAR srcsched.arr"
   type       STRING
   format     %-50s
   }

DATUM   schedule {
      num_rows     1
      num_columns  73
      variable     schedule, 0, 22
      leader       "Scheduling activity: "
      helpfile     help/schdmtr.hlp
      }

VARIABLE  srcorig   {
   foundin    "VARCHAR srcorig.arr"
   type       STRING
   format     %-50s
   }

DATUM   origmtr {
   num_rows  1
   num_columns   73
   variable      srcorig, 0, 22
   leader        "Originating activity:"
   helpfile      help/origmtr.hlp
   }

- MTR Attributes
- ---------------
```

```
VARIABLE  curartcc {                          - CURRENT Set
   foundin "VARCHAR curartcc.arr"
   type     STRING
   format   %3s
   }

DATUM curartcc {
      num_rows       1
      num_columns    27
      variable       curartcc,  0,16
      leader         "ARTCC:          "
      helpfile       help/curartcc.hlp
      }

VARIABLE  curwidright {
   type   INTEGER
   format %2d
   }

DATUM   curwidright {
      num_rows       1
      num_columns    25
      variable       curwidright,  0,16
      leader         "Width (right): "
      helpfile       help/curwidrt.hlp
      }

VARIABLE  curwidleft {
   type   INTEGER
   format %2d
   }

DATUM   curwidleft {
      num_rows       1
      num_columns    25
      variable       curwidleft,  0,16
      leader         "Width (left):  "
      helpfile       help/curwidlf.hlp
      }

VARIABLE  curhighalt {
   foundin   "ALTSPEC curhighalt.spec"
   type      STRING
   format    %9s
   }

DATUM   curhighalt {
      num_rows       1
      num_columns    27
      variable       curhighalt, 0,16
      leader         "High altitude: "
      helpfile       help/curhialt.hlp
      }

VARIABLE  curlowalt {
      foundin   "ALTSPEC curlowalt.spec"
      type      STRING
      format    %9s
}

VARIABLE  preartcc {                          - PREVIOUS Set
   foundin "VARCHAR preartcc.arr"
   type     STRING
   format   %3s
   }

   DATUM preartcc {
      num_rows       1
      num_columns    8
      variable       preartcc,  0,4
      pickable       NO
      }

VARIABLE  prewidright {
   type   INTEGER
   format %2d
   }

DATUM  prewidright {
      num_rows       1
      num_columns    8
      variable       prewidright,  0,4
```

```
        pickable       NO
        }

 VARIABLE  prewidleft {
    type           INTEGER
    format         %2d
    }

  DATUM  prewidleft {
        num_rows       1
        num_columns    8
        variable       prewidleft,  0,4
        pickable       NO
        }

 VARIABLE  prehighalt {
    foundin "ALTSPEC prehighalt.spec"
    type      STRING
    format    %9s
    }

 DATUM  prehighalt {
        num_rows       1
        num_columns    10
        variable       prehighalt, 0,1
        pickable       NO
        }

 VARIABLE  prelowalt {
      foundin    "ALTSPEC prelowalt.spec"
      type        STRING
      format      %9s
 }

 - Coordinates
 - -----------

 VARIABLE entlat  {                          - CURRENT or ENTER Set
      foundin     "COORDINATE ent.lat"
      type        STRING
      format      %-13s
 }

 VARIABLE entlong  {
      foundin     "COORDINATE ent.lon"
      type        STRING
      format      %-13s
 }

 VARIABLE shwlat  {                          - PREVIOUS or SHOW Set
      foundin     "COORDINATE show.lat"
      type        STRING
      format      %-13s
 }

 VARIABLE shwlong  {
      foundin     "COORDINATE show.lon"
      type        STRING
      format      %-13s
 }

 - Navigation Point Properties
 - ---------------------------

 VARIABLE  curfixtyp {                       - CURRENT or ENTER Set
      foundin    "VARCHAR curfixtyp.arr"
      type        STRING
      format      %12s
 }

 VARIABLE  curfixdist {
      type           INTEGER
      format         %3d
 }

 VARIABLE  curfixrad {
      type           INTEGER
      format         %3d
 }

 VARIABLE  curfixid {
      foundin     "VARCHAR curfixid.arr"
```

```
      type          STRING
      format        %5s
  }

  VARIABLE  curnavpt {
      foundin     "VARCHAR curnavpt.arr"
      type          STRING
      format        %3s
  }

  VARIABLE  prefixtyp {                          - PREVIOUS or DISPLAY Set
      foundin    "VARCHAR prefixtyp.arr"
      type        STRING
      format      %12s
  }

  VARIABLE  prefixdist {
      type          INTEGER
      format        %3d
  }

  VARIABLE  prefixrad {
      type          INTEGER
      format        %3d
  }

  VARIABLE  prefixid {
      foundin     "VARCHAR prefixid.arr"
      type          STRING
      format        %5s
  }

  VARIABLE  prenavpt {
      foundin     "VARCHAR prenavpt.arr"
      type          STRING
      format        %3s
  }

  - Aircraft Parameters
  - --------------------

  VARIABLE  acname  {
     foundin  "VARCHAR ac_name.arr"
     type     STRING
     format   %-12s
     }

  DATUM   acname {
       num_rows  1
       num_columns  25
       variable     acname, 0, 10
       leader       "Aircraft:"
       pickable     NO
       }

  VARIABLE  acnamel  {
     foundin "VARCHAR tid.arr"
     type     STRING
     format   %-12s
     }

  DATUM   acnamel {
       num_rows  1
       num_columns 35
       variable    acnamel, 0, 16
       leader      "Aircraft name:"
       helpfile    help/mtrainer.hlp
   }

  - Operations Specification
  - -------------------------

  VARIABLE   day  {                              - GENERIC or JANUARY
      foundin   "OPERATIONS ops[0].day"
      type      INTEGER
      format        %4d
  }

  VARIABLE   janday  {
      foundin   "OPERATIONS ops[0].day"
      type      INTEGER
      format        %4d
```

```
}

VARIABLE   night  {
     foundin   "OPERATIONS ops[0].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   jannite  {
     foundin   "OPERATIONS ops[0].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   febday  {
     foundin   "OPERATIONS ops[1].day"
     type      INTEGER
     format        %4d
}

VARIABLE   febnite  {
     foundin   "OPERATIONS ops[1].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   marday  {
     foundin   "OPERATIONS ops[2].day"
     type      INTEGER
     format        %4d
}

VARIABLE   marnite  {
     foundin   "OPERATIONS ops[2].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   aprday  {
     foundin   "OPERATIONS ops[3].day"
     type      INTEGER
     format        %4d
}

VARIABLE   aprnite  {
     foundin   "OPERATIONS ops[3].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   mayday  {
     foundin   "OPERATIONS ops[4].day"
     type      INTEGER
     format        %4d
}

VARIABLE   maynite  {
     foundin   "OPERATIONS ops[4].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   junday  {
     foundin   "OPERATIONS ops[5].day"
     type      INTEGER
     format        %4d
}

VARIABLE   junnite  {
     foundin   "OPERATIONS ops[5].nite"
     type      INTEGER
     format        %4d
}

VARIABLE   julday  {
     foundin   "OPERATIONS ops[6].day"
     type      INTEGER
     format        %4d
}

VARIABLE   julnite  {
     foundin   "OPERATIONS ops[6].nite"
```

```
        type        INTEGER
        format        %4d
    }

    VARIABLE   augday  {
        foundin   "OPERATIONS ops[7].day"
        type       INTEGER
        format        %4d
    }

    VARIABLE   augnite  {
        foundin   "OPERATIONS ops[7].nite"
        type       INTEGER
        format        %4d
    }

    VARIABLE   sepday  {
        foundin   "OPERATIONS ops[8].day"
        type       INTEGER
        format        %4d
    }

    VARIABLE   sepnite  {
        foundin   "OPERATIONS ops[8].nite"
        type       INTEGER
        format        %4d
    }

    VARIABLE   octday  {
        foundin   "OPERATIONS ops[9].day"
        type       INTEGER
        format        %4d
    }

    VARIABLE   octnite  {
        foundin   "OPERATIONS ops[9].nite"
        type       INTEGER
        format        %4d
    }

    VARIABLE   novday  {
        foundin   "OPERATIONS ops[10].day"
        type       INTEGER
        format        %4d
    }

    VARIABLE   novnite  {
        foundin   "OPERATIONS ops[10].nite"
        type       INTEGER
        format        %4d
    }

    VARIABLE   decday  {
        foundin   "OPERATIONS ops[11].day"
        type       INTEGER
        format        %4d
    }

    VARIABLE   decnite  {
        foundin   "OPERATIONS ops[11].nite"
        type       INTEGER
        format        %4d
    }

    -  Missions
    - ----------

    VARIABLE   misname  {
       foundin "VARCHAR misslabl.arr"
       type    STRING
       format  %-7s
       }

    DATUM  misnamel {
          num_rows      1
          num_columns   34
          variable      misname, 0, 26
          leader        "Name of current mission: "
          pickable      NO
          }

    VARIABLE   newmisname  {
```

```
        foundin       "VARCHAR aid.arr"
        type          STRING
        format        %-7s
        }

 DATUM  misname  {
        num_rows      1
        num_columns   24
        variable      newmisname, 0, 14
        leader        "Mission name:"
        helpfile      help/misname.hlp
       }

 TEXTLINE select  {
    "Right now you can type ? for help, <CTRL> C to quit, or move the cursor"}
 TEXTLINE envass   {"CONDUCT AN ENVIRONMENTAL ASSESSMENT" }
 TEXTLINE housekp  {"PERFORM DATABASE HOUSEKEEPING" }
 TEXTLINE introtxt {"VIEW GENERAL INFORMATION ABOUT THIS PROGRAM"}

 TEXTLINE selacmistxt   {"Select aircraft and mission for MTR"}
 TEXTLINE modcurmtrtxt {"Modify current MTR"}
 TEXTLINE chgmtrtxt    {"Select another MTR"}

 TEXTLINE nextnavptxt  {"Enter next navigation point"}
 TEXTLINE curprotxt
 { "CURRENT       PREVIOUS                    CURRENT       PREVIOUS"}
 TEXTLINE curprotxt3
 { "--------      --------                    -------       ---------"}
 TEXTLINE cancelmtrtxt {"Cancel this MTR data entry"}
 TEXTLINE savemtrtxt   {"Save this MTR"}

 TEXTLINE showmoremtrtxt {"Show more MTR names (if any)"}
 TEXTLINE recallmtrtxt   {"Recall one of the following MTRs:"}
 TEXTLINE stnewmtrtxt    {"Start new MTR "}
 TEXTLINE definemtrtxt  {"Enter route waypoints"}

 TEXTLINE actxt  {"Aircraft:"}

 TEXTLINE opseasontxt {"Operations are seasonal"}
 TEXTLINE opthruyrtxt {"Operations are even throughout year"}
 TEXTLINE instr1txt   {"Please enter day and night operations by month"}
 TEXTLINE daynite     {"  DAY   NIGHT  DAY   NIGHT  DAY   NIGHT  DAY  NIGHT"}
 TEXTLINE instr2txt   {"Please enter daytime and night operations per month"}
 TEXTLINE cancmistxt  {"Abandon this mission"}
 TEXTLINE entmisdbtxt {"Enter this mission into database"}
 TEXTLINE stnewmistxt  {"Start new mission"}

 TEXTBLOCK  introtxt  {
    num_rows      14
    num_columns   76
    border        YES
    filename      txtblk/intro.txt
  }

 BUTTON  assessment   {
    num_rows      1
    num_columns   38
    textline      envass, 0, 2
    helpfile      help/assess.hlp
    }

 BUTTON  chgmtr {
    num_rows       1
    num_columns    35
    textline       chgmtrtxt,  0, 2
    helpfile       help/selmtr.hlp
    }

 BUTTON  housekeeping {
    num_rows      1
    num_columns   33
    textline      housekp, 0, 2
    helpfile      help/housekp.hlp
    }

 BUTTON  introtxt {
    num_rows      1
    num_columns   56
    textline      introtxt, 0, 2
    helpfile      help/introhel.hlp
    }
```

```
BUTTON  modcurmtr {
   num_rows    1
   num_columns 35
   textline    modcurmtrtxt,  0, 2
   helpfile    help/nohelp.hlp
      - no help here cuz this button doesn't do anything
   }

WINDOW password {
    num_rows     1
    num_columns  60
    datum        password, 0, 5
   }

TEXTLINE tit1txt
{"Developed for Noise and Sonic Boom Impact Technology Program"}

TEXTLINE tit2txt {"under U.S. Air Force Contract F33615-86-C-0530"}

TEXTLINE tit3txt {"by BBN Laboratories, Inc."}

TEXTLINE tit4txt  {"February, 1988"}

TEXTLINE tit5txt
{"Unreleased demonstration of Prototype Version...Not for General Use"}

TEXTLINE diafo {"Done viewing general information on ASAN"}

BUTTON diafo {
    num_rows    1
    num_columns 76
    textline    diafo,  0, 2
}

WINDOW  introduction  {
    num_rows     15
    num_columns  78
    textblock    introtxt,   0, 1
    button       diafo, 14, 1, "REMOVE_WINDOW"
 }

WINDOW  introwindow1 {
    num_rows     21
    num_columns  78
    line         1, 0, 1, 77
    title        "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAN)"
    textline     tit1txt, 2, 9
    textline     tit2txt, 3, 16
    textline     tit3txt, 4, 27
    textline     tit4txt,  5, 32
    textline     tit5txt,  7, 6
    line         8,0, 8,77
    datum        plnnrnum, 12,  1,
                 "CALL VCAPITAL &plnnrnum",
                 "ADD_WINDOW password 15 3",
                 "UPDATE_DATUM password",
                 "REMOVE_WINDOW",
                 "CALL pwcheck"
 }

WINDOW  introwindow2  {
    num_rows     9
    num_columns  78
    line         5, 0, 5, 77
    textline     select,        6, 3
    button       introtxt,      7,15, "ADD_WINDOW introduction 3 1"
    button       assessment,    8, 2, "CALL peprobst"
    button       housekeeping,  8,44, "CALL pedbhsek"
 }

SCREEN  firstscreen {
   titlescreen  YES
   border       YES
   title        "ASSESSMENT SYSTEM FOR AIRCRAFT NOISE (ASAN)"
   window       introwindow1,  1,1
   window       introwindow2, 13,1
   }

BUTTON   opseason {
   num_rows    3
   num_columns 41
   textline    opseasontxt,  1, 3
```

```
    border       YES
    helpfile     help/vzblops.hlp
    }

  BUTTON   opsteady {
    num_rows     3
    num_columns 41
    textline     opthrwyrtxt,  1, 3
    border       YES
    helpfile     help/evenops.hlp
    }

  BUTTON mulbut00  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut01  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut02  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut03  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut04  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut05  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut06  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut07  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut08  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut09  {
    num_rows     1
    num_columns 1
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut10  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut11  {
    num_rows     1
    num_columns 2
    helpfile     help/mulbut.hlp
```

```
    }

  BUTTON  mulbut12  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
     }

  BUTTON  mulbut13  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
     }

  BUTTON  mulbut14  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
     }

  BUTTON  mulbut15  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
     }

  BUTTON  mulbut16  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut17  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut18  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
    }

  BUTTON  mulbut19  {
     num_rows     1
     num_columns 2
     helpfile     help/mulbut.hlp
    }

--------------------------------------------------------------------------------
-         Declarations for MTR DATA ENTRY SCREEN
--------------------------------------------------------------------------------

BUTTON    stnewmis {
     num_rows        1
     num_columns     40
     textline        stnewmistxt, 0 ,2
     helpfile        help/stnewmis.hlp
    }

BUTTON   selacmis {
     num_rows        1
     num_columns     40
     textline        selacmistxt,  0, 2
     helpfile        help/selacm.hlp
     }

TEXTLINE  specnwmistxt   {"Specify new mission"}

BUTTON   specnwmis {
     num_rows        1
     num_columns     30
     textline        specnwmistxt,  0, 2
     helpfile        help/newmiss.hlp
     }

WINDOW  mtrdatentry   {
   num_rows     8
   num_columns 78
   button       chgmtr,     0, 2, "CALL pschgmtr"
   button       selacmis,   2, 2, "CALL pmtrmis"
```

```
    button      modcurmtr, 4, 2, "CALL dummy"
    button      specnwmis, 6, 2, "ADD_WINDOW newmisnm  13 3",
                                 "UPDATE_DATUM misname",
                                 "CALL WHEREAMI Screen Window Datum Button",
                                 "CALL strcpy oldscreen Screen",
                                 "CALL psnwmisn cid.arr"

    }

WINDOW  mtrnamcom   {
    num_rows      3
    num_columns   78
    line          2,0, 2,77
    datum         mtrnam,  0, 2
    datum         mtrdesc, 1, 2
    }

WINDOW  mtrnamcom1   {
    num_rows      4
    num_columns   78
    line          3,0, 3,77
    datum         mtrnam,  0, 2
    datum         mtrdesc, 1, 2
    datum         misname1,  2, 2
    }

-----------------------------------------------------------------------------
-         Declarations for DEFINE/MODIFY SCREEN   (defmodmtrscreen)
-----------------------------------------------------------------------------

DATUM   prelowalt {
     num_rows      1
     num_columns   10
     variable      prelowalt, 0,1
     pickable      NO
  }

DATUM   curlowalt {
     num_rows      1
     num_columns   29
     variable      curlowalt,   0, 16
     leader        "Low altitude:  "
     helpfile      help/curloalt.hlp
  }

DATUM   prefixtype {
     num_rows      1
     num_columns   13
     variable      prefixtyp,  0,0
     pickable      NO
  }

DATUM   curfixtype {
     num_rows      1
     num_columns   24
     variable      curfixtyp,  0,11
     leader        "Fix type: "
     helpfile      help/curfixtp.hlp
  }

-----------------------------------------------------------------------------
-         Declarations for entering coordinates
-----------------------------------------------------------------------------

DATUM   entlong {
     num_rows   1
     num_columns    25
     variable   entlong, 0 ,11
     leader     "Longitude:"
     helpfile   help/markmap.hlp
  }

DATUM   entlat {
     num_rows   1
     num_columns    25
     variable   entlat, 0 ,11
     leader     "Latitude: "
     helpfile   help/markmap.hlp
  }

-----------------------------------------------------------------------------
-         Declarations for show coordinates
```

```
-----------------------------------------------------------------------

DATUM   shwlat {
     num_rows   1
     num_columns    14
     variable   shwlat, 0 ,0
     pickable   NO
     helpfile   help/combmap.hlp
  }

DATUM   shwlong {
     num_rows   1
     num_columns    14
     variable   shwlong, 0 ,0
     pickable   NO
     helpfile   help/combmap.hlp
  }

DATUM   prefixdist {
     num_rows      1
     num_columns   8
     variable      prefixdist,  0,4
     pickable      NO
  }

DATUM   curfixdist {
     num_rows      1
     num_columns   26
     variable      curfixdist,  0,16
     leader        "Fix distance: "
     helpfile      help/curfixdi.hlp
  }

DATUM   prefixrad {
     num_rows      1
     num_columns   8
     variable      prefixrad,  0,4
     pickable      NO
  }

DATUM   curfixrad {
     num_rows      1
     num_columns   26
     variable      curfixrad,  0,16
     leader        "Fix radial:   "
     helpfile      help/curfixra.hlp
  }

DATUM   prefixid {
     num_rows      1
     num_columns   10
     variable      prefixid, 0,4
     pickable      NO
  }

DATUM   curfixid {
     num_rows  1
     num_columns   26
     variable      curfixid, 0,16
     leader        "Fix ID:       "
     helpfile      help/curfixid.hlp
  }

DATUM   prenavpt {
     num_rows      1
     num_columns   10
     variable      prenavpt, 0,4
     pickable      NO
}

DATUM   curnavpt {
     num_rows      1
     num_columns   24
     variable      curnavpt,   0, 16
     leader        "Nav. Point:   "
     helpfile      help/curnavpt.hlp
  }

DATUM   newmtrnm  {
     num_rows      1
     num_columns   40
     variable      newmtrnm,   0,7
```

```
    leader        "Name:"
    helpfile      help/nxtmtrnm.hlp
  }

BUTTON  nextnavpt {
    num_rows   1
    num_columns    35
    textline   nextnavptxt,  0, 2
    helpfile   help/nxtnav.hlp
    }
```

```
TEXTLINE  retmtrentxt   {"Continue without selecting MFR"}
TEXTLINE  retmtrentxt1  {"Continue without creating mission"}
TEXTLINE  retmtrentxt2  {"Save mission in database"}
TEXTLINE  retnomis      {"Continue without selecting mission"}

BUTTON retmtrent {
   num_rows     1
   num_columns  40
   textline     retmtrentxt, 0 ,2
   helpfile     help/nomsmtr.hlp
   }

BUTTON retmtrent1 {
   num_rows     1
   num_columns  40
   textline     retmtrentxt1, 0 , 2
   helpfile     help/nomiss.hlp
   }

BUTTON retmtrent2 {
   num_rows    1
   num_columns 30
   textline    retmtrentxt2, 0 , 2
   helpfile    help/savemiss.hlp
   }

BUTTON  retnomis {
   num_rows    1
   num_columns 40
   textline    retnomis, 0 , 2
   helpfile    help/nomsmis.hlp
   }
```

```
WINDOW  defmodmtr   {
      num_rows      15
      num_columns   78
      textline     curpretxt,     0, 16
      textline     curpretxt3,    1, 16
      datum        curnavpt,      2,  1, "CALL VCAPITAL &curnavpt",
                                          "NEWVALS"
      datum        prenavpt,      2, 26
      datum        curfixid,      3,  1, "CALL VCAPITAL &curfixid",
                                          "NEWVALS"
      datum        prefixid,      3, 26
      datum        curfixrad,     4,  1
      datum        prefixrad,     4, 26
      datum        curfixdist,    5,  1
      datum        prefixdist,    5, 26
      datum        entlat,        2, 38,  "CALL lat2dec &ent",
                                          "NEWVALS"
      datum        shwlat,        2, 64
      datum        entlong,       3, 38,  "CALL lon2dec &ent",
                                          "NEWVALS"
      datum        shwlong,       3, 64
      datum        curfixtype,    4, 38,  "CALL VCAPITAL &curfixtyp",
                                          "NEWVALS"
      datum        prefixtype,    4, 64
      datum        curlowalt,     7,  2,  "CALL alt2dec &curlowalt",
                                          "NEWVALS"
      datum        prelowalt,     7, 26
      datum        curhighalt,    8,  2,  "CALL alt2dec &curhighalt",
                                          "NEWVALS"
      datum        prehighalt,    8, 26
      datum        curwidleft,    9,  2
      datum        prewidleft,    9, 27
      datum        curwidright,  10, 2
      datum        prewidright,  10, 27
```

```
      datum        curartcc,     11,  2,  "CALL VCAPITAL &curartcc",
                                           "NEWVALS"
      datum        preartcc,     11, 26
      button       nextnavpt,    13,  2,  "CALL nxtmtrpt"
     }

BUTTON    cancelmtr {
     num_rows   1
     num_columns    35
     textline   cancelmtrtxt,  0, 2
     helpfile   help/canclmtr.hlp
     }

BUTTON   savemtr {
     num_rows   1
     num_columns    20
     textline   savemtrtxt,  0, 2
     helpfile   help/savemtr.hlp
     }

WINDOW  mtraction   {
      num_rows      2
      num_columns   78
      line          0,0, 0,77
      button        savemtr,    1, 2, "CALL savemtr"
      button        cancelmtr,  1,40, "CALL cancmtr"
     }

SCREEN  defmodmtrscreen {
     title      "DEFINE/MODIFY MTR"
     window     mtrnamcom,  2, 1
     window     defmodmtr,  6, 1
     window     mtraction, 20, 1
     border     YES
     }

-----------------------------------------------------------------------
-                Declarations for SELECT ANOTHER MTR SCREEN
-----------------------------------------------------------------------

BUTTON    showmoremtr {
     num_rows   1
     num_columns    37
     textline   showmoremtrtxt, 0 ,2
     helpfile   help/shmrmtrs.hlp
  }

BUTTON    stnewmtr {
     num_rows   1
     num_columns    40
     textline   stnewmtrtxt, 0 ,2
     helpfile   help/startmtr.hlp
    }

WINDOW    newmtrnm {
   num_rows      1
   num_columns   40
   datum         newmtrnm,  0, 2
   }

WINDOW    strecalmtr {
   num_rows      4
   num_columns   78
   button        stnewmtr, 1, 2,
                 "ADD_WINDOW newmtrnm 6  3",
                 "UPDATE_DATUM newmtrnm",
                 "CALL psnwmtrn n2bv.arr"
   button        showmoremtr,  1, 41,
                 "CALL msbunch"
   textline      recallmtrtxt, 3, 2
   }

WINDOW  curdat  {
    num_rows     10
    num_columns  32
    datum  muldat00, 0, 0
    datum  muldat01, 1, 0
    datum  muldat02, 2, 0
    datum  muldat03, 3, 0
    datum  muldat04, 4, 0
    datum  muldat05, 5, 0
    datum  muldat06, 6, 0
```

```
    datum  muldat07, 7, 0
    datum  muldat08, 8, 0
    datum  muldat09, 9, 0
  }

 WINDOW    curdat1  {
    num_rows      10
    num_columns  32
    datum  muldat10, 0, 0
    datum  muldat11, 1, 0
    datum  muldat12, 2, 0
    datum  muldat13, 3, 0
    datum  muldat14, 4, 0
    datum  muldat15, 5, 0
    datum  muldat16, 6, 0
    datum  muldat17, 7, 0
    datum  muldat18, 8, 0
    datum  muldat19, 9, 0
  }

 WINDOW  curmtrbut  {
    num_rows       10
    num_columns   2
    button  mulbut00, 0, 0, "CALL MTRconn dsplmult[0].arr"
    button  mulbut01, 1, 0, "CALL MTRconn dsplmult[1].arr"
    button  mulbut02, 2, 0, "CALL MTRconn dsplmult[2].arr"
    button  mulbut03, 3, 0, "CALL MTRconn dsplmult[3].arr"
    button  mulbut04, 4, 0, "CALL MTRconn dsplmult[4].arr"
    button  mulbut05, 5, 0, "CALL MTRconn dsplmult[5].arr"
    button  mulbut06, 6, 0, "CALL MTRconn dsplmult[6].arr"
    button  mulbut07, 7, 0, "CALL MTRconn dsplmult[7].arr"
    button  mulbut08, 8, 0, "CALL MTRconn dsplmult[8].arr"
    button  mulbut09, 9, 0, "CALL MTRconn dsplmult[9].arr"
  }

 WINDOW    curmtrbut1  {
    num_rows       12
    num_columns   2
    button  mulbut10,   0, 0, "CALL MTRconn dsplmult[10].arr"
    button  mulbut11,   1, 0, "CALL MTRconn dsplmult[11].arr"
    button  mulbut12,   2, 0, "CALL MTRconn dsplmult[12].arr"
    button  mulbut13,   3, 0, "CALL MTRconn dsplmult[13].arr"
    button  mulbut14,   4, 0, "CALL MTRconn dsplmult[14].arr"
    button  mulbut15,   5, 0, "CALL MTRconn dsplmult[15].arr"
    button  mulbut16,   6, 0, "CALL MTRconn dsplmult[16].arr"
    button  mulbut17,   7, 0, "CALL MTRconn dsplmult[17].arr"
    button  mulbut18,   8, 0, "CALL MTRconn dsplmult[18].arr"
    button  mulbut19,   9, 0, "CALL MTRconn dsplmult[19].arr"
    }

 WINDOW retmtrent {
    num_rows     1
    num_columns 45
    button       retmtrent, 0, 1, "CALL mslistc",
                                  "CALL psmtrent"
    }

 SCREEN  chgcurmtrscreen {
     title   "SELECT ANOTHER MTR"
     window  mtrnamcom,  2, 1
     window  strecalmtr, 5, 1
     window  curmtrbut,  9, 3
     window  curmtrbut1, 9,42
     window  curdat,  9, 5
     window  curdat1, 9,44
     window  retmtrent, 20, 2
     border  YES
     }

-----------------------------------------------------------------------------
-                        Declarations for  new  mtr name
-----------------------------------------------------------------------------

DATUM  datepubl {
     num_rows  1
     num_columns   50
     variable      datepubl, 0, 22
     leader        "Date of publication: "
     helpfile  help/datepubl.hlp
  }

BUTTON  definemtr {
```

```
     num_rows   1
     num_columns    40
     textline   definemtrtxt, 0 ,2
     helpfile   help/getmap.hlp
    }

WINDOW  defnumtr   {
      num_rows      9
      num_columns   77
      datum         origmtr,    0, 1
      datum         schedule,   2, 1
      datum         srcds,      4, 1
      datum         datepubl,   6, 1
      button        definemtr,  8, 1,        "CALL entmtrpt"
     }

SCREEN  mtrdefinescreen  {
     title      "MTR DEFINITION"
     window     mtrnamcom,       3,1
     window     defnumtr,        7,2
     window     mtraction,      20,1
     border     YES
     }

-----------------------------------------------------------------------
-                  Declarations for  MISSION REQUIREMENTS WINDOW
-----------------------------------------------------------------------

VARIABLE   ac_in_form  {
     type          INTEGER
     format         %2d
     lowlimit       1
     uplimit        16
     default        2
  }

VARIABLE   pwrstunt  {
     foundin       "VARCHAR pr_pwr_u.arr"
     type          STRING
     format        %-6s
  }

DATUM   pwrstunt {
   num_rows       1
   num_columns    10
   variable       pwrstunt, 0, 2
   leader         "["
   trailer        "]"
   pickable       NO
   }

VARIABLE   mistype  {
     type          STRING
     format         %-1s
  }

DATUM   mistype1  {
     num_rows       1
     num_columns    18
     variable       mistype, 0, 14
     leader         "Mission type:"
     helpfile       help/mistype.hlp
  }

DATUM   mistype {
     num_rows  1
     num_columns    18
     variable       mistype, 0, 14
     leader         "Mission type:"
     pickable       NO
  }

VARIABLE  prealtlev   {
     foundin        "ALTSPEC prelowalt.spec"
     type           STRING
     format         %9s
  }

DATUM   prealtlev {
     num_rows       1
     num_columns    10
     variable       prealtlev, 0, 0
```

```
      pickable      NO
    }

  VARIABLE  curaltlev   {
      foundin       "ALTSPEC curlowalt.spec"
      type          STRING
      format        %9s
    }

  DATUM   curaltlev {
      num_rows      1
      num_columns   21
      variable      curaltlev, 0, 11
      leader        "Alt:      "
      helpfile      help/altmtr.hlp
    }

  VARIABLE  ac_pre_pwr   {
      type          DOUBLE
      format        %12.31f
      lowlimit      0.0
      uplimit       4000.0
      default       100.0
    }

  DATUM   prepwrset {
      num_rows      1
      num_columns   12
      variable      ac_pre_pwr, 0, 0
      pickable      NO
    }

  VARIABLE  ac_cur_pwr   {
      type          DOUBLE
      format        %10.31f
      lowlimit      0.0
      uplimit       4000.0
      default       100.0
    }

  DATUM   curpwrset {
      num_rows      1
      num_columns   21
      variable      ac_cur_pwr, 0, 11
      leader        "Power:    "
      helpfile      help/pwrspec.hlp
    }

  VARIABLE  ac_pre_spd   {
      type          INTEGER
      format        %3d
    }

  DATUM   prespeed {
      num_rows      1
      num_columns   9
      variable      ac_pre_spd, 0, 0
      trailer       "KTS"
      pickable      NO
    }

  VARIABLE  ac_cur_spd   {
      type          INTEGER
      format        %3d
      lowlimit      100
      uplimit       600
      default       450
    }

  DATUM   curspeed {
      num_rows      1
      num_columns   21
      variable      ac_cur_spd, 0, 11
      leader        "Speed:    "
      trailer       "KTS"
      helpfile      help/mtrspeed.hlp
    }

  DATUM   premavpt1 {
      num_rows      1
      num_columns   3
      variable      premavpt, 0,0
```

```
      picktable      NO
   }

 DATUM   curnavpt1 {
      num_rows       1
      num_columns    21
      variable       curnavpt,   0, 11
      leader         "Navpoint:"
      helpfile       help/navname.hlp
   }

 TEXTLINE  cancelparatxt  {"Cancel data entry for this mission"}

 BUTTON    cancelpara {
      num_rows       1
      num_columns    35
      textline       cancelparatxt,  0, 2
      helpfile       help/cancelmtr.hlp
      }

 TEXTLINE  saveparatxt  {"Save this mission's data"}

 BUTTON   savepara {
      num_rows       1
      num_columns    30
      textline       saveparatxt,  0, 2
      helpfile       help/savemiss.hlp
      }

 TEXTLINE  operdatatxt  {"Ready to enter operational data"}

 BUTTON   operdata {
      num_rows       1
      num_columns    33
      textline       operdatatxt,  0, 2
      helpfile       help/entrops.hlp
      }

 TEXTBLOCK  mtr  {
    num_rows     10
    num_columns  38
    filename     txtblk/mtr.txt
 }

 TEXTLINE  mtrlabeltxt  {"NAV     FIX        FIX TYPE      WIDTH"}

 TEXTLINE  mtrtxt  {"REFERENCE INFORMATION FOR DATA ENTRY"}

 TEXTLINE  curpretxt1  {"CURRENT     PREVIOUS"}

 TEXTLINE  curpretxt2  {"--------     ---------"}

 WINDOW opentry {
    num_rows          10
    num_columns       78
    button opsteady,   2, 19, "CALL psaddops  1"
    button opseasca,   7, 19, "CALL psaddops 12"
 }

 WINDOW  fltpara  {
      num_rows       17
      num_columns    78
      datum          mistype,      0, 2
      datum          acname,       1, 2
      textline       mtrtxt,       0,40
      textline       mtrlabeltxt,  2,38
      textline       curpretxt1,   3,14
      textline       curpretxt2,   4,14
      datum          curnavpt1,    5, 1,  "CALL VCAPTEAL &curnavpt",
                                            "RENVALS"
      datum          prenavpt1,    5,25
      datum          curspeed,     7, 1
      datum          prespeed,     7,25
      datum          curpwrset,    9, 2
      datum          prepwrset,    9,22
      datum          pwrstunt,     10, 15
      datum          curaltlev,    12, 2, "CALL alt2dec &curlowalt",
                                            "RENVALS"
      datum          prealtlev,    12,23
      box            1, 37, 14, 77
      line           3, 38,  3, 76
      line           15, 0, 15, 77
```

```
      button        nextnavpt,  14, 2, "CALL nxtmispt"
      button        opardata,   16, 2, "NEW_SCREEN  opsntry"
      button        canclpara,  16,40, "CALL cancmis"
      textblock     mtr,         4,38
     }

SCREEN  mtrflt  {
     title      "FLIGHT PARAMETER ENTRY"
     window     mtrnamcom,      2,1
     window     fltpara,        5,1
     border     YES
     }

SCREEN  opsntry  {
     title      "FLIGHT OPERATION DATA ENTRY FOR MTR"
     window     mtrnamcom,      3,1
     window     opsntry,        6,1
     border     YES
     }

-----------------------------------------------------------------------
-              Declarations for seasonal MTR and REQUIREMENTS
-----------------------------------------------------------------------

DATUM   decnite {
     num_rows  1
     num_columns   6
     variable      decnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   decday {
     num_rows  1
     num_columns   10
     variable      decday, 0, 5
     leader        "DEC:"
     helpfile      help/mnthday.hlp
  }

DATUM   novnite {
     num_rows  1
     num_columns   6
     variable      novnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   novday {
     num_rows  1
     num_columns   10
     variable      novday, 0, 5
     leader        "NOV:"
     helpfile      help/mnthday.hlp
  }

DATUM   octnite {
     num_rows  1
     num_columns   6
     variable      octnite, 0, 0
     helpfile      help/mnthnite.hlp
  }
DATUM   octday {
     num_rows  1
     num_columns   10
     variable      octday, 0, 5
     leader        "OCT:"
     helpfile      help/mnthday.hlp
  }

DATUM   sepnite {
     num_rows  1
     num_columns   6
     variable      sepnite, 0, 0
     helpfile      help/mnthnite.hlp
  }

DATUM   sepday {
     num_rows  1
     num_columns   10
     variable      sepday, 0, 5
     leader        "SEP:"
     helpfile      help/mnthday.hlp
  }
```

```
DATUM   augnite {
        num_rows  1
        num_columns   6
        variable      augnite, 0, 0
        helpfile      help/mnthnite.hlp
    }

DATUM   augday {
        num_rows  1
        num_columns   10
        variable      augday, 0, 5
        leader        "AUG:"
        helpfile      help/mnthday.hlp
    }

DATUM   julnite {
        num_rows  1
        num_columns   6
        variable      julnite, 0, 0
        helpfile      help/mnthnite.hlp
    }

DATUM   julday {
        num_rows  1
        num_columns   10
        variable      julday, 0, 5
        leader        "JUL:"
        helpfile      help/mnthday.hlp
    }

DATUM   junnite {
        num_rows  1
        num_columns   6
        variable      junnite, 0, 0
        helpfile      help/mnthnite.hlp
    }

DATUM   junday {
        num_rows  1
        num_columns   10
        variable      junday, 0, 5
        leader        "JUN:"
        helpfile      help/mnthday.hlp
    }

DATUM   maynite {
        num_rows  1
        num_columns   6
        variable      maynite, 0, 0
        helpfile      help/mnthnite.hlp
    }

DATUM   mayday {
        num_rows  1
        num_columns   10
        variable      mayday, 0, 5
        leader        "MAY:"
        helpfile      help/mnthday.hlp
    }

DATUM   aprnite {
        num_rows  1
        num_columns   6
        variable      aprnite, 0, 0
        helpfile      help/mnthnite.hlp
    }
DATUM   aprday {
        num_rows  1
        num_columns   10
        variable      aprday, 0, 5
        leader        "APR:"
        helpfile      help/mnthday.hlp
    }

DATUM   marnite {
        num_rows  1
        num_columns   6
        variable      marnite, 0, 0
        helpfile      help/mnthnite.hlp
    }

DATUM   marday {
```

```
    num_rows  1
    num_columns   10
    variable      marday, 0, 5
    leader        "MAR:"
    helpfile      help/mnthday.hlp
  }

DATUM   febnite {
    num_rows  1
    num_columns   6
    variable      febnite, 0, 0
    helpfile      help/mnthnite.hlp
  }

DATUM   febday {
    num_rows  1
    num_columns   10
    variable      febday, 0, 5
    leader        "FEB:"
    helpfile      help/mnthday.hlp
  }

DATUM   jannite {
    num_rows  1
    num_columns   6
    variable      jannite, 0, 0
    helpfile      help/mnthnite.hlp
  }

DATUM   janday {
    num_rows  1
    num_columns   10
    variable      janday, 0, 5
    leader        "JAN:"
    helpfile      help/mnthday.hlp
  }

WINDOW    month  {
    num_rows   14
    num_columns    77
    textline   instritxt,  1, 1
    textline   daynite,    3, 0
    datum  janday,     4, 1
    datum  jannite,    4,12
    datum  febday,     5, 1
    datum  febnite,    5,12
    datum  marday,     6, 1
    datum  marnite,    6,12
    datum  aprday,     4,20
    datum  apnaite,    4,31
    datum  mayday,     5,20
    datum  maynite,    5,31
    datum  junday,     6,20
    datum  junnite,    6,31
    datum  julday,     4,39
    datum  julnite,    4,50
    datum  augday,     5,39
    datum  augnite,    5,50
    datum  sepday,     6,39
    datum  sepnite,    6,50
    datum  octday,     4,58
    datum  octnite,    4,69
    datum  novday,     5,58
    datum  novnite,    5,69
    datum  decday,     6,58
    datum  decnite,    6,69
    button savepara,  13, 2, "CALL savemis"
    button canclpara, 13,40, "CALL cancmis"
    }

-      Declarations for new SELECT AIRCRAFT AND MISSION FOR MFR

TEXTLINE  shwmoremistxt  {"Show more mission names (if any)"}

BUTTON   shwmoremis {
    num_rows       1
    num_columns    38
    textline       shwmoremistxt,  0, 2
    helpfile       help/shmrmiss.hlp
    }
```

```
TEXTLINE  recalmis  {"Recall one of the following missions"}

WINDOW  newmisnm {
   num_rows       1
   num_columns    40
   datum          misname,  0, 2
   }

WINDOW  curmiss  {
    num_rows      16
    num_columns   78
    button  stnewmis,     1, 2, "ADD_WINDOW newmisnm  7 3",
                                "UPDATE_DATUM misname",
                                "CALL psnewmisn cid.arr"
    button  shumoremis,  1, 40, "CALL ssbunch"
    textline   recalmis, 3, 2
    datum  muldat00,     4, 4
    datum  muldat01,     5, 4
    datum  muldat02,     6, 4
    datum  muldat03,     7, 4
    datum  muldat04,     8, 4
    datum  muldat05,     9, 4
    datum  muldat06,     10, 4
    datum  muldat07,     11, 4
    datum  muldat08,     12, 4
    datum  muldat09,     13, 4
    datum  muldat10,     4,42
    datum  muldat11,     5,42
    datum  muldat12,     6,42
    datum  muldat13,     7,42
    datum  muldat14,     8,42
    datum  muldat15,     9,42
    datum  muldat16,     10,42
    datum  muldat17,     11,42
    datum  muldat18,     12,42
    datum  muldat19,     13,42
    button  mulbut00,    4, 2, "CALL MISscan displmult[0].arr"
    button  mulbut01,    5, 2, "CALL MISscan displmult[1].arr"
    button  mulbut02,    6, 2, "CALL MISscan displmult[2].arr"
    button  mulbut03,    7, 2, "CALL MISscan displmult[3].arr"
    button  mulbut04,    8, 2, "CALL MISscan displmult[4].arr"
    button  mulbut05,    9, 2, "CALL MISscan displmult[5].arr"
    button  mulbut06,    10,2, "CALL MISscan displmult[6].arr"
    button  mulbut07,   11, 2, "CALL MISscan displmult[7].arr"
    button  mulbut08,   12, 2, "CALL MISscan displmult[8].arr"
    button  mulbut09,   13, 2, "CALL MISscan displmult[9].arr"
    button  mulbut10,    4,40, "CALL MISscan displmult[10].arr"
    button  mulbut11,    5,40, "CALL MISscan displmult[11].arr"
    button  mulbut12,    6,40, "CALL MISscan displmult[12].arr"
    button  mulbut13,    7,40, "CALL MISscan displmult[13].arr"
    button  mulbut14,    8,40, "CALL MISscan displmult[14].arr"
    button  mulbut15,    9,40, "CALL MISscan displmult[15].arr"
    button  mulbut16,    10,40, "CALL MISscan displmult[16].arr"
    button  mulbut17,   11,40, "CALL MISscan displmult[17].arr"
    button  mulbut18,   12,40, "CALL MISscan displmult[18].arr"
    button  mulbut19,   13,40, "CALL MISscan displmult[19].arr"
    button  retnomis,   15, 2, "CALL sslistc",
                                "CALL NEW_SCREEN oldscreen"
    }

TEXTLINE  selmisstxt  {"Select mission"}

BUTTON   selmiss {
     num_rows       1
     num_columns    38
     textline       selmisstxt,  0, 2
     helpfile       help/selmiss.hlp
     }

TEXTLINE  fltparinftxt  {"Specify flight parameter information"}

BUTTON   fltparinf {
     num_rows       1
     num_columns    37
     textline       fltparinftxt,  0, 2
     helpfile       help/fltparam.hlp
     }

WINDOW    selnomiss  {
     num_rows           15
     num_columns        77
     datum  acnamel,    1,1, "CALL vfynomtr tid.arr"
```

```
     button selmiss,    3,2, "CALL pschgmis"
     button fltparinf,  5,2, "CALL psmtrflt"
     button retmtrent1, 13,2, "CALL cancmis"
     }

SCREEN  spemtrmis  {
     title      "SELECT AIRCRAFT AND MISSION FOR MTR"
     window     mtrnamcom1, 3,1
     window     selacmiss,  7,1
     border     YES
     }

SCREEN  chgmis  {
     title      "SELECT AIRCRAFT AND MISSION FOR MTR"
     window     mtrnamcom1, 2,1
     window     curmiss,    6,1
     border     YES
     }

-----------------------------------------------------------------------------
-        Declarations for Day-Night window
-----------------------------------------------------------------------------

DATUM   night {
     num_rows      1
     num_columns   16
     variable      night, 0, 8
     leader        "Night:"
     helpfile      help/niteops.hlp
  }

DATUM   day {
     num_rows      1
     num_columns   16
     variable      day, 0, 6
     leader        "Day:"
     helpfile      help/dayops.hlp
  }

WINDOW    daynite  {
     num_rows    12
     num_columns 77
     datum    acname,     1, 1
     textline instr2txt,  3, 1
     datum    day,        5, 2
     datum    night,      5,15
     button   savepara,  10, 2, "CALL expmops",
                                "CALL savemis"
     button   canclpara, 10,40, "CALL cancmis"
     border   YES
     }

-----------------------------------------------------------------------------
-        Declarations for mission specification window
-----------------------------------------------------------------------------

DATUM   sortie {
     num_rows      1
     num_columns   37
     variable      ac_in_form, 0, 35
     leader        "Number of aircraft in formation:  "
     helpfile      help/numform.hlp
  }

VARIABLE  misdesc {
   foundin "VARCHAR misdesc.arr"        ~this description is for mission
   type     STRING
   format   %-60s
   }

DATUM   misdesc1 {
   num_rows      1
   num_columns   76
   variable      misdesc, 0, 7
   leader        "Descr:"
   pickable      NO
   }

DATUM   misdesc {
   num_rows      1
   num_columns   76
   variable      misdesc, 0, 14
```

```
    leader        "Description:"
    helpfile      help/misdesc.hlp
    }

 WINDOW  mtrnamcom3   {
    num_rows      3
    num_columns   78
    line          2,0, 2,77
    datum         misname1,  0, 2
    datum         misdesc1, 1, 2
    }

 WINDOW    misspec  {
    num_rows              16
    num_columns           77
    datum   misdesc,      1,1
    datum   mistype1,     3,2
    datum   sortie,       5,2
    button  retmtrent1, 14,2,  "CALL NEW_SCREEN oldscreen"
    button  retmtrent2, 14,47, "CALL entermis"
  }

 SCREEN  misspec  {
      title      "MISSION SPECIFICATION FOR AN MTR"
      window     mtrnamcom3, 3,1
      window     misspec,    6,1
      border     YES
      }

 ~01-14-88  st   will try to fix this file so it wil be only for assessments
 ~01-14-88  st   changes will be commented with date
 ~01-16-88  st   change title on first screen, change lat. long. format lon
 ~               map control
 ~01-17-88  st   con't to fix sdf file  commented with 1-17-88

 ------------------------------------------------------------------------
 ~ DECLARATIONS FOR SCREEN HEADER
 ------------------------------------------------------------------------

 VARIABLE assename {
     foundin  "ASANHEADR ASSESSMENT.name"
     type     STRING
     format   %-30s
  }

 DATUM  assename {
     num_rows     1
     num_columns  50
     variable     assename, 0 , 18
     leader       "Assessment name: "
     pickable     NO
  }

 DATUM  assename1 {
     num_rows     1
     num_columns  75
     variable     assename, 0 , 30
     leader       "Name of current assessment: "
     pickable     NO
  }

 VARIABLE comment {
     foundin  "ASANHEADR ASSESSMENT.desc"
     type     STRING
     format   %-66s
  }

 DATUM  comment {
     num_rows     1
     num_columns  76
     variable     comment, 0 , 10
     leader       "Comment: "
     pickable     NO
  }

 VARIABLE  curmapnam {
     type   STRING
     format %-30s
  }

 DATUM  curmapnam {
     num_rows    1
```

```
    num_columns 77
    variable    curmapnam, 0 , 19
    leadar      "Current map name: "
    pickable    NO
 }

WINDOW lin {
    num_rows     1
    num_columns  78
    line         0, 0, 0,77
 }

WINDOW assnmcom {
    num_rows     3
    num_columns  78
    datum        assnamel,  0,1
    datum        comment,  1, 2
    line         2, 0, 2,77
 }

WINDOW assnmcoml {
    num_rows     4
    num_columns  78
    datum        assnamel, 0, 0
    datum        comment, 1, 1
    datum        curmapnam, 2, 1
    line         3, 0, 3,77
 }

-----------------------------------------------------------------------
-   Declarations for buttons used in majoraction footer
-----------------------------------------------------------------------

TEXTLINE probstatxt { "REVIEW CURRENT ASSESSMENT STATUS" }

BUTTON probstat {
    num_rows     1
    num_columns  35
    textline     probstatxt,  0, 2
    helpfile     help/probstat.hlp
 }

TEXTLINE probstatxt2 { "REVIEW ASSESSMENT STATUS" }

BUTTON probstat2 {
    num_rows     1
    num_columns  26
    textline     probstatxt2,  0, 2
    helpfile     help/revstat.hlp
 }

TEXTLINE probdef {"ADD INFORMATION TO CURRENT ASSESSMENT" }

BUTTON probdef {
    num_rows     1
    num_columns  39
    textline     probdef,  0, 2
    helpfile     help/probdef.hlp
 }

TEXTLINE probdef1 {"ADD TO ASSESSMENT DEFINITION" }

BUTTON probdef1 {
    num_rows     1
    num_columns  31
    textline     probdef1,  0, 2
    helpfile     help/probdef.hlp
 }

TEXTLINE analysis { "ANALYZE DATA" }

BUTTON analysis {
    num_rows     1
    num_columns  15
    textline     analysis,  0, 2
    helpfile     help/datanal.hlp
 }

TEXTLINE reportgen { "MAKE A REPORT" }

BUTTON reportgen {
    num_rows     1
```

```
    num_columns  16
    textline     reportgen,  0, 2
    helpfile     help/reportgen.hlp
 }

 TEXTLINE vwchklisttxt { "VIEW CHECKLIST FOR CURRENT ASSESSMENT" }

 BUTTON  vwchklist  {
    num_rows     1
    num_columns  42
    textline     vwchklisttxt,  0, 2
    helpfile     help/viewlist.hlp
 }

 ---------------------------------------------------------------------------
 -         Declarations for majoraction footers
 ---------------------------------------------------------------------------

 WINDOW  majoraction  {        -header for problem status screen
    num_rows      4
    num_columns   78
    line          0, 0, 0,77
    title         "Alternative actions you can now take:"
    button        probdef,      2, 2, "NEW_SCREEN probdefscreen"
    button        analysis,     2,56, "NEW_SCREEN analysiscreen"
    button        vwchklist,    3, 2, "NEW_SCREEN viewchecklistscreen"
    button        reportgen,    3,56, "CALL dummy2"
 }

 WINDOW  majoraction1  {       -header for problem definition screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       analysis,     2, 2, "NEW_SCREEN analysiscreen"
    button       probstat2,    2,48, "CALL peprobst"
    button       vwchklist,    3, 2, "NEW_SCREEN viewchecklistscreen"
    button       reportgen,    3,48, "CALL dummy2"
 }

 WINDOW  majoraction2  {       -header for data analysis screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       probdef1,     2, 2, "NEW_SCREEN probdefscreen"
    button       reportgen,    2,47, "CALL dummy2"
    button       vwchklist,    3, 2, "NEW_SCREEN viewchecklistscreen"
    button       probstat2,    3,47, "CALL peprobst"
 }

 WINDOW  majoraction3  {       -header for report generation screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       vwchklist,    2, 2, "NEW_SCREEN viewchecklistscreen"
    button       probstat2,    2,47, "CALL peprobst"
    button       probdef1,     3, 2, "NEW_SCREEN probdefscreen"
    button       analysis,     3,47, "NEW_SCREEN analysiscreen"
 }

 WINDOW  majoraction4  {       -header for view checklist screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       probstat2,    2, 2, "CALL peprobst"
    button       reportgen,    2,47, "CALL dummy2"
    button       probdef1,     3, 2, "NEW_SCREEN probdefscreen"
    button       analysis,     3,47, "NEW_SCREEN analysiscreen"
 }


TEXTLINE loadglobal {"LOAD LOCAL DATA FOR GENERAL ACCESS"}

BUTTON  loadglobal   {
   num_rows      1
   num_columns   39
   textline      loadglobal, 0, 2
   helpfile      help/loadglob.hlp
   }
```

```
WINDOW  majoraction5  {        ~footer for database housekeeping screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       loadglobal,  2, 2, "CALL dummy"
    button       assessment,  3, 2, "CALL psprobst"
 }

SCREEN  mtrntry {
   title     "MTR DATA ENTRY"
   window    mtrnamcom,      3,1
   window    mtrdatentry,    7,1
   window    majoraction3, 18,1
   border    YES
   }

~------------------------------------------------------------------------
~                Declarations for problem status screen
~------------------------------------------------------------------------

VARIABLE  startdate {
    foundin "VARCHAR startdate.arr"
    type     STRING
    format   %-21s
 }

DATUM  startdate {
    num_rows     1
    num_columns 40
    variable     startdate, 0 ,15
    leader       "Date started: "
    pickable     NO
 }

VARIABLE  lastdate {
    foundin  "VARCHAR lastdate.arr"
    type     STRING
    format   %-21s
 }

DATUM  lastdate {
    num_rows     1
    num_columns  55
    variable     lastdate, 0 , 28
    leader       "Date of last modification: "
    pickable     NO
 }

VARIABLE  planrlast {
    foundin  "VARCHAR planrlst.arr"
    type     STRING
    format   %-30s
 }

DATUM  planrlast {
    num_rows     1
    num_columns 55
    variable     planrlast, 0 ,19
    leader       "Last modified by: "
    pickable     NO
 }

WINDOW  probstat {
    num_rows     5
    num_columns  78
    datum        startdate,  0, 2
    datum        lastdate,   2, 2
    datum        planrlast,  4, 2
 }

TEXTLINE  chgasstxt  {"Work on a different assessment"}

BUTTON  chgass  {                  ~1-16-88
    num_rows     1                 ~1-16-88
    num_columns  70                ~1-16-88
    textline     chgasstxt,  0, 2    ~1-16-88
    helpfile     help/probdef.hlp    ~1-16-88
 }


WINDOW  chgassess  {          ~change to different assessment
```

```
    num_rows    1
    num_columns 78
    button      chgass, 0, 2, "CALL pschgass"
 }

-----------------------------------------------------------------------------
~        Declarations for ENVIRONMENTAL ASSESSMENT DEFINITION SCREEN
-----------------------------------------------------------------------------

TEXTLINE  modmoatxt  {"Work with MOA information (number or type of aircraft, missions, etc.)"}

BUTTON  modmoa {
    num_rows   1
    num_columns    76
    textline   modmoatxt, 0 ,2
    helpfile   help/moawork.hlp
 }

TEXTLINE  modmtrtxt  {"Work with MTR information (number or type of aircraft, missions, etc.)"}

BUTTON  modmtr {
    num_rows    1
    num_columns 76
    textline    modmtrtxt, 0 ,2
    helpfile    help/mtrwork.hlp
 }

TEXTLINE  modmaptxt  {"Work with map information (designate land uses, update maps)"}

BUTTON  modmap  {
    num_rows    1
    num_columns 76
    textline    modmaptxt, 0 ,2
    helpfile    help/mapwork.hlp
 }

TEXTLINE  selectxt  {"Actions you can now take to add information to this assessment:"}

WINDOW  selaction  {
    num_rows    7
    num_columns 77
    textline    selectxt, 0, 1
    button      modmap,   2, 1, "CALL dummy2"
    button      modmtr,   4, 1, "CALL pmtrent"
    button      modmoa,   6, 1, "CALL dummy"
 }

SCREEN  probdefscreen {
    title   "ENVIRONMENTAL ASSESSMENT DEFINITION"
    window  assnamcom,  3,1
    window  selaction,  7, 2
    window  majoraction1, 18 , 1
    border  YES
 }

-----------------------------------------------------------------------------
~        Declaration for entering coordinates window
-----------------------------------------------------------------------------

TEXTLINE  entupplf
  {"Enter upper-left corner coordinates of area of current interest"}

TEXTLINE  entlowrt
  {"Enter lower right corner coordinates of area of current interest"}

DATUM   shwlat2 {
    num_rows   1
    num_columns    25
    variable   shwlat, 0 ,11
    leader     "Latitude: "
    helpfile   help/combmap.hlp
  }

DATUM   shwlong2 {
    num_rows   1
    num_columns    25
    variable   shwlong, 0 ,12
    leader     "Longitude: "
    helpfile   help/combmap.hlp
  }

WINDOW  entcoor  {
```

```
    num_rows       10
    num_columns    78
    textline       emtupplf,  2, 2
    datum          emtlat,    3, 1,   "CALL lat2dec &emt","NEWVALS"
    datum          emtlong,   3, 38,  "CALL lon2dec &emt","NEWVALS"
    textline       emtlowrt,  5, 2
    datum          shwlat2,   6, 1,   "CALL lat2dec &show","NEWVALS"
    datum          shwlong2,  6, 38,  "CALL lon2dec &show","NEWVALS"
 }

-----------------------------------------------------------------------------
-        Declarations for DATA ANALYSIS SCREEN
-----------------------------------------------------------------------------

TEXTLINE  geodatinqtxt {"Make geodata inquiries on map screen"}

BUTTON  geodatinq {
    num_rows     1
    num_columns 76
    textline     geodatinqtxt, 0 ,2
    helpfile     help/geoinq.hlp
 }

TEXTLINE  compnoiseftxt  {"Compare noise effects"}  -1-17-88

BUTTON  compnoisef  {                          -1-17-88
    num_rows     1                             -1-17-88
    num_columns 76                             -1-17-88
    textline     compnoiseftxt, 0 ,2                -1-17-88
    helpfile     help/compnois.hlp                  -2-06-88
 }                                  -1-17-88

TEXTLINE  calcnoiseftxt  {"Calculate noise effects in specified area"}

BUTTON  calcnoisef  {
    num_rows     1
    num_columns 76
    textline     calcnoiseftxt, 0 ,2
    helpfile     help/effcalc.hlp
 }

TEXTLINE  calcnoiseaxptxt  {"Calculate noise exposure in specified area"}

BUTTON  calcnoiseaxp  {
    num_rows     1
    num_columns 76
    textline     calcnoiseaxptxt, 0 ,2
    helpfile     help/expcalc.hlp
 }

TEXTLINE  calcqklooktxt  {"Calculate quicklook (point) exposure estimate"}

BUTTON  calcqklook {
    num_rows     1
    num_columns 50
    textline     calcqklooktxt, 0 ,2
    helpfile     help/qklook.hlp
 }

TEXTLINE  emtcoortxt  {"Enter coordinates from keyboard"}

BUTTON  emtcoor {
    num_rows     1
    num_columns 40
    textline     emtcoortxt, 0 ,2
    helpfile     help/kbdntry.hlp
 }

TEXTLINE  usemaptxt  {"Use map screen"}

BUTTON  usemap {
    num_rows     1
    num_columns 30 textline    usemaptxt, 0 ,2
    helpfile     help/usemap.hlp
 }

TEXTLINE  selectanstxt  {"Actions you can now take to analyze environmental assessment data:"}

TEXTLINE  defgeoareatxt  {"Specify a geographic area of interest:"}

WINDOW  datanactica  {
    num_rows       11
```

```
    num_columns   78
    textline      selectanatxt,  0, 2
    button        calcqklook,    2, 2, "CALL dummy"
    button        calcnoisexp,   3, 2, "CALL dummy"
    button        calcnoisef,    4, 2, "CALL dummy"
    button        compnoisef,    5, 2, "CALL dummy"
    button        geodatinq,     6, 2, "CALL dummy"
    textline      defgeoareatxt, 8, 2
    button        usemap,        9, 2, "CALL dummy2"
    button        entcoor,       10,2, "ADD_WINDOW entcoor  5 1"
 }

SCREEN  analysiscreen {
    title   "DATA ANALYSIS"
    window  assnamcom,  2,1
    window  datanaction,  6,1
    window  majoraction2, 18 , 1
    border  YES
 }

TEXTLINE   showmoretxt {"Show more assessments (if any)"}

BUTTON    showmore {
    num_rows   1
    num_columns    37
    textline   showmoretxt, 0 ,2
    helpfile   help/getassmnt.hlp
 }

TEXTLINE  recallasstxt   {"Recall one of the following assessments:"}

TEXTLINE  stnewasstxt   {"Start a new assessment"}

BUTTON    stnewass {
    num_rows       1
    num_columns    28
    textline       stnewasstxt, 0 ,2
    helpfile       help/newassmnt.hlp
    }

VARIABLE newasnam  {
    foundin    "VARCHAR n2bv.arr"
    type       STRING
    format     %-30s
 }

DATUM   newasnam {
    num_rows       1
    num_columns    40
    variable       newasnam,   0 , 7
    leader         "Name:"
    helpfile       help/stnewass.hlp
 }

WINDOW  newasnam  {
    num_rows       1
    num_columns    40
    datum          newasnam,  0,1
    }

WINDOW  stnewass  {
    num_rows       1
    num_columns    78
    button         stnewass,  0, 2, "ADD_WINDOW newasnam, 6  3",
                              "UPDATE_DATUM newasnam",
                              "CALL psnwasem n2bv.arr"
    button         showmore,  0, 41,    "CALL ubunch"
    }

WINDOW  recalass  {
    num_rows       1
    num_columns    78
    textline       recallasstxt,  0, 2
    }

WINDOW    curassbut  {
    num_rows       10
    num_columns    2
    button   mulbut00,  0, 0,    "CALL ASANcoan dsplmult[0].arr"
    button   mulbut01,  1, 0,    "CALL ASANcoan dsplmult[1].arr"
    button   mulbut02,  2, 0,    "CALL ASANcoan dsplmult[2].arr"
    button   mulbut03,  3, 0,    "CALL ASANcoan dsplmult[3].arr"
```

```
        button   mulbut04,  4, 0,    "CALL ASANocean displmult[4].arr"
        button   mulbut05,  5, 0,    "CALL ASANocean displmult[5].arr"
        button   mulbut06,  6, 0,    "CALL ASANocean displmult[6].arr"
        button   mulbut07,  7, 0,    "CALL ASANocean displmult[7].arr"
        button   mulbut08,  8, 0,    "CALL ASANocean displmult[8].arr"
        button   mulbut09,  9, 0,    "CALL ASANocean displmult[9].arr"
      }

 WINDOW    curassbut1  {
        num_rows       10
        num_columns    2
        button   mulbut10,  0, 0,    "CALL ASANocean displmult[10].arr"
        button   mulbut11,  1, 0,    "CALL ASANocean displmult[11].arr"
        button   mulbut12,  2, 0,    "CALL ASANocean displmult[12].arr"
        button   mulbut13,  3, 0,    "CALL ASANocean displmult[13].arr"
        button   mulbut14,  4, 0,    "CALL ASANocean displmult[14].arr"
        button   mulbut15,  5, 0,    "CALL ASANocean displmult[15].arr"
        button   mulbut16,  6, 0,    "CALL ASANocean displmult[16].arr"
        button   mulbut17,  7, 0,    "CALL ASANocean displmult[17].arr"
        button   mulbut18,  8, 0,    "CALL ASANocean displmult[18].arr"
        button   mulbut19,  9, 0,    "CALL ASANocean displmult[19].arr"
      }

 TEXTLINE  retnoass  {"Continue without selecting assessment"}

 BUTTON retassent {
    num_rows      1
    num_columns   40
    textline      retnoass, 0 ,2
    helpfile      help/noassess.hlp
    }

 WINDOW  retwoass {
    num_rows      1
    num_columns   76
    button        retassent, 0,1, "CALL wlistc", "CALL psprobst"
    }

 SCREEN  chgcurasscreen {
      title       "SELECT ANOTHER ASSESSMENT"
      window      assasmocm,  2,1
      window      stnewass,   6,1
      window      recalass,   8,1
      window      curassbut,  9,3
      window      curassbut1, 9,42
      window      curdat,     9,5
      window      curdat1,    9,44
      window      retwoass,   20,2
      border      YES
      }

-----------------------------------------------------------------------
-        Declarations for NEW ASSESSMENT DEFINITION
-----------------------------------------------------------------------

VARIABLE emtdesc1  {
      foundin     "VARCHAR60 emtdesc[0].arr"
      type    STRING
      format      %-60s
  }

DATUM   emtdesc1 {
      num_rows    1
      num_columns     75
      variable    emtdesc1, 0 ,0
      helpfile    help/emtdesc.hlp
  }

VARIABLE emtdesc2  {
      foundin     "VARCHAR60 emtdesc[1].arr"
      type        STRING
      format      %-60s
  }

DATUM   emtdesc2 {
      num_rows    1
      num_columns     75
      variable    emtdesc2, 0 ,0
      helpfile    help/emtdesc.hlp
  }

VARIABLE emtdesc3  {
```

```
        foundin     "VARCHAR60 amtdesc[2].arr"
        type        STRING
        format      %-60s
    }

  DATUM   amtdesc3 {
        num_rows    1
        num_columns     75
        variable    amtdesc3, 0 ,0
        helpfile    help/amtdesc.hlp
    }

  VARIABLE amtdesc4  {
        foundin     "VARCHAR60 amtdesc[3].arr"
        type        STRING
        format      %-60s
    }

  DATUM   amtdesc4 {
        num_rows    1
        num_columns     75
        variable    amtdesc4, 0 ,0
        helpfile    help/amtdesc.hlp
    }

  TEXTLINE amtdesctxt  {"Please enter a brief description for this assessment"}

  DATUM   newassnam {
        num_rows    1
        num_columns     77
        variable    newasnam, 0 ,35
        leader      "Name of new assessment definition:"
        pickable    NO
    }

  WINDOW  newasdesc  {
        num_rows     9
        num_columns 70
        datum       newassnam,  1,3
        textline    amtdesctxt, 3,4
        datum       amtdesc1,   4,4
        datum       amtdesc2,   5,4
        datum       amtdesc3,   6,4
        datum       amtdesc4,   7,4
        border      YES
        }

-----------------------------------------------------------------------
-           Declarations for database housekeeping screen
-----------------------------------------------------------------------

  TEXTLINE  updateinfotxt  {"Update information"}

  BUTTON    updateinfo {
        num_rows    1
        num_columns     50
        textline    updateinfotxt, 0 ,2
        helpfile    help/updatein.hlp
        }

  TEXTLINE  asstabletxt2  {"Print list of all columns in an assessment's tables"}

  BUTTON    asstable2 {
        num_rows    1
        num_columns     55
        textline    asstabletxt2, 0 ,2
        helpfile    help/asstable.hlp
        }

  TEXTLINE  asstabletxt1  {"Print list of an assessment's tables"}

  BUTTON    asstable1 {
        num_rows    1
        num_columns     50
        textline    asstabletxt1, 0 ,2
        helpfile    help/asstable.hlp
        }

  TEXTLINE  assesstxt  {"Print list of all assessments"}

  BUTTON    assess {
        num_rows    1
```

```
     num_columns    50
     textline   assesstxt, 0 ,2
     helpfile   help/nohelp.hlp
     }

TEXTLINE  dbdatetxt  {"Print all database dates"}

BUTTON    dbdate {
     num_rows   1
     num_columns    50
     textline   dbdatetxt, 0 ,2
     helpfile   help/dbdate.hlp
     }

TEXTLINE  dbhsekpgtxt  {"WARNING: Actions you take on this screen affect
                         ASAN'S permanent databases!"}

WINDOW    dbhsekpgaction   {
          num_rows      13
          num_columns   78
          line          0,0,  0,77
          textline      dbhsekpgtxt,   2,2
          button        dbdate,       4,2,  "CALL dummy"
          button        assess,       6,2,  "CALL prntabc"
          button        asstable1,    8,2,  "CALL SUpropt 1"
          button        asstable2,   10,2,  "CALL SUpropt 2"
          button        updateinfo,  12,2,  "CALL dummy"
     }

SCREEN  dbhsekpgscreen {
     title      "DATABASE HOUSEKEEPING"
     window     dbhsekpgaction, 2,1
     window     majoractions,   18,1
     border     YES
     }


WINDOW  quickassbut  {
       num_rows      10
       num_columns   2
       button   mulbut00,  0, 0,    "CALL SUprint displmult[0].arr"
       button   mulbut01,  1, 0,    "CALL SUprint displmult[1].arr"
       button   mulbut02,  2, 0,    "CALL SUprint displmult[2].arr"
       button   mulbut03,  3, 0,    "CALL SUprint displmult[3].arr"
       button   mulbut04,  4, 0,    "CALL SUprint displmult[4].arr"
       button   mulbut05,  5, 0,    "CALL SUprint displmult[5].arr"
       button   mulbut06,  6, 0,    "CALL SUprint displmult[6].arr"
       button   mulbut07,  7, 0,    "CALL SUprint displmult[7].arr"
       button   mulbut08,  8, 0,    "CALL SUprint displmult[8].arr"
       button   mulbut09,  9, 0,    "CALL SUprint displmult[9].arr"
     }

WINDOW    quickassbut1  {
       num_rows      10
       num_columns   2
       button   mulbut10,  0, 0,    "CALL SUprint displmult[10].arr"
       button   mulbut11,  1, 0,    "CALL SUprint displmult[11].arr"
       button   mulbut12,  2, 0,    "CALL SUprint displmult[12].arr"
       button   mulbut13,  3, 0,    "CALL SUprint displmult[13].arr"
       button   mulbut14,  4, 0,    "CALL SUprint displmult[14].arr"
       button   mulbut15,  5, 0,    "CALL SUprint displmult[15].arr"
       button   mulbut16,  6, 0,    "CALL SUprint displmult[16].arr"
       button   mulbut17,  7, 0,    "CALL SUprint displmult[17].arr"
       button   mulbut18,  8, 0,    "CALL SUprint displmult[18].arr"
       button   mulbut19,  9, 0,    "CALL SUprint displmult[19].arr"
     }

TEXTLINE  susselect  {"Print list of SUPERUSER's tables"}
TEXTLINE  hqselect  {"Print list of HEADQUARTERS' tables"}

BUTTON    susselect {
     num_rows   1
     num_columns    50
     textline   susselect, 0 ,2
     helpfile   help/nohelp.hlp
     }
BUTTON    hqselect {
     num_rows   1
     num_columns    50
     textline   hqselect, 0 ,2
     helpfile   help/nohelp.hlp
     }
```

```
WINDOW  retwoprn {
   num_rows      1
   num_columns   76
   button        retassesmt, 0,1, "CALL ulistc", "NEW_SCREEN dbhsakpgscreen"
   }


WINDOW  hqruselect {
   num_rows      3
   num_columns   76
   button        hqselect, 0,1, "CALL SUpropt 3"
   button        suselect, 2,1, "CALL SUpropt 4"
   }

SCREEN  sloasscreen {
     title       "SELECT ASSESSMENT FOR DATABASE PRINTOUT"
     window      hqruselect,    4,1
     window      recslass,      8,1
     window      quickassbut,   9,3
     window      quickassbut1,  9,42
     window      curdat,        9,5
     window      curdat1,       9,44
     window      retwoprn,     20,2
     border      YES
     }


---------------------------------------------------------------------------
-         Declarations for view CHECK-LIST SCREEN
---------------------------------------------------------------------------

TEXTLINE  otherchklst  {"View another checklist"}

BUTTON    otherchklst {
     num_rows        1
     num_columns     25
     textline        otherchklst,  0, 2
     helpfile        help/othrchk.hlp
     }

TEXTBLOCK  fcasi2  {
     filename        txtblk/fcasi2.txt
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW    fcasi2   {
     num_rows        13
     num_columns     78
     textblock        fcasi2,   0,1
     button          otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  fcasi1  {
     filename        txtblk/fcasi1.txt
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW    fcasi1   {
     num_rows        13
     num_columns     78
     textblock        fcasi1,   0,1
     button          otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  cstax3  {
     filename        txtblk/cstax3.txt
     num_rows        10
     num_columns     76
     border          YES
     }
```

```
WINDOW    catex3   {
     num_rows       13
     num_columns    78
     textblock       catex3,   0,1
     button         otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  catex2  {
     filename       txtblk/catex2.txt
     num_rows       10
     num_columns    76
     border         YES
     }

WINDOW    catex2   {
     num_rows       13
     num_columns    78
     textblock       catex2,   0,1
     button         otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTBLOCK  catex1  {
     filename       txtblk/catex1.txt
     num_rows       10
     num_columns    76
     border         YES
     }

WINDOW    catex1   {
     num_rows       13
     num_columns    78
     textblock       catex1,   0,1
     button         otherchklst, 12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  fonsi2txt  {
"Show documentation necessary for finding of no significant impact"}

BUTTON    fonsi2 {
     num_rows   1
     num_columns    72
     textline   fonsi2txt,  0, 2
     helpfile   help/fonsi2.hlp
     }

TEXTLINE  fonsi1txt  {
"Show NEPA bases for finding of no significant impact (FONSI)"}

BUTTON    fonsi1 {
     num_rows   1
     num_columns    65
     textline   fonsi1txt,  0, 2
     helpfile   help/fonsi1.hlp
     }

TEXTLINE  catex3txt  {"Show documentation needed for categorical exclusions"}

BUTTON    catex3 {
     num_rows   1
     num_columns    55
     textline   catex3txt,  0, 2
     helpfile   help/catex3.hlp
     }

TEXTLINE  catex2txt  {
"Show examples of proposed actions qualifying for categorical exclusions"}

BUTTON    catex2 {
     num_rows   1
     num_columns    74
     textline   catex2txt,  0, 2
     helpfile   help/catex2.hlp
     }

TEXTLINE  catex1txt  {"Show NEPA bases for categorical exclusions (CATEX)"}

BUTTON    catex1 {
     num_rows   1
     num_columns    55
     textline   catex1txt,  0, 2
     helpfile   help/catex1.hlp
     }
```

```
WINDOW    vwchklist   {
       num_rows       11
       num_columns    78
       line      0,0 ,  0,77
       button     catax1,         2,2, "ADD_WINDOW catax1  3  1"
       button     catax2,         4,2, "ADD_WINDOW catax2  3  1"
       button     catax3,         6,2, "ADD_WINDOW catax3  3  1"
       button     fonsi1,         8,2, "ADD_WINDOW fonsi1  3  1"
       button     fonsi2,         10,2,     "ADD_WINDOW fonsi2  3  1"
      }

SCREEN  viewchecklistscreen {
     title       "VIEW CHECKLIST"
     window      vwchklist,  2,1
     window      majoractions4,   18,1
     border      YES
     }

-----------------------------------------------------------------------

SCREEN  probstatscreen {
   title       "ENVIRONMENTAL ASSESSMENT STATUS"
   mainscreen YES
   border      YES
   window      assessmoom,      3,1
   window      probstat,        7,1
   window      chgassess,      16,1
   window      majoractions,   18,1
   }
```

A.3 Screen Description File for Report-Related Portions of ASAN

```
-----------------------------------------------------------------------------
-         Declarations for REPORT GENERATION SCREEN
-----------------------------------------------------------------------------

TEXTLINE  viewnoisetxt  ("View text on calculated noise effects for current assessment")

BUTTON   viewnoisetx {
    num_rows     1
    num_columns 72
    textline     viewnoisetxt, 0 ,2
    helpfile     help/viewtext.hlp
 }

TEXTLINE  viewbplatetxt  ("View boilerplate")

BUTTON   viewbplate {
    num_rows     1
    num_columns 72
    textline     viewbplatetxt, 0 ,2
    helpfile     help/rolodex.hlp
 }

TEXTLINE  prnoisewbptxt  ("Print above text with associated boilerplate")

BUTTON  prnoisewbp {
    num_rows     1
    num_columns 72
    textline     prnoisewbptxt, 0 ,2
    helpfile     help/rolodex.hlp
 }

TEXTLINE  prnoisetxt  ("Print above text")

BUTTON   prnoisetx {
    num_rows     1
    num_columns 72
    textline     prnoisetxt, 0 ,2
    helpfile     help/crestxt.hlp
 }
```

```
TEXTLINE  stdreprtxt  ("Print standard report")

BUTTON   stdreprt {
    num_rows    1
    num_columns 72
    textline    stdreprtxt, 0 ,2
    helpfile    help/creatxt.hlp
  }

TEXTLINE selectreptxt ("Actions you can now take to produce text and graphics:")

WINDOW  reportaction  {
        num_rows     15
        num_columns  77
        line         0, 1, 0, 76
        textline     selectreptxt, 4, 1
        button       stdreprt,     6, 1, "CALL callrpt"
        button       viewnoisetx,  8, 1, "CALL dummy"
        button       prnoisetx,    10, 1, "CALL callrpt3"
        button       prnoisewbp,   12, 1, "CALL callrpt4"
        button       viewbplate,   14,1, "NEW_SCREEN bplatarevscreen"
 }

SCREEN  reportgenscreen {
    title        "MAKE A REPORT"
    mainscreen   YES
    window       reportaction,  2, 1
    border       YES
 }

-----------------------------------------------------------------------------
-        Declarations for boilerplate review screen
-----------------------------------------------------------------------------

TEXTLINE  otherbplatetxt  ("View other boilerplate text")

BUTTON    otherbplate {
     num_rows       1
     num_columns    30
     textlinb       otherbplatetxt,  0, 2
     helpfile       help/blrplate.hlp
     }

TEXTBLOCK  heardam  {
     filename       blrplt/hriagdmg.bpl
     num_rows       10
     num_columns    76
     border         YES
     }

WINDOW  heardam   {
     num_rows       13
     num_columns    78
     textblock      heardam,    0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  heardamtxt  ("Hearing damage risk")

BUTTON    heardam {
     num_rows       1
     num_columns    35
     textline       heardamtxt,  0, 2
     helpfile       help/blrplate.hlp
     }

TEXTBLOCK  structdam  {
     filename       blrplt/strctdmg.bpl
     num_rows       10
     num_columns    76
     border         YES
     }

WINDOW  structdam   {
     num_rows       13
     num_columns    78
     textblock      structdam,  0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  structdamtxt  ("Structural damage")
```

```
BUTTON    structdam {
     num_rows        1
     num_columns     35
     textline        structdamtxt,  0, 2
     helpfile        help/blrplate.hlp
     }

TEXTBLOCK  actintfer  {
     filename        blrplt/elpintf.bpl
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW  actintfer   {
     num_rows       13
     num_columns    78
     textblock      actintfer,  0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  actintfertxt {"Activity interference"}

BUTTON    actintfer {
     num_rows        1
     num_columns     35
     textline        actintfertxt,  0, 2
     helpfile        help/blrplate.hlp
     }

TEXTBLOCK  wildlife  {
     filename        blrplt/wildlife.bpl
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW  wildlife   {
     num_rows       13
     num_columns    78
     textblock      wildlife,   0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  wildlifetxt  {"Wildlife"}

BUTTON    wildlife {
     num_rows        1
     num_columns     35
     textline        wildlifetxt,  0, 2
     helpfile        help/blrplate.hlp
     }

TEXTBLOCK  comintfer  {
     filename        blrplt/spchintf.bpl
     num_rows        10
     num_columns     76
     border          YES
     }

WINDOW  comintfer   {
     num_rows       13
     num_columns    78
     textblock      comintfer,  0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  comintfertxt  {"Communication interference"}

BUTTON    comintfer {
     num_rows        1
     num_columns     35
     textline        comintfertxt,  0, 2
     helpfile        help/blrplate.hlp
     }

TEXTBLOCK  livestock  {
     filename        blrplt/livestck.bpl
     num_rows        10
     num_columns     76
     border          YES
     }
```

```
WINDOW livestock  {
     num_rows       13
     num_columns    78
     textblock      livestock,  0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  livestocktxt  {"Economic damage to livestock"}

BUTTON    livestock {
     num_rows       1
     num_columns    35
     textline       livestocktxt,  0, 2
     helpfile       help/blrplate.hlp
     }

TEXTBLOCK  annyance  {
     filename       blrplt/annyance.bpl
     num_rows       10
     num_columns    76
     border         YES
     }

WINDOW  annoy  {
     num_rows       13
     num_columns    78
     textblock      annyance,    0,1
     button         otherbplate,  12,2,  "REMOVE_WINDOW"
     }

TEXTLINE  annoytxt  {"Human annoyance"}

BUTTON    annoy {
     num_rows       1
     num_columns    35
     textline       annoytxt,  0, 2
     helpfile       help/blrplate.hlp
     }

TEXTLINE   bplatetxt  {"Select one of the following to view available
                        noise effects boilerplate:"}


TEXTLINE returnrpt {"Return to report selection screen"}

BUTTON returnrpt {
   num_rows  1
   num_columns  42
   textline  returnrpt, 0, 2
   helpfile help/nohelp.txt
   }

WINDOW nulltext {
   num_rows  3
   num_columns 77
   button    returnrpt, 1, 1, "NEW_SCREEN  reportgensscreen"
   }


WINDOW  bplate  {
     num_rows       11
     num_columns    78
     line           0,0,  0,77
     textline       bplatetxt,  2,2
     button         annoy,       4,2,   "ADD_WINDOW annoy  3 1"
     button         ocmintfer,   6,2,   "ADD_WINDOW ocmintfer 3 1"
     button         actintfer,   8,2,   "ADD_WINDOW actintfer 3 1"
     button         heardam,     10,2,  "ADD_WINDOW heardam 3 1"
     button         livestock,   4,39,  "ADD_WINDOW livestock 3 1"
     button         wildlife,    6,39,  "ADD_WINDOW wildlife 3 1"
     button         structdam,   8,39,  "ADD_WINDOW structdam 3 1"
     }

SCREEN  bplatereviewscreen {
     title      "VIEW BOILERPLATE TEXT"
     window     bplate, 2,1
     window     nulltext, 18, 1
     border     YES
     }
```

A.4 Screen Description File for Graphic Portion of ASAN

```
  ---------------------------------------------------------------------------
-INCLUDE STATEMENT FOR THE ASAN & GRAPHICS typedef DEFINITIONS NEEDED TO
-        COMPILE USUBS.C WITHOUT INCURRING THE WRATH OF THE COMPILER
---------------------------------------------------------------------------

INCLUDE ASANTYPE.H
INCLUDE ASAN.H

---------------------------------------------------------------------------
-  DECLARATIONS IN NON-LEXICAL OBJECT SEQUENCE ................
---------------------------------------------------------------------------

---------------------------------------------------------------------------
-       Data structures for holding list of map names                     -
---------------------------------------------------------------------------

VARIABLE dlnam[0] {
    type        STRING
    format      %-8s
  }
  DATUM dlnam0 {
    num_rows    1
    num_columns 8
    variable    dlnam[0] , 0 , 0
    pickable    NO
  }

VARIABLE dlnam[1] {
    type        STRING
    format      %-8s
  }
  DATUM dlnam1 {
    num_rows    1
    num_columns 8
    variable    dlnam[1] , 0 , 0
    pickable    NO
  }

VARIABLE dlnam[2] {
    type        STRING
    format      %-8s
  }
  DATUM dlnam2 {
    num_rows    1
    num_columns 8
    variable    dlnam[2] , 0 , 0
    pickable    NO
  }

VARIABLE dlnam[3] {
    type        STRING
    format      %-8s
  }
  DATUM dlnam3 {
    num_rows    1
    num_columns 8
    variable    dlnam[3] , 0 , 0
    pickable    NO
  }

VARIABLE dlnam[4] {
    type        STRING
    format      %-8s
  }
  DATUM dlnam4 {
    num_rows    1
    num_columns 8
    variable    dlnam[4] , 0 , 0
    pickable    NO
  }

VARIABLE dlnam[5] {
    type        STRING
    format      %-8s
  }
  DATUM dlnam5 {
```

```
    num_rows     1
    num_columns  8
    variable     dlnam[5] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[6] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam6 {
    num_rows     1
    num_columns  8
    variable     dlnam[6] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[7] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam7 {
    num_rows     1
    num_columns  8
    variable     dlnam[7] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[8] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam8 {
    num_rows     1
    num_columns  8
    variable     dlnam[8] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[9] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam9 {
    num_rows     1
    num_columns  8
    variable     dlnam[9] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[10] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam10 {
    num_rows     1
    num_columns  8
    variable     dlnam[10] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[11] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam11 {
    num_rows     1
    num_columns  8
    variable     dlnam[11] , 0 , 0
    pickable     NO
  }

VARIABLE dlnam[12] {
    type         STRING
    format       %-8s
  }
  DATUM dlnam12 {
    num_rows     1
    num_columns  8
    variable     dlnam[12] , 0 , 0
    pickable     NO
  }
```

```
VARIABLE dlsam[13] {
    type        STRING
    format      %-8s
  }
  DATUM dlsam13 {
    num_rows    1
    num_columns 8
    variable    dlsam[13] , 0 , 0
    pickable    NO
  }

VARIABLE dlsam[14] {
    type        STRING
    format      %-8s
  }
  DATUM dlsam14 {
    num_rows    1
    num_columns 8
    variable    dlsam[14] , 0 , 0
    pickable    NO
  }

-----------------------------------------------------------------------
--          Declaration for entering coordinates
-----------------------------------------------------------------------

VARIABLE entlat  {                          - CURRENT or ENTER Set
     foundin    "COORDINATE ent.lat"
     type       STRING
     format     %-13s
}

DATUM   entlat {
     num_rows   1
     num_columns    25
     variable   entlat, 0 ,11
     leader     "Latitude: "
     helpfile   help/combmap.hlp
  }

VARIABLE entlong  {
     foundin    "COORDINATE ent.lon"
     type       STRING
     format     %-13s
}

DATUM   entlong {
     num_rows   1
     num_columns    25
     variable   entlong, 0 ,11
     leader     "Longitude:"
     helpfile   help/combmap.hlp
  }

-----------------------------------------------------------------------
--          Declaration for show coordinates
-----------------------------------------------------------------------

VARIABLE shwlat  {
     foundin    "COORDINATE show.lat"
     type       STRING
     format     %-13s
}

DATUM   shwlat {
     num_rows   1
     num_columns    25
     variable   shwlat, 0 ,11
     leader     "Latitude: "
     pickable   NO
     helpfile   help/combmap.hlp
  }

VARIABLE shwlong  {
     foundin    "COORDINATE show.lon"
     type       STRING
     format     %-13s
}

DATUM   shwlong {
     num_rows   1
     num_columns    25
```

```
     variable   shwlong, 0 ,11
     leader     "Longitude:"
     pickable   NO
     helpfile   help/combmap.hlp
  }

VARIABLE shwdist  {
     type       DOUBLE
     uplimit    10000.
     lowlimit   0.
     format     %10.21f
}

DATUM   shwdist {
     num_rows   1
     num_columns    13
     variable   shwdist, 0 , 0
     pickable   NO
     trailer    "km"
  }


-----------------------------------------------------------------------
- DECLARATIONS FOR SCREEN HEADER
-----------------------------------------------------------------------

VARIABLE assename {
    foundin  "ASANHEADR ASSESSMENT.name"
    type     STRING
    format   %-30s
 }

DATUM  assename1 {
    num_rows     1
    num_columns  78
    variable     assename, 0 , 30
    leader       "Name of current assessment: "
    pickable     NO
    helpfile     help/nohelp.hlp
 }

VARIABLE comment {
    foundin  "ASANHEADR ASSESSMENT.desc"
    type     STRING
    format   %-66s
 }

DATUM  comment {
    num_rows     1
    num_columns  77
    variable     comment, 0 , 10
    pickable     NO
 }

WINDOW  lin  {
    num_rows     1
    num_columns  78
    line         0, 0, 0,77
 }

WINDOW  assnamcom  {
    num_rows      3
    num_columns   78
    datum         assename1,  0,0
    datum         comment,  1, 1
    line          2, 0, 2,77
 }

-----------------------------------------------------------------------
-    Declaration for buttons used in majoraction footer
-----------------------------------------------------------------------

TEXTLINE probsta { "REVIEW CURRENT ASSESSMENT STATUS" }

BUTTON  probstat  {
    num_rows     1
    num_columns  35
    textline     probsta,  0, 2
    helpfile     help/nohelp.hlp
 }

TEXTLINE probsta1 { "WORK ON ANOTHER ASSESSMENT" }
```

```
BUTTON  probstat1  {
    num_rows      1
    num_columns   28
    textline      probsta1,  0, 2
    helpfile      help/nohelp.hlp
 }

TEXTLINE probsta2 { "REVIEW ASSESSMENT STATUS" }

BUTTON  probstat2  {
    num_rows      1
    num_columns   26
    textline      probsta2,  0, 2
    helpfile      help/nohelp.hlp
 }

TEXTLINE probdef {"ADD INFORMATION TO CURRENT ASSESSMENT" }

BUTTON  probdef  {
    num_rows      1
    num_columns   39
    textline      probdef,  0, 2
    helpfile      help/probdef.hlp
 }

TEXTLINE probdef1 {"ADD TO ASSESSMENT DEFINITION" }

BUTTON  probdef1  {
    num_rows      1
    num_columns   31
    textline      probdef1,  0, 2
    helpfile      help/probdef.hlp
 }

TEXTLINE analysis { "ANALYZE DATA" }

BUTTON  analysis  {
    num_rows      1
    num_columns   15
    textline      analysis,  0, 2
    helpfile      help/dataanal.hlp
 }

TEXTLINE reportgen { "MAKE A REPORT" }

BUTTON  reportgen  {
    num_rows      1
    num_columns   16
    textline      reportgen,  0, 2
    helpfile      help/reportgen.hlp
 }

TEXTLINE vwchklis { "VIEW CHECKLIST FOR CURRENT ASSESSMENT" }

BUTTON  vwchklist  {
    num_rows      1
    num_columns   42
    textline      vwchklis,  0, 2
    helpfile      help/viewlist.hlp
 }

TEXTLINE databaseinq { "MAKE DATABASE INQUIRIES" }

BUTTON  databaseinq  {
    num_rows      1
    num_columns   26
    textline      databaseinq,  0, 1
    helpfile      help/makinq.hlp
 }

-----------------------------------------------------------------------------
-         Declarations for majorsection footers
-----------------------------------------------------------------------------

WINDOW  majorsection  {        -header for problem status screen
    num_rows      4
    num_columns   78
    line          0, 0, 0,77
    title         "Please select one of the following actions:"
    button        probdef,      2, 0, "CALL dummy"
    button        analysis,     2,43, "CALL dummy"
    button        reportgen,    2,62, "CALL dummy"
```

```
    button        vwchklist,   3, 0, "CALL dummy"
    button        databaseinq, 3,52, "CALL dummy"
  }

WINDOW majoractions3  {        -header for report generation screen
    num_rows     4
    num_columns 78
    line         0, 0, 0,77
    title        "Alternative actions you can now take:"
    button       probstat2,   2, 0, "CALL dummy"
    button       probdef1,    2,29, "CALL dummy"
    button       vwchklist,   3, 0, "CALL dummy"
    button       databaseinq, 3,52, "CALL dummy"
  }

-----------------------------------------------------------------------------
-          Declarations for MAP CONTROL
-----------------------------------------------------------------------------

VARIABLE shwareaLDW  {
     type    INTEGER
     format      4-5d
     uplimit         124
     default         75
     lowlimit    20
  }

DATUM   shwareaLDW {
     num_rows    1
     num_columns     35
     variable    shwareaLDW, 0 ,15
     leader      "Show areas of "
     trailer         "(dB) <= LDW <="
     helpfile    help/nohelp.hlp
  }

VARIABLE shwareaLDW1  {
     type    INTEGER
     format      4-5d
     uplimit         124
     default         75
     lowlimit    20
  }

DATUM   shwareaLDW1 {
     num_rows    1
     num_columns     18
     variable    shwareaLDW1, 0 ,0
     trailer         "(dB)"
     helpfile    help/nohelp.hlp
  }

VARIABLE shwareaPSF  {
     type    INTEGER
     format      4-5d
     uplimit         124
     default         75
     lowlimit    20
  }

DATUM   shwareaPSF {
     num_rows    1
     num_columns      35
     variable    shwareaPSF, 0 ,15
     leader      "Show areas of "
     trailer         "(dB) <= PSF <="
     helpfile    help/nohelp.hlp
  }

VARIABLE shwareaPSF1  {
     type    INTEGER
     format      4-5d
     uplimit         124
     default         75
     lowlimit    20
  }

DATUM   shwareaPSF1 {
     num_rows    1
     num_columns     18
     variable    shwareaPSF1, 0 ,0
     trailer         "(dB)"
```

```
    helpfile   help/nohelp.hlp
  }

TEXTLINE  entidptofint   {"Mark the map at:"}

TEXTLINE  shwidptofint   {"Locate touched point"}

BUTTON    shwidptofint {
    num_rows   1
    num_columns    22
    textline   shwidptofint, 0 ,2
    helpfile   help/nohelp.hlp
    }

TEXTLINE  shwdist   {"Show distance between two touched points:"}

BUTTON    shwdist {
    num_rows   1
    num_columns    43
    textline   shwdist, 0 ,2
    helpfile   help/nohelp.hlp
    }

TEXTLINE  addelmap  {"Add or Remove layers being displayed"}

BUTTON   addelmap {
    num_rows   1
    num_columns    77
    textline   addelmap,  0, 2
    helpfile   help/mapname.hlp
  }

TEXTLINE  addelmapinfo  {"Add or Delete information on a map layer"}

BUTTON   addelmapinfo {
    num_rows   1
    num_columns    77
    textline   addelmapinfo,  0, 2
    helpfile   help/addelmap.hlp
  }

TEXTLINE  showlegend  { "Show the legend" }

BUTTON   showlegend {
    num_rows   1
    num_columns    17
    textline   showlegend,  0, 2
    helpfile   help/nohelp.hlp
  }

TEXTLINE  hidelegend  { "Hide the legend" }

BUTTON   hidelegend {
    num_rows   1
    num_columns    17
    textline   hidelegend,  0, 2
    helpfile   help/nohelp.hlp
  }

TEXTLINE  erasedisplay  { "Erase the map display" }

BUTTON   erasedisplay {
    num_rows   1
    num_columns    77
    textline   erasedisplay,  0, 2
    helpfile   help/nohelp.hlp
  }

TEXTLINE  editcolors  { "Edit the color assignments" }

BUTTON   editcolors {
    num_rows   1
    num_columns    77
    textline   editcolors,  0, 2
    helpfile   help/nohelp.hlp
  }

VARIABLE layersavename  {
    type   STRING
    format     48s
  }
```

```
DATUM   layersavename {
     num_rows   1
     num_columns    77
     variable   layersavename, 0 ,35
     leader     "Save the displayed map as:"
     helpfile   help/savemap.hlp
  }

VARIABLE  vpname {
     type     STRING
     format   %7s
  }

DATUM   vpname {
     num_rows           1
     num_columns        17
     leader             "View ("
     trailer            "):"
     variable           vpname, 0 , 7
     pickable           NO
  }

TEXTLINE  vw_c { "Salls.C" }

BUTTON   vw_c {
     num_rows   1
     num_columns    9
     textline   vw_c,  0, 2
  }

TEXTLINE  vw_ma { "Ajo.M" }

BUTTON   vw_ma {
     num_rows   1
     num_columns    7
     textline   vw_ma,  0, 2
  }

TEXTLINE  vw_ms { "Salls.M" }

BUTTON   vw_ms {
     num_rows   1
     num_columns    9
     textline   vw_ms,  0, 2
  }

TEXTLINE  vw_fa { "Ajo.F" }

BUTTON   vw_fa {
     num_rows   1
     num_columns    7
     textline   vw_fa,  0, 2
  }

TEXTLINE  vw_fs { "Salls.F" }

BUTTON   vw_fs {
     num_rows   1
     num_columns    9
     textline   vw_fs,  0, 2
  }

WINDOW  mapctrlactica  {
    num_rows      13
    num_columns   78
    datum     vpname,          0,  1
    button    vw_c,            0, 21, "CALL new_view 'Salls.C'"
    button    vw_ma,           0, 33, "CALL new_view 'Ajo.M'"
    button    vw_ms,           0, 43, "CALL new_view 'Salls.M'"
    button    vw_fa,           0, 55, "CALL new_view 'Ajo.F'"
    button    vw_fs,           0, 65, "CALL new_view 'Salls.F'"
    button    addalmap,        1,  1, "NEW_SCREEN mapman"
    button    addalmapinfo,    2,  1, "NEW_SCREEN addalmapinfo"
    button    showlegend,      3,  1, "CALL show_legend"
    button    hidelegend,      3, 30, "CALL hide_legend"
    button    erasedisplay,    4,  1, "CALL clear_screen"
    textline  entidptofist,    5,  3
    datum     entlat,          5, 21, "CALL lat2dec &ent" , "NEWVALS"
    datum     entlong,         5, 48, "CALL lon2dec &ent" , "NEWVALS"
    button    editcolors,      6,  1, "CALL edit_colors"
    button    shwidptofist,    7,  1, "CALL show_coords"
    datum     shwlat,          7, 25
```

```
    datum    shwlong,        7, 52
    button   shwdist,        8,  1, "CALL show_dist"
    datum    shwdist,        8, 46
    datum    shwareaLDN,     9,  3, "CALL dummy"
    datum    shwareaLDN1,    9, 40, "CALL dummy"
    datum    shwareaPSF,    10,  3, "CALL dummy"
    datum    shwareaPSF1,   10, 40, "CALL dummy"
    datum    layersavename,12,  1, "CALL store_screen"
 }

SCREEN  mapcontrol {
     title      "MAP DISPLAY CONTROL"
  mainscreen    YES                                  ----- FAKE FAKE FAKE FAKE
     window     assnamcom, 2,1
     window     mapctrlaction,  5,1
     window     majoraction3,   18,1
     border     YES
     }

-----------------------------------------------------------------------
-         Declaration for MAP MANAGER
-----------------------------------------------------------------------

DATUM   curview {
     num_rows           1
     num_columns        22
     leader             "CURRENT VIEW:"
     variable           vpname, 0 , 14
     pickable           NO
  }

TEXTLINE  layertxt_tit { "AVAILABLE MAP LAYERS:" }

TEXTBLOCK  layertxt  {
    num_rows    12
    num_columns 51
    border      YES
    filename    layers.txt
 }

TEXTLINE  layernow_tit { "NOW DISPLAYED:" }

WINDOW  layertxt  {
    num_rows    14
    num_columns 51
    datum       curview , 0 , 0
    textline    layertxt_tit ,  1 , 0
    textline    layernow_tit ,  0 , 37
    textblock   layertxt , 2 , 0
 }

WINDOW  layernow  {
    num_rows    17
    num_columns 10
    border      YES
    datum       dlnam0  ,  1 , 1
    datum       dlnam1  ,  2 , 1
    datum       dlnam2  ,  3 , 1
    datum       dlnam3  ,  4 , 1
    datum       dlnam4  ,  5 , 1
    datum       dlnam5  ,  6 , 1
    datum       dlnam6  ,  7 , 1
    datum       dlnam7  ,  8 , 1
    datum       dlnam8  ,  9 , 1
    datum       dlnam9  , 10 , 1
    datum       dlnam10 , 11 , 1
    datum       dlnam11 , 12 , 1
    datum       dlnam12 , 13 , 1
    datum       dlnam13 , 14 , 1
    datum       dlnam14 , 15 , 1
 }

VARIABLE  layer2add {
    type        STRING
    format      %8s
 }

DATUM  layer2add {
    num_rows    1
    num_columns 34
    leader      "Add map layer"
    trailer     "to display"
```

```
    variable    layer2add , 0 , 15
 }

VARIABLE  layer2del {
    type        STRING
    format      48s
 }

DATUM  layer2del {
    num_rows    1
    num_columns 39
    leader      "Remove map layer"
    trailer     "from display"
    variable    layer2del , 0 , 18
 }

WINDOW  asklayer  {
    num_rows    2
    num_columns 41
    datum       layer2add , 0 ,  1 ,    "CALL add_layer"
    datum       layer2del , 1 ,  1 ,    "CALL del_layer"
 }

TEXTLINE  domel   {"Return to Map Control Screen"}

BUTTON    domel {
     num_rows   1
     num_columns    33
     textline   domel, 0 ,2
     helpfile   help/newassmnt.hlp
 }

WINDOW   domel {
     num_rows   1
     num_columns    50
     button     domel, 0 ,2,        "NEW_SCREEN mapcontrol"
     helpfile   help/newassmnt.hlp
 }

SCREEN  mapman {
     title      "MAP SCREEN MANAGEMENT"
     window     assnamcom ,  2 ,  1
     window     layertxt ,   5 ,  2
     window     layernow ,   5 , 54
     window     asklayer ,  19 ,  1
     window     domel ,     21 ,  1
     border     YES
     }

-----------------------------------------------------------------------
-    Declarations for ADD OR DELETE INFORMATION FROM A MAP
-----------------------------------------------------------------------

TEXTLINE  notavail  {"Sorry, this facility is not yet available"}

WINDOW    addelmapinfo   {
     num_rows   5
     num_columns    78
     textline   notavail,   4,10
     helpfile   help/nohelp.hlp
     }

SCREEN  addelmapinfo {
     title      "ADD OR DELETE INFORMATION ON A MAP LAYER"
     window     assnamcom, 2, 1
     window     addelmapinfo,  5, 1
     window     domel ,     14 , 15
     window     majoraction,    18,1
     border     YES
     }


-----------------------------------------------------------------------
-         Declaration for entering coordinates window
-----------------------------------------------------------------------

TEXTLINE  entupplf
  {"Enter upper-left corner coordinates of area of current interest"}

TEXTLINE  entlowrt
  {"Enter lower right corner coordinates of area of current interest"}
```

```
DATUM   shwlat2 {
     num_rows   1
     num_columns    25
     variable   shwlat, 0 ,11
     leader     "Latitude: "
     helpfile   help/combmap.hlp
  }

DATUM   shwlong2 {
     num_rows   1
     num_columns    25
     variable   shwlong, 0 ,12
     leader     "Longitude: "
     helpfile   help/combmap.hlp
  }

WINDOW  emtocor  {
    num_rows      10
    num_columns   78
    textline      emtupplf,  2, 2
    datum         emtlat,    3, 1,   "CALL lat2dec &emt","NEWVALS"
    datum         emtlong,   3, 38,  "CALL lon2dec &emt","NEWVALS"
    textline      emtlowrt,  5, 2
    datum         shwlat2,   6, 1,   "CALL lat2dec &show","NEWVALS"
    datum         shwlong2,  6, 38,  "CALL lon2dec &show","NEWVALS"
 }

- end of g.sdf
```

Appendix B
PROGRAM LISTINGS

The following listings are the C-language software modules of which ASAN is composed.

```
/*****************************************************************************
*                                                                            *
*          asan.pc  --  ASAN Main Program                                    *
*                                                                            *
*             1. Opens the printer.                                          *
*             2. Calls strtASAN to establish communication with ORACLE       *
*             3. Verifies that ASAN software is valid.                       *
*             4. Calls Uinit to start the screen driver.                     *
*             5. At end of the session closes database and printer           *
*                                                                            *
*****************************************************************************/

#include <stdio.h>
#include <process.h>
#include <string.h>
#include <time.h>


EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE fmharris.h;
EXEC SQL INCLUDE hostvars.h;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;
#define SQLCA_STORAGE_CLASS
#include "asan.h"



main()
{
extern int ERA_DEBUG_FEATURES;

int i;
int CC_u211_spheroid(), closeORA(), logentry(), strtASAN(), vfy_ASAN();
void expOmsg();
static char *legal_notice[11] =
{"\n\n\n\n\t\t\t  RESTRICTED RIGHTS LEGEND\n\n",
"\tUse, duplication, or disclosure is subject to restrictions\n",
"\t      as set forth in subdivision (b)(3)(ii) of the\n",
"\t   Rights in Technical Data and Computer Software Clause\n",
"\t\t    at 52.227-7013 of the DOD FAR Supplement.\n\n",
"\t\t\tBBN LABORATORIES INCORPORATED\n",
"\t\t\t      10 MOULTON STREET\n",
"\t\t\t     Cambridge, MA 02238\n",
"\t\t\t        617-873-3000\n\n\n",
"     User Interface Copyright (C) 1985,  BBN Laboratories Incorporated\n",
"\t\t\t     All Rights Reserved");


FILE *fopen();

prn = fopen("prn","a");
if (prn == NULL) {
    printf("\nCan't open printer!");  exit(128); }
dante = fopen("chromfil.asf","a");
if (dante == NULL) {
    printf("\nCan't open chromfile!");  exit(128); }

printf("%c[2J",27);
for (i = 0; i < 11; i++) printf("%s",legal_notice[i]);

printf("\n\n\n\n\t   Please tap the space bar to continue, \"CTRL-C\" \
to abort.\n\n");
do {
    i = getch();
    } while (i != ' ');

printf("\nASAN starting.....");
```

```
MRA_DEBUG_FEATURES = 1;

utmzone = 12;
CC_u2ll_spheroid("clark66");

strtASAN();
#ifdef CHECKOUT
printf("\aSuccessfully connected as Project %d (%s)\aPrivileges are",
        ASSESSMENT.id, ASSESSMENT.name);
for (i = 0; i < 3; i++)  printf(" %c ", ASSESSMENT.auth[i]);
#endif

/*      You are now successfully CONNECTed to ORACLE.  Compare the      */
/*      software actually running to what is in the validation file.  */

if (vfy_ASAN()) exit(16);

/*      You are now successfully CONNECTed to ASAN.  Control is         */
/*      transferred to Uinit, whence it returns only at the very        */
/*      end unless something untoward happens along the way and         */
/*      ASAN decides to pull the plug in midstream somewhere.           */

Uinit();
logentry();
closeORA();
printf("%c[24;1H",27);
exit(0);
}


/*****************************************************************************
*                                                                            *
*     con2db.pc   --   Routines connecting a USERNAME to ORACLE,  etc.      *
*                                                                            *
*     This file contains:                                                    *
*                                                                            *
*     ASANconn        - connects any ASAN Assessment to ORACLE               *
*     closeORA        - closes ORACLE DBMS                                   *
*     lastsess        - retrieves last session from the logbook              *
*     logentry        - makes an entry in the Assessment's logbook           *
*     myOname         - fills in ORACLE user portion of ASSESSMENT structure *
*     enrollORA       - enrolls a new Assessment as an ORACLE User           *
*     SUconn          - connects ASAN's SUPERUSER to ORACLE.                 *
*     vfyOid          - determines if a username exists in ORACLE.           *
*                                                                            *
*****************************************************************************/

#include <process.h>                    /*  Header for calls to MS-DOS     */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files        */
EXEC SQL BEGIN DECLARE SECTION;         /*  All SQL declarations for this */
EXEC SQL INCLUDE hostvars.h;            /*  are in these two header files */
EXEC SQL INCLUDE fmharris.h;            /*  this one comes from "U"        */
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                       /*  Standard ASAN Header File      */


int ASANconn(name)
/*****************************************************************************
*                                                                            *
*          ASANconn  --   Connect an ASAN Assessment to ORACLE               *
*                                                                            *
*      Routine ROLLS BACK any outstanding transactions of the current       *
*      user and then connects to the requested user.  Returns an error      *
*      code which indicates whether or not the CONNECT was successful.      *
*                                                                            *
*                                                                            *
*     Notes:   1.  The existence of the assessment on the database is       *
*                  assumed.  ( vfyOid(name) is available to check.)          *
*                                                                            *
*              2.  If the name requested is not a "real" assessment         *
*                  but one of the privileged ORACLE names (used for          *
*                  system maintenance, DBA functions, etc.) the CONNECT      *
*                  will fail (sqlca.sqlcode == SQL_BAD_LOGON).  If so,       *
*                  you will be re-CONNECTed to the old assessment and        *
*                  the error code of the failed connect is returned          *
*                  to the calling program.                                   *
*                                                                            *
```

```
*****************************************************************************/

char name[];
{
register int i, j;
int temp, logentry(), loggedon(), myOname(), recon, SUconn();
void expOmsg();
char *clock();

#ifdef CHECKOUT
   printf("\nASANconn: %s", name);
#endif

loggedon();          /* Verify that we know who the person responsible is   */
SLOUT("Please stand by: Switching assessments....");
logentry();          /* Make entry in log book and disconnect previous user */
WHEREAMI(Screen, Window, Datum, Button);
misslabl.arr[0] = misdesc.arr[0] = '\0';
misslabl.len    = misdesc.len = 0;
if ( (sqlca.sqlcode == 0) ||          /* These are reasonable return codes */
     (sqlca.sqlcode == NOT_LOGGED_ON) ||
     (sqlca.sqlcode == NOT_CONNECTED) )   {
   strcpy ( uid.arr, name );          /* Note:  The SQL CONNECT statement   */
   uid.len = strlen ( uid.arr );      /* doesn't like an immediate password */
   strcpy ( pwd.arr, univpwd);        /* and a dynamic username.....        */
   pwd.len = strlen ( pwd.arr );
   EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
   temp = (int) sqlca.sqlcode;
   #ifdef CHECKOUT
   printf(" Connect = %ld", sqlca.sqlcode);
   #endif

   switch(temp) {                        /* What did ORACLE come up with? */
   case 0:                               /* CONNECT Succeeded as planned  */
      fprintf(dante,                     /* Load the ASSESSMENT Structure */
     "\n%s Connected %s to ID = %d (%s)", /* and if that was o.k. you can  */
        clock(), planrnum.arr,           /* Pick up the description from  */
        usernc, uid.arr);                /* the site's table of contents  */
      if (!myOname()) {
         EXEC SQL SELECT description
              FROM   table_of_contents
              INTO   :workspace
              WHERE  idnumber = :usernc;
         #ifdef CHECKOUT
            printf(" %ld", sqlca.sqlcode);
         #endif
         if (!sqlca.sqlcode) {  /* Found a description in table of contents */
            workspace.arr[workspace.len] = '\0';    /* Truncate description */
            j = (sizeof ASSESSMENT.desc) -1;        /* to fit on the screen */
            if (workspace.len >= j) {
               for(i = 0; i < j; i++) ASSESSMENT.desc[i] = workspace.arr[i];
               ASSESSMENT.desc[j] = '\0'; }
            else  strcpy(ASSESSMENT.desc, workspace.arr);
            }
         else {  /* No Description.....  How can this be???? */
            ERROMSG;
            fprintf(dante, "\n%s ASANconn: %s not in T.O.C", clock(), uid.arr);
            fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
            SLOUTEP("Security Violation: Table of contents error");

/* This will become a security violation!
```

```
            closeORA();
            exit(255);        But for now, let it go by   */


            strcpy(ASSESSMENT.desc,
                   "Description missing from table of contents");
            }

         EXEC SQL SELECT TO_CHAR(SYSDATE,'DD-Mon-YY HH24:MI:SS')
                         FROM dual INTO :timest1;
         #ifdef CHECKOUT
         printf("%ld", sqlca.sqlcode);
         #endif

         EXEC SQL SELECT TO_CHAR(start_work,'dd-Mon-yy HH24:MI:SS')
                  FROM   logbook
                  INTO   :startdate
                  WHERE  start_work =
                         (SELECT MIN(start_work) from logbook);
```

```
        #ifdef CHECKOUT
        printf("%ld", sqlca.sqlcode);
        #endif

        if (sqlca.sqlcode) {                    /* If you are in the T of C  */
           if (sqlca.sqlcode == SQL_NOF) {  /* but not here, you are new */
              strcpy(startdate.arr, timest1.arr, timest1.len);
              startdate.len = timest1.len;
              strcpy(lastdate.arr, timest1.arr, timest1.len);
              lastdate.len = timest1.len;
              lastmtr.len = 0;
              lastmoa.len = 0;}
           else {
              ENDOMSG;
              fprintf(dante, "\n%s ASANocan: %s logbook error",
                      clock(), uid.arr);
              fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerm.sqlermc); }
           }
        else {
           EXEC SQL SELECT lastmtr, lastmoa
                    FROM   logbook
                    INTO   :lastmtr, :lastmoa
                    WHERE  stop_work =
                           (SELECT MAX(stop_work) from logbook);
           #ifdef CHECKOUT
           printf("%ld", sqlca.sqlcode);
           #endif
           }
        lastmtr.arr[lastmtr.len] = '\0';
        lastmoa.arr[lastmoa.len] = '\0';

        /* All that's left now is to put up the screen */

        NEW_SCREEN("probestatscreen");
        NEWVALS();
        if (ASSESSMENT.auth[2] != 'Y')
           SLOUTRP("This assessment does not have resource authorization");
        return (int) 0;
        }
     else {
        ENDOMSG;
        fprintf(dante, "\n%s Mysterious failure on %s", clock(), uid.arr);
        fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerm.sqlermc);
        fprintf(dante, "\n\t\t\t  while retrieving info from SYS.V4EXPUSER");
        temp = (int) sqlca.sqlcode;
        SLOUTRP("Undiagnosible system failure. ASAN will restart");
        strcpy(ASSESSMENT.name, "SUPERUSER");
        NEW_SCREEN("firstscreen");
        break;}

 /* Trouble in River City if you get this far down in the switch ....  */

 case SQL_NO_USERNAME:
    SLOUTRP("You cannot use a blank as a name");
    break;

 case SQL_BAD_LOGON:
 case NOT_LOGGED_ON:
    ENDOMSG;
    fprintf(dante, "\n%s Authorization failure on %s", clock(), uid.arr);
    fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerm.sqlermc);
    sprintf( workspace.arr, "Unsuccessful: %s",
             sqlca.sqlerm.sqlermc);
    SLOUTRP(workspace.arr);
    break;

 default:
    ENDOMSG;
    fprintf( dante, "\n%s Unsuccessful logon: %s", clock(), uid.arr);
    fprintf( dante, "\n\t\t\t  ", sqlca.sqlerm.sqlermc);
    sprintf(workspace.arr, "Unsuccessful: %s", sqlca.sqlerm.sqlermc);
    SLOUTRP(workspace.arr);
 }  /*  End SWITCH  */

 strcpy ( uid.arr, ASSESSMENT.name);      /*  Re-CONNECT with   */
 uid.len = strlen ( uid.arr );            /*  previous name.    */
 if ( strcmp("SUPERUSER", uid.arr) ) {
    EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
    recon = (int) sqlca.sqlcode;
    /* Re-initialize the screen. Logentry() call has closed all cursors! */
    if (!recon && (!strcmp($screen,"chgcursscreen")) ) {
       pschgass();
```

```
            return temp; }
         }
      else recon = SUconn();

      if (recon) {            /* Now we're stuck beyond help */
         BRDCMSG;
         SLOUTRP("ASAN got \"stuck\" looking for the data.....");
         fprintf( dante, "\n%s ASANconn stuck: %s", clock(),
                  sqlca.sqlerrm.sqlerrmc);
         fprintf( dante, "\n\t\t\t\t  After attempted re-connect to %s", uid.arr);
         closeORA();
         exit(255);}
      SLOUTRP("ASAN must restart due to an internal error");
      NEW_SCREEN("firstscreen");
      return temp;
      }
   else { emptmsg();                  /* If you get here, you've been doing */
      fprintf(dante,                  /* something  outstandingly bizarre a */
      "\n%s Forced shut-down by %s", /* long time before this function was */
      clock(),                        /* called.  No idea how to recover!   */
      sqlca.sqlerrm.sqlerrmc);
      closeORA();
      exit(255);
      }
}


closeORA()
/*************************************************************************
*                                                                        *
*          closeORA   --   Disconnect an ASAN Assessment from ORACLE     *
*          ==========      and return ORACLE to original state as        *
*                          given by OO_rcode.                            *
*                                                                        *
*      Routine ROLLS BACK any outstanding transactions of the current    *
*      user and then disconnects from ORACLE.  Returns an error code     *
*      which indicates whether or not the call was successful.           *
*                                                                        *
*                                                                        *
*     Notes:   1.  The value of OO_code is set in the main() function    *
*                  as the return code of the initial logon attempt.  If  *
*                  these codes change in subsequent releases of ORACLE   *
*                  this function must be recompiled with the new codes.  *
*                                                                        *
*              2.  Unless the current user is SUPERUSER or this function *
*                  is called because of an unrecoverable ORACLE error,   *
*                  calls to closeORA() should be preceded by logentry();*
*                                                                        *
*************************************************************************/
{
int dcode, fclose(), spawnlp();
char *clock();

#ifdef CHECKOUT
printf("\ncloseORA");
#endif

SLOUT("                          ***  End ASAN ***");
fprintf(dante, "\n%s ASAN Close-down process:", clock());

EXEC SQL ROLLBACK WORK RELEASE;
#ifdef CHECKOUT
printf(" rolled back %ld", sqlca.sqlcode);
#endif
if (OO_rcode == 0) {       /* We're done if ORACLE was up when we started */
   fprintf(dante, " Servers retained intact");
   return 0;
   }
if ((OO_rcode == ORA_UNAVAILABLE) || (OO_rcode == -3120)) {
   dcode = spawnlp(P_WAIT, "ior.exe", "ior.exe", "shut", NULL);
   #ifdef CHECKOUT
   printf(" %d", dcode);
   #endif
   fprintf(dante, "\n%s Shut down ORACLE Server (%d)", clock(), dcode);
   if (OO_rcode != ORA_UNAVAILABLE) {
      dcode = spawnlp(P_WAIT, "remora.exe", "remora.exe", "all", NULL);
      #ifdef CHECKOUT
      printf(" %d", dcode);
      #endif
      fprintf(dante, "\n%s Deinstalled SQLNET (%d)", clock(), dcode);
      return dcode;
      }
```

```
   }
else {  /* Could be ASAN wasn't installed, or ..... */
   fprintf(dante, " Servers left unchanged (Code %d)", OO_rcode);
   }
fprintf(dante,"\n");
fclose(dante);
fclose(prm);
}



int lastsess()
/*************************************************************************
*                                                                        *
*     lastsess  --   Retrieves last entry from the ASAN logbook.  It is  *
*                    used when we need to establish the last activity on *
*                    the system.  (SUPERUSER does not log its activity.) *
*                                                                        *
*     Returns the value of sqlca.sqlcode to calling program              *
*                                                                        *
*************************************************************************/
{
char *clock();

#ifdef CHECKOUT
printf("\nlastsess ");
#endif

SLOUT("Re-loading the last session's parameters");
EXEC SQL SELECT planmar, TO_CHAR(stop_work,'DD-Mon-YY HH24:MI:SS'), idnumber
/*       FROM   lastlog */
         FROM   last_login
         INTO   :planrlst, :lastdate, :usernc;

#ifdef CHECKOUT
printf("%ld", sqlca.sqlcode);
#endif

if (!sqlca.sqlcode) {              /* This is what happens when all is O.K. */
   planrlst.arr[planrlst.len] = '\0';
   lastdate.arr[lastdate.len] = '\0';
   }

else if ( sqlca.sqlcode == SQL_EOF ) { /* When you start the very first time
*/
   strcpy(planrlst.arr, "No work ever done yet");
   planrlst.arr[21] = '\0';
   planrlst.len     =   21;
   lastdate.arr[0]  = '\0';
   lastdate.len     =    0;
   }

else {
   ERROMSG;
   fprintf(dante,
    "\n%s Incomprehensible return code %ld retrieving last session",
          clock(), sqlca.sqlcode);
   fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
   expOmsg();
   }

#ifdef CHECKOUT
printf("\n%s, %s,  %d", planrlst.arr, lastdate.arr, usernc);
#endif
return (int) sqlca.sqlcode;
}


int logentry()
/*************************************************************************
*                                                                        *
*     logentry  --   Makes an entry in the ASAN register for the work just *
*                    performed on the assessment.  It is called by the   *
*                    ASANconn and SUconn routines when they close out the *
*                    previous assessment.                                *
*                                                                        *
*                                                                        *
*     Returns the value of sqlca.sqlcode to calling program              *
*                                                                        *
*************************************************************************/
{
```

```
int rcode;
char *clock();

#ifdef CHECKOUT
   printf("\nlogentry: ");
#endif

if (strcmp(ASSESSMENT.name,"SUPERUSER")) { /* Make entry in log book if the */
                                           /* previous name was not SUPERUSER.  */
   EXEC SQL ROLLBACK WORK;                 /* Terminate whatever is outstanding */
   #ifdef CHECKOUT                         /* One normally expects 0 here, but  */
      printf(" %ld", sqlca.sqlcode);       /* occasionally something horrid has */
   #endif                                  /* been known to pop up here.        */
   if (sqlca.sqlcode != 0) {
      ENDOMSG;
      if ((sqlca.sqlcode != NOT_LOGGED_ON) &&
          (sqlca.sqlcode != NOT_CONNECTED)) {
         sprintf(workspace.arr, "Logbook choked on: %s",
                 sqlca. sqlerrm.sqlerrmc);
         fprintf(dante,"\n%s %s\t\t\t  making logbook entry for %s",
                 clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.name);
         SLOUTBP(workspace.arr);}
      }
   else {     /* All is well and we are ready to make the logbook entry     */
      userno = ASSESSMENT.id;
      EXEC SQL SELECT TO_CHAR(SYSDATE,'dd-mon-yy HH24:MI:SS')
               FROM DUAL INTO :timest2;
      EXEC SQL INSERT INTO  register(planner,    lastmtr,   lastmon,
                                     start_work, stop_work, idnumber)
                     VALUES (:planrnam, :lastmtr, :lastmon,
                             TO_DATE(:timest1,'dd-mon-yy HH24:MI:SS'),
                             TO_DATE(:timest2,'dd-mon-yy HH24:MI:SS'),
                             :userno);
      #ifdef CHECKOUT
         printf("INSERT %ld", sqlca.sqlcode);
      #endif
      if (rcode = sqlca.sqlcode) {     /* Just in case it fails.....        */
         ENDOMSG;
         fprintf(dante, "\n%s Logbook entry failed: %s",
                     clock(), sqlca.sqlerrm.sqlerrmc);
         fprintf(dante, "\t\t\t  %s (ID = %d)", ASSESSMENT.name, userno);
         fprintf(dante, "n\t\t\t %s, %s, %s",
                 planrnam.arr, timest1.arr, timest2.arr);
         EXEC SQL ROLLBACK WORK RELEASE;
         #ifdef CHECKOUT
            printf(" Rolled back %ld", sqlca.sqlcode);
         #endif
         }
      else {
         EXEC SQL COMMIT WORK RELEASE;
         #ifdef CHECKOUT
            printf(" Committed %ld", sqlca.sqlcode);
         #endif
         fprintf(dante, "\n%s Disconnected %s from ID = %d (%s)",
                     clock(), planrnam.arr, userno, ASSESSMENT.name);
         }
      return rcode;
      }
   }
else {                                     /* If you are SUPERUSER, just make  */
   EXEC SQL ROLLBACK WORK RELEASE;         /* a call to log off and return     */
   #ifdef CHECKOUT
      printf(" %ld (SU Bypassed)", sqlca.sqlcode);
   #endif
   }
return (int) sqlca.sqlcode;
}


int myOname()
/*****************************************************************************
*                                                                           *
*     myOname   --   To set the ORACLE USER information in the ASSESSMENT   *
*                    structure so it is available to all ASAN routines.     *
*                                                                           *
*                    Looks in view SYS.V4EXPUSER and retrieves USERID,      *
*                    MYNAME, TIMESTAMP and authorization codes.             *
*                                                                           *
*     Returns the value of sqlca.sqlcode to calling program                 *
*                                                                           *
*****************************************************************************/
```

```
{
#ifdef CHECKOUT
printf("\nmyOname: ");
#endif

EXEC SQL SELECT myid, myprivs, myname
                FROM sys.v4expuser
                INTO :usernc, :O_auth, :my_name;
my_name.arr[my_name.len] = '\0';

#ifdef CHECKOUT
printf(" %ld - %d (%s)\n", sqlca.sqlcode, my_name.len, my_name.arr);
#endif
strcpy(ASSESSMENT.name, my_name.arr);        /*  Store results in global */
ASSESSMENT.id = usernc;                      /*  ASSESSMENT structure    */
ASSESSMENT.auth[0] = O_auth.arr[0];
ASSESSMENT.auth[1] = O_auth.arr[1];
ASSESSMENT.auth[2] = O_auth.arr[2];

return (int) sqlca.sqlcode;
}


int nrollORA(name)
/*************************************************************************
*                                                                        *
*     nrollORA -- ASAN's SUPERUSER Grants CONNECT and RESOURCE           *
*                 to new ORACLE user                                     *
*                                                                        *
*     Routine   a).  Signs on as SUPERUSER, if current user not DBA.     *
*               b).  Enrolls a new ORACLE USERNAME.                      *
*               c).  Signs on under the new assessment's name.           *
*                                                                        *
*   Returns to calling program with sqlca.sqlcode of last step executed. *
*   Routine terminates immediately if a sub-step returns a non-zero code. *
*                                                                        *
*     Returns (ZERO) if successful or (NON-ZERO) if not successful.      *
*             Most likely, in order of appearance                        *
*             -0001 - Name already exists                                *
*             -0954 - No IDENTIFIED BY clause (i.e. blanks in name)      *
*             -0987 - Illegal character in username                      *
*             -nnnn - Any other ORACLE error that can occur              *
*                                                                        *
*************************************************************************/

char name[];
{
int dlen, rcode, vfyOid(), SUcomm();
register int i, j;
char *clock();
static char inputline[] = "emtdesc ";

#ifdef CHECKOUT
printf("\nnrollORA");
#else
SLOUT("A moment, please, while ASAN updates its database.......");
#endif

if ( rcode = SUcomm() ) {    /* SUPERUSER's services are required to do this */
   #ifdef CHECKOUT
      printf(" %ld", sqlca.sqlcode);
   #endif
   sprintf(workspace.arr, "SUPERUSER LOGON REQUEST CRASHED Error %d.",
           rcode);
   SLOUTEP(workspace.arr);
   SLOUTP("You need serious help!.......");
   fprintf(dante, "\n%s NROLLORA: Superuser logon crashed", clock());
   ENDCMSG;
   fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerm.sqlermc);
   closeORA();
   exit(255);
   }

j = n2bv.len = strlen(name);         /* Convert name to uppercase because  */
for (i = 0; i < j; i++)              /* ORACLE keeps USERNAME in uppercase */
   n2bv.arr[i] = name[i] = toupper(name[i]);
n2bv.arr[n2bv.len] = '\0';
strcpy(workspace.arr, "GRANT CONNECT, RESOURCE TO "); /* The GRANT statement */
strcat(workspace.arr, name);                          /* next is not part of */
strcat(workspace.arr, " IDENTIFIED BY ");             /* ANSI standard SQL!  */
strcat(workspace.arr, univpwd);
workspace.len = strlen(workspace.arr);
```

```
EXEC SQL EXECUTE IMMEDIATE :workspace;

#ifdef CHECKOUT
printf("\n%s %ld", workspace.arr, sqlca.sqlcode);
#endif

if (sqlca.sqlcode) { /* Who knows what evil lurks in the heart of ORACLE? */
   fprintf(dmnte, "\n%s Failed to enrol %s\n\t\t\t %s",
                  clock(), name, sqlca.sqlerrm.sqlerrmc);
   return (int) sqlca.sqlcode;}      /* This traps any illegal characters */
                                     /* and other strange things....      */

fprintf(dmnte, "\n%s Successfully enrolled %s", clock(), name);

EXEC SQL SELECT userid
                FROM sysuserlist
                INTO :uid
                WHERE username = :n2bv;

ADD_WINDOW( "newusdesc" , 5, 5);       /*  User may enter up to 240 */
workspace.arr[0] = '\0';               /*  characters of drivel to  */
for (i = 0; i < 4; i++) {              /*  identify this assessment */
   inputline[7] = i+49;
   UPDATE_DATUM(inputline);
   emtdesc[i].len = strlen(emtdesc[i].arr);
   if (emtdesc[i].len == 0) break;
   strcat(workspace.arr, emtdesc[i].arr);
   if (emtdesc[i].len < 50) break;
   }
workspace.len = strlen(workspace.arr);
REMOVE_WINDOW();

EXEC SQL INSERT INTO table_of_contents (idnumber, description)
                VALUES (:uid, :workspace);

if (sqlca.sqlcode) {
   ENDCOMM;
   fprintf(dmnte, "\n%s Table of contents entry failed\n\t\t\t %s ",
                  clock(), sqlca.sqlerrm.sqlerrmc);
   SLOUTP(sqlca.sqlerrm.sqlerrmc);
   rcode = (int) sqlca.sqlcode;        /*  This will cause the assessment  */
   EXEC SQL ROLLBACK WORK;             /*  to become invisible to ASAN or  */
   #ifdef CHECKOUT                     /*  may lead to security violation  */
   printf(" %ld", sqlca.sqlcode);      /*  errors that will stop execution */
   #endif
   }
else {
   EXEC SQL COMMIT WORK;
   rcode = ASANcomm(name);             /*  Once you get this far, sign on. */
   #ifdef CHECKOUT                     /*  This better never be non-zero!  */
   printf(" %ld", sqlca.sqlcode);
   #endif
   }
return rcode;
}


int SUcomm()
/*****************************************************************************
*                                                                            *
*          SUcomm    --     Connect ASAN's SUPERUSER to ORACLE              *
*                                                                            *
*                                                                            *
*       Returns SQLCA.SQLCODE                                                *
*                                                                            *
*****************************************************************************/
{
static char dbauid[] = {0123,0125,0120,0105,0122,0125,0123,0105,0122,057,
                        0115,0105,0120,0110,0111,0123,0124,0117,0120,0110,
                        0105,0114,0105,0123,'\0'};

int temp, logentry(), myname();

#ifdef CHECKOUT
printf("\nSUcomm: ");
#endif

logentry();    /* Make logbook entry and terminate whatever is outstanding */

strcpy( uid.arr, dbauid );            /* Connect */
uid.len = strlen( uid.arr );
EXEC SQL CONNECT :uid;
```

```
#ifdef CHECKOUT
printf(" %ld", sqlca.sqlcode);
#endif

if (!sqlca.sqlcode) myOname();
strcpy(ASSESSMENT.desc,"Owner of ASAN's Administrative Information");
return (int) sqlca.sqlcode;
}


int vfyOid(name)
/*************************************************************************
*                                                                        *
*    vfyOid   --   Verify that a USERNAME exists in ORACLE's Dictionary  *
*                                                                        *
*                                                                        *
*     Determines if the username exists from the SYSUSERLIST View.       *
*     Returns SQLCA.SQLCODE of the SELECT statement.                     *
*                                                                        *
*************************************************************************/

char  name[];            /*  Argument is pointer to character string  */
                         /*  containing the username to validate      */
{
register int i;

#ifdef CHECKOUT
printf("\nvfyOid");
#endif

n2bv.len = strlen(name);          /*  Set up for WHERE clause of the SELECT  */
for (i = 0; i < n2bv.len; i++)    /*  ORACLE likes uppercase names */
     n2bv.arr[i]  = toupper(name[i]);

EXEC SQL SELECT usrid
         FROM   sysuserlist
         INTO   :usrno
         WHERE  username = :n2bv ;

#ifdef CHECKOUT
     printf(" %ld", sqlca.sqlcode);
#endif

return (int) sqlca.sqlcode;
}
/*************************************************************************
*                                                                        *
*      housarpt.pc  --  Set of routines to print "housekeeping" data from *
*      ============      the ORACLE Data Dictionary.                     *
*                                                                        *
*    Routines in this file:                                              *
*                                                                        *
*      SUprint      -  Allows SUPERUSER to print an assessments tables   *
*      SUpropt      -  Selects level of detail for SUprint or prints     *
*                      HEADQUARTERS or SUPERUSER tables with that option *
*      prntabs()    -  Prints names and comments on tables for assessment *
*      prntcol()    -  Prints names of all tables and names and comments *
*                      of columns within tables                          *
*      prntabc()    -  Prints TABLE_OF_CONTENTS entries                  *
*                                                                        *
*************************************************************************/


#include <stdio.h>                /*  The usual stuff, of course */
#include <process.h>              /*  Header for calls to MS-DOS */
#include <string.h>               /*  String manipulation header */

#define SQLCA_STORAGE_CLASS extern
EXEC SQL INCLUDE SQLCA;           /*  SQL Communication Area     */
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE hostvars.h;
EXEC SQL END DECLARE SECTION;
#include "asantype.h"
#include "asan.h"                 /*  Standard ASAN Header file  */

static printoption = 0;


int SUprint(name)
/*************************************************************************
*                                                                        *
```

```
*     SUprint()   ---   Print tables for an assessment (called from the     *
*     ==========        housekeeping screen while SUPERUSER is connected    *
*                                                                           *
*     Routine logs on as the named assessment, prints tables as that        *
*     assessment, returns to SUPERUSER, puts up housekeeping screen again.  *
*                                                                           *
*****************************************************************************/
char name [];
{
char *clock();

strcpy (uid.arr, name);
uid.len = strlen(uid.arr);
strcpy (pwd.arr, univpwd);
pwd.len = strlen(pwd.arr);
EXEC SQL ROLLBACK WORK RELEASE;
EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
if (sqlca.sqlcode) {
   ENDOMSG;
   fprintf(dante, "\n%s Could not connect to %s for printing tables",
                  clock(), name);
   fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
   SLOUTRP("Could not connect to assessment");
   SLOUTP(sqlca.sqlerrm.sqlerrmc);
   }
else {
   myOname();
   if (printoption == 2) prntcol();
   else prntabs();
   }
strcpy(ASSESSMENT.name,"SUPERUSER");
SUcoan();
if (sqlca.sqlcode) {
   ENDOMSG;
   fprintf(dante, "\n%s Could not reconnect after printing tables for %s",
                   clock(), name);
   fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
   SLOUTRP("Could not reconnect to SUPERUSER");
   SLOUTP(sqlca.sqlerrm.sqlerrmc);
   exit(255);
   }
NEW_SCREEN("dbhsekpgscreen");
return (int) 0;
}



int SUpropt(value)
/*****************************************************************************
*                                                                           *
*     SUpropt()   ---   Set options for printing assessments or print the   *
*     ==========        HEADQUARTERS or SUPERUSER tables                    *
*                                                                           *
*     Routine logs on as the named assessment, prints tables as that        *
*     assessment, returns to SUPERUSER, puts up housekeeping screen again.  *
*                                                                           *
*****************************************************************************/
int value;
{
char *clock();
#ifdef CHECKOUT
   printf("\nSUpropt %d", value);
#endif


switch (value) {
case 1:
   printoption = 1;
   if (ulisto()) {
      ENDOMSG;
      SLOUTP(sqlca.sqlerrm.sqlerrmc);
      return (int) -1;}
   blnkdspl();
   NEW_SCREEN("slcasscreen");
   ubunch();
   return (int) 0;
case 2:
   printoption = 2;
   if (ulisto()) {
      ENDOMSG;
      SLOUTP(sqlca.sqlerrm.sqlerrmc);
```

```
        return (int) -1;}
    blnkdspl();
    NEW_SCREEN("slcasscreen");
    ubunch();
    return (int) 0;

 case 3:
 ulistc();
 cptr1 = hqconn;
 EXEC SQL ROLLBACK WORK RELEASE;
 EXEC SQL CONNECT :cptr1;
 if (sqlca.sqlcode) {
    ENDOMSG;
    fprintf(dante, "\n%s Could not connect to HEADQUARTERS for printing tables",
                   clock());
    fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
    SLOUTBP("Could not connect HEADQUARTERS");
    SLOUTP(sqlca.sqlerrm.sqlerrmc);
    }
 else {
    myOname();
    if (printoption == 2) prntcol();
    else prntabs();
    }
 strcpy(ASSESSMENT.name,"SUPERUSER");
 SUconn();
 if (sqlca.sqlcode) {
    ENDOMSG;
    fprintf(dante,"\n%s Could not reconnect after printing HEADQUARTERS tables",
                    clock());
    fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
    SLOUTBP("Could not reconnect to SUPERUSER");
    SLOUTP(sqlca.sqlerrm.sqlerrmc);
    exit(255);
    }
 NEW_SCREEN("dbhsekpgscreen");
 return (int) 0;

 default:
 ulistc();
 if (printoption == 2) prntcol();
 else prntabs();
    }
 NEW_SCREEN("dbhsekpgscreen");
 return (int) 0;
 }


 int prntabs()
 /*****************************************************************************
 *                                                                           *
 *     prntabs()   ---   Print a list of all tables for this assessment     *
 *     ========                                                              *
 *                                                                           *
 *     Routine opens cursor U2 then fetches rows until SQL_EOF is found.     *
 *     Each row printed on the system printer.  Then the cursor is closed.   *
 *                                                                           *
 *     Note:   Modifications to this function may impact the related         *
 *             functions tlisto(), tlistf() and tlistc() that open the       *
 *             cursor, fetch rows using it respectively, and close it        *
 *                                                                           *
 *****************************************************************************/
 {
 #define PAGESIZE 54
 #undef COMMENTSPACE
 #define COMMENTSPACE 54
 int i, j, line, lcp, ncp, page;              /* Counters for lines, etc.    */

 int tlisto();                                /* Database Utilities Used     */
 int tlistf();
 int tlistc();

 SLOUT("Printing tables");
 if ( ! tlisto() )  {                        /* Open sys.v4exptab query      */
    j = 0;
    page = 1;
    fprintf( prn, "\t\tInventory of Tables for %s\t\tPage %d\n",
             ASSESSMENT.name, page);
    line = 2;

    while ( sqlca.sqlcode != SQL_EOF )  {
```

```
        if ( tlistf() == SQL_EOF ) break;      /* Get name of the next table */
        if ( line >= PAGESIZE ) {              /* Make sure that you have space */
           fprintf( prn,"\f\t\t\tInventory of Tables for %s\t\tPage %d\n",
                    ASSESSMENT.name, ++page);
           line = 2;}

        j++;                                   /* Print name of the table */
        fprintf(prn, "\n%4d %-20s ", j, tid.arr);   /* Assumes <= 20 char! */

        lcp = 0; /* Print as many characters as the comment field contains */
        while (lcp < workspace.len) {
           ncp = (lcp+COMMENTSPACE < workspace.len) ?
                     lcp+COMMENTSPACE : workspace.len;
           for ( ; ncp > lcp; ncp--)      /* Find good spot to end this line */
              if (workspace.arr[ncp] == ' ' ||
                  workspace.arr[ncp] == '\0') break;

           if (lcp == ncp) {  /* In case we have a line without white space */
              fprintf(prn, "          ");
              for ( ; ncp < lcp+COMMENTSPACE; ncp++)
                 fprintf(prn,"%c",workspace.arr[ncp]);
                 fprintf(prn,"\n\t\t\t   ");
                 line++; }

           else { /* The normal case when we can find a white space somewhere */
              workspace.arr[ncp] = '\0';
              fprintf(prn, "%s\n\t\t\t   ", &workspace.arr[lcp]);
              line++;
              lcp = ncp+1; }
           line++;
           }  /* End WHILE (lcp)*/
        }. /* End WHILE (sqlca) */
    fprintf(prn, "\n\n\t\t\t\t==  END  ==\f");
    tlistc();
    }

#ifdef CHECKOUT
else {
   printf("\n\nOUCH!  My cursor did not open!");
   expOmsg();
   }
#endif
}


int prntool()
/*************************************************************************
*                                                                        *
*     prntool()    ---    Print a list of all columns CREATED BY this    *
*     =========           assessment sorted by table in which they occur *
*                                                                        *
*     Routine opens cursor U4 then fetches rows until SQL_EOF is found.  *
*     Each row printed on the system printer.  Then the cursor is closed. *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions tclisto(), tclistf() and tclistc() that open the *
*             cursor, fetch rows using it respectively, and close it     *
*                                                                        *
*************************************************************************/
{
#define PAGESIZE 54
#undef COMMENTSPACE
#define COMMENTSPACE 40

int tclisto();                          /* Database Utilities Used    */
int tclistf();
int tclistc();

int i, j, k, line, lcp, ncp, page;      /* Counters for lines, etc.    */
char qual_col[31];

SLOUT("Printing tables and columns");
if ( ! tclisto() )  {                   /* Open "COL" DataDict Query   */
   j = k = 0;
   page = 1;
   fprintf( prn, "\tInventory of Columns by Table for %s\t\tPage %d",
            ASSESSMENT.name, page);
   line = 3;
   strcpy(qual_col, "q");

   while ( sqlca.sqlcode != SQL_EOF )  {
```

```
        if ( tclistf() == SQL_EOT ) break;     /* Get name of the next column */

        if ( line >= PAGESIZE ) {              /* Make sure that you have space */
           fprintf(prn,"\f\tInventory of Columns by Table for %s\t\tPage %d\n",
                  ASSESSMENT.name, ++page);
           line = 3;}
                                                /* Print name of the table */
        if ( strncmp (qual_col, tid.arr, (unsigned int) tid.len))  {
           fprintf(prn, "\n\n%4d %s", ++j, tid.arr);
           k = 0;
           strcpy(qual_col, tid.arr);}
        else if (line == 3)  fprintf(prn, "\n\n%4d %s (Cont'd)", j, tid.arr);

                                               /* Print name of the column  */
        fprintf(prn, "\n\t%4d %-15s ", ++k, cid.arr);

        lcp = 0; /* Print as many characters as the comment field contains */
        if (workspace.len == 0) {
           fprintf(prn, "\n");
           line++;}
        else {
           while (lcp < workspace.len) {
              ncp = (lcp+COMMENTSPACE < workspace.len) ?
                        lcp+COMMENTSPACE : workspace.len;
              for ( ; ncp > lcp; ncp--)     /* Find good spot to end this line */
                 if (workspace.arr[ncp] == ' ' ||
                     workspace.arr[ncp] == '\0') break;

              if (lcp == ncp) {  /* In case we have a line without white space */
                 fprintf(prn, "        ");
                 for ( ; ncp < lcp+COMMENTSPACE; ncp++)
                    fprintf(prn, "\t");
                    fprintf(prn,"%c",workspace.arr[ncp]);
                    fprintf(prn,"\n\t\t\t\t  ");
                    line++; }

              else { /* Normally we can find a white space somewhere */
                 workspace.arr[ncp] = '\0';
                 fprintf(prn, "\t%s\n\t\t\t\t  ", &workspace.arr[lcp]);
                 line++;
                 lcp = ncp+1; }
              }  /* End WHILE (lcp)*/
           }  /* End IF(null string) */
        line++;
        }  /* End WHILE (sqlca) */
     fprintf(prn, "\n\n\t\t\t\t==  END  ==\f");
     tclistc();
     }

#ifdef CHECKOUT
else {
   printf("\n\nOUCH!  My cursor did not open!");
   expOmsg();
   }
#endif
}


int prntabc()
/*************************************************************************
*                                                                        *
*     prntabc()   ---   Print ASAN's TABLE_OF_CONTENTS                   *
*     ==========                                                         *
*                                                                        *
*     Routine opens cursor U5 then fetches rows until SQL_EOT is found.  *
*     Each row printed on the system printer.  Then the cursor is closed.*
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions vlisto(), vlistf() and vlistc() that open the    *
*             cursor, fetch rows using it respectively, and close it     *
*                                                                        *
*************************************************************************/
{
#define PAGESIZE 54
#undef COMMENTSPACE
#define COMMENTSPACE 54
int i, j, line, lcp, ncp, page;              /* Counters for lines, etc.    */

int vlisto();                                /* Database Utilities Used     */
int vlistf();
int vlistc();
```

```
char *now, *clock();

SLOUT("Printing ASAN's Table of Contents");
now = clock();
if ( ! vlisto() )  {                          /* Open sysuserlist query       */
   j = 0;
   page = 1;
   fprintf(prn,"\tInventory of ASAN Assessments %s\tPage %d\n", now, page);
   line = 2;

   while ( sqlca.sqlcode != SQL_NOF )  {
      if ( vlistf() == SQL_NOF ) break;     /* Get the next assessment name  */
      if ( line >= PAGESIZE ) {             /* Make sure that you have space */
         fprintf(prn,"\f\tInventory of Assessments %s\tPage %d\n", now, ++page);
         line = 2;}

      j++;
      fprintf(prn, "\n%4d %-20s ", j, uid.arr);   /* Assumes <= 20 char! */

      lcp = 0; /* Print as many characters as the comment field contains */
      while (lcp < workspace.len) {
         ncp = (lcp+COMMENTSPACE < workspace.len) ?
                  lcp+COMMENTSPACE : workspace.len;
         for ( ; ncp > lcp; ncp--)     /* Find good spot to end this line */
            if (workspace.arr[ncp] == ' ' ||
                workspace.arr[ncp] == '\0') break;

         if (lcp == ncp) {  /* In case we have a line without white space */
            fprintf(prn, "        ");
            for ( ; ncp < lcp+COMMENTSPACE; ncp++)
               fprintf(prn,"%c",workspace.arr[ncp]);
               fprintf(prn,"\n\t\t\t\t  ");
               line++; }

         else { /* The normal case when we can find a white space somewhere */
            workspace.arr[ncp] = '\0';
            fprintf(prn, "%s\n\t\t\t\t  ", &workspace.arr[lcp]);
            line++;
            lcp = ncp+1; }
         line++;
         }  /* End WHILE (lcp)*/
      }  /* End WHILE (sqlca) */
   fprintf(prn, "\n\n\t\t\t\t==  END  ==\f");
   vlistc();
   }

#ifdef CHECKOUT
else {
   printf("\n\nOUCH!  My cursor did not open!");
   sqpOmsg();
   }
#endif
}
/*************************************************************************
*                                                                        *
*     intlze   --   Create standard set of ASAN Assessment tables.       *
*     ======                                                             *
*                                                                        *
*     This file generates:                                               *
*                                                                        *
*                 Name                Index               Cluster        *
*                 ----------------    ----------------    ---------------*
*     CLUSTERS:   GEO_CLUS                                               *
*                                                                        *
*                                                                        *
*     TABLES:     R_LDN                                   GEO_CLUS       *
*                 R_LDNNR                                 GEO_CLUS       *
*                 R_LEQ                                   GEO_CLUS       *
*                 R_PEF                                   GEO_CLUS       *
*                                                                        *
*                 MISSIONS                                               *
*                 ACTIVITIES                                             *
*                 OPERATIONS                                             *
*                                                                        *
*                 MPR_FLIGHT_PARAM                                       *
*                 TAB_MPR_EXP                                            *
*                                                                        *
*************************************************************************/


#include <stdio.h>
#define SQLCA_STORAGE_CLASS extern  /*  Switch for header files            */
```

```
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL INCLUDE fmharris.h;
EXEC SQL INCLUDE hostvars.h;
EXEC SQL END DECLARE SECTION;
#include "asan.h"                    /*  Standard ASAN Header File          */



int imtlse()
{
/*************************************************************************
*                                                                        *
*               Create General Geographic cluster GEO_CLUS               *
*                                                                        *
*************************************************************************/

#ifdef CHECKOUT
     printf("\nimtlse ");
#endif

char *clock();

SLOUT("Reserving space for data");

EXEC SQL CREATE CLUSTER GEO_CLUS (X_COORD NUMBER, Y_COORD NUMBER);
#ifdef CHECKOUT
   printf("\nGeo_clus %ld", workspace.arr, sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   EXECMSG;
   SLOUTER(sqlca.sqlerrm.sqlerrmc);
   fprintf(data, "\n%s %s\n\t\t\t  Creating GEO_CLUS (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}


/*************************************************************************
*                                                                        *
*            Create table to receive data points of Ldn values           *
*                                                                        *
*                                                                        *
*     Note:  Only the clustered (X,Y) pairs are identified here.  The    *
*            computational routines will create the remaining (real)     *
*            columns and decide at run time what their names will be.    *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE R_LDN (
                X_COORD NUMBER NOT NULL,  /* X-Coordinate of Point */
                Y_COORD NUMBER NOT NULL)  /* Y-Coordinate of Point */
                CLUSTER GEO_CLUS (X_COORD, Y_COORD);
#ifdef CHECKOUT
     printf("\nR_LDN %ld", sqlca.sqlcode);
#else
   SLOUT("Leq and derivatives");
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   EXECMSG;
   SLOUTER(sqlca.sqlerrm.sqlerrmc);
   fprintf(data, "\n%s %s\n\t\t\t  Creating Table R_LDN (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}



/*************************************************************************
*                                                                        *
*            Create table to receive data points of Ldnmr values         *
*                                                                        *
*                                                                        *
*     Note:  Only the clustered (X,Y) pairs are identified here.  The    *
*            computational routines will create the remaining (real)     *
*            columns and decide at run time what their names will be.    *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE R_LDNMR (
```

```
                X_COORD NUMBER NOT NULL,  /* X-Coordinate of Point */
                Y_COORD NUMBER NOT NULL)  /* Y-Coordinate of Point */
                CLUSTER GEO_CLUS (X_COORD, Y_COORD);
#ifdef CHECKOUT
     printf("\nR_LDNMR %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   SLOUTER(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table R_LDNMR (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}


/*****************************************************************************
*                                                                            *
*            Create table to receive data points of Leq values               *
*                                                                            *
*                                                                            *
*     Note:  Only the clustered (X,Y) pairs are identified here.  The        *
*            computational routines will create the remaining (real)         *
*            columns and decide at run time what their names will be.        *
*                                                                            *
*****************************************************************************/

EXEC SQL CREATE TABLE R_LEQ (
                X_COORD NUMBER NOT NULL,  /* X-Coordinate of Point */
                Y_COORD NUMBER NOT NULL)  /* Y-Coordinate of Point */
                CLUSTER GEO_CLUS (X_COORD, Y_COORD);
#ifdef CHECKOUT
     printf("\nR_LEQ %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   ERROMSG;
   SLOUTER(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table R_LEQ (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}



/*****************************************************************************
*                                                                            *
*            Create table to receive data points of pdf values               *
*                                                                            *
*                                                                            *
*     Note:  Only the clustered (X,Y) pairs are identified here.  The        *
*            computational routines will create the remaining (real)         *
*            columns and decide at run time what their names will be.        *
*                                                                            *
*****************************************************************************/

EXEC SQL CREATE TABLE R_PDF (
                X_COORD NUMBER NOT NULL,  /* X-Coordinate of Point */
                Y_COORD NUMBER NOT NULL)  /* Y-Coordinate of Point */
                CLUSTER GEO_CLUS (X_COORD, Y_COORD);
#ifdef CHECKOUT
     printf("\nR_PDF %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   ERROMSG;
   SLOUTER(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table R_PDF (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}



/*****************************************************************************
*                                                                            *
*           MISSIONS:  Missions Flown in this Assessment                     *
*                                                                            *
*****************************************************************************/

EXEC SQL CREATE TABLE MISSIONS (
```

```
                MISSION     CHAR(7) NOT NULL, /* Mission Identifier      */
                TYPE        CHAR(1),          /* Mission Type            */
                DESCR       CHAR(60),         /* Description of Mission  */
                SORTIE_SIZE NUMBER);          /* Number of A/C in sortie */
#ifdef CHECKOUT
     printf("\nMISSIONS %ld", sqlca.sqlcode);
#else
   SLOUT("Missions and Operations");
#endif


if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   EXECMSG;
   SLOUTEP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table MISSIONS (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}



/*************************************************************************
*                                                                        *
*      ACTIVITIES:  Population of Sources by Mission by Aircraft          *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE ACTIVITIES (
                S_LABEL   CHAR(9),          /* Label of the "Source"  */
                M_IDENT   CHAR(7),          /* Identifier of mission  */
                AIRCRAFT  CHAR(12),         /* Specific Aircraft Name */
                ACTIVITY  NUMBER);          /* Internal Identifier    */
#ifdef CHECKOUT
     printf("\nACTIVITIES %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   EXECMSG;
   SLOUTEP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table ACTIVITIES (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}



/*************************************************************************
*                                                                        *
*       OPERATIONS: Number of day and night sorties by month             *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE OPERATIONS (
                ACTIVITY  NUMBER,       /* Internal Miss/Aircraft/Route Tag */
                MONTH     NUMBER(2,0),  /* Month for this entry  */
                DAY       NUMBER,       /* Daytime Operations &  */
                NIGHT     NUMBER,       /* Blokes Flying  Night */
                LASTUPD   DATE);        /* When last modified    */
#ifdef CHECKOUT
     printf("\nOPERATIONS %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   EXECMSG;
   SLOUTEP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table OPERATIONS (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);
   return (int) sqlca.sqlcode;}



/*************************************************************************
*                                                                        *
*      MTR_FLIGHT_PARAM: Flight parameters for each activity by segment  *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE MTR_FLIGHT_PARAM (
```

```
            ACTIVITY   NUMBER,          /* Internal Miss/Aircraft/Route Tag */
            FIX_LABEL  CHAR(12),        /* MTR Navigation Point Id.         */
            ALT_REF    CHAR(3),         /* Reference for altitude           */
            ALT        NUMBER,          /* Altitude segment is flown        */
            PWR        NUMBER,          /* Power setting for the segment    */
            SPD        NUMBER,          /* Speed at which it is flown       */
            SEQ        NUMBER(3,0));    /* Sequence number of the nav point */
#ifdef CHECKOUT
     printf("\nMTR_FLIGHT_PARAM %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   HNDCMSG;
   SLOUTRP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table MTR_FLIGHT_PARAM (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc,  ASSESSMENT.id);
   return (int) sqlca.sqlcode;}


/*************************************************************************
*                                                                        *
*      MTR_EXP_TAB: Tabulated Leq's for each activity.  For the moment   *
*                   there is a single table.  When segments can have     *
*                   different flight parameters the fate of this table   *
*                   should be reviewed.                                  *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE MTR_EXP_TAB (SIDELINE number(5,0));

/*
strcpy(workspace.arr,"CREATE TABLE MTR_EXP_TAB (SIDELINE number(5,0)) AS \
SELECT SIDELINE FROM SUPERUSER.MTR_EXP_TAB");

EXEC SQL EXECUTE IMMEDIATE :workspace;
#ifdef CHECKOUT
     printf("\nMTR_EXP_TAB %ld ", sqlca.sqlcode);
#endif
*/
if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   HNDCMSG;
   SLOUTRP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t  Creating Table MTR_EXP_TAB (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);}

EXEC SQL DECLARE T1 CURSOR FOR SELECT sideline FROM SUPERUSER.MTR_EXP_TAB;
EXEC SQL OPEN T1;
if ( sqlca.sqlcode ) {
   HNDCMSG;
   SLOUTRP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s %s\n\t\t\t Opening MTR_EXP_TAB (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);}
else {
   for (;;) {
      EXEC SQL FETCH T1 INTO :sideline;
      if (sqlca.sqlcode) break;
      EXEC SQL INSERT INTO mtr_exp_tab (sideline) VALUES (:sideline);
      if ( sqlca.sqlcode ) {
         HNDCMSG;
         SLOUTRP(sqlca.sqlerrm.sqlerrmc);
         fprintf(dante,"\n%s %s\n\t\t\t Inserting Table MTR_EXP_TAB (ID = %d)",
         clock(), sqlca.sqlerrm.sqlerrmc, ASSESSMENT.id);}
      EXEC SQL COMMIT WORK;
      }
   EXEC SQL CLOSE T1;
   }


/*************************************************************************
*                                                                        *
*  QUAL_CIT1 and QUAL_CIT2  Temporary tables for citation database queries  *
*                                                                        *
*************************************************************************/

EXEC SQL CREATE TABLE QUAL_CIT1(entry_num char(5));
```

```
#ifdef CHECKOUT
     printf("\nQUAL_CITY1 %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   ERROMSG;
   SLOUTRP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dmste, "\n%s %s\n\t\t\t  Creating Table QUAL_CITY1 (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc,  ASSESSMENT.id);
   return (int) sqlca.sqlcode;}

EXEC SQL CREATE TABLE QUAL_CITY2(entry_num char(5));

#ifdef CHECKOUT
     printf("\nQUAL_CITY2 %ld", sqlca.sqlcode);
#endif

if ( sqlca.sqlcode && sqlca.sqlcode != DUPLICATE_OBJECT) {
   ERROMSG;
   SLOUTRP(sqlca.sqlerrm.sqlerrmc);
   fprintf(dmste, "\n%s %s\n\t\t\t  Creating Table QUAL_CITY2 (ID = %d)",
      clock(), sqlca.sqlerrm.sqlerrmc,  ASSESSMENT.id);
   return (int) sqlca.sqlcode;}


return (int) sqlca.sqlcode;
}


/*************************************************************************
*                                                                        *
*     missions.pc  --  Routines for missions                             *
*     ============                                                       *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*        pschgmis  --  Set up routine to select a new mission            *
*        pmstrmis  --  Set up routine to select a mission and aircraft   *
*                      combination for a given MTR                       *
*        entermis  --  Enter a mission for this assessment               *
*        pmstrflt  --  Set up routine to process a mission and aircraft  *
*                      combination for a given MTR                       *
*        pnewmiss  --  Set up routine for entering a new mission         *
*        mstmispt  --  Store an MTR Navigation Point actually flown on a *
*                      particular activity and advance all pointers.     *
*        insmispt  --  Insert an MTR Navigation Point actually traversed *
*                      during a particular mission in the database table *
*                      of MTR segments flown                             *
*        psaddops  --  Put up the add operations window and accept data. *
*        expmops   --  Expand steady yearly operations over all months.  *
*        vfyMmis   --  Verify the existence of a mission as part of an   *
*                      assessment's definition                           *
*        cancmis   --  Cancel mission currently pending on the database  *
*        savemis   --  Commit mission currently pending on the database  *
*        MISocan   --  Connect an MTR with a Mission                     *
*                                                                        *
*                                                                        *
*                                                                        *
*                                                                        *
*     ORACLE access routines:                                            *
*                                                                        *
*       1. ...listo()   Open Cursor  (Opens a "logical file")            *
*       2. ...listf()   Fetch Cursor (Reads next record from logical file) *
*          ...bunch()   Fetch Cursor (Reads next bunch from logical file *
*                       applicable only for multiple choice options)     *
*       4. ...listc()   Close Cursor (Closes the logical file)           *
*                                                                        *
*     WHERE:                                                             *
*                                                                        *
*     ... =  ss  -  Names of all Missions defined for this assessment    *
*                   (Multiple choice display option is supported)        *
*     ... =  so  -  Operations of a particular activity                  *
*                                                                        *
*************************************************************************/

#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern    /*  Switch for header files       */
EXEC SQL INCLUDE SQLCA;
```

```
EXEC SQL BEGIN DECLARE SECTION;          /*  All SQL declarations are in   */
EXEC SQL INCLUDE hostvars.h;             /*  these header files            */
EXEC SQL INCLUDE fmharris.h;

EXEC SQL END DECLARE SECTION;
#include "asan.h"                        /*  Standard ASAN Header File     */

static missflg = 0;


int pschgmis()
/*************************************************************************
*                                                                        *
*          pschgmis  --  Set up routine to select a new mission          *
*          ========                                                      *
*                                                                        *
*     Routine (1) opens cursor for fetch of all missions known (in this  *
*                 version missions are always local to the assessment)   *
*             (2) fetches the first batch into memory                    *
*             (3) puts up "chgcurmtrscreen"                              *
*                                                                        *
*************************************************************************/
{
int blnkdspl(), sslisto(), ssbunch(), rcode;

#ifdef CHECKOUT
printf("\npschgmis ");
#endif

WHEREAMI(Screen, Window, Button, Datum);  /* Save where you were when you */
strcpy(oldscreen,Screen);                 /* called this "pseudo pop-up"  */

blnkdspl();
NEW_SCREEN("chgmis");
SLOUT("Retrieving list of missions");
rcode = sslisto();
return (rcode ? rcode : ssbunch());
}


int pmstrmis()
/*************************************************************************
*                                                                        *
*          pmstrmis  --  Set up routine to select a mission and          *
*          ========      aircraft combination for a given MTR            *
*                                                                        *
*     Routine (1) sets mission flags to false                            *
*             (2) puts up "spmstrmis"                                    *
*                                                                        *
*************************************************************************/
{
missflg= 0;
strcpy(tid.arr,ac_name.arr);
tid.len = ac_name.len;
NEW_SCREEN("spmstrmis");
return (int) 0;
}


int entermis()
/*************************************************************************
*                                                                        *
*          entermis  --  Enter a mission for this assessment             *
*          ========                                                      *
*                                                                        *
*************************************************************************/
{
int rcode;
void expOmsg();
char *clock();

#ifdef CHECKOUT
   printf("\nentermis ");
#endif

if (ac_in_form <= 0) {
   SLOUTRP("? Sortie size");
   UPDATE_DATUM("numac");}

misstype = toupper(misstype);
```

```
misdesc.len = strlen(misdesc.arr);

EXEC SQL INSERT INTO missions (mission,   type,      descr,    sortie_size)
                        VALUES (:misslabl, :misstype, :misdesc, :ac_in_form);
#ifdef CHECKOUT
   printf(" %ld", sqlca.sqlcode);
#endif

if (rcode = (int) sqlca.sqlcode) {
   expOmsg();
   fprintf(dante,"\n%s entermis: %s", clock(), sqlca.sqlerrm.sqlerrmc);
   EXEC SQL ROLLBACK WORK;
   missflg = 0;}
else {
   EXEC SQL COMMIT WORK;
   missflg = 1;}
NEW_SCREEN(oldscreen);
NEWVALS();
return rcode;
}



int psmtrflt()
/*****************************************************************************
*                                                                           *
*          psmtrflt  --  Set up routine to process a mission and            *
*          ========      aircraft combination for a given MTR               *
*                                                                           *
*****************************************************************************/
{
int cancmis(); solisto(), solistf(), solistc(), vfyacmtr();
void expOmsg();
char *clock();

if (vfyacmtr(ac_name.arr))          /* Do we know how to calculate this? */
   return (int) sqlca.sqlcode;

if (!missflg) {                    /* Has a mission been entered? */
   SLOUTBP("Please Select Mission First");
   return (int) SQL_EOF;}

#ifdef CHECKOUT
   printf("\nPrevious? %s %s %s", srcid.arr, misslabl.arr, ac_name.arr);
#endif

EXEC SQL SELECT activity FROM activities INTO :activity
                WHERE S_LABEL  = :srcid
                  AND M_IDENT  = :misslabl
                  AND AIRCRAFT = :ac_name;
#ifdef CHECKOUT
   printf(" = %ld (code = %ld)", activity, sqlca.sqlcode);
   SLOUTP("ack");
#endif

if (sqlca.sqlcode == SQL_EOF) {   /* This is a unique combination */

   EXEC SQL SELECT MAX(activity) FROM activities INTO :activity;
   #ifdef CHECKOUT
      printf("\nLast activity = %ld code = %ld", activity, sqlca.sqlcode);
   #endif
   if (sqlca.sqlcode)
      if (sqlca.sqlcode == SQL_EOF) activity = 0;
      else {
         expOmsg();
         fprintf(dante,"\n%s PSmtrflt2: %s", clock(), sqlca.sqlerrm.sqlerrmc);
         return (int) sqlca.sqlcode;}
   activity++;

   EXEC SQL INSERT INTO activities (S_LABEL, M_IDENT, AIRCRAFT, ACTIVITY)
                   VALUES(:srcid, :misslabl, :ac_name, :activity);
   #ifdef CHECKOUT
      printf(" Insert = %ld", sqlca.sqlcode);
      SLOUTP("ack");
   #endif
   if (sqlca.sqlcode) {
      expOmsg();
      fprintf(dante,"\n%s PSmtrflt3: %s", clock(), sqlca.sqlerrm.sqlerrmc);
      return (int) sqlca.sqlcode;}
   curnavpt.arr[curnavpt.len = 0] = '\0';
   prenavpt.arr[prenavpt.len = 0] = '\0';
```

```
    lptr1 = &curlowalt.altitude;
    optr1 = curlowalt.units;
    NEWVALS();
    NEW_SCREEN("mtrflt");
    }
 else {
    if (sqlca.sqlcode) {
       expOmsg();
       fprintf(dante,"\n%s PSmtrflt1: %s", clock(), sqlca.sqlerrm.sqlerrmc);
       cancmis();
       return (int) sqlca.sqlcode;}
    solistc();
    if (sqlca.sqlcode) {
        expOmsg();
        sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
        fprintf(dante, "\n%s OPEN %s", clock(), workspace.arr);
        fprintf(dante, "\n\t\t\t  Retrieving Operations for activity %ld",
                       activity);
        cancmis();
        return (int) sqlca.sqlcode;}
    for (;;) {
       solistf();
       if (sqlca.sqlcode) {
          if (sqlca.sqlcode != SQL_EOF) {
             expOmsg();
             sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
             fprintf(dante, "\n%s SELECT %s", clock(), workspace.arr);
             fprintf(dante, "\n\t\t\t  Retrieving Month %d Operations for %ld",
                            active_mo, activity);
             cancmis();
             }
          break;
          }
       ops[active_mo-1].day  = ops_day;
       ops[active_mo-1].nite = ops_nite;
       }
    if (sqlca.sqlcode == SQL_EOF) {
       solistc();
       EXEC SQL DELETE FROM operations WHERE activity = :activity;
       #ifdef CHECKOUT
          printf("\nPrevious data deleted %ld", sqlca.sqlcode);
          SLOUTP("ack");
       #endif
       }
    else solistc();
    ADD_WINDOW("month", 7, 1);
    SLOUTP("You already have data!");
    }
 return (int) 0;
 }



 int psnwmisn(name)
 /*****************************************************************************
 *                                                                           *
 *          psnwmisn  --  Set up routine for entering a new mission          *
 *          ========                                                         *
 *                                                                           *
 *     Routine (1) verifies that the new name is unique.                     *
 *             (2) puts up the mission definition screen and starts it       *
 *                                                                           *
 *****************************************************************************/

 char name[];
 {
 char *clock();
 int rcode, mslistc(), msbunch();
 void expOmsg();

 #ifdef CHECKOUT
 printf("\npsnwmisn ");
 #endif

 REMOVE_WINDOW();
 SLOUT("Processing New Mission Request");

 if (( rcode = vfyXmis(name)) != SQL_EOF) {  /* Check for strange things */
    if ( rcode == 0) {
       sprintf(workspace.arr,"Sorry, but %s already exists", name);
       SLOUTBP(workspace.arr);
```

```
      return (-1);}
   expOmsg();
   fprintf(dante,"\n%s PSnewmis: %s", clock(), sqlca.sqlerrm.sqlerrmc);
   return rcode;}
else {
   sslistc();                          /* Close list of choices    */
   strcpy (misslabl.arr, n2bv.arr);    /* This is the new mission  */
   misslabl.len = n2bv.len;
   misdesc.len = 0;
   misdesc.arr[0] = '\0';
   NEW_SCREEN("misspec");
   NEWVALS();
/*   NEXT_DATUM("misdesc");   Removed for possible conflict with U Bug */
   return (int) 0;    }
}



int nxtmispt()
/*****************************************************************************
*                                                                           *
*     nxtmispt  --  Store an MTR Navigation Point actually flown on a       *
*     ========      particular activity and advance all pointers.           *
*                                                                           *
*****************************************************************************/
{
register int i;
int insmtrpt();

#ifdef CHECKOUT
printf("\nnxtmispt ");
#endif

if (!insmispt()){

     /* Advance the Nav Point Parameters */

   for (i = 0; i <  3; i++) prenavpt.arr[i]    = curnavpt.arr[i];
   for (i = 0; i < 10; i++) prelowalt.spec[i]  = curlowalt.spec[i];
   ac_pre_spd = ac_cur_spd;
   ac_pre_pwr = ac_cur_pwr;

   curnavpt.arr[0]    = '\0';
   NEWVALS();
   NEXT_DATUM("curnavpt1");
   }
return (int) sqlca.sqlcode;

}



int insmispt()
/*****************************************************************************
*                                                                           *
*     insmispt  --  Insert an MTR Navigation Point actually traversed       *
*     ========      during a particular mission in the database             *
*                                                                           *
*****************************************************************************/
{
register int i;
void expOmsg();

#ifdef CHECKOUT
printf("\ninsmispt ");
#endif

strcpy(n2bv.arr, srcid.arr);
strcat(n2bv.arr, curnavpt.arr);
n2bv.len = strlen(n2bv.arr);
fltseq += 1;

EXEC SQL INSERT INTO MTR_FLIGHT_PARAM
                        (ACTIVITY,     FIX_LABEL,    ALT_REF,     ALT,
                         PWR,          SPD,          SEQ)
               VALUES(   :activity,    :n2bv,        :cptr1,      :lptr1,
                         :ac_cur_pwr,  :ac_cur_spd,  :fltseq);
#ifdef CHECKOUT
  printf(" Insert = %ld", sqlca.sqlcode);
```

```
#endif
if (sqlca.sqlcode) {
   if (sqlca.sqlcode == EXISTS) SLOUTER("Duplicate Entry");
   else expOmsg();
   cancmis();}  /* This is necessary since ORACLE automatically rolls back */
                /* the inserts already done at this point ! */

return (int) sqlca.sqlcode;

}

int psaddops(count)
/*************************************************************************
*                                                                        *
*     psaddops  --  Put up the add operations window and accept data.    *
*     ========                                                           *
*                                                                        *
*************************************************************************/

int count;
{
if (count == 1) ADD_WINDOW("daynite", 6, 1);
else if (count == 12) ADD_WINDOW("month", 6, 1);
else SLOUTP("PSADDOPS called with bad argument");
return (int) 0;
}



int expmops()
/*************************************************************************
*                                                                        *
*     expmops  --  Expand steady yearly operations over all months.      *
*     =======                                                            *
*                                                                        *
*************************************************************************/
{
register int i;

for (i = 1; i < 12; i++) {
   ops[i].day  = ops[0].day;
   ops[i].nite = ops[0].nite;}
return (int) 0;
}



int vfyMmis(name)
/*************************************************************************
*                                                                        *
*          vfyMmis   --   Verify the existence of a mission as part      *
*          =======         of an assessment                              *
*                                                                        *
*     Routine looks in the assessment's MISSIONS table, loads mission    *
*     into :cid and returns sqlca.sqlcode for the query.                 *
*                                                                        *
*************************************************************************/

char name[];
{
register int i;
#ifdef CHECKOUT
printf("\nvfyMmis ");
#endif

n2bv.len = strlen(name);
for ( i=0; i < n2bv.len; i++)
    n2bv.arr[i] = toupper(name[i]);
n2bv.arr[i] = '\0';

EXEC SQL SELECT mission
         FROM   missions
         INTO   :cid
         WHERE  mission = :n2bv;

cid.arr[cid.len] = '\0';
#ifdef CHECKOUT
   printf(" %ld ", sqlca.sqlcode);
#endif
```

```
return (int) sqlca.sqlcode;
}


int cancmis()
/*************************************************************************
*                                                                        *
*       cancmis  --  Cancel the mission currently pending on the database *
*       ========                                                         *
*                                                                        *
*     Routine always issues a ROLLBACK                                   *
*                                                                        *
*************************************************************************/
{
int pemtrent();

#ifdef CHECKOUT
printf("\ncancmis ");
#endif

EXEC SQL ROLLBACK WORK;
#ifdef CHECKOUT
   printf("%ld", sqlca.sqlcode);
#endif
sprintf(workspace.arr, "Entry for mission %s CANCELLED", misslabl.arr);
SLOUTEP(workspace.arr);
misslabl.arr[misslabl.len = 0] = '\0';
pemtrent();
return (int) sqlca.sqlcode;
}


int savemis()
/*************************************************************************
*                                                                        *
*       savemis  --  Commit the mission currently pending on the database *
*       ========                                                         *
*                                                                        *
*************************************************************************/
{
int mtrtabl(), pemtrent();

#ifdef CHECKOUT
printf("\nsavemis ");
#endif

for (active_mo = 1; active_mo < 13; active_mo++) {
    ops_day  = ops[active_mo-1].day;
    ops_nite = ops[active_mo-1].nite;
    EXEC SQL INSERT INTO operations ( ACTIVITY, MONTH, DAY, NIGHT, LASTUPD)
             VALUES (:activity, :active_mo, :ops_day, :ops_nite, SYSDATE);
    if (sqlca.sqlcode) {
       EHDOMSG;
       sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
       fprintf(dante, "\n%s INSERT %s", clock(), workspace.arr);
       fprintf(dante, "\n\t\t\t  Storing Operations %d %ld",
                      active_mo, activity);
       EXEC SQL ROLLBACK WORK;
       pemtrent();
       return (int) sqlca.sqlcode;}
   }

EXEC SQL COMMIT WORK;

mtr_tabl();

pemtrent();


return (int) sqlca.sqlcode;
}



int MISconn(name)
/*************************************************************************
*                                                                        *
```

```
*        MIScoan  --  Connect an MTR with a Mission                        *
*        ========                                                          *
*                                                                          *
*     Routine SELECTS the Mission from the database and loads parameter    *
*     block.  Returns an error code which indicates whether or not         *
*     the SELECT was successful.                                           *
*                                                                          *
*     Note:  The existence of the mission on the database is assumed.      *
*            SQLCA.SQLCODE is returned and if nonzero an error             *
*            message is displayed on the status line.                      *
*                                                                          *
*****************************************************************************/
char name[];
{
register int i;
int NEW_SCREEN(), NEWVALS(), SLOUT();
char *clock();

#ifdef CHECKOUT
   printf("\nMIScoan ");
#endif

SLOUT("Retrieving Mission");
n2bv.len = strlen(name);
for (i = n2bv.len; i >=0; i--) n2bv.arr[i] = name[i] = toupper(name[i]);

EXEC SQL SELECT mission, type, descr, sortie_size
         FROM   missions
         INTO   :misslabl, :misstype, :misdesc, :ac_in_form
         WHERE  mission = :n2bv;

#ifdef CHECKOUT
   printf("%ld ", sqlca.sqlcode);
#endif

if (!sqlca.sqlcode) {                    /*  Everything is O.K. */
   misslabl.arr[misslabl.len] = '\0';
   misdesc.arr[misdesc.len]   = '\0';
   sslistc();
   missflg = 1;
   NEWVALS();
   NEW_SCREEN(oldscreen);
   }
else {        /*  This should never happen!  But,.....  */
   missflg = 0;
   ENDOMSG;
   sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n%s SELECT %s", clock(), workspace.arr);
   fprintf(dante, "\n\t\t\t  Retrieving Mission %s", n2bv.arr);}
return (int) sqlca.sqlcode;
}



int sslisto()
/*****************************************************************************
*                                                                          *
*     sslisto   ---   Open cursor S1 for a list of missions that           *
*     ========        belong to the current assessment                     *
*                                                                          *
*    Routine executes an open cursor command for cursor S1 and then        *
*    returns to the calling program with the ORACLE status code.           *
*                                                                          *
*    Note:   Modifications to this function may impact the related         *
*            functions sslistf(), ssbunch() and sslistc() that fetch       *
*            rows and close the cursor and, possibly, functions that       *
*            call these utility routines.                                  *
*                                                                          *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\nsslisto ");
#endif

EXEC SQL DECLARE S1 CURSOR FOR SELECT mission
                          FROM missions ORDER BY mission;

EXEC SQL OPEN S1;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
```

```
#endif
return (int) sqlca.sqlcode;
}



int sslistf()
/*************************************************************************
*                                                                        *
*     sslistf   ---   Fetch a row using the opened cursor S1 for missions *
*     =======                                                            *
*                                                                        *
*     Routine executes an fetch command for cursor S1, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions sslisto(), ssbunch() and sslistc() that open,    *
*             fetch groupwise and close the cursor and, likely,          *
*             functions that call these utility programs                 *
*                                                                        *
*************************************************************************/
{
EXEC SQL FETCH S1 INTO :misslabl;
misslabl.arr[misslabl.len] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int ssbunch()
/*************************************************************************
*                                                                        *
*     ssbunch   ---   Fetch a bunch (20 or whatever the size of dsplmult) *
*     =======         using the opened cursor S1 for mission list        *
*                                                                        *
*     Routine executes an fetch command for cursor S1, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions sslisto(), sslistf() and sslistc() that open,    *
*             fetch and close the cursor and, likely, functions that     *
*             call these utility programs                                *
*                                                                        *
*************************************************************************/
{
register int i;
int arows, mrows, rcode;
char *clock();

EXEC SQL FETCH S1 INTO :dsplmult;
arows = (int) sqlca.sqlerrd[2];
mrows = (sizeof dsplmult) / 34;
rcode = (int) sqlca.sqlcode;

#ifdef CHECKOUT
printf(" ssBunch: %ld returns %d of %d rows", sqlca.sqlcode, arows, mrows);
#endif
if ((rcode == SQL_FETCH_OUT_OF_ORDER) || (rcode == SQL_NOF)) {
   if ((rcode == SQL_NOF) && (arows > 0)) {
      for (i = 0; i < arows; i++) dsplmult[i].arr[dsplmult[i].len] = '\0';
      if (arows < mrows)
         for (i = arows; i < mrows; i++)  dsplmult[i].arr[0] = '\0';
      NEWVALS();
      SLOUTP("The last mission in the list is on the screen");    }
   else {
      NEWVALS();
      SLOUTP("You are already as far down in the list as you can go");  }
   }
else {
   if (rcode) {
      ERDOMSG;
      fprintf(dante, "\n%s SSBUNCH: %s", clock(), sqlca.sqlerrm.sqlerrmc);}
   NEWVALS();}
return rcode;
}
```

```
int sslistc()
/*************************************************************************
*                                                                        *
*          sslistc   ---   Close cursor S1 for mission list              *
*          =======                                                       *
*                                                                        *
*     Routine executes a close cursor command for cursor S1 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions sslistc(), sabunch, and sslistf() that open the  *
*             cursor and fetch rows using it and, possibly, functions    *
*             that call these utilities                                  *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nsslistc ");
#endif

EXEC SQL CLOSE S1;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int solistc()
/*************************************************************************
*                                                                        *
*      solistc   ---   Open cursor S2 for a list of operations that      *
*      =======         belong to the current activity                    *
*                                                                        *
*     Routine executes an open cursor command for cursor S2 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions solistf() and solistc() that fetch rows and      *
*             close the cursor and, possibly, functions that call        *
*             these utility routines.                                    *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nsolistc ");
#endif

EXEC SQL DECLARE S2 CURSOR FOR SELECT month, day, night, lastupd
             FROM operations WHERE activity = :activity ORDER BY month;

EXEC SQL OPEN S2;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int solistf()
/*************************************************************************
*                                                                        *
*     solistf  ---  Fetch a row using the opened cursor S2 for operations *
*     =======                                                            *
*                                                                        *
*     Routine executes an fetch command for cursor S2, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions solistc() and solistc() that open and close      *
*             the cursor and, likely, functions that call these utilities *
*                                                                        *
*************************************************************************/
{
EXEC SQL FETCH S2 INTO :active_mo, :ops_day, :ops_nite, :timest2;
```

```
timest2.arr[timest2.len] = '\0';
#ifdef CHECKOUT
printf(" Solistf: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int solistc()
/*************************************************************************
*                                                                        *
*          solistc   ---   Close cursor S2 for operations list           *
*          ========                                                      *
*                                                                        *
*     Routine executes a close cursor command for cursor S2 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions solisto() and solistf() that open and fetch      *
*             rows from the cursor and, possibly, functions that call    *
*             these utilities                                            *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nsolistc ");
#endif

EXEC SQL CLOSE S2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}
/*************************************************************************
*                                                                        *
*     mtr.pc   --   Routines calculating noise exposure from MTRs.       *
*     ======                                                             *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*     isperpen  -  Return distance of point to line if perpendicular is  *
*                  inside line segment, NULL otherwise                   *
*     g_mtrsel  -  Get the 2-parameter SEL distance approximation for    *
*                  an aircraft                                           *
*     mtr_seel  -  Calculate the Single Event Exposure Level for an MTR  *
*                  segment.                                              *
*     mtr_tabl  -  Calculate the noise/distance table for an MTR segment *
*                  when the computational parameters have been loaded    *
*     MTRocam   -  Connect an assessment to an MTR                       *
*                                                                        *
*************************************************************************/

#include <process.h>                /*  Header for calls to MS-DOS    */
#include <stdio.h>
#include <math.h>

#define SQLCA_STORAGE_CLASS extern  /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;     /*  All SQL declarations are in   */
EXEC SQL INCLUDE hostvars.h;        /*  these header files            */
EXEC SQL INCLUDE fmharris.h;

VARCHAR rowid[25];

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                   /*  Standard ASAN Header File     */

double delta_pwr;       /* Global: adjustment for actual power setting  */
double delta_spd;       /* Global: adjustment for actual aircraft speed */
double X0, Y0;          /* Global: coordinates of point of observation  */
double X1, Y1;          /* Global: coordinates of a segment end point   */
double X2, Y2;          /* Global: coordinates of a segment end point   */


double isperpen()
/*************************************************************************
*                                                                        *
*          isperpen  --  Routine to determine if the perpendicular of    *
```

```
*          ========      point (X0,Y0) to a line segment (X1,Y1)-(X2,Y2)  *
*                        falls inside the line segment.                  *
*                                                                        *
*                                                                        *
*     Routine performs a translation of the coordinate system so that    *
*     (X0,Y0) is at the origin of the new coordinate system.  It then    *
*     rotates this new coordinate system around (X0,Y0) to a second      *
*     new coordinate system in which the X-axis is parallel to the line  *
*     segment.  If the perpendicular to the line segment is inside the   *
*     segment, then in this coordinate system the X-coordinates of the   *
*     end points of the segment will have different signs.  In this      *
*     system, the absolute value of the Y-coordinate of either point is  *
*     the distance from the point to the segment.                        *
*                                                                        *
*     Notes:   1.  If the Y'' value (in the second coordinate system)    *
*                  is positive, a person moving from (X1,Y1) to (X2,Y2)  *
*                  will observe (X0,Y0) as being to his/her right.       *
*                                                                        *
*              2.  The original coordinates are (X1,Y1), etc.            *
*                  The translated coordinates are (X1p,Y1p), etc.        *
*                  The translated and rotated coordinates are (X1dp,Y1dp) *
*                  etc. to mimic the usual notational convention of      *
*                  primed and double-primed coordinate pairs.            *
*                                                                        *
**************************************************************************/


{

double Xp0,  X1p,  X2p,  Yp0,  Y1p,  Y2p;
double Xdp0, X1dp, X2dp, Ydp0, Y1dp, Y2dp;

double sqrt();
double hypothenuse, sinetheta, cosinetheta;

#ifdef CHECKOUT
printf("\nisparpen ");
#endif

hypothenuse = sqrt( (X2 - X1)*(X2 - X1) + (Y2 - Y1)*(Y2 - Y1) );
sinetheta   = (X2 - X1) / hypothenuse;
cosinetheta = (Y2 - Y1) / hypothenuse;

X1p = X1 - X0;
Y1p = Y1 - Y0;
X2p = X2 - X0;
Y2p = Y2 - Y0;

X1dp =  X1p * cosinetheta + Y1p * sinetheta;
Y1dp = -X1p * sinetheta   + Y1p * cosinetheta;
X2dp =  X2p * cosinetheta + Y2p * sinetheta;
Y2dp = -X2p * sinetheta   + Y2p * cosinetheta;

#ifdef CHECKOUT
printf("\nSin = %7.5f  Cos = %7.5f  Hyp = %6.2f",
       sinetheta, cosinetheta, hypothenuse);
printf("\nX0  = (%5.1f,%5.1f)", X0, Y0);
printf("\nX   = (%5.1f,%5.1f)  Y   = (%5.1f,%5.1f)", X1, Y1, X2, Y2);
printf("\nX ' = (%5.1f,%5.1f)  Y'  = (%5.1f,%5.1f)", X1p, Y1p, X2p, Y2p);
printf("\nX'' = (%5.1f,%5.1f)  Y'' = (%5.1f,%5.1f)", X1dp, Y1dp, X2dp, Y2dp);
#endif
return ( ( (X1dp > 0) && (X2dp > 0) )  ||
         ( (X1dp < 0) && (X2dp < 0) )
         ? NULL
         : fabs(Y2dp) );
}


int g_mtrsel()
/**************************************************************************
*                                                                        *
*       g_mtrsel  --  Routine to get the MTR two-parameter approximation *
*       ==========      to the SKL - Distance curve from the database    *
*                                                                        *
*     Routine selects the proper profile from the database and loads     *
*     the host variables.  It returns the sqlca.sqlcode of the select.   *
*                                                                        *
*       Note:  An error will result if the entries in the database are   *
*              not unique!  At this time no provision is made for a      *
*              pick-and-choose approach to SKL profiles.                 *
*                                                                        *
```

```
*************************************************************************/
{
int temp;
#ifdef CHECKOUT
printf("\ng_mtr_sel ");
#endif
EXEC SQL SELECT  power_units, power,       power_scale,
                 slope,       intercept,    speed
         INTO    :pr_pwr_u,   :pr_power,    :pwr_scale,
                 :pwr_slope,  :pwr_intcpt,  :pr_speed
         FROM    mtrseltab
         WHERE   aircraft = :ac_name;
#ifdef CHECKOUT
fprintf(prn, "\n%s %lf %s  speed = %lf, sqlcode %ld\f",
      ac_name.arr, pr_power, pr_pwr_u.arr, pr_speed, sqlca.sqlcode);
printf("%ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



double mtr_seal(lateral)
/*************************************************************************
*                                                                        *
*       mtr_seal  --  Routine to calulate noise exposure at a point that *
*       ========      is lateral to an MTR for a SINGLE specific activity *
*                     on that MTR.  (Operational adjustments not made.)  *
*                                                                        *
*************************************************************************/

double lateral;                    /* Lateral distance to track */
{
double log_angle;                  /* LOG10(angle above the horizon)   */
double onset;                      /* Rise time parameter              */
double onset_pen;                  /* Penalty assoc'd therewith        */
double saladj;                     /* SEL adjusted for power and speed */
double atan2(), log10(), pow();    /* math.h routines used             */


          /*  Calculate "Pure" Propagation-adjusted SEL */

#ifdef CHECKOUT
   printf("\nAltitude %lf    Lateral %lf", ac_alt, lateral);
   fprintf(prn, "\nAltitude %5.01f  Lateral %5.01f", ac_alt, lateral);
   fprintf(prn, " intercept %7.31f slope %7.31f delta P %7.31f   S %7.31f",
       pwr_intcpt, pwr_slope, delta_pwr, delta_spd);
#endif


log_angle =  1.758122632 + log10(atan2(ac_alt, lateral));

saladj    =  pwr_intcpt
             + pwr_slope * ( 0.5 * log10 (ac_alt*ac_alt + lateral*lateral) )
             + delta_pwr                    /* Power Adjustment              */
             + delta_spd                    /* Speed Adjustment              */
             - (log_angle < 1.477121255 ?   /* Ground adjustment if < 30 deg */
                  (6.995 - 6.606 * log_angle
                         + 1.566 * log_angle * log_angle) : 0.0);

             /*  Calculate Rise Time Penalty */

onset  = 100.0 / (1+ pow(2.7183, 10.01 - 3.62 * log10(ac_speed)
                                 + 2.48 * log10(ac_alt)
                                 + 0.15 * log10(lateral)
                                 - 0.0542 * saladj));
if (onset < 15.0)          onset_pen =  0.0;
else if (onset <= 30.0)    onset_pen = 16.6 * log10(onset/15.0);
else                       onset_pen =  5.0;

#ifdef CHECKOUT
   fprintf(prn, "saladj %10.4lf  onset %10.4lf", saladj, onset);
#endif
return (saladj + onset_pen);
}



int mtr_tabl()
```

```
/*****************************************************************************
*                                                                            *
*       mtr_tabl  --  Routine to calulate noise exposure - perpendicular     *
*       ========      distance table (MTR_EXP_TAB) for a specific activity   *
*                                                                            *
*****************************************************************************
*****************************************************************************
***    NOTE:   This routine currently assumes that operations parameters  ***
***            are fixed over an MTR.  It must be changed when that is    ***
***            no longer true.  Also INFLEX.PC must then be changed!      ***
*****************************************************************************
*****************************************************************************
*****************************************************************************/

{
char   *clock();
int    clisto(), clistf(), clistc(), i;
int    g_mtrseq();             /* Finds profile for this aircraft      */
double mtr_seel();             /* Single event calculation routine     */
void   expOmsg();

SLOUT("Calculating MTR Exposure Table");
strcpy(tid.arr,"MTR_EXP_TAB");
tid.len = 11;
sprintf(a2bv.arr,"ACT%04ld",activity);
a2bv.len = strlen(a2bv.arr);  /* Just in case activity > 9999 */

/*****************************************************************************
*                                                                            *
*    This query only computes things for the 1st segment.  If we start      *
*    allowing different operations parameters per segment, then this        *
*    must be changed.  Also see the note in INFLEX.PC!                      *
*                                                                            *
*****************************************************************************/

EXEC SQL SELECT alt,     pwr,       spd       FROM mtr_flight_parms
           INTO :ac_alt, :ac_power, :ac_speed
           WHERE activity = :activity AND seq = 1;

#ifdef CHECKOUT
   printf("\nAircraft is  %lf, %lf KTS, power = %lf sqlcode = %ld",
             ac_alt, ac_speed, ac_power, sqlca.sqlcode);
   fprintf(prn, "\nAircraft is  %lf, %lf KTS, power = %lf sqlcode = %ld",
             ac_alt, ac_speed, ac_power, sqlca.sqlcode);
#endif


clisto();   /* Open database cursor */
for (;;) {
   clistf();    /* Fetch the next instance */
   if (!sqlca.sqlcode) {
      cid.arr[cid.len] = '\0';
      if (!strcmp (cid.arr, a2bv.arr)) break;      /* Have it already!  */
      else cid.arr[0] = '\0'; cid.len = 0;}        /* No, not that one  */
   else if (sqlca.sqlcode == SQL_NOF) break;       /* Have to create it */
   else {                                          /* Trouble!          */
      expOmsg();
      fprintf(dante, "\n%s %s", clock(), sqlca.sqlerrm.sqlerrmc);
      fprintf(dante, "\n\t\t\t  Retrieving MTR_EXP_TAB %s", a2bv.arr);
      SLOUTRP("Calculation Aborted");
      closeORA();
      return (int) sqlca.sqlcode;}
   }
clistc();        /* Close database cursor */

if (!cid.len) {             /* Create a new column in the table */
   sprintf(workspace.arr,"ALTER TABLE MTR_EXP_TAB ADD (%s NUMBER)",a2bv.arr);
   workspace.len = strlen(workspace.arr);
   #ifdef CHECKOUT
      printf("\n%s", workspace.arr);
      fprintf(prn, "\n%s", workspace.arr);
   #endif
   EXEC SQL EXECUTE IMMEDIATE :workspace;
   #ifdef CHECKOUT
      printf("\nAlter Table %s  %ld", a2bv.arr, sqlca.sqlcode);
      fprintf(prn, "\nAlter Table %s  %ld", a2bv.arr, sqlca.sqlcode);
   #endif
   if (sqlca.sqlcode) {
      expOmsg();
      fprintf(dante, "\n%s %s", clock(), sqlca.sqlerrm.sqlerrmc);
      fprintf(dante, "\n\t\t\t  Creating MTR_EXP_TAB %s", a2bv.arr);
      SLOUTRP("Calculation Aborted");
```

```
      closeORA();
      return (int) sqlca.sqlcode;
      }
   sprintf(workspace.arr,
   "COMMENT ON COLUMN MTR_EXP_TAB.%s IS 'Operations on %s by %s on mission %s'",
   a2bv.arr, srcid.arr, ac_name.arr, misslabl.arr);
   workspace.len = strlen(workspace.arr);
   #ifdef CHECKOUT
      printf("\n%s", workspace.arr);
   #endif
   EXEC SQL EXECUTE IMMEDIATE :workspace;
   }
g_mtrseal();
delta_pwr  = (ac_power - pr_power) * pwr_scale;
delta_spd  = 10.0 * log10(pr_speed / ac_speed);
sprintf(workspace.arr,
    "SELECT rowid, sideline, %s FROM mtr_exp_tab FOR UPDATE OF %s",
    a2bv.arr, a2bv.arr);
workspace.len = strlen(workspace.arr);
#ifdef CHECKOUT
   printf("\nPREPARE %s", workspace.arr);
#endif

EXEC SQL PREPARE D1 FROM :workspace;
#ifdef CHECKOUT
   printf("\n returns %ld", sqlca.sqlcode);
   fprintf(prn, "\n%s returns %ld", workspace.arr, sqlca.sqlcode);
#endif
EXEC SQL DECLARE C1 CURSOR FOR D1;
EXEC SQL OPEN C1;
#ifdef CHECKOUT
   printf("\nopen returns %ld", sqlca.sqlcode);
   fprintf(prn,"\nopen returns %ld", sqlca.sqlcode);
#endif


for (;;) {
   EXEC SQL FETCH C1 INTO :rowid, :sideline, :exposure;
   #ifdef CHECKOUT
      printf("\nFetch sideline %lf returns %ld", sideline, sqlca.sqlcode);
      fprintf(prn,
      "\nFetch sideline %lf returns %ld", sideline, sqlca.sqlcode);
   #endif
   if ((sqlca.sqlcode) && (sqlca.sqlcode != NULL_FETCHED)) break;
   exposure = mtr_seal(sideline);
   if (rowid.len != 18) {
      fprintf(prn, "\nrowid length = %d", rowid.len);
      rowid.len = 18;
      }
   sprintf(workspace.arr,
         "UPDATE MTR_EXP_TAB SET %s = %lf WHERE rowid = '%s'",
         a2bv.arr, exposure, rowid.arr);
   workspace.len = strlen(workspace.arr);
   EXEC SQL EXECUTE IMMEDIATE :workspace;
   #ifdef CHECKOUT
      printf("\n%s RETURNS %ld", workspace.arr, sqlca.sqlcode);
      fprintf(prn, "\nEXE IMM %s RETURNS %ld", workspace.arr, sqlca.sqlcode);
   #endif
   }
if (sqlca.sqlcode == SQL_EOF) {   /*  We finsished the list */
   EXEC SQL COMMIT WORK;
   }
else {
   expOmsg();
   fprintf(dante, "\n%s  %s", clock(), sqlca.sqlerrm.sqlerrmc);
   fprintf(dante, "\n\t\t\t  Fetching MTR_EXP_TAB for activity %ld  %lf",
            activity, sideline);
   EXEC SQL ROLLBACK WORK;
   }
EXEC SQL CLOSE C1;
}




int MTRconn(name)
/*****************************************************************************
*                                                                            *
*         MTRconn  --  Connect an ASAN assessment with an MTR                *
*         ========                                                           *
*                                                                            *
```

```
*       Routine SELECTS the MTR from the database and loads parameter        *
*       block.  Returns an error code which indicates whether or not         *
*       the SELECT was successful.                                           *
*                                                                            *
*       Note:  The existence of the MTR on the database is assumed.  If      *
*              any SQL error is found, the routine returns sqlca.sqlcode     *
*              with an error on the status line.  When the MTR exists        *
*              the MTR Data Entry Screen will be displayed.  To verify       *
*              the existence of an MTR use vfyMtr().                         *
*                                                                            *
******************************************************************************/
char name[];
{
register int i;
int NEW_SCREEN(), NEWVALS(), SLOUT(), SLOUTRP();
char *clock();

#ifdef CHECKOUT
   printf("\nMTRcomm ");
#endif

SLOUT("Switching MTRs ....");
n2bv.len = strlen(name);
for (i = n2bv.len; i >=0; i--) n2bv.arr[i] = name[i] = toupper(name[i]);
EXEC SQL SELECT label, status, type, descr, orig, sched, owner,
                TO_CHAR(date_pub, 'dd-Mon-YYYY'),
                TO_CHAR(timestamp,'dd-Mon-yy HH24:MI:SS')
         FROM   sources
         INTO   :srcid, :srcstat, :srctype, :srcdesc, :srcorig, :srcsched,
                :srcowner, :srcpdate, :srcmdate
         WHERE  label = :n2bv AND type = '1';

#ifdef CHECKOUT
   printf("%ld ", sqlca.sqlcode);
#endif

if (!sqlca.sqlcode) {                        /*  Everything is O.K. */
   srcid.arr[srcid.len]       = '\0';
   srcdesc.arr[srcdesc.len]   = '\0';
   srcorig.arr[srcorig.len]   = '\0';
   srcsched.arr[srcsched.len] = '\0';
   strcpy(lastmtr.arr, srcid.arr);
   lastmtr.len = srcid.len;
   mslists();
   mkmtrtxt();
   NEW_SCREEN("mtrentry");
   NEWVALS();
   if ((srcstat != 'A')) {
      sprintf(workspace.arr, "Note: Status of this MTR is \"%c\"", srcstat);
      SLOUTRP(workspace.arr);}
   }
else {        /*  This should never happen!  But,.....  */
   ERROMSG;
   sprintf(workspace.arr, "FAILED - %s", sqlca.sqlerrm.sqlerrmc);
   if (sqlca.sqlcode != SQL_EOF)    SLOUTRP(workspace.arr);
   else  SLOUTRP("This MTR exists but is not available to you");
   fprintf(dante, "\n%s SELECT %s", clock(), workspace.arr);
   fprintf(dante, "\n\t\t\t  Retrieving MTR %s", n2bv.arr);}
return (int) sqlca.sqlcode;
}
/*****************************************************************************
*                                                                            *
*     screensa.pc  --  Routines to set up the screens for assessments.       *
*     ============                                                           *
*                                                                            *
*     This file contains:                                                    *
*                                                                            *
*     loggedon    - Answers the question: "Did the planner log on?"          *
*     psdbhsek    - Puts up database housekeeping screen                     *
*     pschgass    - Puts up change assessment screen                         *
*     psnwasen    - Puts up new assessment screen                            *
*     psprobst    - Puts up problem status screen                            *
*     blnkdspl    - Blanks out the common display area                       *
*                                                                            *
******************************************************************************/

#include <process.h>                  /*  Header for calls to MS-DOS     */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern    /*  Switch for header files        */
EXEC SQL INCLUDE SQLCA;
```

```
EXEC SQL BEGIN DECLARE SECTION;          /*  All SQL declarations are in   */
EXEC SQL INCLUDE hostvars.h;             /*  these header files            */
EXEC SQL INCLUDE fmharris.h;
EXEC SQL END DECLARE SECTION;
#include "asam.h"                        /*  Standard ASAM Header File     */


int loggedon()
/*************************************************************************
*                                                                        *
*     loggedon    --   Routine to determine if the planner filled in the  *
*                      username on the first screen.                     *
*                                                                        *
*************************************************************************/
{
#define CHAR_ZERO '0'

char *clock();
int i, j, closeORA();

#ifdef CHECKOUT
printf("\nloggedon ");
#endif

WHERERAMI(Screen, Window, Datum, Button);

#ifdef CHECKOUT
printf("called from %s", Screen);
#endif

planrnam.len = strlen(planrnam.arr);        /* Who is this person?   */
for (i = 0; i < planrnam.len; i++) {
    if (planrnam.arr[i] > CHAR_ZERO) break;
    planrnam.len--;
    for (j = 0; j < planrnam.len; j++) planrnam.arr[j] = planrnam.arr[j+1];
}

#ifdef CHECKOUT
printf(" Username = %s is %d characters", planrnam.arr, planrnam.len);
#endif
if ( planrnam.len == 0 ) {  /* Did not enter a name!  (or we lost it) */
   if (strcmp(Screen,"firstscreen")) {
      SLOUTP("TROUBLE!  ASAM has forgotten your name.....");
      fprintf(dante, "\n%s Loggedon: ASAM \"lost\" planner's name", clock());
      fprintf(dante, "\n\t\t\t  ASSESSMENT.name was %s", ASSESSMENT.name);
      fprintf(dante, "\n\t\t\t  n2bv.arr was %s", n2bv.arr);
      fprintf(dante, "\n\t\t\t  Screen %s", Screen);
      fprintf(dante, "\n\t\t\t  Window %s", Window);
      fprintf(dante, "\n\t\t\t  Datum  %s", Datum);
      fprintf(dante, "\n\t\t\t  Button %s", Button);
      NEW_SCREEN("firstscreen");
/*
      closeORA();
      exit(255);
*/
      return (int) 255;
      }
   else {
      SLOUTP("I don't know yet who you are. Please enter your name");
      UPDATE_DATUM("planrnam");
      VCAPITAL(&planrnam);
      ADD_WINDOW("password", 15, 3);
      UPDATE_DATUM("password");
      VCAPITAL(&password);
      REMOVE_WINDOW();
      }
   return (int) SQL_NOF;
   }
return (int) 0;
}



int psdbhsek()
/*************************************************************************
*                                                                        *
*          psdbhsek  --  Set up routine for housekeeping screen          *
*                                                                        *
*                                                                        *
*     Routine (1) makes sure that user is known                          *
```

```
*             (2) makes an entry in the audit file                          *
*             (3) puts up "dbhsekpgscreen"                                  *
*                                                                           *
*****************************************************************************/
int closeORA(), loggedon();
char *clock();


loggedon();
fprintf(dante,
      "\n\n\n%s  PSEDREEK: %s signed on as SUPERUSER for File Maintenance\n\n",
                    clock(), planrnm.arr);
if (strcmp(ASSESSMENT.name, "SUPERUSER") ) {
   if (SUcon()) {
      fprintf(dante, "%s  PSEDREEK: SUPERUSER connect failed\n\n", clock());
      fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
      SLOUTER("SUPERUSER access denied.  Not good.....");
      closeORA();
      exit(256);
      }
   }
NEW_SCREEN("dbhsekpgscreen");
return 0;
}


pschgass()
/*****************************************************************************
*                                                                           *
*          pschgass  --  Set up routine for change assessment screen        *
*          ========                                                         *
*                                                                           *
*     Routine (1) opens cursor for fetch of ORACLE usernames                *
*             (2) fetches the first batch into memory                       *
*             (3) puts up "chgcurasscreen"                                  *
*                                                                           *
*****************************************************************************/
{
int blnkdspl(), ulisto(), ubunch(), rcode;

#ifdef CHECKOUT
printf("\npschgass ");
#endif

blnkdspl();
NEW_SCREEN("chgcurasscreen");
SLOUT("Retrieving ASAN's table of contents");
rcode = ulisto();
if (!rcode) rcode = ubunch();
return rcode;
}



int pcnwasen(name)
/*****************************************************************************
*                                                                           *
*          pcnwasen  --  Set up routine for new assessment screen           *
*          ========                                                         *
*                                                                           *
*     Routine (1) verifies that the new name is unique.                     *
*             (2) creates the new ORACLE user and tables for this           *
*                 assessment.                                               *
*                                                                           *
*     Notes:   1. When routine determines that it is stuck, closeORA        *
*                 is called to terminate the execution.                     *
*              2. Execution is terminated when this routine is called       *
*                 without being connected to ORACLE.  (One should not       *
*                 be able to get oneself into this predicament.)            *
*                                                                           *
*****************************************************************************/

char name[];
{
char *clock();
int rcode, ASANconn(), intlse(), arollORA(), SUcon();
int ulisto(), ubunch(), ulistc();
void expOmsg();

#ifdef CHECKOUT
```

```
printf("\npsnewasm ");
#endif

REMOVE_WINDOW();
SLOUT("Processing New Assessment Request");

if (( rcode = vfyOid(name)) != SQL_EOF) {  /* Check for strange things */
   if ( rcode == NOT_LOGGED_ON || rcode == SQL_BAD_LOGON) {
      sprintf(workspace.arr,
         "PSnewass: You are not connected to ORACLE!  --  FATAL");
      SLOUTB(workspace.arr);
      fprintf(dante,"\n%s %s", clock(), workspace.arr);
      exit(255);}
   if ( rcode == 0) {
      sprintf(workspace.arr,"Sorry, but %s already exists", name);
      SLOUTBP(workspace.arr);
      return (-1);}
   expOmsg();
   return rcode;}
else {                    /* This is where you land when everything is O.K. */
   ulisto();                          /* Close the list of choices */
   rcode = arollORA(name);            /* Try to get this one added */
   if ( rcode ) {
      sprintf(workspace.arr, "FAILED -- %s", sqlca.sqlerrm.sqlerrmc);
      fprintf(dante, "\n%s %s FAILED to enroll", clock(), name);
      fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
      if (SUcoon()) {
         sprintf( workspace.arr, "PSNWASEN stuck: %s!",
               sqlca.sqlerrm.sqlerrmc);
         SLOUTBP(workspace.arr);
         fprintf(dante, "\n%sPSNWASEN stuck: SUPERUSER re-connect failed",
               clock());
         fprintf(dante, "\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
         closeORA();
         exit(255);}
      if (!ulisto()) {              /* Recover by reinitializing screen */
         ubunch();
         NEWVALS(); }
      if ((rcode == SQL_BAD_LOGON) || (rcode == NO_IDENTIFIED_BY)) {
         sprintf(workspace.arr,
            "FAILED -- Are there perhaps spaces in \"%s\"?", name);}
      SLOUTBP(workspace.arr);
      }
   else  {                       /* arollORA() executed O.K.   Start */
      if (!intlze()) return 0;   /* create the ORACLE tables needed */
      #ifdef CHECKOUT
         printf("\nSomething major is wrong.  Code = %d", rcode);
      #endif
      SLOUTBP("Initialization terminated abnormally");
      SLOUTP(sqlca.sqlerrm.sqlerrmc);
      SLOUTP("This should be fixed before entering data");
      fprintf(dante,"\n%s Initialization did not complete normally", clock());
      fprintf(dante," due to\n\t\t\t  %s", sqlca.sqlerrm.sqlerrmc);
      fprintf(dante,"\n\t\t\t  This problem should be fixed before proceeding");
      fprintf(dante," with\n\t\t\t  data entry for %s", name);
      }
   }
}


int psprobst()
/*************************************************************************
*                                                                        *
*     psprobst    --   Routine to set up the ASAN Status Screen.  This  *
*     ========         displays the ASSESSMENT structure.  If SUPERUSER  *
*                      is the current user, the system loads the last    *
*                      active assessment first.                          *
*                                                                        *
*************************************************************************/
{
int closeORA(), lastsess(), loggedon(), rcode, statusflag;
char *clock();

#ifdef CHECKOUT
printf("\npsprobst ");
#endif
loggedon();
if (!strcmp(ASSESSMENT.name, "SUPERUSER") ) { /* Not CONNECTed ?        */
   statusflag = lastsess();                   /* Get last session       */
   if (statusflag == 0)  {                    /* Found it!              */
```

```
      EXEC SQL SELECT username                    /* Find the name of the  */
               FROM   sysuserlist                 /* last assessment       */
               INTO   :m2bv                       /* worked on             */
               WHERE  userid = :usernc;

      #ifdef CHECKOUT
         printf(" %ld", sqlca.sqlcode);
      #endif
      if (sqlca.sqlcode == SQL_NOF){  /* Trouble in River City.... */
         SLOUTRP("PSPROBST detected logbook violation.  ID = %d", usernc);
         fprintf(dante,"\n%s PSPROBST detected logbook violation.  ID = %d",
                 clock(), usernc);
         closeORA();
         exit(255);}
      rcode = ASANcoan(m2bv.arr);  /* Sign on as that last assessment */

      if (rcode) {
         SLOUTRP("ASANcoan error not trapped");
         ENDCMSG;
         SLOUTP(sqlca.sqlerrm.sqlerrmc);
         }
      return rcode;  /* ASANcoan will have put up the proper screen   */

      }
   else {
      if (statusflag == SQL_NOF) {
         SLOUTRP("No work ever done yet: You can only start a new one");
         blakdspl();
         NEW_SCREEN("chgcurrnscreen");
         return 0;
         }
      expCmsg();
      return statusflag;
      }
   }
else {      /* When you are already connected to ASAN as a regular user */
            /* You YOURSELF are now the last user and the time is NOW!  */

   strcpy(plnnrlst.arr, plnnrnam.arr);
   EXEC SQL SELECT TO_CHAR(SYSDATE,'dd-Mon-yy HH24:MI:SS')
                  FROM dual into :lastdate;
   NEWVALS();
   NEW_SCREEN("probstatscreen");
   return 0;
   }
}


int blakdspl()
/*************************************************************************
*                                                                        *
*     blakdspl    --   Routine to blank out the common display area.    *
*     ========         This is a safety precaution, since if the list   *
*                      to be displayed is empty, whatever was left from *
*                      the previous list would show.  This is because   *
*                      the "bunch" routines don't "remember" how often  *
*                      they have been called and so don't blank out a   *
*                      display when they run out of data!               *
*                                                                        *
*************************************************************************/
{
register int m, i;
m = (sizeof dsplmult) / 34;
for (i = 0; i < m; i++) {
   dsplmult[i].arr[0] = '\0';
   dsplmult[i].len    = 0;}
NEWVALS();
}
/*************************************************************************
*                                                                        *
*     screenmm.pc  --  Routines to set up the screens for MTRs.          *
*     ============                                                       *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*     pmntrmnt    - Puts up screen to work with MTRs                     *
*     pschgmtr    - Puts up screen to select a new MTR                   *
*     pnewmtrn    - Puts up screen to define a new mtr                   *
*     pmnisreq    - Puts up screen to define mission requirements        *
*     mkmtrtxt    - Make a textblock of an MTR's Navigation Point user data *
*     entmtrpt    - Starts the MTR entry process                         *
```

```
*     nxtmtrpt   - Store an MTR Navigation Point and advance entry one step *
*     insmtrpt   - Insert an MTR Navigation Point into the database        *
*     vfyacmtr   - Verify existence of aircraft data for MTR calculation   *
*     vfymtr     - Verify existence of a MTR on the system                 *
*     cancmtr    - Cancel the MTR currently pending on the database        *
*     savemtr    - Commit the MTR currently pending on the database        *
*                                                                          *
*     ORACLE access routines:                                              *
*                                                                          *
*       1. ...listo()   Open Cursor  (Opens a "logical file")              *
*       2. ...listf()   Fetch Cursor (Reads next record from logical file) *
*          ...bunch()   Fetch Cursor (Reads next bunch from logical file   *
*                       applicable only for multiple choice options)       *
*       4. ...listc()   Close Cursor (Closes the logical file)             *
*                                                                          *
*     WHERE:                                                               *
*                                                                          *
*     ... =  ms* -  Names of all accessible MTRs (Alphabetical order)      *
*     ... =  mt* -  Segments of a particular MTR                           *
*     ... =  ma* -  All user information for a particular MTR              *
*                                                                          *
*             * -  Asterisk indicates that multiple choice option is       *
*                  supported for this set of routines.                     *
*                                                                          *
****************************************************************************/

#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern     /*  Switch for header files       */
EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;        /*  All SQL declarations are in   */
EXEC SQL INCLUDE hostvars.h;           /*  these header files            */
EXEC SQL INCLUDE fmharris.h;

EXEC SQL END DECLARE SECTION;
#include "asan.h"                      /*  Standard ASAN Header File     */


pmntrent()
/****************************************************************************
*                                                                          *
*          pmntrent  --  Set up routine for "MTR Data Entry Screen"        *
*                                                                          *
*                                                                          *
*     Routine finds last MTR known to have been worked on that is "good"   *
*     (i.e. not one that has been abandoned during entry) makes that the   *
*     current noise source and mtrid and puts up the screen.               *
*                                                                          *
****************************************************************************/
{
char *clock();
void expOmsg();

#ifdef CHECKOUT
printf("\npmntrent ");
#endif


EXEC SQL SELECT label, type, status, descr, orig,   /* The laundry list */
               sched, owner, TO_CHAR(date_pub,'dd-Mon-yy'),
               TO_CHAR(timestamp,'dd-Mon-yy HH24:MI:SS')
         FROM  sources                /*  the list of accessible sources */
         INTO  :srcid, :srctype, :srcdesc, :srcorig,
               :srcsched, :srcowner, :srcpdate, :srcedate
         WHERE label = :lastmtr                 /*  for this name and */
               AND type = '1';                  /*  which is an MTR   */
#ifdef CHECKOUT
   printf(" %ld owner %d", sqlca.sqlcode, srcowner);
#endif

if (sqlca.sqlcode) {
   if (sqlca.sqlcode == SQL_NOF) {
      sprintf(srcdesc.arr,"There are no MTRs active for %s", ASSESSMENT.name);
      srcdesc.len = strlen(srcdesc.arr);}
   else {
      ERROMSG;
      fprintf(dante,
      "\n%s %s\n\t\t\t  looking for MTR %s in Source Library for %d",
      clock(), sqlca.sqlerrm.sqlerrmc, n2bv.arr, ASSESSMENT.id);
      expOmsg();
```

```
      return (int) sqlca.sqlcode;}
   }
else {
   srodesc.arr[srodesc.len] = '\0';
   srcedate.arr[srcedate.len] = '\0';
   srcpdate.arr[srcpdate.len] = '\0';
   srcsched.arr[srcsched.len] = '\0';
   srcorig.arr[srcorig.len] = '\0';
   srcid.arr[srcid.len] = '\0';}
NEWVALS();
NEW_SCREEN("mtrentry");

return (int) sqlca.sqlcode;
}




pschgmtr()
/*************************************************************************
*                                                                        *
*          pschgmtr  --  Set up routine for screen to change to new MTR  *
*          ========                                                      *
*                                                                        *
*     Routine (1) opens cursor for fetch of all MTR names known to system *
*             (2) fetches the first batch into memory                    *
*             (3) puts up "chgcurmtrscreen"                              *
*                                                                        *
*************************************************************************/
{
int blakdspl(), mslisto(), msbunch(), rcode;

#ifdef CHECKOUT
printf("\npschgmtr ");
#endif

blakdspl();
NEW_SCREEN("chgcurmtrscreen");
SLOUT("Retrieving ASAN's list of accessible MTRs");
rcode = mslisto();
return (rcode ? rcode : msbunch());
}




int psnewmtrn(name)
/*************************************************************************
*                                                                        *
*          psnewmtrn  --  Set up routine for new MTR entry screen        *
*          ========                                                      *
*                                                                        *
*     Routine (1) verifies that the new name is unique.  If not, the     *
*                 reason (whether it is already on your list or another  *
*                 assessment already has exclusive use of it) is shown   *
*             (2) puts up the MTR definition screen and starts it        *
*                                                                        *
*************************************************************************/

char name[];
{
char *clock();
int rcode, mslisto(), msbunch();
void expOmsg();

#ifdef CHECKOUT
printf("\npsnewmtrn ");
#endif

REMOVE_WINDOW();
SLOUT("Processing New MTR Request");

if (( rcode = vfyNmtr(name)) != SQL_EOF) {  /* Check for strange things */
   if ( rcode == 0) {
      sprintf(workspace.arr,"Sorry, but %s already exists", name);
      SLOUTRP(workspace.arr);
      return (-1);}
   if ( rcode == DUPLICATE_OBJECT) {
      EXEC SQL SELECT username
```

```
                FROM   syrusarlist
                INTO   :tid
                WHERE userid = :srcowner;
        tid.arr[tid.len] = '\0';
        sprintf(workspace.arr,
           "Sorry!  %s already \"owns\" %s", tid.arr, name);
        SLOUTBP(workspace.arr);
        return (-1);}
    ampOmsg();
    fprintf(dante,"\n%s P$newmtr: %s", clock(), sqlca.sqlerrm.sqlerrmc);
    srostat = 'U';
    return rcode;}
 else {
    mslistc();                                  /* Close list of choices    */
    strcpy (srcid.arr, a2bv.arr);               /* This is now the mtr name */
    srcid.len = a2bv.len;
    EXEC SQL SELECT TO_CHAR(SYSDATE, 'dd-Mon-yyyy')
       FROM dual INTO :srcpdate;                /* Suggest a date of Publ.  */
    srcpdate.arr[srcpdate.len] = '\0';
    #ifdef CHECKOUT
       printf("\nDATE (sqlcode = %ld) %u %s", sqlca.sqlcode,
                        srcpdate.len,  srcpdate.arr);
    #endif
    srcdesc.len = 0;
    srcdesc.arr[0] = '\0';
    srcstat = 'A';
    NEW_SCREEN("mtrdefinescreen");              /* Put up data entry screen */
    NEWVALS();
    return (int) 0;   }
 }


 int pmisreq()
 /*************************************************************************
 *                                                                        *
 *     pmisreq   --  Routine to set up the Mission Requirements Screen.   *
 *                                                                        *
 *************************************************************************/
 {
 int NEW_SCREEN();

 #ifdef CHECKOUT
 printf("\npmisreq ");
 #endif

 NEW_SCREEN("missreq");
 return 0;

 }


 int mkmtrtxt()
 /*************************************************************************
 *                                                                        *
 *     mkmtrtxt  --  Create a textblock of an MTR's Navigation Points     *
 *     ---------     user information (Identifier, Fix info, etc.)        *
 *                                                                        *
 *************************************************************************/
 {
 char *clock();
 FILE *fopen();

 int i, j, mtlistc(), mtlistf(), mtlistc(), fclose();

 #ifdef CHECKOUT
 printf("\nmkmtrtxt ");
 #endif
 if ((txtblkf = fopen("txtblk\\mtr.txt","w") ) == NULL) {
    SLOUTBP("Error opening file for help window");
    fprintf(dante, "\n%s could not open MTR textblock file for %d",
            clock(), ASSESSMENT.id);
    return (int) DUPLICATE_OBJECT;}

 fprintf(txtblkf, "   Navigation Points for %s are:\n\n", srcid.arr);
 mtlistc();
 if (!sqlca.sqlcode) {
    for (;;) {
       mtlistf();
       if (sqlca.sqlcode) break;
       for (i=srcid.len, j = 0; i < a2bv.len; i++, j++)
```

```
            prenavpt.arr[j] = n2bv.arr[i];
        prenavpt.arr[j] = '\0';
        fprintf(txtblkf, "%-4s%-5s %03d/%03d  %-12s  %02d/%02d\n",
                prenavpt.arr, prefixid.arr, prefixrad, prefixdist,
                prefixtyp.arr, prewidleft, prewidright);
        }
    if (sqlca.sqlcode != SQL_NOF) {
        ERROMSG;
        fprintf(dante,
            "\n%s %s\n\t\t\t  looking for MTR segments in MTRSEGMENTS for %d",
            clock(), sqlca.sqlerrm.sqlerrmc, n2bv.arr, ASSESSMENT.id);}
    fprintf(txtblkf,"\n      ==  END  ==\n\n\n\0");
    mtlists();
    }
fclose(txtblkf);
return (int) sqlca.sqlcode;
}



int emtmtrpt()
/*****************************************************************************
*                                                                            *
*      emtmtrpt  --  Inserts the originating and scheduling activity in      *
*      ========      the SOURCELIST and pops up the window for entering      *
*                    MTR navigation points.                                  *
*                                                                            *
*****************************************************************************/
{
char *clock();

#ifdef CHECKOUT
printf("\nemtmtrpt ");
#endif

EXEC SQL SELECT TO_CHAR(SYSDATE,'dd-Mon-yy HH24:MI:SS')
                FROM dual INTO :srcodate;            /* Now  */
#ifdef CHECKOUT
printf(" DATE = %ld ", sqlca.sqlcode);
#endif
srodesc.len  = strlen(srodesc.arr);   /* While the interface knows,    */
srcorig.len  = strlen(srcorig.arr);   /* ORACLE doesn't that know yet! */
srcsched.len = strlen(srcsched.arr);

if (strcmp(ASSESSMENT.name, "SUPERUSER")) {
   srcowner = ASSESSMENT.id;
   EXEC SQL INSERT INTO mysources (LABEL, TYPE, STATUS,
                    DESCR, ORIG, SCHED, OWNER, DATE_PUB, TIMESTAMP)
            VALUES (:srcid,          /* Name given to this MTR          */
                    '1',             /* Code identifying MTRs           */
                   :srcstat,         /* Obsolescence/Applicability      */
                   :srodesc,         /* English Description             */
                   :srcorig,         /* Originating Activity            */
                   :srcsched,        /* Scheduling Activity             */
                   :srcowner,        /* This assessment's id            */
                   TO_DATE(:srcpdate,'dd-Mon-YYYY'),  /* Date of publ.  */
                   TO_DATE(:srcodate,'dd-Mon-yy HH24:MI:SS'));}
else {
   srcowner = 0;
   EXEC SQL INSERT INTO noise_sources (LABEL, TYPE, STATUS,
                    DESC, ORIG, SCHED, OWNER, DATE_PUB, TIMESTAMP)
            VALUES (:srcid,          /* Name given to this MTR          */
                    '1',             /* Code identifying MTRs           */
                   :srcstat,         /* Obsolescence/Applicability      */
                   :srodesc,         /* English Description             */
                   :srcorig,         /* Originating Activity            */
                   :srcsched,        /* Scheduling Activity             */
                   NULL,             /* SUPERUSER doesn't mess around!  */
                   TO_DATE(:srcpdate,'dd-Mon-YYYY'),  /* Date of publ.  */
                   TO_DATE(:srcodate,'dd-Mon-yy HH24:MI:SS'));}

#ifdef CHECKOUT
printf(" Insert = %ld", sqlca.sqlcode);
#endif

if (sqlca.sqlcode) {
   sprintf(workspace.arr,"FAILED %s", sqlca.sqlerrm.sqlerrmc);
   SLOUTEP(&workspace.arr[7]);  /* Track errors other than date conversions */
   if ((sqlca.sqlcode < -1899) || (sqlca.sqlcode > -1800))
      ERROMSG;
      fprintf(dante,
```

```
        "\n%s %s\n\t\t\t  entering MTR %s into Source Library for %d",
        clock(), workspace.arr, srcid.arr, srcownar);
    srcstat = '0';        /* Leave a special code to prevent committing MTR! */}
 else {
    srcstat = '1';        /* (Still) looks like a good MTR to me! */
    NEW_SCREEN("defmodmtrscreen");
    prenavpt.arr[0]    = curnavpt.arr[1]    = '\0';  /* Initialize variables  */
    prenavpt.len       = curnavpt.len       =   0 ;
    prefixid.arr[0]    = curfixid.arr[0]    = '\0';
    prefixid.len       = curfixid.len       =   0 ;
    prefixrad          = curfixrad          =   0 ;
    prefixdist         = curfixdist         =   0 ;
    prefixtyp.arr[0]   = curfixtyp.arr[0]   = '\0';
    curnavpt.arr[0] = 'A';
    strcpy(show.lat," ..x...'..\"N");
    strcpy(show.lon,"...x...'..\"W");
    ent.lat[0]         = ent.lon[0]         = '\0';

    strcpy(prehighalt.spec,"..... MSL");
    STRCPY(prelowalt.spec, "..... AGL");

    prewidleft         = curwidleft         =   0 ;
    prewidright        = curwidright        =   0 ;
    preartcc.arr[0]    = curartcc.arr[0]    = '\0';

    NEWVALS();

    lptr1 = &curlowalt.altitude;   /*  Set pointers for ORACLE INSERTs later */
    lptr2 = &curhighalt.altitude;
    cptr1 = curlowalt.units;
    cptr2 = curhighalt.units;
    cptr3 = ent.lat;
    cptr4 = ent.lon;
    dptr1 = &ent.eastings;
    dptr2 = &ent.northings;
    dptr3 = &ent.latitude;
    dptr4 = &ent.longitude;
    }
 return (int) sqlca.sqlcode;
 }


int antmtrpt()
/*************************************************************************
*                                                                        *
*     antmtrpt  --  Store an MTR Navigation Point and advance the        *
*     ========      current NAV POINT to be the previous.                *
*                                                                        *
*************************************************************************/
{
register int i;
int insmtrpt();

#ifdef CHECKOUT
printf("\nantmtrpt ");
#endif

if (!insmtrpt()){

     /* Advance the Nav Point Parameters */

   for (i = 0; i <  3; i++) prenavpt.arr[i]    = curnavpt.arr[i];
   for (i = 0; i <  3; i++) preartcc.arr[i]    = curartcc.arr[i];
   for (i = 0; i <  5; i++) prefixid.arr[i]    = curfixid.arr[i];
   for (i = 0; i < 10; i++) prelowalt.spec[i]  = curlowalt.spec[i];
   for (i = 0; i < 10; i++) prehighalt.spec[i] = curhighalt.spec[i];
   for (i = 0; i < 12; i++) prefixtyp.arr[i]   = curfixtyp.arr[i];
   for (i = 0; i < 14; i++) show.lat[i]        = ent.lat[i];
   for (i = 0; i < 14; i++) show.lon[i]        = ent.lon[i];

   prefixrad    = curfixrad;
   prefixdist   = curfixdist;
   prewidleft   = curwidleft;
   prewidright  = curwidright;


    /* Blank out the Nav Point Parameters */

   curnavpt.arr[0]    = '\0';
   ent.lat[0]         = '\0';
   ent.lon[0]         = '\0';
```

```
    emt.latitude        =  OFF_EARTH;
    emt.longitude       =  OFF_EARTH;
    curfixrad           =   0 ;
    curfixdist          =   0 ;

    NEWVALS();
    }
return (int) sqlca.sqlcode;

}



int insmtrpt()
/*************************************************************************
*                                                                        *
*     insmtrpt  --  Insert an MTR Navigation Point into the database     *
*     ========                                                           *
*                                                                        *
*************************************************************************/
{
int cancmtr();
register int i;
void expOmsg();

#ifdef CHECKOUT
printf("\ninsmtrpt ");
#endif

if (COOR2utm(&emt)) return (int) -1;

strcpy(n2bv.arr, srcid.arr);
strcat(n2bv.arr, curnavpt.arr);
n2bv.len = strlen(n2bv.arr);

EXEC SQL INSERT INTO MTRSEGMENTS
                        (FIX_LABEL,     FLOOR_REF,  CEILING_REF, FIX_ID,
                         FIX_TYPE,      ARTCC,      FIX_LAT,     FIX_LON,
                         FIX_RAD,       FIX_DIST,   FLOOR,       CEILING,
                         WIDTH_LEFT,    WIDTH_RIGHT)
                    VALUES(:n2bv,       :optr1,     :optr2,      :curfixid,
                         :curfixtyp,    :curartcc,  :optr3,      :optr4,
                         :curfixrad,    :curfixdist, :lptr1,     :lptr2,
                         :curwidleft,   :curwidright);
#ifdef CHECKOUT
  printf(" Insert1 = %ld", sqlca.sqlcode);
#endif
if (sqlca.sqlcode) {
   if (sqlca.sqlcode == EXISTS) SLOUTEP("Duplicate Navpoint Id");
   else expOmsg();
   cancmtr();}  /* This is necessary since ORACLE automatically rolls back */
                /* the inserts already done at this point ! */
else {
   EXEC SQL INSERT INTO NAVPOINTS( FIX_LABEL, X, Y, LAT, LON )
                        VALUES( :n2bv, :dptr1, :dptr2, :dptr3, :dptr4);
   #ifdef CHECKOUT
      printf(" Insert2 = %ld", sqlca.sqlcode);
   #endif
   if (sqlca.sqlcode) {
      expOmsg();
      if (sqlca.sqlwarn[0]) expOwrn();
      cancmtr();}  /* ORACLE automatically rolls back */
                   /* inserts after this occurs !     */
   }
return (int) sqlca.sqlcode;

}



int vfyacmtr(name)
/*************************************************************************
*                                                                        *
*         vfyacmtr   --   Verify the existence of aircraft data for      *
*         ========        MTR calculations in the database               *
*                                                                        *
*     Routine looks in MTRSELTAB (a view on a HEADQUARTERS table), loads *
*     pr_pwr_u with the power units for which we have data fpr the aircraft *
*     and returns sqlca.sqlcode for the query.  Zero means MTR exists,   *
*     SQL_EOT means it does not, all other values indicate an SQL error. *
```

```
*                                                                          *
*************************************************************************/

char name[];
{
register int i;
char *clock();

#ifdef CHECKOUT
printf("\nvfyacmtr ");
#endif

n2bv.len = strlen(name);
for ( i=0; i < n2bv.len; i++)
    n2bv.arr[i] = name[i] = toupper(name[i]);
pr_pwr_u.arr[0] = '\0';
pr_pwr_u.len = 0;

EXEC SQL SELECT power_units          /*  See if aircraft power units */
         FROM   mtrseltab            /*  are in the list of aircraft */
         INTO   :pr_pwr_u            /*  for which we have MTR data  */
         WHERE  aircraft = :n2bv;

#ifdef CHECKOUT
   printf(" %ld %s power in %s", sqlca.sqlcode, n2bv.arr, pr_pwr_u.arr);
#endif

if (sqlca.sqlcode) {
   if (sqlca.sqlcode == SQL_EOF) {
      sprintf(workspace.arr, "MTR Calculation not supported for %s", n2bv.arr);
      SLOUTBP(workspace.arr);}
   else {
      ENDOMSG;
      fprintf(dante,
      "\n%s %s\n\t\t\t  looking for %s in MTRSELTAB for %d",
      clock(), sqlca.sqlerrm.sqlerrmc, n2bv.arr, ASSESSMENT.id);}
   }
else {
   pr_pwr_u.arr[pr_pwr_u.len] = '\0';
   for ( i=0; i < n2bv.len; i++)
      ac_name.arr[i] = n2bv.arr[i];
   ac_name.len = i;
   ac_name.arr[i] = '\0';
   EXEC SQL SELECT power, speed       /*  Load the reference conditions */
         FROM   mtrseltab             /*  as initialization for this    */
         INTO   :ac_cur_pwr, :ac_cur_spd          /* Aircraft/Mission   */
         WHERE  aircraft = :ac_name;
   ac_pre_pwr = ac_cur_pwr;
   ac_pre_spd = ac_cur_spd;
#ifdef CHECKOUT
   printf(" %ld %s power in %s", sqlca.sqlcode, n2bv.arr, pr_pwr_u.arr);
#endif


   NEWVALS(); }
return (int) sqlca.sqlcode;
}



int vfyXmtr(name)
/*************************************************************************
*                                                                        *
*          vfyXmtr   --   Verify the existence of a MTR on the system    *
*          ________                                                      *
*                                                                        *
*     Routine looks in SOURCELIST (a view on a SUPERUSER table), loads   *
*     srcowner and returns sqlca.sqlcode for the query.  Zero means MTR  *
*     exists, SQL_EOF means it does not, all other values indicate an SQL *
*     error except that DUPLICATE_OBJECT means that the object exists on *
*     the system but is not accessible to the present assessment         *
*                                                                        *
*************************************************************************/

char name[];
{
register int i;
#ifdef CHECKOUT
printf("\nvfyXmtr ");
#endif
```

```
a2bv.len = strlen(name);
for ( i=0; i < a2bv.len; i++)
    a2bv.arr[i] = toupper(name[i]);

EXEC SQL SELECT owner                  /*  Identify the owner from */
         FROM   sourcelist             /*  the list of all sources */
         INTO   :srcowner
         WHERE  label = :a2bv          /*  which has this name and */
                AND type = '1';        /*  is an MTR               */

#ifdef CHECKOUT
   printf(" %ld owner %d", sqlca.sqlcode, srcowner);
#endif

if (sqlca.sqlcode == NULL_FETCHED) return (int) 0;
if ((sqlca.sqlcode == 0) && (srcowner != ASSESSMENT.id))
   return (int) DUPLICATE_OBJECT;
return (int) sqlca.sqlcode;
}



int cancmtr()
/*************************************************************************
*                                                                        *
*       cancmtr  --  Cancel the MTR currently pending on the database    *
*       ========                                                         *
*                                                                        *
*     Routine always issues a ROLLBACK                                   *
*                                                                        *
*************************************************************************/
{
int pemtrest();

#ifdef CHECKOUT
printf("\ncancmtr ");
#endif

EXEC SQL ROLLBACK WORK;
#ifdef CHECKOUT
   printf("%ld", sqlca.sqlcode);
#endif
sprintf(workspace.arr, "Entry for MTR %s CANCELLED", srcid.arr);
SLOUTB(workspace.arr);
srcid.arr[srcid.len = 0] = '\0';
pemtrest();
return (int) sqlca.sqlcode;
}



int savemtr()
/*************************************************************************
*                                                                        *
*       savemtr  --  Commit the MTR currently pending on the database    *
*       ========                                                         *
*                                                                        *
*     Routine checks if there is a valid MTR being constructed then     *
*     issues a COMMIT to the database                                    *
*                                                                        *
*************************************************************************/
{
register int i;

#ifdef CHECKOUT
printf("\nsavemtr ");
#endif

mtlistc();
if (srcstat == '0') {
   SLOUTB("MTR had errors and was not saved");
   EXEC SQL ROLLBACK WORK;
   #ifdef CHECKOUT
      printf("ROLLED MTR BACK!..... %ld", sqlca.sqlcode);
   #endif
   srcid.arr[srcid.len = 0] = '\0';
   }
else {   /* First check if we have to enter one more point! */

   if (curnavpt.arr[0] != '\0')
```

```
      if (insmtrpt()) return (int) sqlca.sqlcode;

   EXEC SQL COMMIT WORK;
   #ifdef CHECKOUT
      printf("Committed MTR %ld", sqlca.sqlcode);
   #endif
   }
pmstrent();
return (int) sqlca.sqlcode;
}


int mslisto()
/*************************************************************************
*                                                                        *
*     mslisto   ---   Open cursor S1 for a list of MTRs on the system    *
*     ========        that are accessible to an assessment               *
*                                                                        *
*    Routine executes an open cursor command for cursor S1 and then      *
*    returns to the calling program with the ORACLE status code.         *
*                                                                        *
*    Note:   Modifications to this function may impact the related       *
*            functions mslistf(), msbunch() and mslistc() that fetch     *
*            rows and close the cursor and, possibly, functions that     *
*            call these utility routines.                                *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nmslisto ");
#endif

EXEC SQL DECLARE S1 CURSOR FOR
        SELECT label                /* These are unique identifiers     */
        FROM   sources              /* View of all accessible "sources" */
        WHERE  type = '1'           /* MTRs are type == 1               */
        ORDER BY label;

EXEC SQL OPEN S1;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int mslistf()
/*************************************************************************
*                                                                        *
*    mslistf   ---   Fetch a row using the opened cursor S1 for MTRs     *
*    =======                                                             *
*                                                                        *
*    Routine executes an fetch command for cursor S1, which is assumed   *
*    to have been opened, and then returns to the calling program with   *
*    the ORACLE status code.                                             *
*                                                                        *
*    Note:   Modifications to this function may impact the related       *
*            functions mslisto(), msbunch() and mslistc() that open,     *
*            fetch groupwise and close the cursor and, likely,           *
*            functions that call these utility programs                  *
*                                                                        *
*************************************************************************/
{
EXEC SQL FETCH S1 INTO :srcid;
srcid.arr[srcid.len] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int msbunch()
/*************************************************************************
*                                                                        *
*    msbunch   ---   Fetch a bunch (20 or whatever the size of dsplmult) *
*    =======         using the opened cursor S1 for mtrlist             *
```

```
*                                                                          *
*     Routine executes an fetch command for cursor S1, which is assumed    *
*     to have been opened, and then returns to the calling program with    *
*     the ORACLE status code.                                              *
*                                                                          *
*     Note:   Modifications to this function may impact the related        *
*             functions mslisto(), mslistf() and mslistc() that open,      *
*             fetch and close the cursor and, likely, functions that      *
*             call these utility programs                                  *
*                                                                          *
*****************************************************************************/
{
register int i;
int nrows, mrows, rcode;
char *clock();

EXEC SQL FETCH S1 INTO :dsplmult;
nrows = (int) sqlca.sqlerrd[2];
mrows = (sizeof dsplmult) / 34;
rcode = (int) sqlca.sqlcode;

#ifdef CHECKOUT
printf(" msBunch: %ld returns %d of %d rows", sqlca.sqlcode, nrows, mrows);
#endif
if ((rcode == SQL_FETCH_OUT_OF_ORDER) || (rcode == SQL_EOF)) {
   if ((rcode == SQL_EOF) && (nrows > 0)) {
      for (i = 0; i < nrows; i++) dsplmult[i].arr[dsplmult[i].len] = '\0';
      if (nrows < mrows)
         for (i = nrows; i < mrows; i++)  dsplmult[i].arr[0] = '\0';
      NEWVALS();
      SLOUTP("The last MTR in the list is on the screen");    }
   else {
      NEWVALS();
      SLOUTP("You are already as far down in the list as you can go");  }
   }
else {
   if (rcode) {
      ENDOMSG;
      fprintf(dante, "\n%s MSBUNCH: %s", clock(), sqlca.sqlerrm.sqlerrmc);}
   NEWVALS();}
return rcode;
}


int mslistc()
/*****************************************************************************
*                                                                          *
*          mslistc   ---   Close cursor S1 for mtrlist                     *
*          ========                                                        *
*                                                                          *
*     Routine executes a close cursor command for cursor S1 and then       *
*     returns to the calling program with the ORACLE status code.          *
*                                                                          *
*     Note:   Modifications to this function may impact the related        *
*             functions mslisto(), msbunch, and mslistf() that open the    *
*             cursor and fetch rows using it and, possibly, functions      *
*             that call these utilities                                    *
*                                                                          *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\mmslistc ");
#endif

EXEC SQL CLOSE S1;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int mtlisto()
/*****************************************************************************
*                                                                          *
*     mtlisto   ---   Open cursor S2 for a list of user information        *
*     ========        of navigation points on a particular MTR             *
*                                                                          *
*     Routine executes an open cursor command for cursor S2 and then       *
```

```
 *     returns to the calling program with the ORACLE status code.          *
 *                                                                          *
 *     Note:   Modifications to this function may impact the related        *
 *             functions mtlistf() and mtlistc() that fetch rows and        *
 *             close the cursor and, possibly, functions that call          *
 *             these utility routines.                                      *
 *                                                                          *
 ****************************************************************************/
{
#ifdef CHECKOUT
printf("\nmtlisto ");
#endif

strcpy( cid.arr, srcid.arr);
strcat(cid.arr,"%");
cid.len = strlen(cid.arr);

EXEC SQL DECLARE S2 CURSOR FOR
         SELECT   fix_label, fix_id,     fix_type,    fix_rad,
                  fix_dist,  width_left, width_right
         FROM     mtrsegments
         WHERE    fix_label LIKE :cid
         ORDER BY fix_label;

EXEC SQL OPEN S2;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int mtlistf()
/****************************************************************************
 *                                                                          *
 *     mtlistf   ---   Fetch a row using the opened cursor S2 for MTR       *
 *     =======         Navigation Point User Information                    *
 *                                                                          *
 *     Routine executes an fetch command for cursor S2, which is assumed    *
 *     to have been opened, and then returns to the calling program with    *
 *     the ORACLE status code.                                              *
 *                                                                          *
 *     Note:   Modifications to this function may impact the related        *
 *             functions mtlisto() and mtlistc() that open and close the    *
 *             cursor and, likely, functions that call these utilities      *
 *                                                                          *
 ****************************************************************************/
{


EXEC SQL FETCH S2 INTO :n2bv, :prefixid, :prefixtyp, :prefixrad,
                       :prefixdist, :prewidleft, :prewidright;

prefixid.arr[prefixid.len] = '\0';
prefixtyp.arr[prefixtyp.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int mtlistc()
/****************************************************************************
 *                                                                          *
 *          mtlistc   ---   Close cursor S2 for Navigation Points           *
 *          =======                                                         *
 *                                                                          *
 *     Routine executes a close cursor command for cursor S2 and then       *
 *     returns to the calling program with the ORACLE status code.          *
 *                                                                          *
 *     Note:   Modifications to this function may impact the related        *
 *             functions mtlisto() and mtlistf() that open the cursor       *
 *             and fetch rows using it and, possibly, functions that        *
 *             call these utilities                                         *
 *                                                                          *
 ****************************************************************************/
```

```
{
#ifdef CHECKOUT
printf("\mmtlistc ");
#endif

EXEC SQL CLOSE S2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int malisto()
/*************************************************************************
*                                                                        *
*     malisto   ---   Open cursor S3 for a list of user information      *
*     ========        of navigation points on a particular MTR           *
*                                                                        *
*    Routine executes an open cursor command for cursor S3 and then      *
*    returns to the calling program with the ORACLE status code.         *
*                                                                        *
*    Note:   Modifications to this function may impact the related       *
*            functions mtlistf() and mtlistc() that fetch rows and       *
*            close the cursor and, possibly, functions that call         *
*            these utility routines.                                     *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\mmalisto ");
#endif

strcpy( cid.arr, srcid.arr);
strcat(cid.arr,"%");
cid.len = strlen(cid.arr);

EXEC SQL DECLARE S3 CURSOR FOR
        SELECT   fix_label, floor_ref, ceiling_ref, fix_id,   fix_type,
                 artcc,     fix_lat,   fix_lon,     fix_rad,  fix_dist,
                 floor,     ceiling,   width_left,  width_right
        FROM     mtrsegments
        WHERE    fix_label LIKE :cid
        ORDER BY fix_label;

EXEC SQL OPEN S3;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int malistf()
/*************************************************************************
*                                                                        *
*    malistf   ---   Fetch a row using the opened cursor S3 for MTR      *
*    ========        Navigation Point User Information                   *
*                                                                        *
*    Routine executes an fetch command for cursor S3, which is assumed   *
*    to have been opened, and then returns to the calling program with   *
*    the ORACLE status code.                                             *
*                                                                        *
*    Note:   Modifications to this function may impact the related       *
*            functions mtlisto() and mtlistc() that open and close the   *
*            cursor and, likely, functions that call these utilities     *
*                                                                        *
*************************************************************************/
{

strcpy(cptr3,"              ");  /* If you don't use VARCHAR, you have to  */
strcpy(cptr4,"              ");  /* clear the space or weird things happen */

EXEC SQL FETCH S3 INTO :a2bv,        :cptr1,       :cptr2,       :curfixid,
                       :curfixtyp,   :curartcc,    :cptr3,       :cptr4,
                       :curfixrad,   :curfixdist,  :lptr1,       :lptr2,
                       :curwidleft,  :curwidright;
```

```
prefixid.arr[prefixid.len] = '\0';
prefixtyp.arr[prefixtyp.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int malistc()
/*************************************************************************
*                                                                        *
*          malistc   ---   Close cursor S3 for Navigation Points         *
*          =======                                                       *
*                                                                        *
*     Routine executes a close cursor command for cursor S3 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions mtlistc() and mtlistf() that open the cursor      *
*             and fetch rows using it and, possibly, functions that      *
*             call these utilities                                       *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nmtlistc ");
#endif

EXEC SQL CLOSE S3;
#ifdef CHECKOUT
printf(" Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}
/*************************************************************************
*                                                                        *
*     startup.pc  --  ASAN Initialization Code.  This program is only    *
*                     used to determine the status of ORACLE and to      *
*                     load those pieces which are needed.  The space     *
*                     it occupies can be relinquished after execution.   *
*                                                                        *
*************************************************************************/

#include <stdio.h>                /*  The usual stuff, of course */
#include <process.h>              /*  Header for calls to MS-DOS */
#include <string.h>               /*  String manipulation header */
#include <time.h>

#define SQLCA_STORAGE_CLASS extern

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE hostvars.h;
EXEC SQL INCLUDE fmharris.h;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;
#include "asan.h"


strtASAN()
/*************************************************************************
*                                                                        *
*     strtASAN   --  start ASAN function                                 *
*                                                                        *
*         Starts by attempting to connect to ORACLE as "SUPERUSER".      *
*         If ORACLE is not up it will attempt to install it.  If         *
*         unsuccessful, execution will terminate with appropriate        *
*         diagnostic information (i.e. what piece of the ORACLE          *
*         database manager it cannot find).                              *
*                                                                        *
*                                                                        *
*    Note:  This program is totally dependent on how ORACLE and its      *
*           associated programs behave.  It should be tested (with       *
*           #define CHECKOUT here and in other files with ORACLE calls)   *
*           when a new ORACLE release is installed.  Particularly the    *
*           return codes for uninstalled systems, since they are not     *
```

```
*          according to specification in release 5.1.  (e.g., -3120        *
*          should be -3121).                                               *
*                                                                          *
****************************************************************************/
{
int rcode, xcode;                    /* Oracle return code temporaries */
int dcode = 0;                       /* MS-DOS return code */
int spawnlp(), SUcomm();

char *clock();

strcpy(ASSESSMENT.name, "SUPERUSER");

OO_rcode = SUcomm();                 /* Try SUPERUSER and see what happens */

if (rcode = OO_rcode) {             /* 0 == ORACLE up and running  */

   switch (rcode) {
   case  -3120:       /* 3120  ==  SQLPME is not installed */
/*                      ---------------------------------- */

      dcode = spawnlp(P_WAIT,"sqlpme.exe","sqlpme.exe","/noior",NULL);
      if ( dcode < 0 ) {
         printf("\nSystem not properly installed!  ASAN can't find");
         printf("\nORACLE's Protected Mode Executive SQLPME.EXE");
         printf("\nYour Data Administrator should be able to help\n\n");
         fprintf(dante, "\n%s SQLPME Failed (%d)", clock(), dcode);
         exit(16);}
      else  if ( dcode < 768 ) {
         printf("\nORACLE's Protected Mode Executive SQLPME.EXE");
         printf("\nAbnormally terminated.  Return code was %d", dcode);
         printf("\nYour Data Administrator should be able to help");
         fprintf(dante, "\n%s SQLPME Failed (%d)", clock(), dcode);
         exit(16);}
      fprintf(dante, "\n%s SQLPME Installed", clock());


   case ORA_UNAVAILABLE:   /*  Not available: Do IOR first   */
/*                            --------------------------   */
      dcode = spawnlp(P_WAIT, "ior.exe", "ior.exe", "warm", NULL);
      if ( dcode < 0 ) {
         printf("\nSystem not properly installed!  ASAN can't find");
         printf("\nORACLE's Startup routine IOR.EXE");
         printf("\nYour Data Administrator should be able to help");
         fprintf(dante, "\n%s ORACLE Server Failed(%d)", clock(), dcode);
         exit(16);}

      else  if ( dcode > 0 ) {
         printf("\nORACLE's Initialization Routine IOR.EXE has");
         printf("\nAbnormally terminated.  Return code was %d", dcode);
         printf("\nYour Data Administrator should be able to help");
         fprintf(dante, "\n%s ORACLE Server Failed (%d)", clock(), dcode);
         exit(16);}

      else {          /*    Give CONNECT one more try    */
         fprintf(dante, "\n%s ORACLE Server Started", clock());
         if ( xcode = SUcomm() ) {
            printf("\nASAN has attempted to install the ORACLE Database");
            printf("server, but after what \nappears to have been a ");
            printf("\successful installation ASAN doesn't want to start");
            printf("\nCause: %s", sqlca.sqlerrm.sqlerrmc);
            if (sqlca.sqlcode == SQL_BAD_LOGON)
        printf("\n\tASAN installation program has probably not been run yet");
            printf("\nYour Data Administrator should be able to help\n\n");
            system("pause");
            closeORA();
            fprintf(dante, "\n%s Initial Connect failed second time (%d)",
                    clock(), xcode);
            exit(16);}
         }
         break;

   default: {
      fprintf(dante, "\n%s ASAN cannot establish communication with ORACLE",
              clock());
      fprintf(dante, "\nCause: %s", sqlca.sqlerrm.sqlerrmc);
      printf("\n\nASAN cannot establish communication with the ORACLE ");
      printf("Database Server.  \nCause: %s", sqlca.sqlerrm.sqlerrmc);
      if (sqlca.sqlcode == SQL_BAD_LOGON)
      printf("\n\tASAN installation program has probably not been run yet");
      printf("\nYour Data Administrator should be able to help\n\n");
      system("pause");
```

```
        closeORA();
        exit(16);}
    }  /* End Switch */
    }  /* End If */
fprintf(dante, "\n%s ASAN Started ", clock());
return 0;
}



int vfy_ASAN()
/*****************************************************************************
*                                                                            *
*     vfy_ASAN  --  Routine to check ASAN release levels and other system    *
*                   integrity functions.  This function is not implemented   *
*                   in nor meaningful for the prototype version of ASAN      *
*                                                                            *
*****************************************************************************/
{
return (int) 0;
}



int pwcheck()
/*****************************************************************************
*                                                                            *
*      pwcheck  --  Routine to check password and/or user name validity      *
*                   This function is not implemented in the prototype        *
*                                                                            *
*****************************************************************************/
{
return (int) 0;
}
/*****************************************************************************
*                                                                            *
*       tables.pc  --  Set of routines to access the data dictionary's       *
*       ==========     inventory of users, tables, columns etc.              *
*                                                                            *
*    For each ORACLE cursor there are three routines as follows:             *
*       1. ...listo()   Open Cursor  (Opens a "logical file")                *
*       2. ...listf()   Fetch Cursor (Reads next record from logical file)   *
*          ...bunch()   Fetch Cursor (Reads next bunch from logical file     *
*                       applicable only for multiple choice options)         *
*       4. ...listc()   Close Cursor (Closes the logical file)               *
*                                                                            *
*    Routines in this file:                                                  *
*     ... =   u* -  Names of assessments (LIFO order on date started).       *
*             t  -  Names and comments of all tables CREATED by current      *
*                   assessment only.                                         *
*             c  -  Names and comments of columns in specific table [:tid]   *
*                   (retrieved in order of column in the table).             *
*            tc  -  Names and comments of columns in all tables CREATED by   *
*                   current assessment (sorted alphabetically on table and   *
*                   column within table)                                     *
*             v  -  ASAN table_of_contents in alphabetic order.              *
*                                                                            *
*             *  -  Asterisk indicates that multiple choice option is        *
*                   supported for this set of routines.                      *
*                                                                            *
*****************************************************************************/

#include <stdio.h>                /*  The usual stuff, of course */
#include <process.h>              /*  Header for calls to MS-DOS */
#include <string.h>               /*  String manipulation header */

#define SQLCA_STORAGE_CLASS extern
EXEC SQL INCLUDE SQLCA;           /*  SQL Communication Area     */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE hostvars.h;
EXEC SQL INCLUDE fsharris.h;

EXEC SQL END DECLARE SECTION;

#include "asan.h"                 /*  Standard ASAN Header file  */

int ulisto()
/*****************************************************************************
```

```
*                                                                        *
*     ulisto   ---   Open cursor U1 for userlist (ASAN assessments)      *
*     ======                                                             *
*                                                                        *
*    Routine executes an open cursor command for cursor U1 and then      *
*    returns to the calling program with the ORACLE status code.         *
*                                                                        *
*    Note:   Modifications to this function may impact the related       *
*            functions ulistf(), ubunch() and ulistc() that fetch rows   *
*            and close the cursor and, possibly, functions that call     *
*            these utility programs                                      *
*                                                                        *
**************************************************************************/
{
#ifdef CHECKOUT
printf("\nulisto ");
#endif

EXEC SQL DECLARE U1 CURSOR FOR
                    SELECT username          /*  Assessments are ORACLE Users  */
                    FROM sysuserlist,        /*  Standard Data Dictionary view */
                         table_of_contents   /*  ASAN's List of assessments.   */
                    WHERE sysuserlist.userid = table_of_contents.idnumber
                    ORDER BY timestamp DESC;  /* LIFO Listing Order */

EXEC SQL OPEN U1;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int ulistf()
/**************************************************************************
*                                                                        *
*     ulistf   ---   Fetch a row using the opened cursor U1 for userlist *
*     ======                                                             *
*                                                                        *
*     Routine executes an fetch command for cursor U1, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions ulisto(), ubunch() and ulistc() that open,       *
*             fetch groupwise and close the cursor and, likely, any      *
*             functions that call these utility programs                 *
*                                                                        *
**************************************************************************/
{
EXEC SQL FETCH U1 INTO :uid;
uid.arr[uid.len] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int ubunch()
/**************************************************************************
*                                                                        *
*     ubunch   ---   Fetch a bunch (20 or whatever the size of dsplmult) *
*     ======         using the opened cursor U1 for sysuserlist          *
*                                                                        *
*     Routine executes an fetch command for cursor U1, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions ulisto(), ulistf() and ulistc() that open and    *
*             close the cursor and, likely, functions that call these    *
*             utility programs                                           *
*                                                                        *
**************************************************************************/
{
register int i;
int nrows, mrows, rcode;
char *clock();
```

```
EXEC SQL FETCH U1 INTO :displmult;
nrows = (int) sqlca.sqlerrd[2];
mrows = (sizeof displmult) / 34;
rcode = (int) sqlca.sqlcode;

#ifdef CHECKOUT
printf(" Bunch: %ld returns %d of %d rows", sqlca.sqlcode, nrows, mrows);
#endif
if ((rcode == SQL_FETCH_OUT_OF_ORDER) || (rcode == SQL_EOF)) {
   if ((rcode == SQL_EOF) && (nrows > 0)) {
      for (i = 0; i < nrows; i++) displmult[i].arr[displmult[i].len] = '\0';
      if (nrows < mrows)
         for (i = nrows; i < mrows; i++)  displmult[i].arr[0] = '\0';
      NEWVALS();
      SLOUTP("The last assessment in the list is on the screen");    }
   else {
      NEWVALS();
      SLOUTP("You are already as far down in the list as you can");  }
   }
else if (rcode) fprintf(dante, "\n%s UBUNCH: %s",
                        clock(), sqlca.sqlerrm.sqlerrmc);
NEWVALS();
return rcode;
}


int ulistc()
/*************************************************************************
*                                                                        *
*          ulistc   ---   Close cursor U1 for userlist                   *
*          ======                                                        *
*                                                                        *
*     Routine executes a close cursor command for cursor U1 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions ulistc(), ulistf() and ubunch() that open the    *
*             cursor and fetch rows using it and, possibly, functions    *
*             that call these utilities                                  *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nulistc ");
#endif

EXEC SQL CLOSE U1;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int tlistc()
/*************************************************************************
*                                                                        *
*     tlistc   ---   Open cursor U2 for list of tables in an assessment  *
*     ======                                                             *
*                                                                        *
*     Routine executes an open cursor command for cursor U2 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions tlistf() and tlistc() that fetch rows and        *
*             close the cursor respectively and, possibly, functions     *
*             that call these utilities                                  *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\ntlistc");
#endif

EXEC SQL DECLARE U2 CURSOR FOR
                 SELECT name, cmnt      /* That is what table name and */
                 FROM   sys.v4axptab    /* comment are called in this view */
                 ORDER BY name;

EXEC SQL OPEN U2;
```

```
#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int tlistf()
/*************************************************************************
*                                                                        *
*    tlistf   ---   Fetch a row using the opened cursor U2 for tablelist *
*    ======                                                              *
*                                                                        *
*     Routine executes an fetch command for cursor U2, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions tlisto() and tlistc() that open and close        *
*             the cursor respectively and, possibly, functions that      *
*             call these utilities                                       *
*                                                                        *
*************************************************************************/
{
EXEC SQL FETCH U2 INTO :tid, :workspace;
tid.arr[tid.len] = '\0';
workspace.arr[workspace.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int tlistc()
/*************************************************************************
*                                                                        *
*          tlistc   ---   Close cursor U2 for tablelist                  *
*          ======                                                        *
*                                                                        *
*     Routine executes a close cursor command for cursor U2 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions tlisto() and tlistf() that open the cursor and   *
*             fetch rows using it respectively and, possibly, functions  *
*             that call these utilities                                  *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("  tlistc ");
#endif

EXEC SQL CLOSE U2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int clisto()
/*************************************************************************
*                                                                        *
*     clisto   ---   Open cursor U3 for list of columns in a table       *
*     ======                                                             *
*                                                                        *
*     Routine executes an open cursor command for cursor U3 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions clistf() and clistc() that fetch rows and        *
*             close the cursor respectively and, possibly, functions     *
*             that call these utilities                                  *
*                                                                        *
*************************************************************************/
{
```

```
#ifdef CHECKOUT
printf("\nclisto");
#endif

EXEC SQL DECLARE U3 CURSOR FOR
                SELECT   cname          /* That is what the column is  */
                FROM     col            /* called in this view         */
                WHERE tname = :tid      /* For the current table       */
                ORDER BY colno;         /* In order of creation        */

EXEC SQL OPEN U3;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int clistf()
/*****************************************************************************
*                                                                            *
*    clistf   ---   Fetch a row using the opened cursor U3 for column list  *
*    ======                                                                  *
*                                                                            *
*     Routine executes an fetch command for cursor U3, which is assumed     *
*     to have been opened, and then returns to the calling program with     *
*     the ORACLE status code.                                                *
*                                                                            *
*     Note:   Modifications to this function may impact the related         *
*             functions clisto() and clistc() that open and close           *
*             the cursor respectively and, possibly, functions that         *
*             call these utilities                                           *
*                                                                            *
*****************************************************************************/
{
EXEC SQL FETCH U3 INTO :cid;
tid.arr[tid.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int clistc()
/*****************************************************************************
*                                                                            *
*           clistc   ---   Close cursor U3 for tablelist                     *
*           ======                                                           *
*                                                                            *
*     Routine executes a close cursor command for cursor U3 and then        *
*     returns to the calling program with the ORACLE status code.            *
*                                                                            *
*     Note:   Modifications to this function may impact the related         *
*             functions clisto() and clistf() that open the cursor and      *
*             fetch rows using it respectively and, possibly, functions     *
*             that call these utilities                                      *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("  clistc ");
#endif

EXEC SQL CLOSE U3;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int tclisto()
/*****************************************************************************
*                                                                            *
*   tclisto   ---   Open cursor U4 for list of all columns sorted on table  *
```

```
*   ------                                                                 *
*                                                                          *
*     Routine executes an open cursor command for cursor U4 and then       *
*     returns to the calling program with the ORACLE status code.          *
*                                                                          *
*     Note:   Modifications to this function may impact the related        *
*             functions tclistf() and tclistc() that fetch rows and        *
*             close the cursor respectively and, possibly, functions       *
*             that call these utilities                                    *
*                                                                          *
****************************************************************************/

{
#ifdef CHECKOUT
printf("\ntclisto");
#endif

EXEC SQL DECLARE U4 CURSOR FOR
                    SELECT tname,          /* TNAME is the table name  */
                           cname,          /* CNAME that of the column */
                           remarks         /* in the COL view of the   */
                    FROM col               /* ORACLE Data Dictionary   */
                    ORDER BY tname, cname;

EXEC SQL OPEN U4;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int tclistf()
/****************************************************************************
*                                                                          *
*   tclistf   ---   Fetch a row using the opened cursor U4                 *
*   ------          (column names sorted by table name)                    *
*                                                                          *
*     Routine executes an fetch command for cursor U4, which is assumed    *
*     to have been opened, and then returns to the calling program with    *
*     the ORACLE status code.                                              *
*                                                                          *
*     Note:   Modifications to this function may impact the related        *
*             functions tclisto() and tclistc() that open and close        *
*             the cursor respectively and, possibly, functions that        *
*             call these utilities                                         *
*                                                                          *
****************************************************************************/
{
EXEC SQL FETCH U4 INTO :tid, :cid, :workspace;
tid.arr[tid.len] = '\0';
cid.arr[cid.len] = '\0';
workspace.arr[workspace.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int tclistc()
/****************************************************************************
*                                                                          *
*       tclistc   ---   Close cursor U4 for column list sorted by table    *
*       -------                                                            *
*                                                                          *
*     Routine executes a close cursor command for cursor U4 and then       *
*     returns to the calling program with the ORACLE status code.          *
*                                                                          *
*     Note:   Modifications to this function may impact the related        *
*             functions tclisto() and tclistf() that open the cursor and   *
*             fetch rows using it respectively and, possibly, functions    *
*             that call these utilities                                    *
*                                                                          *
****************************************************************************/
{
#ifdef CHECKOUT
```

```
printf("  talistc ");
#endif

EXEC SQL CLOSE U4;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int vlisto()
/*************************************************************************
*                                                                        *
*      vlisto   ---   Open cursor U5 for list of ASAN's assessments      *
*      ======                                                            *
*                                                                        *
*     Routine executes an open cursor command for cursor U5 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions vlistf() and vlistc() that fetch rows and close  *
*             the cursor and, possibly, functions that call these        *
*             utility programs                                           *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nvlisto ");
#endif

EXEC SQL DECLARE U5 CURSOR FOR
                    SELECT username, description
                      FROM sysuserlist,
                           table_of_contents
                     WHERE sysuserlist.userid = table_of_contents.idnumber
                  ORDER BY username DESC;

EXEC SQL OPEN U5;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int vlistf()
/*************************************************************************
*                                                                        *
*     vlistf   ---   Fetch a row using the opened cursor U5 for list of  *
*     ======         ASAN assessments and their descriptions             *
*                                                                        *
*     Routine executes an fetch command for cursor U5, which is assumed  *
*     to have been opened, and then returns to the calling program with  *
*     the ORACLE status code.                                            *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions vlisto() ulistc() that open close the cursor     *
*             and, likely, any functions that call these utility programs *
*                                                                        *
*************************************************************************/
{
EXEC SQL FETCH U5 INTO :uid, :workspace;
uid.arr[uid.len] = '\0';
workspace.arr[workspace.len] = '\0';
#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


int vlistc()
/*************************************************************************
*                                                                        *
*          ulistc   ---   Close cursor U5 for ASAN table_of_contents     *
*          ======                                                        *
```

```
*                                                                        *
*     Routine executes a close cursor command for cursor U5 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions ulisto() and ulistf() open the cursor and fetch  *
*             rows using it and, possibly, functions that call these     *
*             utilities                                                  *
*                                                                        *
**************************************************************************/
{
#ifdef CHECKOUT
printf("\nulisto ");
#endif

EXEC SQL CLOSE U5;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}


/**************************************************************************
*                                                                        *
*     util.pc   --   A few general utilities                             *
*     ========                                                           *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*     expOmsg        - displays expansion of ORACLE error message        *
*     expOwrn        - displays expansion of ORACLE warning message      *
*     clock          - returns time/date information                     *
*                                                                        *
**************************************************************************/

#include <process.h>                  /*  Header for calls to MS-DOS    */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern    /*  Switch for header files       */
EXEC SQL BEGIN DECLARE SECTION;       /*  All SQL declarations for this */
EXEC SQL INCLUDE hostvars.h;          /*  are in these two header files */
EXEC SQL INCLUDE fmharris.h;          /*  this one comes from "U"       */
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                     /*  Standard ASAN Header File     */



void expOmsg()
/**************************************************************************
*                                                                        *
*      expOmsg  --  Expand Oracle Error Message                          *
*                                                                        *
**************************************************************************/

{
#ifdef CHECKOUT
   printf("\n%.70s\n", sqlca.sqlerrm.sqlerrmc);
#else
   ENDOMSG;
   SLOUTP(sqlca.sqlerrm.sqlerrmc);
#endif
}


void expOwrn()
/**************************************************************************
*                                                                        *
*          expOwrn  --  Expand Oracle Warning Message                    *
*          ========                                                      *
*                                                                        *
**************************************************************************/
{
if (sqlca.sqlwarn[1] == 'W')
          SLOUTRP("SQLWARNING: Column Truncated");
if (sqlca.sqlwarn[2] == 'W')
          SLOUTRP("SQLWARNING: Null in aggregate");
if (sqlca.sqlwarn[3] == 'W')
          SLOUTRP("SQLWARNING: INTO var count != col count");
```

```
if (sqlca.sqlwarn[4] == 'W')
        SLOUTEP("SQLWARNING: Update or Delete w/o WHERE");
if (sqlca.sqlwarn[5] == 'W')
        SLOUTEP("SQLWARNING: ?????");
if (sqlca.sqlwarn[6] == 'W')
        SLOUTEP("SQLWARNING: Rollback Required");
if (sqlca.sqlwarn[7] == 'W')
        SLOUTEP("SQLWARNING: Change after query for UPDATE");
}


char *aclock()
/*************************************************************************
*                                                                        *
*          aclock  --  Routine that returns the location of the string  *
*          ======      with the date and time of this very moment       *
*                                                                        *
*************************************************************************/
{
long now, time();
char *ctime(), *text;

#ifdef CHECKOUT
   printf("\nclock ");
#endif

now = time(NULL);
text = ctime(&now);
text[24] = ' ';
return text;
}


VCAPITAL(x)
/*************************************************************************
*                                                                        *
*         VCAPITAL  --  Convert a VARCHAR entered by user to uppercase   *
*         ========      and store len                                    *
*                                                                        *
*************************************************************************/

VARCHAR *x;
{
register int i, j;
j = x->len = strlen(x->arr);
for (i=0; i<j; i++) x->arr[i] = toupper(x->arr[i]);
}



/*************************************************************************
*                                                                        *
*     These are a set of dummy routines while the program is under       *
*     development.  They will be removed in the production version.      *
*                                                                        *
*************************************************************************/

dummy()
{
SLOUTP("This feature is not available in this prototype version");
}

dummy2()
{
SLOUTP("This feature currently runs as a separate program.");
}
/*************************************************************************
*                                                                        *
*   utmconv.pc  --   Routines dealing with the wonderful world of UTM,   *
*   ==========       DLG, GRASS and such                                 *
*                                                                        *
*                                                                        *
*   This file contains:                                                  *
*                                                                        *
*   1. utm to lat/lon conversion (CC_u2ll) routines                      *
*   2. lat/lon to utm conversion (CC_ll2u) routines                      *
*                                                                        *
*      CC_u2ll_spheroid (spheroid_name)  must be called first. sets the  *
*             spheroid parameters for the ellipse 'spheroid_name' (see   *
```

```
*           get_spheroid.c for known spheroids).   Function is used for    *
*           conversions in either direction.                                *
*                                                                           *
*     CC_u2ll_spheroid_parameters (a,e)  called by CC_u2ll_spheroid() to    *
*           set the ellipsoid major axis 'a' and eccentricity squared 'e'   *
*           can be called directly for unknown ellipsoids.                  *
*                                                                           *
*     CC_u2ll_zone (zone)  must be called before CC_u2ll_north().  Set      *
*           utm 'zone' (must be non-zero).  Negative means southern         *
*           hemisphere.  Used to set the longitude of the central           *
*           meridian (only used for utm to lat/lon conversions)             *
*                                                                           *
*     CC_u2ll_north (north)  set the utm north.  Must be called before      *
*           CC_u2ll.                                                        *
*                                                                           *
*     CC_u2ll (east, lat, lon) computes lat, lon from east after            *
*           CC_u2ll_north() has already been called with north.             *
*                                                                           *
*                                                                           *
*  3. miscellaneous user interface service routines                         *
*                                                                           *
*     lon2dec   - Convert longitude of COORDINATE variable to seconds       *
*                 decimal                                                   *
*     lat2dec   - Convert latitude of COORDINATE variable to seconds        *
*                 decimal                                                   *
*     alt2dec   - Convert an ALTSPEC VARIABLE's spec to units and altitude  *
*     convcoord - Convert the spherical coordinate character string of      *
*                 a COORDINATE variable to decimal                          *
*     COOR2utm  - Convert COORDINATE decimal lat/lon to utms                *
*                                                                           *
*****************************************************************************/

#include <process.h>                    /*  Header for calls to MS-DOS     */
#include <stdio.h>
#include <ctype.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files        */

EXEC SQL BEGIN DECLARE SECTION;         /*  All SQL declarations are in    */
EXEC SQL INCLUDE hostvars.h;            /*  these header files             */
EXEC SQL INCLUDE fmharris.h;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                       /*  Standard ASAN Header File      */


#define abs(x)  ((x)<0?-(x):(x))

#define RADIANS_TO_SECONDS 206264.8062470964
#define SECONDS_TO_RADIANS 4.848136811095360e-6

double sqrt(), sin(), cos();

static double a1,a2,a3,a4;       /* lat coef: geodetic to rectifying */
static double a5 = 5.0e5;        /* false easting (UTM) */
static double a6;                /* false northing */
static double a7 = 0;            /* NO? */
static double a8 = .9996;        /* UTM scale factor at central meridian */
static double a9;                /* central meridian in seconds */
static double a10;               /* radius of curvature */
static double a11,a12,a13,a14;   /* lat coef: rectifying to geodetic */
static double a15;               /* major axis */
static double a16;               /* eccentricity squared */

static double b1,b2,b3,b4;       /* intermediate values */
static double b5,b6,b7,b8;
static double b9,b10,b11,b12;

static struct {  char *name;
                 double a;             /* semi-major axis */
                 double e;             /* eccentricity squared */
                 } spheroid[] =
                     {  "australian",     6378160.0,    0.0066945419,
                        "bessel",         6377739.155,  0.0066743722,
                        "clark66",        6378206.4,    0.006768658,
                        "clark80",        6378249.145,  0.0068035113,
                        "everest",        6377276.345,  0.0066378466,
                        "international",  6378388.0,    0.00672267,
                        "wgs72",          6378135.0,    0.006694317778
                      };
```

```
int CC_get_spheroid (name, a, e)
/*************************************************************************
*                                                                        *
*      CC_get_spheroid   --   Returns axis and excentricity squared      *
*      ================       for the named spheroid.  Return value      *
*                             1 for success, 0 for failure to find       *
*                             the spheroid in the list                   *
*                                                                        *
*************************************************************************/

char *name;
double *a, *e;
{
int n;

n = sizeof(spheroid)/sizeof(spheroid[0]);

#ifdef CHECKOUT
printf("\nget_spheroid %s from %d entries ", name, n);
#endif

while (n--)
   if (equal (name, spheroid[n].name)) {
      *a = spheroid[n].a;
      *e = spheroid[n].e;
      return 1;
      #ifdef CHECKOUT
         printf(" found %d", n);
      #endif
      }

   #ifdef CHECKOUT
      printf(" not found");
   #endif
   return 0 ;
}



char * CC_spheroid_name (n)
/*************************************************************************
*                                                                        *
*     CC_spheroid_name   --   Find the name of spheriod n in the list    *
*     ================                                                   *
*                                                                        *
*************************************************************************/

int n;

{
if (n < 0 || n >= sizeof(spheroid)/sizeof(spheroid[0]))  return 0;
else  return spheroid[n].name;
}


static equal (a, b)
/*************************************************************************
*                                                                        *
*         equal    --    comparison between two character strings      *
*         =====          (Forced to lowercase)                          *
*                                                                        *
*************************************************************************/
char *a, *b;
{
char lcase();

while (*a) if (lcase (*a++) != lcase (*b++))  return 0;
return *b == 0;
}



static char lcase (c)
/*************************************************************************
*                                                                        *
*      lcase    --   Force character to lowercase                       *
*      =====                                                            *
*                                                                        *
```

```
*****************************************************************************/
char c;
{
if (c >= 'A' && c <= 'Z')  c += 'a' - 'A';
return c;
}



int CC_u2ll_spheroid (spheroid_name)
/*****************************************************************************
*                                                                            *
*    CC_u2ll_spheroid  --  Set up spheroid parameters for UTM to Lat/Long    *
*    ==================    given a spheroid name                             *
*                                                                            *
*****************************************************************************/

char *spheroid_name ;
{
double a,e;

if (CC_get_spheroid (spheroid_name, &a, &e))        /* Known Spheroid */
   return CC_u2ll_spheroid_parameters (a, e);
return -1;                                          /* Unknown Spheroid */
}



int CC_u2ll_spheroid_parameters (a,e)
/*****************************************************************************
*                                                                            *
*     CC_u2ll_spheroid_parameters -- Store the conversion parameters         *
*     ============================   given axis and excentricity             *
*                                                                            *
*****************************************************************************/
double a,e;
{
double x,x2,x3,x4;

if (a < 0.0 || e < 0.0 || e > 1.0) return -2;       /* illegal values */

a15 = a;
a16 = e;

x  = (((e * (7.0/32.0) + 5.0/16.0) * e + .5) * e + 1.0) * e * .25;
x2 = x * x;
x3 = x * x2;
x4 = x * x3;

/* coefficients to convert geodetic to rectifying latitude */

a1  = -(((x * (195.0/64.0) + 3.25) * x + 3.75) * x + 3.0) * x ;
a2  =  (((1455.0/32.0) * x + 70.0/3.0) * x + 7.5) * x2;
a3  = -((70.0/3.0) + x * (945.0/8.0)) * x3;
a4  =  (315.0/4.0) * x4;


/* coefficients to convert rectifying to geodetic latitude */

a11 =  (((7.75 - (657.0/64.0) * x) * x - 5.25) * x + 3.0) * x;
a12 =  (((5045.0/32.0) * x - (151.0/3.0)) * x + 10.5) * x2;
a13 =  ((151.0/3.0) - (3291.0/8.0) * x) * x3;
a14 =  (109.0/4.0) * x4;


/* radius of curvature */

a10 =  (((225.0/64.0) * x2 + 2.25) * x2 + 1.0) * (1.0 - x2) * (1.0 - x) * a;
return 1;
}



int CC_u2ll_zone (zone)
/*****************************************************************************
*                                                                            *
*     CC_u2ll_zone                                                           *
```

```
*     ____________                                                      *
*                                                                       *
*************************************************************************/
{
double utz;

/* set false northing (a6), compute central meridian (a9) */

if (zone < 0)  {
   a6 = 10.0e6;
   utz = 30.0 + zone;
   }
else {
   a6 = 0.0;
   utz = 30.0 - zone;
   }
a9 = (utz * 6.0 + 3.0) * 3600.0;
}



int CC_u211_north (north)
/*************************************************************************
*                                                                       *
*     CC_u211_north                                                     *
*     _____________                                                     *
*                                                                       *
*************************************************************************/
double north;
{
double sinw, cosw;
double t, ts, rn, rn2, rn4, rn6, rn8;
double etas;

b10 = ((north - a6) / a8 + a7) / a10;
if (abs(b10) > 1.47)
   return -1; /* rectifying lat exceeds 1.47 radians, -84.15.30 */

sinw = sin(b10);
cosw = cos(b10);
b12  = cosw * cosw;
b11  = (((a14 * b12 + a13) * b12 + a12) * b12 + a11) * sinw * cosw + b10;

sinw = sin(b11);
cosw = cos(b11);
rn   = sqrt (1.0 - a16*sinw*sinw) * 1.0e6 / a15;
rn2  = rn * rn ;
rn4  = rn2 * rn2 ;
rn6  = rn2 * rn4 ;
rn8  = rn4 * rn4 ;
t    = sinw/cosw;
ts   = t * t;

b12  = cosw * cosw;
etas = a16 * b12 / (1.0 - a16);

b1   = rn/cosw;
b2   = -t * (1.0 + etas) * rn2 / 2.0 ;
b3   = - (1.0 + ts + ts + etas) * b1 * rn2 / 6.0 ;
b4   = ((( -6.0 - etas * 9.0) * etas + 3.0) * ts +
          (6.0 - etas * 3.0) * etas + 5.0 ) * t * rn4 / 24.0 ;
b5   = ((ts * 24.0 + etas * 8.0 + 28.0) * ts
                  + etas * 6.0 +  5.0) * b1 * rn4 / 120.0 ;
b6   = (((etas *  45.0 - 45.0) * ts + etas * 162.0 -90.0) * ts
        -etas * 107.0 - 61.0) * t * rn6 / 720.0 ;
b7   = -(((ts * 720.0 + 1320.0) * ts
                     + 662.0) * ts + 61.0) * b1 * rn6 / 5040.0 ;
b8   = (((ts * 1575.0 + 4095.0) * ts
                    + 3633.0) * ts + 1385.0) * t * rn8 / 40320.0;

return 1;
}



int CC_u211 (east, lat, lon)
/*************************************************************************
*                                                                       *
*     CC_u211                                                           *
*     _______                                                           *
```

```
*                                                                        *
**************************************************************************/

double east;
double *lat, *lon;
{
double b9, b10;

b9 = ((a5 - east) * 1.0e-6) / a8 ;
if (abs(b9) > a15 * 2.0e-7)
   return -1;   /* utm easting to far from center of zone */

b10 = b9 * b9 ;
*lat = ((((b8 * b10 + b6) * b10 + b4) * b10 + b2) * b10 + b11)
                                          * RADIANS_TO_SECONDS ;
*lon = (((b7 * b10 + b5) * b10 + b3) * b10 + b1) * b9
                                          * RADIANS_TO_SECONDS + a9 ;

return 1;
}



int CC_112u (lat, lon, east, north, zone)
/**************************************************************************
*                                                                        *
*     CC_112u                                                            *
*     =======                                                            *
*                                                                        *
**************************************************************************/

double lat, lon;
double *east, *north;
int *zone;
{
int deg;
double sinp, cosp;
double etas, t, ts, rn;
double c1,c2,c3;

/* if zone is != 0, force into this zone, otherwise compute the zone */

if (*zone == 0)  {
   if (lon < 0)  {            /* eastern zones */
      deg = -lon / 3600;
      *zone = 31 + deg / 6;
      }
   else  {                    /* western zones */
      deg = lon / 3600 ;
      *zone = 30 - deg / 6;
      }
   if (lat < 0)     *zone = -(*zone) ;
   }

      /* now, set a6,a9 */

CC_u211_zone (*zone);
if (abs(lat) > 302400.0)  return -1;   /* latitude above 84 degrees */

b10 = (a9 - lon) * SECONDS_TO_RADIANS;
if (abs(b10) > .16)  return -2; /* longitude to far from center of utm zone */

b9    = lat * SECONDS_TO_RADIANS;
sinp  = sin (b9);
cosp  = cos (b9);
rn    = a15 / sqrt (1.0 - a16 * sinp * sinp);
t     = sinp / cosp;
ts    = t * t;
c1    = cosp * cosp;
c2    = c1 * c1;
c3    = c1 * c2;
etas  = a16 * c1 / (1.0 - a16);

b1    = rn * cosp;
b3    = (1.0 - ts + etas) * b1 * c1 / 6.0;
b5    = ((ts - 18.0) * ts + 5.0 + (14.0 - 58.0 * ts) * etas) * b1 * c2 / 120.0;
b7    = (((179.0 -ts) * ts - 479.0) * ts + 61.0) * b1 * c3 / 5040.0 ;
b12   = b10 * b10;

*east = (((b7 * b12 + b5) * b12 + b3) * b12 + b1) * b10 * a8 + a5;
```

```
b2     = zm * c1 * t / 2.0;
b4     = (etas * (9.0 + 4.0 * etas) + 5.0 - ts) * b2 * c1 / 12.0;
b6     = ((ts - 58.0) * ts + 61.0 +
                (270.0 - 333.0 * ts) * etas) * b2 * c2 / 360.0;
b8     = (((543.0 - ts) * ts - 3111.0) * ts + 1385.0) * b2 * c3 / 20160.0;

*north = (((b8 * b12 + b6) * b12 + b4) * b12 + b2) * b12 +
            ((((a4 * c1 + a3) * c1 + a2) * c1 + a1) * sinp * cosp + b9)
            * a10;

*north = (*north - a7) * a8 + a6;

return 1;
}


int lon2dec(x)
/*************************************************************************
*                                                                        *
*          lon2dec   --  Convert a COORDINATE VARIABLE's longitude to    *
*          ========      decimal (seconds of arc)                        *
*                                                                        *
*       Routine unpacks longitude of the structure, reformats it and then *
*       stores the reformatted and the converted value in the structure  *
*                                                                        *
*************************************************************************/

COORDINATE *x;

{

double convcoord();
register int i;

#ifdef CHECKOUT
printf("\nlon2dec ");
#endif

for (i = 0; i <14; i++) x->lon[i] = toupper(x->lon[i]);
if ((x->longitude = convcoord(x->lon)) < OFF_EARTH) {
   if (x->longitude > 0.0)   x->lon[12] = 'W';
   else x->lon[12] = 'E';
   }
else x->eastings = -1.0e75;
}


lat2dec(x)
/*************************************************************************
*                                                                        *
*          lat2dec   --  Convert a COORDINATE VARIABLE's latitude to     *
*          ========      decimal (seconds of arc)                        *
*                                                                        *
*       Routine unpacks latitude of the structure, reformats it and then *
*       stores the reformatted and the converted value in the structure  *
*                                                                        *
*************************************************************************/

COORDINATE *x;

{
double convcoord();
register int i;

for (i = 0; i < 14; i++) x->lat[i] = toupper(x->lat[i]);
x->latitude = convcoord(x->lat);
if (x->latitude < OFF_EARTH) {
   x->lat[0] = ' ';
   if (x->latitude > 0.0) {
      if (x->latitude > 324000.0) {
         ELOUTRP(" > 90  ");
         x->northings = -1.0e75;
         }
      else  x->lat[12] = 'N';
      }
   else {
      if (x->latitude < -324000.0) {
         ELOUTRP(" > 90  ");
         x->northings = -1.0e75;
         }
      else x->lat[12] = 'S';
      }
```

```
   }
else x->northings = -1.0e75;
}



int alt2dec(x)
/*****************************************************************************
*                                                                            *
*      alt2dec   --  Convert an ALTSPEC VARIABLE's spec to units and         *
*      =======        long integer value                                     *
*                                                                            *
*     Note:  Use only to convert values at the time of entry or the status  *
*            line message will make you wonder with great amazement ......  *
*                                                                            *
*****************************************************************************/
ALTSPEC *x;
{
register int i, j;

for (i = 0; i < 10; i++) x->spec[i] = toupper(x->spec[i]);

j = strlen(x->spec);
x->altitude = 0;

for (i = 0; i < j; i++) {
   if (isspace(x->spec[i])) continue;
   if (isdigit(x->spec[i])) break;
   if (x->spec[i] == 'A') {
      strcpy(x->spec, "AS ASSG'D ");
      strcpy(x->units, "TBD");}
   else  SLOUTEP("Not altitude");
   return 0;}
if (i == j) {
   SLOUTEP("No value");
   return 0;}
do {
   x->altitude = (10 * x->altitude) + x->spec[i] - 48;
   i++;} while (isdigit(x->spec[i]) && i < j);
for (; i<j; i++)
   if (!isspace(x->spec[i])) break;
if (i == j) {
   SLOUTEP("Need Reference (MSL,AGL,SFC) or ASSIGNED");
   return 0;}
if      (x->spec[i] == 'A') strcpy(x->units, "AGL");
else if (x->spec[i] == 'M') strcpy(x->units, "MSL");
else if (x->spec[i] == 'S') strcpy(x->units, "SFC");
else return 0;
sprintf(x->spec,"%04ld %s", x->altitude, x->units);
return 0;
}



double convcoord(x)
/*****************************************************************************
*                                                                            *
*        convcoord   --  Unpack a character string with a spherical          *
*        ==========      coordinate and return seconds of arc as a double    *
*                                                                            *
*     Note: The character string will be reformatted to nnnxnn'nn.n"X       *
*           (as in 112x30'24.5"W)                                            *
*                                                                            *
*****************************************************************************/

char x[];
{
int i, j, ideg = 0, imin = 0;
double t = 0.0;

j = strlen(x);
for (i = 0; i < j; i++) {                    /* Degrees */
   if (isspace(x[i])) continue;
   if (isdigit(x[i])) break;
   SLOUTEP("Not a coordinate");
   return (double) OFF_EARTH;}
if (i == j) {
   SLOUTEP("No value");
   return (double) OFF_EARTH;}
```

```
do {
   ideg = (10 * ideg) + x[i] - 48;
   i++;} while (isdigit(x[i]) && i < j);

if ((ideg > 180) ||
      ((x[i] != ' ') && (x[i] != 'D') && (x[i] != '\0') && (x[i] != -8))) {
   SLOUTEP("? Degrees"); return (double) OFF_EARTH;}

if (i < j-1) {
   i++;
   for (; i < j; i++) {                        /* Minutes */
      if (isspace(x[i])) continue;
      if (isdigit(x[i])) break;
      SLOUTEP("? Minutes");
      return (double) OFF_EARTH;}
   for ( ;isdigit(x[i]) && i < j; i++) {
      imin = (10 * imin) + x[i] - 48;
      }

   if ((imin > 60) ||
       ((x[i] != ' ') && (x[i] != 'M') && (x[i] != '\'') && (x[i] != '\0'))) {
      SLOUTEP("? Minutes"); return (double) OFF_EARTH;}
   }

if (i < j-1) {
   i++;
   for (; i < j; i++) {                        /* Seconds */
      if (isspace(x[i])) continue;
      if (isdigit(x[i])) break;
      SLOUTEP("? Seconds");
      return (double) OFF_EARTH;}
   for (;isdigit(x[i]) && i < j;i++) {
      t = 10.0 * t + (double)(x[i]-48);
      }
   if (x[i] == '.') {
      i++;
      if (i < j) {
         if (isdigit(x[i])) t += ((double)(x[i]-48))/10.0;
         i++;}
      }

   if ((t > 60.0) ||
      ((x[i] != '\0') && (x[i] != ' ') && (x[i] != 'S') && (x[i] != '"'))) {
      SLOUTEP("? Seconds"); return (double) OFF_EARTH;}
   }

if (i < j) i++;
for (; i < j; i++) {                       /* Hemisphere */
   if (!isspace(x[i])) break;}

if ((i == j) || (x[i] == 'N') || (x[i] == 'W')) {
   sprintf(x,"%03d %02d'%04.1lf\" ", ideg, imin, t);
   return (3600.0 * (double)(ideg) + 60.0 * (double)(imin) + t);}
else if ((x[i] == 'S') || (x[i] == 'E')) {
   sprintf(x,"%03d %02d'%04.1lf\" ", ideg, imin, t);
   return (-(3600.0 * (double)(ideg) + 60.0 * (double)(imin) + t));}
SLOUTEP("? Hemisphere");

return (double) OFF_EARTH;
}



int COOR2utm(x)
/*************************************************************************
*                                                                        *
*      COOR2utm  --  Convert a decimal latitude and longitude to the     *
*      ==========    corresponding utm northing/eastings for an ASAN     *
*                    COORDINATE structure                                *
*                                                                        *
*************************************************************************/

COORDINATE *x;

{
char *clock();

if ( (x->latitude < OFF_EARTH) &&  (x->longitude < OFF_EARTH) ) {
   if (CC_ll2u (x->latitude, x->longitude,
               &x->eastings, &x->northings, &utmzone) == 1) return (int) 0;
```

```
/*  If it didn't work, then complain to the user */

   else {
      SLOUTEP("Coordinate conversion error");
      fprintf(dante,"\n%s COORDINATE Conversion error", clock());
      fprintf(dante,"\n\t\t\tLat  = %12.21f [%s]", x->latitude,  x->lat);
      fprintf(dante,"\n\t\t\tLong = %12.21f [%s]", x->longitude, x->lon);
      }
   }
else {
   SLOUTEP("Fix bad coordinates first");
   }
return (int) -1;
}
/*************************************************************************/
/*                                                                       */
/*         asan.pc  --  ASAN Main Program                                */
/*                                                                       */
/*            1. Opens the printer.                                      */
/*            2. Calls strtASAN to establish communication with ORACLE   */
/*            3. Verifies that ASAN software is valid.                   */
/*            4. Calls Uinit to start the screen driver.                 */
/*            5. At end of the session closes database and printer       */
/*                                                                       */
/*                                                                       */
/*************************************************************************/

#include <stdio.h>                  /*  The usual stuff, of course */
#include <process.h>                /*  Header for calls to MS-DOS */
#include <string.h>                 /*  String manipulation header */
#include <time.h>


EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;             /*  SQL Communication Area      */
#include "asan.h"                   /*  Standard ASAN Header file   */
                                    /*  In asan.pc it must be here  */
                                    /*  since we need to initialize */
                                    /*  character arrays!           */

main()
{
extern int ERA_DEBUG_FEATURES;
int i;
static char *legal_notice[11] =
{"\n\n\n\n\t\t\t  RESTRICTED RIGHTS LEGEND\n\n",
"\tUse, duplication, or disclosure is subject to restrictions\n",
"\t      as set forth in subdivision (b)(3)(ii) of the\n",
"\t   Rights in Technical Data and Computer Software Clause\n",
"\t\t    at 52.227-7013 of the DOD FAR Supplement.\n\n",
"\t\t\tBBN LABORATORIES INCORPORATED\n",
"\t\t\t      10 MOULTON STREET\n",
"\t\t\t     Cambridge, MA 02238\n",
"\t\t\t        617-873-3000\n\n\n",
"     User Interface Copyright (C) 1985,  BBN Laboratories Incorporated\n",
"\t\t\t     All Rights Reserved"};


FILE *fopen();

prn = fopen("prn","a");
if (prn == NULL) {
    printf("\nCan't open printer!");  exit(128); }


dante = fopen("chronfil.asf","a");
if (dante == NULL) {
    printf("\nCan't open chronfile!");  exit(128); }

printf("%c[2J",27);
for (i = 0; i < 11; i++) printf("%s",legal_notice[i]);

printf("\n\n\n\n\n\t   Please tap the space bar to continue, \"CTRL-C\"\
 to abort.\n\n");
do {
    i = getch();
    } while (i != ' ');
```

```
printf("ASAN Citation module starting.....");
NEA_DEBUG_FEATURES = 1;


strcpy(cituid.arr,"INQUIRER/ALIGHIERI");
cituid.len = strlen(cituid.arr);

EXEC SQL CONNECT :cituid;

if (sqlca.sqlcode) {
   ERDCMSG;
   printf("\nCannot Start Citation Database Inquiry due to\n\n%s\n\n",
          sqlca.sqlerrm.sqlerrmc);
   exit (16);}

EXEC SQL SELECT COUNT(entry_num)
                FROM headquarters.animal_list
                INTO :maxanimals;

Uinit();
EXEC SQL ROLLBACK WORK RELEASE;
exit(0);
}
/*************************************************************************
*                                                                        *
*   query1.pc -- Routines for citation database retrieval without keywords *
*   ----------                                                           *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*     qsetup   -  Set up for a new search                                *
*     queryh   -  Select citations from the Human area                   *
*     querya   -  Select citations from the Animal area                  *
*     querys   -  Select citations from the Structures area              *
*     querym   -  Select citations from the Modeling area                *
*     queryw   -  Select based on (first few letters of) Writer's name   *
*     queryd   -  Select based on the range of Dates given               *
*     queryt   -  Select based on the occurrence of a phrase in the Title *
*                                                                        *
*************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS    */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                       /*  Standard ASAN Header File     */

EXEC SQL DECLARE C100 CURSOR FOR
                      SELECT entry_num
                        FROM headquarters.citation_search
                       WHERE human_area = 'T';

EXEC SQL DECLARE C101 CURSOR FOR
                      SELECT q.entry_num
                        FROM headquarters.citation_search s, qual_cit1 q
                       WHERE s.entry_num = q.entry_num
                         AND s.human_area = 'T';

EXEC SQL DECLARE C102 CURSOR FOR
                      SELECT q.entry_num
                        FROM headquarters.citation_search s, qual_cit2 q
                       WHERE s.entry_num = q.entry_num
                         AND s.human_area = 'T';



EXEC SQL DECLARE C200 CURSOR FOR
                      SELECT entry_num
                        FROM headquarters.citation_search
                       WHERE animal_area = 'T';

EXEC SQL DECLARE C201 CURSOR FOR
```

```
                    SELECT q.entry_num
                      FROM headquarters.citation_search s, qual_cit1 q
                     WHERE s.entry_num = q.entry_num
                       AND s.animal_area = 'T';

EXEC SQL DECLARE C202 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.citation_search s, qual_cit2 q
                     WHERE s.entry_num = q.entry_num
                       AND s.animal_area = 'T';



EXEC SQL DECLARE C300 CURSOR FOR
                    SELECT entry_num
                      FROM headquarters.citation_search
                     WHERE struc_area = 'T';

EXEC SQL DECLARE C301 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.citation_search s, qual_cit1 q
                     WHERE s.entry_num = q.entry_num
                       AND s.struc_area = 'T';

EXEC SQL DECLARE C302 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.citation_search s, qual_cit2 q
                     WHERE s.entry_num = q.entry_num
                       AND s.struc_area = 'T';



EXEC SQL DECLARE C400 CURSOR FOR
                    SELECT entry_num
                      FROM headquarters.citation_search
                     WHERE model_area = 'T';

EXEC SQL DECLARE C401 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.citation_search s, qual_cit1 q
                     WHERE s.entry_num = q.entry_num
                       AND s.model_area = 'T';

EXEC SQL DECLARE C402 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.citation_search s, qual_cit2 q
                     WHERE s.entry_num = q.entry_num
                       AND s.model_area = 'T';



EXEC SQL DECLARE C50s CURSOR FOR
                    SELECT authornum
                      FROM headquarters.author_list a
                     WHERE UPPER(author) LIKE :pattern;

EXEC SQL DECLARE C500 CURSOR FOR
                    SELECT entry_num
                      FROM headquarters.author_citation_link
                     WHERE authornum = :authornum;

EXEC SQL DECLARE C501 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.author_citation_link l, qual_cit1 q
                     WHERE l.entry_num = q.entry_num
                       AND l.authornum = :authornum;

EXEC SQL DECLARE C502 CURSOR FOR
                    SELECT q.entry_num
                      FROM headquarters.author_citation_link l, qual_cit2 q
                     WHERE l.entry_num = q.entry_num
                       AND l.authornum = :authornum;



EXEC SQL DECLARE C600 CURSOR FOR
                    SELECT entry_num
                      FROM headquarters.citation_search
```

```
                        WHERE date_pub >= :date1
                          AND date_pub <= :date2;

EXEC SQL DECLARE C601 CURSOR FOR
                       SELECT q.entry_num
                         FROM headquarters.citation_search s, qual_cit1 q
                        WHERE s.entry_num = q.entry_num
                          AND s.date_pub >= :date1
                          AND s.date_pub <= :date2;

EXEC SQL DECLARE C602 CURSOR FOR
                       SELECT q.entry_num
                         FROM headquarters.citation_search s, qual_cit2 q
                        WHERE s.entry_num = q.entry_num
                          AND s.date_pub >= :date1
                          AND s.date_pub <= :date2;


EXEC SQL DECLARE C700 CURSOR FOR
                       SELECT entry_num
                         FROM headquarters.citation_titles
                        WHERE UPPER(title) LIKE :pattern;

EXEC SQL DECLARE C701 CURSOR FOR
                       SELECT q.entry_num
                         FROM headquarters.citation_titles c, qual_cit1 q
                        WHERE c.entry_num = q.entry_num
                          AND UPPER(c.title) LIKE :pattern;

EXEC SQL DECLARE C702 CURSOR FOR
                       SELECT q.entry_num
                         FROM headquarters.citation_titles c, qual_cit2 q
                        WHERE c.entry_num = q.entry_num
                          AND UPPER(c.title) LIKE :pattern;


int qsetup()
/*************************************************************************
*                                                                        *
*        qsetup   --  start-up routine that clears all temporary tables  *
*        ------       for intermediate pointers in queries              *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqsetup ");
#endif

SLOUT("Purging obsolete qualifier lists");

EXEC SQL DELETE FROM QUAL_CIT1;
if (sqlca.sqlcode)
   if (sqlca.sqlcode != SQL_NOF) {
      ENDOMSG;
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      exit(16);}
#ifdef CHECKOUT
   else  printf("\nQualifying table1 cleared %ld", sqlca.sqlcode);
#endif

EXEC SQL DELETE FROM QUAL_CIT2;
if (sqlca.sqlcode)
   if (sqlca.sqlcode != SQL_NOF) {
      ENDOMSG;
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      exit(16);}
#ifdef CHECKOUT
   else printf("\nQualifying table2 cleared %ld", sqlca.sqlcode);
#endif
EXEC SQL COMMIT WORK;
qual_entries = 0;                    /* No qualifying entries */
temptabl = 0;                        /* No temporary table    */
dates1 = dates2 = 0;                 /* No date */
selcrit[0] = '\0';                   /* No search path        */
animals = maxanimals;                /* ALL Animals ...       */
olddepth = 0;

contsum[0] = '\0';
```

```
majorst[0] = '\0';
minorst[0] = '\0';
address[0] = '\0';
setaff(6);

NEWVALS();
return (int) 0;
}



int queryh()
/*****************************************************************************
*                                                                            *
*    queryh  --  Create the subset of citations for the titles that          *
*    ======      pertain to the HUMAN area                                   *
*                                                                            *
*****************************************************************************/
{
int i;

#ifdef CHECKOUT
   printf("\nqueryh ");
#endif

SLOUT("Human Area Search");
strcat(selcrit,"Human ");
switch (temptabl) {
   case 0:
      EXEC SQL OPEN C100;
      if (sqlca.sqlcode) {
         ERDOMSG;
         SLOUTP("CURSOR C100");
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C100 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)   {
            ERDOMSG;
            SLOUTP("CURSOR C100");
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         #ifdef CHECKOUT
            printf("\nAccepted ");
            for (i = 0; i < 5; i++) printf("%c",enumb.arr[i]);
         #endif
         qual_entries++;
         NEWVALS();
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C100;
      temptabl = 1;
      break;

   case 1:
      EXEC SQL DELETE FROM QUAL_CIT2;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C101;
      if (sqlca.sqlcode) {
         ERDOMSG;
         SLOUTP("CURSOR C101");
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C101 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            ERDOMSG;
            SLOUTP("CURSOR C101");
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
```

```
            if (sqlca.sqlcode) {
               ENDOMSG;
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               exit(16);
               }
            #ifdef CHECKOUT
               printf("\nAccepted ");
               for (i = 0; i < 5; i++) printf("%c",enumb.arr[i]);
            #endif
            qual_entries++;
            NEWVALS();
            }
         EXEC SQL COMMIT WORK;
         EXEC SQL CLOSE C101;
         temptabl = 2;
         break;

      case 2:
         EXEC SQL DELETE FROM QUAL_CIT1;
         EXEC SQL COMMIT WORK;
         EXEC SQL OPEN C102;
         if (sqlca.sqlcode) {
            ENDOMSG;
            SLOUTP("CURSOR C102");
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         qual_entries = 0;
         for (;;) {
            EXEC SQL FETCH C102 into :enumb;
            if (sqlca.sqlcode == SQL_EOF) break;
            if (sqlca.sqlcode)  {
               ENDOMSG;
               SLOUTP("CURSOR C102");
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               exit(16);
               }
            EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
            if (sqlca.sqlcode) {
               ENDOMSG;
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               exit(16);
               }
            #ifdef CHECKOUT
               printf("\nAccepted ");
               for (i = 0; i < 5; i++) printf("%c",enumb.arr[i]);
            #endif
            NEWVALS();
            qual_entries++;
            }
         EXEC SQL COMMIT WORK;
         EXEC SQL CLOSE C102;
         temptabl = 1;
         break;

      default:
         bad_temp();
      }

   #ifdef CHECKOUT
      printf(" %d entries in table %d", qual_entries, temptabl);
      SLOUTP("Ack");
   #endif
   return (int) qual_entries;
   }



int querya()
/*****************************************************************************
*                                                                           *
*    querya  --  Create the subset of citations for the titles that         *
*    ======      pertain to the ANIMAL area                                 *
*                                                                           *
*****************************************************************************/
{

#ifdef CHECKOUT
   printf("\nquerya ");
#endif
```

```
SLOUT("Animal Area Search");
strcat(selcrit,"Animal ");
switch (temptabl) {
   case 0:
      EXEC SQL OPEN C200;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C200");
         ENDCMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C200 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)   {
            SLOUTP("CURSOR C200");
            ENDCMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDCMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C200;
      temptabl = 1;
      break;

   case 1:
      EXEC SQL DELETE FROM QUAL_CIT2;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C201;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C201");
         ENDCMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C201 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            ENDCMSG;
            SLOUTP("CURSOR C201");
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDCMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C201;
      temptabl = 2;
      break;

   case 2:
      EXEC SQL DELETE FROM QUAL_CIT1;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C202;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C202");
         ENDCMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C202 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C202");
            ENDCMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDCMSG;
```

```
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
          qual_entries++;
          }
       EXEC SQL COMMIT WORK;
       EXEC SQL CLOSE C202;
       temptabl = 1;
       break;

    default:
       bad_temp();
    }

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}




int querys()
/*************************************************************************
*                                                                        *
*    querys  --  Create the subset of citations for the titles that      *
*    =======     pertain to the STRUCTURES area                          *
*                                                                        *
*************************************************************************/
{

#ifdef CHECKOUT
   printf("\nquerys ");
#endif
SLOUT("Structures Area Search");
strcat(selcrit,"Struct ");
switch (temptabl) {
   case 0:
      EXEC SQL OPEN C300;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C300");
         ERDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C300 into :entnb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)   {
            ERDOMSG;
            SLOUTP("CURSOR C300");
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:entnb);
         if (sqlca.sqlcode) {
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C300;
      temptabl = 1;
      break;

  case 1:
      EXEC SQL DELETE FROM QUAL_CIT2;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C301;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C301");
         ERDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C301 into :entnb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C301");
            ERDOMSG;
```

```
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C301;
      temptabl = 2;
      break;

   case 2:
      EXEC SQL DELETE FROM QUAL_CIT1;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C302;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C302");
         ENDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C302 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C302");
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C302;
      temptabl = 1;
      break;

   default:
      bad_temp();
   }

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}




int querym()
/*****************************************************************************
*                                                                            *
*    querym  --  Create the subset of citations for the titles that          *
*    ======      pertain to the MODELING area                                *
*                                                                            *
*****************************************************************************/
{

#ifdef CHECKOUT
   printf("\nquerym ");
#endif
SLOUT("Model Area Search");
strcat(selcrit,"Model ");

switch (temptabl) {
   case 0:
      EXEC SQL OPEN C400;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C400");
         ENDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
```

```
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C400 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)   {
            SLOUTP("CURSOR C400");
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CITI (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C400;
      temptabl = 1;
      break;

   case 1:
      EXEC SQL DELETE FROM QUAL_CITI2;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C401;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C401");
         EHDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C401 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)   {
            SLOUTP("CURSOR C401");
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CITI2 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C401;
      temptabl = 2;
      break;

   case 2:
      EXEC SQL DELETE FROM QUAL_CITI1;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C402;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C402");
         EHDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C402 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)   {
            SLOUTP("CURSOR C402");
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CITI (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C402;
      temptabl = 1;
      break;

   default:
```

```
      bad_temp();
   }

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}




int queryw()
/*************************************************************************
*                                                                        *
*    queryw  --  Create the subset of citations for the author(s) whose  *
*    ------      name(s) look like the pattern given in "authornam"      *
*                                                                        *
*************************************************************************/
{
register int i, j;
#ifdef CHECKOUT
   printf("\nqueryw ");
#endif

for (i = 0; i < authornam.len; i++) {
   if (authornam.arr[i] != ' ') break;
   for (j = i; j < authornam.len; j++)
      authornam.arr[j-1] = authornam.arr[j];
   authornam.len--;
   authornam.arr[authornam.len] = '\0';
   #ifdef CHECKOUT
      printf("\n%s (%u)", authornam.arr, authornam.len);
      SLOUTP("Ack");
   #endif
   }
if (!authornam.len) return (int) -1;

strcpy(pattern.arr, authornam.arr);
strcat(pattern.arr,"%");
pattern.len = strlen(pattern.arr);
for (i = 0; i < pattern.len; i++) pattern.arr[i] = toupper(pattern.arr[i]);
SLOUT("Author Search");
strcat(selcrit,pattern.arr);


EXEC SQL OPEN CS0s;
if (sqlca.sqlcode) {
   SLOUTP("CURSOR CS0s");
   ENDCMSG;
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   exit(16);
   }

qual_entries = 0;
if (temptabl == 1) {
   EXEC SQL DELETE FROM QUAL_CIT2;
   }
else {
   EXEC SQL DELETE FROM QUAL_CIT1;
   }
EXEC SQL COMMIT WORK;

EXEC SQL FETCH CS0s into :authornum;
if (sqlca.sqlcode) {
   if (sqlca.sqlcode != SQL_NOF) {
      ENDCMSG;
      SLOUTP("CURSOR CS0s");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      exit(16);
      }
   }
else {
   do {
      #ifdef CHECKOUT
         printf("\nComparing table %d for Author number ", temptabl);
         for (i = 0; i < 5; i++) printf("%c",authornum.arr[i]);
      #endif
      switch (temptabl) {
      case 0:
```

```
      EXEC SQL OPEN C500;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C500");
         ERDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         exit(16);
         }
      for (;;) {
         EXEC SQL FETCH C500 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode) {
            SLOUTP("CURSOR C500");
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         EXEC SQL INSERT INTO QUAL_CITI (entry_num) VALUES (:enumb);
         if (sqlca.sqlcode) {
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         #ifdef CHECKOUT
            printf("\nAccepted ");
            for (i = 0; i < 5; i++) printf("%c",enumb.arr[i]);
         #endif
         EXEC SQL COMMIT WORK;
         if (sqlca.sqlcode) {
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         qual_entries++;
         RENVALS();
         }
      EXEC SQL CLOSE C500;
      break;

   case 1:
      EXEC SQL OPEN C501;
      if (sqlca.sqlcode) {
         ERDOMSG;
         SLOUTP("CURSOR C501");
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      for (;;) {
         EXEC SQL FETCH C501 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C501");
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         EXEC SQL INSERT INTO QUAL_CITI2 (entry_num) VALUES (:enumb);
         if (sqlca.sqlcode) {
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         #ifdef CHECKOUT
            printf("\nAccepted ");
            for (i = 0; i < 5; i++) printf("%c",enumb.arr[i]);
         #endif
         EXEC SQL COMMIT WORK;
         if (sqlca.sqlcode) {
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            exit(16);
            }
         qual_entries++;
         RENVALS();
         }
      EXEC SQL CLOSE C501;
      break;

   case 2:
      EXEC SQL OPEN C502;
      if (sqlca.sqlcode) {
         ERDOMSG;
         SLOUTP("CURSOR C502");
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         exit(16);
```

```
            }
         for (;;) {
            EXEC SQL FETCH C502 into :enumb;
            if (sqlca.sqlcode == SQL_EOF) break;
            if (sqlca.sqlcode)  {
               SLOUTP("CURSOR C502");
               ERDOMSG;
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               exit(16);
               }
            EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
            if (sqlca.sqlcode) {
               ERDOMSG;
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               exit(16);
               }
            #ifdef CHECKOUT
               printf("\nAccepted ");
               for (i = 0; i < 5; i++) printf("%c",enumb.arr[i]);
            #endif
            EXEC SQL COMMIT WORK;
            if (sqlca.sqlcode) {
               ERDOMSG;
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               exit(16);
               }
            qual_entries++;
            NEWVALS();
            }
         EXEC SQL CLOSE C502;
         break;

      default:
         bad_temp();
      }
      EXEC SQL FETCH C50s INTO :authoraum;
      } while (!sqlca.sqlcode);
   if (sqlca.sqlcode != SQL_EOF) {
      ERDOMSG;
      SLOUTP("CURSOR C50s");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      exit(16);
      }
   }
EXEC SQL CLOSE C50s;
#ifdef CHECKOUT
   printf("\nOld table %d", temptabl);
#endif
temptabl = (temptabl % 2) + 1;
#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}



int queryd()
/*************************************************************************
*                                                                        *
*    queryd  --  Create the subset of citations for citations that were  *
*    ======      published between two years                             *
*                                                                        *
*************************************************************************/
{

#ifdef CHECKOUT
   printf("\nqueryd ");
#endif
if (dates1 > dates2) {                  /*  Check on the order of the dates  */
   if (dates2) {
      sprintf(date1.arr,"%4d",dates2);            /* They are out of order */
      sprintf(date2.arr,"%4d",dates1);}
   else {
      sprintf(date1.arr,"%4d",dates1);            /* Only one date entered */
      strcpy(date2.arr,"2000");}
   }
else {
   if (dates2) {                        /*  There is a second date specified */
      sprintf(date1.arr,"%4d",dates1);                /* They are in order */
      sprintf(date2.arr,"%4d",dates2);}
```

```
   else return (int) -1;                /* They are both zero! (or negative) */
   }
date1.len = date2.len = 4;
strcat(selcrit,"/");
strcat(selcrit,date1.arr);
strcat(selcrit,"/");
strcat(selcrit,date2.arr);
SLOUT("Date Search");

#ifdef CHECKOUT
   printf(" Dates %s and %s", date1.arr, date2.arr);
#endif

switch (temptabl) {
   case 0:
      EXEC SQL OPEN C600;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C600");
         EHDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C600 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C600");
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C600;
      temptabl = 1;
      break;

   case 1:
      EXEC SQL DELETE FROM QUAL_CIT2;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C601;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C601");
         EHDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C601 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C601");
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            EHDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C601;
      temptabl = 2;
      break;

   case 2:
      EXEC SQL DELETE FROM QUAL_CIT1;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C602;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C602");
         EHDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
```

```
        for (;;) {
           EXEC SQL FETCH C602 into :enumb;
           if (sqlca.sqlcode == SQL_EOF) break;
           if (sqlca.sqlcode)    {
              SLOUTP("CURSOR C602");
              ERDOMSG;
              SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
              }
           EXEC SQL INSERT INTO QUAL_CITI (entry_num)  VALUES (:enumb);
           if (sqlca.sqlcode) {
              ERDOMSG;
              SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
              }
           qual_entries++;
           }
        EXEC SQL COMMIT WORK;
        EXEC SQL CLOSE C602;
        temptabl = 1;
        break;

     default:
        bad_temp();
     }

 #ifdef CHECKOUT
    printf(" %d entries in table %d", qual_entries, temptabl);
 #endif
 return (int) qual_entries;
 }


 int queryt()
 /*************************************************************************
 *                                                                        *
 *    queryt  --  Create the subset of citations for the titles that      *
 *    ------      tain the phrase given in "title_frag"                   *
 *                                                                        *
 *************************************************************************/
 {
 register int i, j;

 #ifdef CHECKOUT
    printf("\nqueryt ");
 #endif
 titlefrag.len = strlen(titlefrag.arr);
 for (i = 0; i < titlefrag.len; i++) {
    if (titlefrag.arr[i] != ' ') break;
    for (j = i; j < titlefrag.len; j++)
       titlefrag.arr[j-i] = titlefrag.arr[j];
    titlefrag.len--;
    titlefrag.arr[titlefrag.len] = '\0';
    #ifdef CHECKOUT
       printf("\n%u %s", titlefrag.len, titlefrag.arr);
    #endif
    }
 if (!titlefrag.len) return (int) -1;
 strcpy(pattern.arr, "%");
 strcat(pattern.arr, titlefrag.arr);
 strcat(pattern.arr,"%");
 pattern.len = strlen(pattern.arr);
 for (i = 0; i < pattern.len; i++) pattern.arr[i] = toupper(pattern.arr[i]);
 strcat(selcrit,pattern.arr);
 #ifdef CHECKOUT
    printf("\%u %s", pattern.len, pattern.arr);
 #endif

 SLOUT("Title Search");
 switch (temptabl) {
    case 0:
       EXEC SQL OPEN C700;
       if (sqlca.sqlcode) {
          SLOUTP("CURSOR C700");
          ERDOMSG;
          SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
          }
       qual_entries = 0;
       for (;;) {
          EXEC SQL FETCH C700 into :enumb;
          if (sqlca.sqlcode == SQL_EOF) break;
          if (sqlca.sqlcode)    {
             SLOUTP("CURSOR C700");
```

```
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C700;
      temptabl = 1;
      break;

   case 1:
      EXEC SQL DELETE FROM QUAL_CIT2;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C701;
      if (sqlca.sqlcode) {
         SLOUTP("CURSOR C701");
         ENDOMSG;
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C701 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C701");
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C701;
      temptabl = 2;
      break;

   case 2:
      EXEC SQL DELETE FROM QUAL_CIT1;
      EXEC SQL COMMIT WORK;
      EXEC SQL OPEN C702;
      if (sqlca.sqlcode) {
         ENDOMSG;
         SLOUTP("CURSOR C702");
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      qual_entries = 0;
      for (;;) {
         EXEC SQL FETCH C702 into :enumb;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode)  {
            SLOUTP("CURSOR C702");
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
         if (sqlca.sqlcode) {
            ENDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         qual_entries++;
         }
      EXEC SQL COMMIT WORK;
      EXEC SQL CLOSE C702;
      temptabl = 1;
      break;

   default:
      bad_temp();
   }

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
```

```
return (int) qual_entries;
}


int bad_temp()
{
char msg[60];
sprintf(msg,"Bad temporary table identifier %5d used!", temptabl);
SLOUTER(msg);
exit(16);
}
/*************************************************************************
*                                                                        *
*      queryal.pc -- Routines for citation database retrieval in the     *
*      ==========     animal area using the taxonomy table               *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS    */
#include <stdio.h>
#include <ctype.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;
VARCHAR foo[14];                        /* Dummy for uniqueness fetches   */

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                       /*  Standard ASAN Header File     */


EXEC SQL DECLARE CA001 CURSOR FOR
                       SELECT anml_id, anml_name
                         FROM headquarters.animal_list
                        WHERE anml_name LIKE :animal;

EXEC SQL DECLARE CA002 CURSOR FOR
                       SELECT anml_name, anml_id
                         FROM headquarters.animal_list
                        WHERE anml_id LIKE :pattern
                        ORDER BY anml_id, anml_name;


EXEC SQL DECLARE CA100 CURSOR FOR
                       SELECT anml_id
                         FROM headquarters.animal_list
                        WHERE anml_id LIKE :pattern;

EXEC SQL DECLARE CA110 CURSOR FOR
                       SELECT e.entry_num
                         FROM headquarters.animal_effects e
                        WHERE anml_id = :nextanml;

EXEC SQL DECLARE CA111 CURSOR FOR
                       SELECT e.entry_num
                         FROM headquarters.animal_effects e, qual_cit1 q
                        WHERE e.entry_num = q.entry_num
                          AND anml_id = :nextanml;

EXEC SQL DECLARE CA112 CURSOR FOR
                       SELECT e.entry_num
                         FROM headquarters.animal_effects e, qual_cit2 q
                        WHERE e.entry_num = q.entry_num
                          AND anml_id = :nextanml;



int mkanmlst()
/*************************************************************************
*                                                                        *
```

```
*    mkanmlst --  Create the subset of animals that belong to the next      *
*    ========     level down in the taxonomy table                           *
*                                                                            *
******************************************************************************/
{
char *clock();
FILE *txtblkf, *fopen();
int account, depth, i, lowdepth;

static char *dots[] = {" ", ". ", ".. ", "... "};

#ifdef CHECKOUT
   printf("\nmkanmlst ");
#endif

if ( (txtblkf = fopen("txtblk\\varaniml.txt","w")) == NULL) {
   SLOUTP("Error creating next level help window");
   fprintf(dante, "\n%s could not open animal textblock file", clock());
   return (int) DUPLICATE_OBJECT;
   }

EXEC SQL OPEN CA001;                        /* Find the "object" */
if (sqlca.sqlcode) {
   ERDCMSG;
   SLOUTP("CURSOR CA001");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   }

account = 0;
lowdepth = 16;
for (;;) {
   EXEC SQL FETCH CA001 INTO :anml_id, :thisbeast;
   if (sqlca.sqlcode == SQL_EOF) break;
   if (sqlca.sqlcode)    {
      ERDCMSG;
      SLOUTP("FETCH CA001");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      }
   thisbeast.arr[thisbeast.len] = '\0';

   #ifdef CHECKOUT
      anml_id.arr[anml_id.len] = '\0';
      printf("\nAnimal_id is %s", anml_id.arr, thisbeast.arr);
   #endif

   fprintf(txtblkf, "Animals below %s are:\n\n", thisbeast.arr);
   sprintf(workspace.arr,"Looking for Animals below %s ", thisbeast.arr);
   SLOUT(workspace.arr);

   for (i = 6; i > 0;) {                    /* Where are we? */
      if ((anml_id.arr[i] != '0') || (anml_id.arr[i-1] != '0')) break;
      i -= 2;}
   depth = i;
   if (depth < olddepth) continue;
   if (lowdepth > depth) lowdepth = depth;

   #ifdef CHECKOUT
      printf(" depth  %d", depth);
   #endif
   if (depth < 6) {                         /* You can't go further than that */
      for (i = 0; i <= depth; i++)
         pattern.arr[i] = anml_id.arr[i];
      pattern.arr[i] = '%';
      pattern.len = i+1;

      #ifdef CHECKOUT
         pattern.arr[pattern.len] = '\0';
         printf("  Pattern is %s", pattern.arr);
      #endif

      EXEC SQL OPEN CA002;                  /* What else is there like it */
      if (sqlca.sqlcode) {
         ERDCMSG;
         SLOUTP("CURSOR CA002");
         SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
         }
      for (;;) {
         EXEC SQL FETCH CA002 INTO :anmlname, :nextanml;
         if (sqlca.sqlcode) break;

      #ifdef CHECKOUT
         nextanml.arr[nextanml.len] = '\0';
```

```
        anmlname.arr[anmlname.len] = '\0';
        printf("\nFound %s %s", nextanml.arr, anmlname.arr);
    #endif
        if (!strnicmp(nextanml.arr, anml_id.arr, 10)) continue;

        for (i = 6; i > 0;) {
        if ((nextanml.arr[i] != '0') || (nextanml.arr[i-1] != '0')) break;
        i -= 2;}

    #ifdef CHECKOUT
       printf(" included");
    #endif
        anmlname.arr[anmlname.len] = '\0';
        fprintf(txtblkf, "%s %s\n", dots[i/2], anmlname.arr);
        acount++;
        }
     if (sqlca.sqlcode != SQL_EOF) {
        ENDOMSG;
        fprintf(dante,
                "\n%s %s\n\t\t\t  looking for animals like %s",
                clock(), sqlca.sqlerrm.sqlerrmc, animal.arr);
        }
     fprintf(txtblkf,"\n\n");
     EXEC SQL CLOSE CA002;
     }
   }

if (sqlca.sqlcode != SQL_EOF) {
   ENDOMSG;
   fprintf(dante,
           "\n%s %s\n\t\t\t  looking for animals like %s",
           clock(), sqlca.sqlerrm.sqlerrmc, animal.arr);}
fprintf(txtblkf,"      ==  END  ==\n");
fclose(txtblkf);
sprintf(workspace.arr,"%d entries of %d for %s", acount, animals, animal.arr);
#ifdef CHECKOUT
SLOUTP(workspace.arr);
#endif
if ((lowdepth != 16) && (lowdepth > olddepth)) olddepth = lowdepth;
if (olddepth < 8) olddepth += 2;
return (int) acount;
}


int queryall()
/*****************************************************************************
*                                                                           *
*      queryall  --  Find the animal of "species 1" in the taxonomy table   *
*      ==========                                                           *
*                                                                           *
*****************************************************************************/
{
int i;

#ifdef CHECKOUT
   printf("\nqueryall ");
#endif

olddepth = 0;                    /*  Current Search depth is zero */
anml_id.arr[0] = '\0';           /*  Current animal not yet defined */
anml_id.len    =   0 ;

species1.len = strlen(species1.arr);
for (i = 0; i < species1.len; i++) {
   animal.arr[i] = species1.arr[i];
   if (isspace(species1.arr[i])) break;
   }
if (i == 0) return (int) 0;            /*  Animal not entered */

animal.arr[i] = '%'; i++;
animal.arr[i] = '\0';
animal.len = i;
if (animals = mkanmlst()) {                        /* Find the second level  */
   NEW_SCREEN("akeyalt");
   ADD_WINDOW("spec2ntry", 5, 1);
   return (int) 0; }
}


int queryal2()
```

```
/*************************************************************************
*                                                                        *
*      query12  --  Find the animal of "species 2" in the taxonomy table *
*      ---------                                                         *
*                                                                        *
*************************************************************************/
{
int count, i;

#ifdef CHECKOUT
   printf("\nquery12 ");
#endif


species2.len = strlen(species2.arr);
for (i = 0; i < species2.len; i++) {
   animal.arr[i] = species2.arr[i];
   if (isspace(species2.arr[i])) break;
   }
if (i != 0) {                        /*  Animal entered ?  */
   animal.arr[i] = '%'; i++;
   animal.arr[i] = '\0';
   animal.len = i;
   if (count = mkanmlst()) {
      animals = count;
      REMOVE_WINDOW();
      NEW_SCREEN("akeyalt");
      ADD_WINDOW("spec3ntry", 6, 1);
      return (int) 0; }
   }
REMOVE_WINDOW();
NEW_SCREEN("akeysrch");
}




int query13()
/*************************************************************************
*                                                                        *
*      query13  --  Find the animal of "species 3" in the taxonomy table *
*      ---------                                                         *
*                                                                        *
*************************************************************************/
{
int count, i;

#ifdef CHECKOUT
   printf("\nquery13 ");
#endif

species3.len = strlen(species3.arr);
for (i = 0; i < species3.len; i++) {
   animal.arr[i] = species3.arr[i];
   if (isspace(species3.arr[i])) break;
   }
if (i != 0) {                        /*  Animal entered ?  */
   animal.arr[i] = '%'; i++;
   animal.arr[i] = '\0';
   animal.len = i;
   if (count = mkanmlst()) {
      animals = count;
      REMOVE_WINDOW();
      NEW_SCREEN("akeyalt");
      ADD_WINDOW("spec4ntry", 7, 1);
      return (int) 0;}
   }
REMOVE_WINDOW();
NEW_SCREEN("akeysrch");
}


int query14()
/*************************************************************************
*                                                                        *
*      query14  --  Find the animal of "species 4" in the taxonomy table *
*      ---------                                                         *
*                                                                        *
*************************************************************************/
{
int count, i;
```

```
#ifdef CHECKOUT
   printf("\nquery14 ");
#endif


for (i = 0; i < species4.len; i++) {
   animal.arr[i] = species3.arr[i];
   if (isspace(species4.arr[i])) break;
   }
if (i != 0) {                    /* Animal entered ?  */
   animal.arr[i] = '%'; i++;
   animal.arr[i] = '\0';
   animal.len = i;
   if (count = mkanmlst()) {
      animals = count;
      }
   }
REMOVE_WINDOW();
NEW_SCREEN("akeysrch");
}



int asrch002()
/*************************************************************************
*                                                                        *
*     asrch002  -  Search based on entries where an Animal is selected   *
*     ========                                                           *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nasrch002 ");
#endif
if (queryw()) {
   NEWVALS();
   if (queryt()) {
      NEWVALS();
      if (queryd()) {
         NEWVALS();
         query10();
         }
      }
   }
NEWVALS();
if (!qual_entries) {
   SLOUTRF("No entries matching these criteria were found");
   return (int) 0;}
else {
   if (!mclisto()) {
      if (!shwartcit()) {
         NEW_SCREEN("citdspl");
         ADD_WINDOW("citdispaction", 19, 1);}
      }
   return (int) 0;}
}


int query10()
/*************************************************************************
*                                                                        *
*     query10  -  Select citations that deal with specific animal(s)     *
*     ========                                                           *
*                                                                        *
*************************************************************************/
{
int i;

#ifdef CHECKOUT
   printf("\nquery10 ");
#endif

if (olddepth == 0) return (int) qual_entries;
strncpy(pattern.arr, anml_id.arr, (unsigned int) (olddepth-2));
pattern.arr[olddepth-1] =  '%';
pattern.arr[olddepth] = '\0';
pattern.len = strlen(pattern.arr);
strcat(selcrit,pattern.arr);

#ifdef CHECKOUT
   anml_id.arr[anml_id.len] = '\0';
```

```
   printf("\n%s, use %u characters for %s", anml_id.arr, i, pattern.arr);
#endif


SLOUT("Specific Animal Search");


EXEC SQL OPEN CA100;
#ifdef CHECKOUT
   if (sqlca.sqlcode) {
      SLOUTP("CURSOR CA100");
      ERDOMSG;
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      }
#endif
qual_entries = 0;

EXEC SQL FETCH CA100 into :nextanml;
if (sqlca.sqlcode == SQL_EOF) {
   EXEC SQL CLOSE CA100;
   return (int) qual_entries;}

if (sqlca.sqlcode) {
   ERDOMSG;
   SLOUTP("FETCH CA100");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   }
else {
   if (temptabl == 1) {
      EXEC SQL DELETE FROM QUAL_CIT2;}
   else if (temptabl == 2) {
      EXEC SQL DELETE FROM QUAL_CIT1;}
   }
   EXEC SQL COMMIT WORK;

#ifdef CHECKOUT
   nextanml.arr[nextanml.len] = '\0';
   printf("\nTaxon %s", nextanml.arr);
#endif

do {
   switch (temptabl) {
      case 0:
         EXEC SQL OPEN CA110;
         if (sqlca.sqlcode) {
            SLOUTP("CURSOR CA110");
            ERDOMSG;
            SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
            }
         for (;;) {
            EXEC SQL FETCH CA110 into :enumb;
            if (sqlca.sqlcode == SQL_EOF) break;
            if (sqlca.sqlcode) {
               SLOUTP("FETCH CA110");
               ERDOMSG;
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               }
            EXEC SQL SELECT entry_num
                        FROM qual_cit1
                        INTO :foo
                       WHERE entry_num = :enumb;
            if (sqlca.sqlcode == SQL_EOF) {
               EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
               if (sqlca.sqlcode) {
                  SLOUTP("INSERT Table 1");
                  ERDOMSG;
                  SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
                  }
               #ifdef CHECKOUT
                  enumb.arr[10] = '\0';
                  printf("\n%s qualifies",enumb.arr);
               #endif
               EXEC SQL COMMIT WORK;
               qual_entries++;
               REWVALS();
               }
            }
         EXEC SQL CLOSE CA110;
         break;

      case 1:
         EXEC SQL OPEN CA111;
```

```
            if (sqlca.sqlcode) {
               ERDOMSG;
               SLOUTP("CURSOR CA111");
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               }
            for (;;) {
               EXEC SQL FETCH CA111 into :enumb;
               if (sqlca.sqlcode == SQL_EOF) break;
               if (sqlca.sqlcode)  {
                  SLOUTP("FETCH CA111");
                  ERDOMSG;
                  SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
                  }
               EXEC SQL SELECT entry_num
                           FROM qual_cit2
                           INTO :foo
                          WHERE entry_num = :enumb;
               if (sqlca.sqlcode == SQL_EOF) {
                  EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
                  if (sqlca.sqlcode) {
                     ERDOMSG;
                     SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
                     }
                  #ifdef CHECKOUT
                     enumb.arr[10] = '\0';
                     printf("\n%s qualifies",enumb.arr);
                  #endif
                  EXEC SQL COMMIT WORK;
                  qual_entries++;
                  NEWVALS();
                  }
               }
            EXEC SQL CLOSE CA111;
            break;

         case 2:
            EXEC SQL OPEN CA112;
            if (sqlca.sqlcode) {
               ERDOMSG;
               SLOUTP("CURSOR CA112");
               SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
               }
            for (;;) {
               EXEC SQL FETCH CA112 into :enumb;
               if (sqlca.sqlcode == SQL_EOF) break;
               if (sqlca.sqlcode)  {
                  SLOUTP("FETCH CA112");
                  ERDOMSG;
                  SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
                  }
               EXEC SQL SELECT entry_num
                           FROM qual_cit1
                           INTO :foo
                          WHERE entry_num = :enumb;
               if (sqlca.sqlcode == SQL_EOF) {
                  EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
                  if (sqlca.sqlcode) {
                     ERDOMSG;
                     SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
                     }
                  #ifdef CHECKOUT
                     enumb.arr[10] = '\0';
                     printf("\n%s qualifies",enumb.arr);
                  #endif
                  EXEC SQL COMMIT WORK;
                  qual_entries++;
                  NEWVALS();
                  }
               }
            EXEC SQL CLOSE CA112;
            break;

         default:
            bad_temp();
         }
      EXEC SQL FETCH CA100 INTO :nextaxml;
      } while (!sqlca.sqlcode);
   if (sqlca.sqlcode != SQL_EOF) {
      ERDOMSG;
      SLOUTP("FETCH CA100");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      }
```

```
temptabl = (temptabl % 2) + 1;
EXEC SQL CLOSE CA100;
#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}

/*****************************************************************************
*                                                                            *
*      queryh1.pc -- Routines for citation database retrieval in the         *
*      ==========    human area using keyword set 1                          *
*                                                                            *
*     This file contains:                                                    *
*                                                                            *
*****************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS     */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files        */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                       /*  Standard ASAN Header File      */

static char far *hcursors[] = {

"SELECT entry_num FROM headquarters.citation_search WHERE h_annoyanc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
 WHERE s.entry_num = q.entry_num AND s.h_annoyanc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
 WHERE s.entry_num = q.entry_num AND s.h_annoyanc = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_psychlgy = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
 WHERE s.entry_num = q.entry_num AND s.h_psychlgy = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
 WHERE s.entry_num = q.entry_num AND s.h_psychlgy = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_physical = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
 WHERE s.entry_num = q.entry_num AND s.h_physical = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
 WHERE s.entry_num = q.entry_num AND s.h_physical = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_sleep = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
 WHERE s.entry_num = q.entry_num AND s.h_sleep    = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
 WHERE s.entry_num = q.entry_num AND s.h_sleep    = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_speech = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
 WHERE s.entry_num = q.entry_num AND s.h_speech   = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
 WHERE s.entry_num = q.entry_num AND s.h_speech   = 'T'",

"SELECT entry_num FROM headquarters.citation_search WHERE h_perfrmnc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit1 q\
 WHERE s.entry_num = q.entry_num AND s.h_perfrmnc = 'T'",

"SELECT q.entry_num FROM headquarters.citation_search s, qual_cit2 q\
 WHERE s.entry_num = q.entry_num AND s.h_perfrmnc = 'T'"};
```

```
int hoursoffset;


int queryh1()
/*************************************************************************
*                                                                        *
*    queryh1 --  Create the subset of citations for the titles that      *
*    =======     pertain to the HUMAN area and also select the effect    *
*                at level 1                                              *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh1 ");
#endif
SLOUT("Human Area");
switch (aff2srch[0]) {
   case 0: break;
   case 1: queryh11(); break;
   case 2: queryh12(); break;
   case 3: queryh13(); break;
   case 4: queryh14(); break;
   case 5: queryh15(); break;
   case 6: queryh16(); break;
#ifdef CHECKOUT
   default: sprintf(workspace.arr,"Invalid Human Effect %d", aff2srch[0]);
            SLOUTP(workspace.arr);
#endif
   }
return (int) qual_entries;
}


int queryh11()
/*************************************************************************
*                                                                        *
*    queryh11 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also ANNOYANCE           *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh11 ");
#endif
strcat(selcrit,"Human Ann");
hoursoffset = 0;
return (int) queryhlx();
}


int queryh12()
/*************************************************************************
*                                                                        *
*    queryh12 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also PSYCHOLOGICAL HEALTH *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh12 ");
#endif
strcat(selcrit,"Psychol");

hoursoffset = 3;
return (int) queryhlx();
}


int queryh13()
/*************************************************************************
*                                                                        *
*    queryh13 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also PHYSICAL HEALTH     *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh13 ");
#endif
```

```
strcat(selcrit,"Phys. Health");
hoursoffset = 6;
return (int) queryhlx();
}


int queryh14()
/*************************************************************************
*                                                                        *
*    queryh14 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also SLEEP INTERFERENCE  *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh14 ");
#endif
strcat(selcrit,"Human Sleep");
hoursoffset = 9;
return (int) queryhlx();
}


int queryh15()
/*************************************************************************
*                                                                        *
*    queryh15 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also SPEECH INTERFERENCE *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh15 ");
#endif
strcat(selcrit,"Speech");
hoursoffset = 12;
return (int) queryhlx();
}


int queryh16()
/*************************************************************************
*                                                                        *
*    queryh16 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also TASK PERFORMANCE    *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh16 ");
#endif
strcat(selcrit,"Human Perf");
hoursoffset = 18;
return (int) queryhlx();
}


int queryhlx()
/*************************************************************************
*                                                                        *
*    queryhlx --  Actually do the search using hoursoroffset             *
*    ========                                                            *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf(" queryhlx %d+%d ", hoursoffset, temptabl);
#endif

strcpy(sqlstmnt.arr, hoursors[hoursoffset+temptabl]);
sqlstmnt.len = strlen(sqlstmnt.arr);

EXEC SQL PREPARE D1 FROM :sqlstmnt;
if (sqlca.sqlcode) {
   ERDOMSG;
   SLOUTP("Prepare CMD");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   exit(16);
```

```
   }

EXEC SQL DECLARE CED CURSOR FOR D1;

EXEC SQL OPEN CED;
if (sqlca.sqlcode) {
   ERDOMSG;
   SLOUTP("Open CED");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   }

qual_entries = 0;
for (;;) {
   EXEC SQL FETCH CED into :enumb;
   if (sqlca.sqlcode == SQL_EOF) break;
   if (sqlca.sqlcode)   {
      ERDOMSG;
      SLOUTP("Fetch CED");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      }

   if (temptabl == 1) {
      EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
      }
   else {
      EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
      }
   if (sqlca.sqlcode) {
       ERDOMSG;
       SLOUTP("Insert after CED");
       SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
       exit(16);
       }
   qual_entries++;
   NEWVALS();
   EXEC SQL COMMIT WORK;
   }
EXEC SQL CLOSE CED;
temptabl = (temptabl % 2) + 1;

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}


/*************************************************************************
*                                                                        *
*      queryh2.pc -- Routines for citation database retrieval in the     *
*      ----------     human area using keyword set 2                     *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*************************************************************************/


#include <process.h>                  /*  Header for calls to MS-DOS     */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern    /*  Switch for header files        */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                     /*  Standard ASAN Header File      */

static char far *h2cursors[] = {
"SELECT d.entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search s WHERE d.entry_num = s.entry_num\
 AND d.aircraft = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.aircraft = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
```

```
 headquarters.citation_search s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.aircraft = 'T'",

"SELECT d.entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 d.ns_blast = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_blast   = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_blast   = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 d.ns_seismic = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_seismic = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_seismic = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 d.ns_sonicbm = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_sonicbm = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_sonicbm = 'T'",

"SELECT d.entry_num  FROM headquarters.citation_details d,\
 headquarters.citation_search  s  WHERE d.entry_num = s.entry_num\
 AND d.ns_terrain = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_terrain = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.ns_terrain = 'T'",

"SELECT d.entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 d.traffic = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num  AND d.traffic  = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.traffic = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s  WHERE d.entry_num = s.entry_num\
 d.wind_nse  = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num  AND d.wind_nse   = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.wind_nse   = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s  WHERE d.entry_num = s.entry_num\
 d.othr_nse   = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.othr_nse   = 'T'",
```

```
"SELECT q.entry_num  FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.othr_nse   = 'T'");


int cursoroffset;



int queryh2()
/*****************************************************************************
*                                                                            *
*    queryh2 --  Create the subset of citations for the titles that          *
*    ========    pertain to the HUMAN area and also select the effect        *
*                at level 2                                                  *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh2 ");
#endif
SLOUT("Human Area");
switch (eff2srch[1]) {
   case 0: break;
   case 1: queryh21(); break;
   case 2: queryh22(); break;
   case 3: queryh23(); break;
   case 4: queryh24(); break;
   case 5: queryh25(); break;
   case 6: queryh26(); break;
   case 7: queryh27(); break;
   case 8: queryh28(); break;
#ifdef CHECKOUT
   default: sprintf(workspace.arr,"Invalid Human Effect %d", eff2srch[1]);
            SLOUTP(workspace.arr);
#endif
   }
return (int) qual_entries;
}


int queryh21()
/*****************************************************************************
*                                                                            *
*    queryh21 --  Create the subset of citations for the titles that         *
*    ========     pertain to the HUMAN area and also Aircraft Noise          *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh21 ");
#endif
strcat(selcrit,"Aircraft");
cursoroffset = 0;
return (int) queryh2x();
}


int queryh22()
/*****************************************************************************
*                                                                            *
*    queryh22 --  Create the subset of citations for the titles that         *
*    ========     pertain to the HUMAN area and also Blast Noise             *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh22 ");
#endif
strcat(selcrit,"Blast");
cursoroffset = 3;
return (int) queryh2x();
}
```

```
int queryh23()
/*************************************************************************
*                                                                        *
*    queryh23 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also Seismic             *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh23 ");
#endif
strcat(selcrit,"Seismic");
cursoroffset = 6;
return (int) queryh2x();
}


int queryh24()
/*************************************************************************
*                                                                        *
*    queryh24 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also Sonic Boom          *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh24 ");
#endif
strcat(selcrit,"Boom");
cursoroffset = 9;
return (int) queryh2x();
}


int queryh25()
/*************************************************************************
*                                                                        *
*    queryh25 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also Terrain             *
*                                                                        *
*                                                                        *
*     Revision History:                                                  *
*     1.00  02/04/88   mhr      1. Creation                              *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh25 ");
#endif
strcat(selcrit,"Terrain");
cursoroffset = 12;
return (int) queryh2x();
}


int queryh26()
/*************************************************************************
*                                                                        *
*    queryh26 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also Traffic             *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh26 ");
#endif
strcat(selcrit,"Traffic");
cursoroffset = 15;
return (int) queryh2x();
}


int queryh27()
/*************************************************************************
*                                                                        *
*    queryh27 --  Create the subset of citations for the titles that     *
*    ========     pertain to the HUMAN area and also Wind Noise          *
*                                                                        *
```

```
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh27 ");
#endif
strcat(selcrit,"Wind");
cursoroffset = 18;
return (int) queryh2x();
}


int queryh28()
/*************************************************************************
*                                                                        *
*    queryh28 --  Create the subset of citations for the titles that     *
*    ---------    pertain to the HUMAN area and also Other Noise         *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh28 ");
#endif
strcat(selcrit,"Other");
cursoroffset = 21;
return (int) queryh2x();
}


int queryh2x()
/*************************************************************************
*                                                                        *
*    queryh1x --  Actually do the search using hcursoroffset             *
*    --------                                                            *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
   printf(" queryh1x %d+%d ", cursoroffset, temptabl);
#endif

strcpy(sqlstmnt.arr, h2cursors[cursoroffset+temptabl]);
sqlstmnt.len = strlen(sqlstmnt.arr);


EXEC SQL PREPARE D1 FROM :sqlstmnt;
if (sqlca.sqlcode) {
   ERDOMSG;
   SLOUTP("Prepare CED");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   exit(16);
   }

EXEC SQL DECLARE CED CURSOR FOR D1;

EXEC SQL OPEN CED;
if (sqlca.sqlcode) {
   ERDOMSG;
   SLOUTP("Open CED");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   }

qual_entries = 0;
for (;;) {
   EXEC SQL FETCH CED into :enumb;
   if (sqlca.sqlcode == SQL_EOF) break;
   if (sqlca.sqlcode)   {
      ERDOMSG;
      SLOUTP("Fetch CED");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      }

   if (temptabl == 1) {
      EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
      }
   else {
      EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
      }
   if (sqlca.sqlcode) {
       ERDOMSG;
       SLOUTP("Insert after CED");
```

```
        SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
        exit(16);
        }
    qual_entries++;
    EXEC SQL COMMIT WORK;
    NEWVALS();
    }
EXEC SQL CLOSE CRD;
temptabl = (temptabl % 2) + 1;

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}
/*************************************************************************
*                                                                        *
*      queryh3.pc -- Routines for citation database retrieval in the     *
*      ==========     human area using keyword set 3                     *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS    */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                       /*  Standard ASAN Header File     */

static char far *h3cursors[] = {

"SELECT d.entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 AND d.field_expt = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.field_expt = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.field_expt = 'T'",

"SELECT d.entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 d.lab_expment = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.lab_expment = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.lab_expment = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s WHERE d.entry_num = s.entry_num\
 review_art = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.review_art = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s  WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.review_art = 'T'",

"SELECT entry_num FROM headquarters.citation_details d,\
 headquarters.citation_search  s  WHERE d.entry_num = s.entry_num\
 propos1_ar = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit1 q,\
```

```
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.proposl_ar = 'T'",

"SELECT q.entry_num FROM headquarters.citation_details d, qual_cit2 q,\
 headquarters.citation_search  s WHERE s.entry_num = q.entry_num\
 AND d.entry_num = q.entry_num AND d.proposl_ar = 'T'");


int cursoroffset;




int queryh3()
/*****************************************************************************
*                                                                            *
*    queryh3 --  Create the subset of citations for the titles that          *
*    ========    pertain to the HUMAN area and also select the effect        *
*                at level 3                                                  *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh3 ");
#endif
ELOUT("Human Area");
switch (eff2srch[2]) {
   case 0: break;
   case 1: queryh31(); break;
   case 2: queryh32(); break;
   case 3: queryh33(); break;
   case 4: queryh34(); break;
#ifdef CHECKOUT
   default: sprintf(workspace.arr,"Invalid Human Effect %d", eff2srch[2]);
            ELOUTP(workspace.arr);
#endif
   }
return (int) qual_entries;
}


int queryh31()
/*****************************************************************************
*                                                                            *
*    queryh31 --  Create the subset of citations for the titles that         *
*    =========    pertain to the HUMAN area and also Field Experiment        *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh31 ");
#endif
strcat(selcrit,"Human Fld");
cursoroffset = 0;
return (int) queryh3x();
}


int queryh32()
/*****************************************************************************
*                                                                            *
*    queryh32 --  Create the subset of citations for the titles that         *
*    =========    pertain to the HUMAN area and also Lab experiment          *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh32 ");
#endif
strcat(selcrit,"Human Lab");
cursoroffset = 3;
return (int) queryh3x();
}


int queryh33()
/*****************************************************************************
*                                                                            *
```

```
*    queryh33 --  Create the subset of citations for the titles that        *
*    ========     pertain to the HUMAN area and also Review Article          *
*                                                                            *
******************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh33 ");
#endif
strcat(selcrit,"Human Rev");
cursoroffset = 6;
return (int) queryh3x();
}


int queryh34()
/******************************************************************************
*                                                                            *
*    queryh34 --  Create the subset of citations for the titles that        *
*    ========     pertain to the HUMAN area and also Theoretical             *
*                                                                            *
******************************************************************************/
{
#ifdef CHECKOUT
   printf("\nqueryh34 ");
#endif
strcat(selcrit,"Human Theor");
cursoroffset = 9;
return (int) queryh3x();
}


int queryh3x()
/******************************************************************************
*                                                                            *
*    queryh3x --  Actually do the search using hcursoroffset                 *
*    ========                                                                *
*                                                                            *
******************************************************************************/
{
#ifdef CHECKOUT
   printf(" queryh3x %d+%d ", cursoroffset, temptabl);
#endif

strcpy(sqlstmnt.arr, h3cursors[cursoroffset+temptabl]);
sqlstmnt.len = strlen(sqlstmnt.arr);

EXEC SQL PREPARE D1 FROM :sqlstmnt;
if (sqlca.sqlcode) {
   EXECMSG;
   SLOUTP("Prepare CED");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   exit(16);
   }

EXEC SQL DECLARE CED CURSOR FOR D1;

EXEC SQL OPEN CED;
if (sqlca.sqlcode) {
   EXECMSG;
   SLOUTP("Open CED");
   SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
   }

qual_entries = 0;
for (;;) {
   EXEC SQL FETCH CED into :enumb;
   if (sqlca.sqlcode == SQL_EOF) break;
   if (sqlca.sqlcode)   {
      EXECMSG;
      SLOUTP("Fetch CED");
      SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
      }

   if (temptabl == 1) {
      EXEC SQL INSERT INTO QUAL_CIT2 (entry_num)  VALUES (:enumb);
      }
   else {
      EXEC SQL INSERT INTO QUAL_CIT1 (entry_num)  VALUES (:enumb);
```

```
        }
    if (sqlca.sqlcode) {
        ERDOMSG;
        SLOUTP("Insert after CRD");
        SLOUTP("%s", sqlca.sqlerrm.sqlerrmc);
        exit(16);
        }
    qual_entries++;
    EXEC SQL COMMIT WORK;
    RENVALS();
    }
EXEC SQL CLOSE CRD;
temptabl = (temptabl % 2) + 1;

#ifdef CHECKOUT
   printf(" %d entries in table %d", qual_entries, temptabl);
#endif
return (int) qual_entries;
}
/*************************************************************************
*                                                                        *
*      querypc.pc -- Routines for point of contact retrieval             *
*      ==========                                                        *
*                                                                        *
*     This file contains:                                                *
*                                                                        *
*************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS    */
#include <stdio.h>
#include <ctype.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;
VARCHAR crit1[16];
VARCHAR crit2[46];
VARCHAR crit3[16];
VARCHAR crit4[10];
VARCHAR crit5[26];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                       /*  Standard ASAN Header File     */

static int poc_type;

EXEC SQL DECLARE POC1 CURSOR FOR
                  SELECT first_name, last_name, title, office,
                         agncy_dept, st_add_div, po_box,
                         misc_add, city_base, state, zipcode,
                         mail_code, phone, affiliation,
                         maj_attrib, min_attrib,
                         area, scope_auth
                    FROM superuser.point_of_contact
                   WHERE UPPER(last_name)    LIKE :crit1
                     AND UPPER(min_attrib)   LIKE :crit2
                     AND UPPER(maj_attrib)   LIKE :crit3
                     AND UPPER(affiliation)  LIKE :crit4
                     AND UPPER(state) = :crit5;

EXEC SQL DECLARE POC2 CURSOR FOR
                  SELECT first_name, last_name, title, office,
                         agncy_dept, st_add_div, po_box,
                         misc_add, city_base, state, zipcode,
                         mail_code, phone, affiliation,
                         maj_attrib, min_attrib,
                         area, scope_auth
                    FROM superuser.point_of_contact
                   WHERE UPPER(last_name)    LIKE :crit1
                     AND UPPER(min_attrib)   LIKE :crit2
                     AND UPPER(maj_attrib)   LIKE :crit3
                     AND UPPER(affiliation)  LIKE :crit4
                     AND UPPER(city_base)    LIKE :crit5;
```

```
int pocsrch()
/*************************************************************************
*                                                                        *
*    pocsrch  --  Set up the search in the point of contact database     *
*    ---------                                                           *
*                                                                        *
*************************************************************************/
{
int i, j;

static char *affil[7] = {"CITY%",
                         "COUNTY%",
                         "STATE%",
                         "FEDERAL%",
                         "MILITARY%",
                         "TRIBAL%",
                         "%"};


#ifdef CHECKOUT
   printf("\npocsrch ");
#endif
SLOUT("Point of contact search");

strcpy(crit4.arr,affil[whichafl]);
crit4.len = strlen(crit4.arr);

j = strlen(contnam);
for (i = 0; i < j; i++) crit1.arr[i] = toupper(contnam[i]);
crit1.arr[j] = '\0';
strcat(crit1.arr,"%");
crit1.len = strlen(crit1.arr);

j = strlen(minorst);
for (i = 0; i < j; i++) crit2.arr[i] = toupper(minorst[i]);
crit2.arr[j] = '\0';
strcat(crit2.arr,"%");
crit2.len = strlen(crit2.arr);

j = strlen(majorst);
for (i = 0; i < j; i++) crit3.arr[i] = toupper(majorst[i]);
crit3.arr[j] = '\0';
strcat(crit3.arr,"%");
crit3.len = strlen(crit3.arr);

j = strlen(address);
for (i = 0; i < j; i++) crit5.arr[i] = toupper(address[i]);
crit5.arr[j] = '\0';
crit5.len = strlen(crit5.arr);
if (j  == 2)
   poc_type = 1;
else {
  poc_type = 2;
  strcat(crit5.arr,"%");
  crit5.len++;
  }

if (poc_type == 1) {
   EXEC SQL OPEN POC1;
   if (sqlca.sqlcode) {
      SLOUTP("Cursor POC1");
      expOmsg();
      exit(16);
      }
   }
else {
   EXEC SQL OPEN POC2;
   if (sqlca.sqlcode) {
      SLOUTP("Cursor POC2");
      expOmsg();
      exit(16);
      }
   }
if (nxtpoc()) {
   NEW_SCREEN("contactscream");
   SLOUTP("Sorry, but I cannot find anybody like that...");}
else {
   NEW_SCREEN("poodisplay");
   }
}
```

```
int nxtpoc()
/*************************************************************************
*                                                                        *
*    nxtpoc  --  Find next instance in the point of contact database     *
*    ------                                                              *
*                                                                        *
*************************************************************************/
{
switch (poc_type) {
   case 1:
   EXEC SQL FETCH POC1 INTO :f_name, :l_name, :contitle, :office,
                            :agency_dept, :st_add_div, :po_box,
                            :misc_add, :city_base, :state, :zipcode,
                            :mail_code, :phone, :affiliatio,
                            :major_attrib, :minor_attribute,
                            :area, :scope;
   if (sqlca.sqlcode) {
      if (sqlca.sqlcode == SQL_EOF) {
         poc_type = 0;
         EXEC SQL CLOSE POC1;
         SLOUTP("No more entries!");
         return (int) SQL_EOF; }
      else {
         SLOUTP("Fetch POC1");
         expOmsg();
         exit(16);
         }
      }
   break;

   case 2:
   EXEC SQL FETCH POC2 INTO :f_name, :l_name, :contitle, :office,
                            :agency_dept, :st_add_div, :po_box,
                            :misc_add, :city_base, :state, :zipcode,
                            :mail_code, :phone, :affiliatio,
                            :major_attrib, :minor_attribute,
                            :area, :scope;
   if (sqlca.sqlcode) {
      if (sqlca.sqlcode == SQL_EOF) {
         poc_type = 0;
         EXEC SQL CLOSE POC2;
         SLOUTP("No more entries!");
         return (int) SQL_EOF;}
      else {
         SLOUTP("Fetch POC2");
         expOmsg();
         exit(16);
         }
      }
   break;

   case 0:
         SLOUTP("There REALLY are no more entries!");
         return (int) 0;
   }
f_name.arr[f_name.len] = '\0';
l_name.arr[l_name.len] = '\0';
contitle.arr[contitle.len] = '\0';
office.arr[office.len] = '\0';
agency_dept.arr[agency_dept.len] = '\0';
st_add_div.arr[st_add_div.len] = '\0';
po_box.arr[po_box.len] = '\0';
misc_add.arr[misc_add.len] = '\0';
city_base.arr[city_base.len] = '\0';
state.arr[state.len] = '\0';
zipcode.arr[zipcode.len] = '\0';
mail_code.arr[mail_code.len] = '\0';
phone.arr[phone.len] = '\0';
affiliatio.arr[affiliatio.len] = '\0';
major_attrib.arr[major_attrib.len] = '\0';
minor_attribute.arr[minor_attribute.len] = '\0';
area.arr[area.len] = '\0';
scope.arr[scope.len] = '\0';

RENVALS();
return (int) sqlca.sqlcode;
}


int setaff(i)
/*************************************************************************
*                                                                        *
```

```
*    nxtpoc  --  Find next instance in the point of contact database        *
*    ------                                                                 *
*                                                                           *
*****************************************************************************/
int i;
{
static *affseals[] = {"CITY", "COUNTY", "STATE", "FEDERAL", "MILITARY",
                      "TRIBAL", "NONE"};
whichafl = i;
strcpy(affsaloc, affseals[i]);
NEWVALS();
}
/*****************************************************************************
*                                                                           *
*      queryel.pc -- Routines for citation database retrieval in the        *
*      ----------     structures area using keyword set 1                   *
*                                                                           *
*     This file contains:                                                   *
*                                                                           *
*****************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS    */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE citvars.h;

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

#include "asan.h"                       /*  Standard ASAN Header File     */

int queryel()
{
#ifdef CHECKOUT
   printf("\aqueryel ");
#endif
SLOUTEP("Structures Area Unavailable");
return (int) qual_entries;
}


/*****************************************************************************
*                                                                           *
*    searches.pc  --  Routines that use the query routines to do searches   *
*    ----------                                                             *
*                                                                           *
*     This file contains:                                                   *
*                                                                           *
*     hsrch001  -  Search based on entries on first Human area screen       *
*     asrch001  -  Search based on entries on first Animal area screen      *
*     ssrch001  -  Search based on entries on first Structures area screen  *
*     msrch001  -  Search based on entries on first Modeling area screen    *
*                                                                           *
*     shwnxtcit -  Display the next citation on the screen                  *
*     loadncit  -  Load the next citation from the database FOR DISPLAY     *
*                                                                           *
*****************************************************************************/


#include <process.h>                    /*  Header for calls to MS-DOS    */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern      /*  Switch for header files       */

EXEC SQL BEGIN DECLARE SECTION;         /*  All SQL declarations are in   */
EXEC SQL INCLUDE citvars.h;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                       /*  Standard ASAN Header File     */

/*****************************************************************************
*                                                                           *
*                              SQL Cursors                                  *
*                                                                           *
*****************************************************************************/
```

```
EXEC SQL DECLARE CD00 CURSOR FOR
         SELECT authornum FROM headquarters.author_citation_link
         WHERE  entry_num = :enumb;

EXEC SQL DECLARE CD10 CURSOR FOR
         SELECT q.entry_num, s.suitability, t.title, s.date_pub
         FROM   headquarters.citation_titles t,
                headquarters.citation_search s,
                qual_cit1 q
         WHERE  t.entry_num = q.entry_num
           AND  s.entry_num = q.entry_num;


EXEC SQL DECLARE CD20 CURSOR FOR
         SELECT q.entry_num, s.suitability, t.title, s.date_pub
         FROM   headquarters.citation_titles t,
                headquarters.citation_search s,
                qual_cit2 q
         WHERE  t.entry_num = q.entry_num
           AND  s.entry_num = q.entry_num;

/*****************************************************************************
*                                                                            *
*                              Screen "Memory"                               *
*                                                                            *
*****************************************************************************/
int hsrch000()
{
WHEREAMI(Screen, Window, Datum, Button);
strcpy(oldscreen, Screen);
selcrit[0] = '\0';
}
int asrch000()
{
WHEREAMI(Screen, Window, Datum, Button);
strcpy(oldscreen, Screen);
selcrit[0] = '\0';
}
int ssrch000()
{
WHEREAMI(Screen, Window, Datum, Button);
strcpy(oldscreen, Screen);
selcrit[0] = '\0';
}
int msrch000()
{
WHEREAMI(Screen, Window, Datum, Button);
strcpy(oldscreen, Screen);
selcrit[0] = '\0';
}



int hsrch001()
/*****************************************************************************
*                                                                            *
*     hsrch001  -  Search based on entries on first Human area screen        *
*     ========                                                               *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\nhsrch001 ");
#endif
hsrch000();
if (queryw()) {
   NEWVALS();
   if (queryt()) {
      NEWVALS();
      if (queryd()) {
         NEWVALS();
         queryh();
         }
      }
   }
NEWVALS();
if (!qual_entries) {
   SLOUTRP("No entries matching these criteria were found");
   return (int) 0;}
else {
   if (!mclisto()) {
      if (!showartcit()) {
```

```
            NEW_SCREEN("citdspl");
            ADD_WINDOW("citdispaction", 19, 1);}
         }
      return (int) 0;}
   }


   int hsrch002()
   /*************************************************************************
   *                                                                        *
   *     hsrch002  -  Search based on entries on Human area keyword screen  *
   *     ________                                                           *
   *                                                                        *
   *************************************************************************/
   {
   #ifdef CHECKOUT
   printf("\nhsrch002 ");
   #endif
   if (queryw()) {
      NEWVALS();
      if (queryt()) {
         NEWVALS();
         if (queryd()) {
            NEWVALS();
            if (queryh1()) {
               if (queryh2()) {
                  queryh3();
                  }
               }
            }
         }
      }
   NEWVALS();
   if (!qual_entries) {
      SLOUTPP("No entries matching these criteria were found");
      return (int) 0;}
   else {
      if (!mclisto()) {
         if (!shwartcit()) {
            NEW_SCREEN("citdspl");
            ADD_WINDOW("citdispaction", 19, 1);}
         }
      return (int) 0;}
   }


   int asrch001()
   /*************************************************************************
   *                                                                        *
   *     asrch001  -  Search based on entries on first Animal area screen   *
   *     ________                                                           *
   *                                                                        *
   *************************************************************************/
   {
   #ifdef CHECKOUT
   printf("\nasrch001 ");
   #endif
   if (queryw()) {
      NEWVALS();
      if (queryt()) {
         NEWVALS();
         if (queryd()) {
            NEWVALS();
            queryn();
            }
         }
      }
   NEWVALS();
   if (!qual_entries) {
      SLOUTPP("No entries matching these criteria were found");
      return (int) 0;}
   else {
      if (!mclisto()) {
         if (!shwartcit()) {
            NEW_SCREEN("citdspl");
            ADD_WINDOW("citdispaction", 19, 1);}
         }
      return (int) 0;}
   }
```

```
int ssrch001()
/*****************************************************************************
*                                                                            *
*     ssrch001  -  Search based on entries on first Structures area screen   *
*     ========                                                               *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\n ssrch001 ");
#endif

if (queryw()) {
   NEWVALS();
   if (queryt()) {
      NEWVALS();
      if (queryd()) {
         NEWVALS();
         querys();
         }
      }
   }
NEWVALS();
if (!qual_entries) {
   SLOUTPP("No entries matching these criteria were found");
   return (int) 0;}
else {
   if (!mclisto()) {
      if (!shwartcit()) {
         NEW_SCREEN("citdspl");
         ADD_WINDOW("citdispaction", 19, 1);}
      }
   return (int) 0;}
}



int msrch001()
/*****************************************************************************
*                                                                            *
*     msrch001  -  Search based on entries on first Modeling area screen     *
*     ========                                                               *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\nmsrch001 ");
#endif

if (queryw()) {
   NEWVALS();
   if (queryt()) {
      NEWVALS();
      if (queryd()) {
         NEWVALS();
         querym();
         }
      }
   }
NEWVALS();
if (!qual_entries) {
   SLOUTPP("No entries matching these criteria were found");
   return (int) 0;}
else {
   if (!mclisto()) {
      if (!shwartcit()) {
         NEW_SCREEN("citdspl");
         ADD_WINDOW("citdispaction", 19, 1);}
      }
   return (int) 0;}
}



int mclisto()
/*****************************************************************************
*                                                                            *
*     mclisto  -  Open the cursor for main citation display screen           *
*     ========                                                               *
*                                                                            *
*****************************************************************************/
```

```
{
#ifdef CHECKOUT
   printf("\nmclisto ");
#endif

currcit = 0;
switch (temptabl) {

   case 1:
      EXEC SQL OPEN CD10;
      if (sqlca.sqlcode) {
         SLOUTP("Cursor CD10");
         expOmsg();
         exit(16);}
      break;

   case 2:
      EXEC SQL OPEN CD20;
      if (sqlca.sqlcode) {
         SLOUTP("Cursor CD20");
         expOmsg();
         exit(16);}
      break;

   default:
      bad_temp(temptabl);
   }
return (int) sqlca.sqlcode;
}


int mclistc()
/*****************************************************************************
*                                                                            *
*     mclistc  -  Close the cursor for main citation display screen          *
*     ========                                                               *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
   printf("\nmclistc ");
#endif

switch (temptabl) {
   case 1:
      EXEC SQL CLOSE CD10;
      if (sqlca.sqlcode) {
         SLOUTP("Cursor CD10");
         expOmsg();
         exit(16);}
      break;

   case 2:
      EXEC SQL CLOSE CD20;
      if (sqlca.sqlcode) {
         SLOUTP("Cursor CD20");
         expOmsg();
         exit(16);}
      break;

   default:
      bad_temp(temptabl);
   }
return (int) sqlca.sqlcode;
}


int shwnxtcit()
/*****************************************************************************
*                                                                            *
*     shwnxtcit - Show next citation                                         *
*     =========                                                              *
*                                                                            *
*****************************************************************************/
{
int select;

#ifdef CHECKOUT
   printf("\nshwnxtcit ");
#endif

if (select = loadncit()) {
```

```
      switch (select) {
         case SQL_EOF:

         case SQL_FETCH_OUT_OF_ORDER:
            SLOUTEP("You are at the end of the list");
            REMOVE_WINDOW();
            break;

         default:
            expOmsg();
            exit(16);
         }
      }
   else {
      currcit++;
      RENVALS();
      }
   return (int) sqlca.sqlcode;
   }



   int loadmcit()
   /*****************************************************************************
   *                                                                            *
   *     loadmcit - Load the citation detail display with the next citation     *
   *     ---------                                                              *
   *                                                                            *
   *****************************************************************************/
   {
   int i, j, k, m;

   #ifdef CHECKOUT
      printf("\nloadmcit ");
   #endif

   SLOUT("Retrieving Display Text");

   if (temptabl == 1) {
      EXEC SQL FETCH CD10 INTO :enumb, :suitable, :workspace, :datap;
      if (sqlca.sqlcode)  return (int) sqlca.sqlcode;
      }
   else {
      EXEC SQL FETCH CD20 INTO :enumb, :suitable, :workspace, :datap;
      if (sqlca.sqlcode)  return (int) sqlca.sqlcode;
      }
   enumb.arr[enumb.len] = '\0';
   suitable.arr[1] = '\0';
   workspace.arr[workspace.len] = '\0';
   datap.arr[datap.len] = '\0';
   details_read = 0;

   j = (int) (((float) workspace.len) / ((float) 60)) + 1;
   j = j > 4 ? 4 : j;
   k = 0;
   for ( i = 0; i < j; i++) {
      m = k + 60  > workspace.len ? workspace.len - k : 60;
      strncpy(emtdesc[i].arr, &workspace.arr[60*i], m);
      emtdesc[i].len = m;
      emtdesc[i].arr[m] = '\0';
      k += m;
      }
   for ( ; i < 4;i++) emtdesc[i].arr[0] = '\0';

   EXEC SQL OPEN CD00;
   if (sqlca.sqlcode) {
      SLOUTP("Cursor CD00");
      expOmsg();}
   k = 0;
   for (;;) {
      v40point = &authorlist[k];
      authorlist[k].len = 40;
   /* for (i= 0; i < 40; i++) authorlist[k].arr[i] = ' '; */

      EXEC SQL FETCH CD00 INTO :authornum;
      if (sqlca.sqlcode) break;
      EXEC SQL SELECT author FROM headquarters.author_list
           INTO :v40point WHERE authornum = :authornum;
      if (sqlca.sqlcode) {
         SLOUTP("Author selection...");
         expOmsg();
```

```
        exit(16);}
    authorlist[k].arr[authorlist[k].len] = '\0';
    k++;
    }
 if (sqlca.sqlcode != SQL_EOF) {
    if (sqlca.sqlcode) {
       SLOUTP("Cursor CD00");
       sqpOmsg();
       exit(16);}
    else SLOUTP("Next citation: more authors than screen shows");
    }
 EXEC SQL CLOSE CD00;

 for (i = k; i < 10; i++) {    /* NULL for those not filled this time */
    authorlist[i].arr[0] = '\0';
    authorlist[i].len    =   0 ;}
 if (k > 4)                    /* Truncate if we have more than 5 */
    for (i = 0; i < k; i++) {
       authorlist[k].len = i > 30 ? 30 : i;
       authorlist[k].arr[authorlist[k].len] = '\0';}
 return (int) 0;
 }



 int rescsrch()
 /*************************************************************************
 *                                                                        *
 *     rescsrch - Rescope the search.  If no citations, clear the tables. *
 *     ========                                                           *
 *                                                                        *
 *************************************************************************/
 {
 if (!qual_entries) qsetup();
 NEW_SCREEN(oldscreen);
 }



 int hsrch010()
 /*************************************************************************
 *                                                                        *
 *     hsrch010  -  Setup for entries on Human keyword screen             *
 *     ========                                                           *
 *                                                                        *
 *************************************************************************/
 {
 #ifdef CHECKOUT
 printf("\nhsrch010 ");
 #endif

 hsrch000();
 if (!qual_entries) {aff2srch[0] = aff2srch[1] = aff2srch[2] = 0;}

 NEW_SCREEN("hkeysrch");

 }



 int hsrch011()
 /*************************************************************************
 *                                                                        *
 *     hsrch011  -  Human Effect Descriptor Type Verification             *
 *     ========                                                           *
 *                                                                        *
 *************************************************************************/
 {
 #ifdef CHECKOUT
 printf("\nhsrch011 ");
 #endif
 for(;;) {
    if (desctype.arr[0] == 'A')                    {aff2srch[0] = 1; break;}
    if (desctype.arr[0] == 'P')
        if      (desctype.arr[1] == 'S')           {aff2srch[0] = 2; break;}
        else if (desctype.arr[1] == 'E')           {aff2srch[0] = 3; break;}
    if (desctype.arr[0] == 'S')
        if      (desctype.arr[1] == 'L')           {aff2srch[0] = 4; break;}
        else if (desctype.arr[1] == 'P')           {aff2srch[0] = 5; break;}
```

```
    if (desctype.arr[0] == 'T')                          {aff2srch[0] = 6; break;}
    if ((aff2srch[0]) || (desctype.arr[0] == ' ')) {aff2srch[0] = 0; break;}
    ELOUTB("Not a valid effect");
    UPDATE_DATUM("desctype");
    }
 ADD_WINDOW("noistype", 7, 24);
 NEXT_DATUM("noistype");
 }



 int hsrch012()
 /*****************************************************************************
 *                                                                            *
 *     hsrch012  -  Noise Descriptor Type Verification                        *
 *     ========                                                               *
 *                                                                            *
 *****************************************************************************/
 {
 #ifdef CHECKOUT
 printf("\nhsrch012 ");
 #endif
 for(;;) {
    if (desctype.arr[0] == 'A')                          {aff2srch[1] = 1; break;}
    if (desctype.arr[0] == 'B')                          {aff2srch[1] = 2; break;}
    if (desctype.arr[0] == 'S')
        if      (desctype.arr[1] == 'E')                 {aff2srch[1] = 3; break;}
        else if (desctype.arr[1] == 'O')                 {aff2srch[1] = 4; break;}
    if (desctype.arr[0] == 'T')
        if      (desctype.arr[1] == 'E')                 {aff2srch[1] = 5; break;}
        else if (desctype.arr[1] == 'R')                 {aff2srch[1] = 6; break;}
    if (desctype.arr[0] == 'W')                          {aff2srch[1] = 7; break;}
    if (desctype.arr[0] == 'O')                          {aff2srch[1] = 8; break;}

    if ((aff2srch[0]) || (desctype.arr[0] == ' ')) {aff2srch[1] = 0; break;}
    ELOUTB("Not a valid noise type");
    UPDATE_DATUM("noistype");
    }
 REMOVE_WINDOW();
 ADD_WINDOW("exprtype", 7, 24);
 NEXT_DATUM("expdesc");
 }



 int hsrch013()
 /*****************************************************************************
 *                                                                            *
 *     hsrch013  -  Experimental Type Verification                            *
 *     ========                                                               *
 *                                                                            *
 *****************************************************************************/
 {
 #ifdef CHECKOUT
 printf("\nhsrch013 ");
 #endif
 for(;;) {
    if (desctype.arr[0] == 'F')                          {aff2srch[2] = 1; break;}
    if (desctype.arr[0] == 'L')                          {aff2srch[2] = 2; break;}
    if (desctype.arr[0] == 'R')                          {aff2srch[2] = 3; break;}
    if (desctype.arr[0] == 'T')                          {aff2srch[2] = 4; break;}
    if ((aff2srch[0]) || (desctype.arr[0] == ' ')) {aff2srch[2] = 0; break;}
    ELOUTB("Not valid type");
    UPDATE_DATUM("exprtype");
    }
 REMOVE_WINDOW();
 }



int ssrch010()
/*****************************************************************************
*                                                                            *
*     ssrch010  -  Setup for entries on Structures keyword screen            *
*     ========                                                               *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
```

```
printf("\nsrch010 ");
#endif

srch000();
if (!qual_entries) {aff2srch[0] = aff2srch[1] = aff2srch[2] = 0;}

NEW_SCREEN("skeysrch");

}



int msrch010()
/*****************************************************************************
*                                                                            *
*     msrch010  -  Setup for entries on Modeling keyword screen              *
*                                                                            *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\nmsrch010 ");
#endif

msrch000();
if (!qual_entries) {aff2srch[0] = aff2srch[1] = aff2srch[2] = 0;}

NEW_SCREEN("mkeysrch");

}



int asrch010()
/*****************************************************************************
*                                                                            *
*     asrch010  -  Setup for entries on Animal keyword screen                *
*                                                                            *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\nasrch010 ");
#endif

asrch000();
species1.arr[0] = '\0'; species1.len = 0;
species2.arr[0] = '\0'; species2.len = 0;
species3.arr[0] = '\0'; species3.len = 0;
species4.arr[0] = '\0'; species4.len = 0;
NEWVALS();
NEW_SCREEN("akeysrch");

}
/*****************************************************************************
*                                                                            *
*    textdisplpc  --  Routines that display text fields from "MEMO_FILE"     *
*                                                                            *
*                                                                            *
*     This file contains:                                                    *
*                                                                            *
*                                                                            *
*****************************************************************************/

#include <process.h>                /*  Header for calls to MS-DOS     */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern  /*  Switch for header files        */

EXEC SQL BEGIN DECLARE SECTION;     /*  All SQL declarations are in    */
EXEC SQL INCLUDE citvars.h;
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                   /*  Standard ASAN Header File      */


int displabet()
/*****************************************************************************
*                                                                            *
```

```
 *     dsplabst  --  Display abstract                                        *
 *                                                                           *
 ****************************************************************************/

 {

 #ifdef CHECKOUT
    printf("\n dsplabst ");
 #endif

 if (!details_read) get_details();

 if (abstract.len) {
    strncpy(nextmemo.arr, abstract.arr, 10);
    getmemo();
    NEW_SCREEN("showabs");
    ADD_WINDOW("shwartabs", 21, 1);}
 else {
    SLOUTP("No abstract available");
    }
 }




 int dsplcrit()
 /***************************************************************************
 *                                                                           *
 *     dsplcrit  --  Display critique                                        *
 *                                                                           *
 ****************************************************************************/

 {
 #ifdef CHECKOUT
    printf("\n dsplcrit ");
 #endif

 if (!details_read) get_details();

 switch (critdisp) {

 case 0:
       if (critique1.len == 10) {
          strncpy(nextmemo.arr,critique1.arr, 10);
          getmemo();
          NEW_SCREEN("showabs");
          ADD_WINDOW("shwartrev", 21, 1);
          NEWVALS();
          critdisp = 1;
          return (int) critdisp; }


 case 1:
       if (critique2.len == 10) {
          strncpy(nextmemo.arr,critique2.arr, 10);
          getmemo();
          NEW_SCREEN("showabs");
          ADD_WINDOW("shwartrev", 21, 1);
          NEWVALS();
          critdisp = 2;
          return (int) critdisp; }


 case 2:
       if (critique3.len == 10) {
          strncpy(nextmemo.arr,critique3.arr, 10);
          getmemo();
          NEW_SCREEN("showabs");
          ADD_WINDOW("shwartrev", 21, 1);
          NEWVALS();
          critdisp = 3;
          return (int) critdisp; }


 default:
      SLOUTP("There are no more critical reviews");
      critdisp = 0;
      ADD_WINDOW("shwartabs", 21, 1);
      return (int) critdisp;
 }
 }
```

```
int getmemo()
/*****************************************************************************
 *                                                                           *
 *     getmemo   --   Retrive a memofield from the database                  *
 *                                                                           *
 *****************************************************************************/

{
FILE *fopen(), *memo;
int fwrite();

#ifdef CHECKOUT
   printf(" getmemo ");
#endif

nextmemo.len = 10;

EXEC SQL SELECT memo_text FROM headquarters.memo_file INTO :bigdisplay
                        WHERE block_number = :nextmemo;
if (sqlca.sqlcode) {
   SLOUTP("Cannot find memo block");
   ERDCMSG;
   SLOUTP(sqlca.sqlerrm.sqlerrmc);
   return (int) 0;}


if ((memo = fopen("txtblk\\memotxt.txt","w")) == NULL) {
   SLOUTPP("Error creating memofield textblock");
   return (int) 0;}

fwrite(bigdisplay.arr, bigdisplay.len, 1, memo);
fprintf(memo, "\n\n");
fclose(memo);
return (int) 0;

}


int get_details()
/*****************************************************************************
 *                                                                           *
 *     get_details  --  get the pointers to the detailed memo fields         *
 *                                                                           *
 *****************************************************************************/

{

#ifdef CHECKOUT
   printf(" getdetails ");
#endif


EXEC SQL SELECT ABSTRACT, CRITIQUE_R1, CRITIQUE_R2, CRITIQUE_R3
         FROM headquarters.citation_details WHERE entry_num = :emnmb
         INTO :abstract, :critique1, :critique2, :critique3;

if (sqlca.sqlcode) {
   SLOUTP("Cannot find details");
   ERDCMSG;
   SLOUTP(sqlca.sqlerrm.sqlerrmc);
   return (int) 0;}

#ifdef CHECKOUT
   printf(" Abs %d, Crit %d, %d, %d", abstract.len, critique1.len,
            critique2.len, critique3.len);
#endif


critdisp = 0;
details_read = 1;

}
/*****************************************************************************
 *                                                                           *
 *     ... =  mm* -  All user information for a particular MTR               *
 *                                                                           *
 *****************************************************************************/

#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern    /*  Switch for header files       */
```

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL BEGIN DECLARE SECTION;          /* All SQL declarations are in   */
EXEC SQL INCLUDE hostvars.h;             /* these header files            */
EXEC SQL INCLUDE fmharris.h;

EXEC SQL END DECLARE SECTION;
#include "asan.h"                        /* Standard ASAN Header File     */


int malisto()
/*****************************************************************************
*                                                                            *
*      malisto   ---   Open cursor S2 for a list of user information         *
*      =======          of navigation points on a particular MFR            *
*                                                                            *
*     Routine executes an open cursor command for cursor S2 and then        *
*     returns to the calling program with the ORACLE status code.            *
*                                                                            *
*     Note:   Modifications to this function may impact the related          *
*             functions mtlistf() and mtlisto() that fetch rows and          *
*             close the cursor and, possibly, functions that call            *
*             these utility routines.                                        *
*                                                                            *
*****************************************************************************/
{
#ifdef CHECKOUT
printf("\nmalisto ");
#endif

strcpy( cid.arr, srcid.arr);
strcat(cid.arr,"%");
cid.len = strlen(cid.arr);

EXEC SQL DECLARE S2 CURSOR FOR
         SELECT   fix_label, floor_ref, ceiling_ref, fix_id,   fix_type,
                  artcc,     fix_lat,   fix_lon,     fix_rad,  fix_dist,
                  floor,     ceiling,   width_left,  width_right
         FROM     mtrsegments
         WHERE    fix_label LIKE :cid
         ORDER BY fix_label;

EXEC SQL OPEN S2;

#ifdef CHECKOUT
printf(" Open: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}



int malistf()
/*****************************************************************************
*                                                                            *
*     malistf   ---   Fetch a row using the opened cursor S2 for MFR         *
*     =======         Navigation Point User Information                      *
*                                                                            *
*     Routine executes an fetch command for cursor S2, which is assumed      *
*     to have been opened, and then returns to the calling program with      *
*     the ORACLE status code.                                                *
*                                                                            *
*     Note:   Modifications to this function may impact the related          *
*             functions mtlisto() and mtlisto() that open and close the      *
*             cursor and, likely, functions that call these utilities        *
*                                                                            *
*****************************************************************************/
{

strcpy(cptr3,"              ");  /* If you don't use VARCHAR, you have to  */
strcpy(cptr4,"              ");  /* clear the space or weird things happen */

EXEC SQL FETCH S2 INTO :n2bv,        :cptr1,       :cptr2,       :curfixid,
                       :curfixtyp,   :curartcc,    :cptr3,       :cptr4,
                       :curfixrad,   :curfixdist,  :lptr1,       :lptr2,
                       :curwidleft,  :curwidright;


prefixid.arr[prefixid.len] = '\0';
```

```
prefixtyp.arr[prefixtyp.len] = '\0';

#ifdef CHECKOUT
printf(" Fetch: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}

int malistc()
/*************************************************************************
*                                                                        *
*          malistc   ---   Close cursor S2 for Navigation Points         *
*          ========                                                      *
*                                                                        *
*     Routine executes a close cursor command for cursor S2 and then     *
*     returns to the calling program with the ORACLE status code.        *
*                                                                        *
*     Note:   Modifications to this function may impact the related      *
*             functions mtlistc() and mtlistf() that open the cursor      *
*             and fetch rows using it and, possibly, functions that      *
*             call these utilities                                       *
*                                                                        *
*************************************************************************/
{
#ifdef CHECKOUT
printf("\nmtlistc ");
#endif

EXEC SQL CLOSE S2;
#ifdef CHECKOUT
printf("Close: %ld ", sqlca.sqlcode);
#endif
return (int) sqlca.sqlcode;
}
/*************************************************************************/
/*                                                                       */
/*                  ASAN REPORT GENERATOR MODULE                         */
/*                                                                       */
/*                        January 26, 1987                               */
/*                                                                       */
/*************************************************************************/


#include <process.h>                  /*  Header for calls to MS-DOS     */
#include <stdio.h>

#define SQLCA_STORAGE_CLASS extern    /*  Switch for header files        */

EXEC SQL BEGIN DECLARE SECTION;       /*  All SQL declarations are in    */

EXEC SQL INCLUDE hostvars.h;          /*  these header files             */
EXEC SQL INCLUDE fmharris.h;

VARCHAR comparemis[8];                /* Last mission used */
VARCHAR comparemtr[26];               /* Last MTR used */
VARCHAR compareac[12];                /* Last aircraft used */

double exptab[27];                    /* Space for "ROUTE.BAS" Table */

int day_ops[12], night_ops[12];

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
#include "asan.h"                     /*  Standard ASAN Header File      */


EXEC SQL DECLARE LINDA1 CURSOR FOR
                        SELECT s_label, m_ident, aircraft, activity
                          FROM activities
                       ORDER BY s_label, aircraft, m_ident;

EXEC SQL DECLARE LINDA2 CURSOR FOR
                        SELECT day, night FROM operations
                         WHERE activity = :activity
                       ORDER BY month ASC;

EXEC SQL DECLARE LINDA3 CURSOR FOR
                        SELECT s_label, m_ident, aircraft, activity
                          FROM activities
                       ORDER BY s_label, m_ident, aircraft;
```

```
char    month[13][4] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUN",
                        "JUL", "AUG", "SEP", "OCT", "NOV", "DEC", "ANY"};
int     rta_vals1[10][2], rta_vals2[10][2], rta_vals3[10][2];
int     rta_vals4[10][2], rta_vals5[10][2], total;
int     annoy_flag = -2, hearing_flag = -2;
int     sleep_flag = -2, livestock_flag = -2, mnth[20];
int     speech_flag = -2, glass_flag = -2;
int     compare_flag = -2, species_flag = -2, index;
double  ldnmr1, ldnmr2, ldnmr3, ldnmr4, ldnmr5, ldnmr[20][5];
double  leq1, leq2, leq3, alevel[20][5];


report(name)
char  name[];
{
   FILE    *fopen(), *rpt;
   int     i;
   strcpy(altlev,"   ");
   rpt = fopen(name, "w");
   if (rpt == NULL) {
      printf("\nBad filename %s", name);
      exit(16);}

   table1(rpt);
   tbl2(rpt);
   table3(rpt);
   EXEC SQL OPEN LINDA3;
   if (sqlca.sqlcode) {
      printf("Open 3: %s", sqlca.sqlerrm.sqlerrmc);
      exit(4); }
   for (i=0;;i++) {
      EXEC SQL FETCH LINDA3 INTO :srcid, :misslabl, :ac_name, :activity;
      if (sqlca.sqlcode == SQL_EOF) break;
      if (sqlca.sqlcode) {
         printf("Fetch 3: %s", sqlca.sqlerrm.sqlerrmc);
         exit(4); }
      srcid.arr[srcid.len] = '\0';
      misslabl.arr[misslabl.len] = '\0';
      ac_name.arr[ac_name.len] = '\0';

      fillrtavals(i);
      landuse(rpt,i);
      inconsequential(rpt,i);
      minor(rpt,i,1);
      considerable(rpt,i,1);
      notconsidered(rpt,i);
   }
   EXEC SQL CLOSE LINDA3;
   references(rpt);
   fclose (rpt);
}


report3(name,which)
char  name[];
int   which;
{
   FILE    *fopen(), *rpt;
   int     i;
   strcpy(altlev,"   ");
   rpt = fopen(name, "w");
   if (rpt == NULL) {
      printf("\nBad filename %s", name);
      exit(16);}

   EXEC SQL OPEN LINDA3;
   if (sqlca.sqlcode) {
      printf("Open 3: %s", sqlca.sqlerrm.sqlerrmc);
      exit(4); }
   for (i=0;;i++) {
      EXEC SQL FETCH LINDA3 INTO :srcid, :misslabl, :ac_name, :activity;
      if (sqlca.sqlcode == SQL_EOF) break;
      if (sqlca.sqlcode) {
         printf("Fetch 3: %s", sqlca.sqlerrm.sqlerrmc);
         exit(4); }
      srcid.arr[srcid.len] = '\0';
      misslabl.arr[misslabl.len] = '\0';
      ac_name.arr[ac_name.len] = '\0';

      fillrtavals(i);
      landuse(rpt,i);
      inconsequential(rpt,i);
```

```
        minor(rpt,i,which);
        considerable(rpt,i,which);
        notconsidered(rpt,i);
      }
      EXEC SQL CLOSE LINDA3;
      references(rpt);
      fclose (rpt);
   }


   table1(rpt)
   FILE  *rpt;
   {
      cptr3 = show.lat;
      cptr4 = show.lon;
      comparestr.arr[0] = '\0';

      EXEC SQL OPEN LINDA1;
      if (sqlca.sqlcode) {
         printf("Cursor 1: %s", sqlca.sqlerrm.sqlerrmc);
         exit (4);
         }

      fprintf(rpt,"DESCRIPTION OF PROPOSED ACTION\n\n");
      fprintf(rpt,"The proposed action, known as %s,\n",ASSESSMENT.name);
      fprintf(rpt,"(%s)\n",ASSESSMENT.desc);
      fprintf(rpt,"consists of the use of the MTR segments as described in Table \
   1.\n");
   fprintf(rpt,"The subsonic flight operations proposed for these MTR segments\n");
      fprintf(rpt,"are described in Table 2.\n\n");

      for (;;) {
         EXEC SQL FETCH LINDA1 INTO :srcid, :misslabl, :ac_name, :activity;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode) {
            printf("Fetch 1: %s", sqlca.sqlerrm.sqlerrmc);
            exit(4); }

         if (!strncmp(comparestr.arr, srcid.arr, srcid.len)) continue;
         strncpy(comparestr.arr, srcid.arr, srcid.len);
         srcid.arr[srcid.len] = '\0';
         misslabl.arr[misslabl.len] = '\0';
         ac_name.arr[ac_name.len] = '\0';

   fprintf(rpt,"Table 1:  Description of MTR %s\n\n",srcid.arr);
   fprintf(rpt,"   NAV  \n");
   fprintf(rpt,"  POINT    FIX    RAD/DIS          LATITUDE           LONGITUDE\n");
   fprintf(rpt,"-------------------------------------------------------------\n");

         if (malisto()) {
            expOmsg();
            exit(4);
      }

         while ( !malistf() )  {
            {int i,j;
            for (i=srcid.len, j=0; i < a2bv.len; i++, j++)
               curnavpt.arr[j] = a2bv.arr[i];
            curnavpt.arr[j] = '\0';}
            curfixid.arr[curfixid.len] = '\0';

         show.lat[3] = ' ';
         show.lon[3] = ' ';
         fprintf(rpt,"    %s       %s     %03d/%03d       %s       %s\n",
         curnavpt.arr, curfixid.arr, curfixrad, curfixdist, show.lat, show.lon);

      }
      fprintf(rpt,"\n\n\n");

      if (sqlca.sqlcode != SQL_EOF)  printf("\n%s",sqlca.sqlerrm.sqlerrmc);
      malistc();
      }
   EXEC SQL CLOSE LINDA1;
   }


   tbl2(rpt)
   FILE  *rpt;
   {
      int  i;
```

```
      comparemtr.arr[0] = '\0';
      compareac .arr[0] = '\0';
      EXEC SQL OPEN LINDA1;
      if (sqlca.sqlcode) {
         printf("Cursor 1: %s", sqlca.sqlerrm.sqlerrmc);
         exit(4);}

      for (;;) {
         EXEC SQL FETCH LINDA1 INTO :srcid, :misslabl, :ac_name, :activity;
         if (sqlca.sqlcode == SQL_EOF) break;
         if (sqlca.sqlcode) {
            printf("Fetch 1: %s", sqlca.sqlerrm.sqlerrmc);
            exit(4); }

         if (strncmp(comparemtr.arr, srcid.arr, srcid.len)) {
            srcid.arr[srcid.len] = '\0';
            strncpy(comparemtr.arr, srcid.arr, srcid.len);
            }

         if (strncmp(compareac.arr, ac_name.arr, (unsigned int)12)) {
            ac_name.arr[ac_name.len] = '\0';
            strncpy(compareac.arr, ac_name.arr, (unsigned int) 12);

            EXEC SQL SELECT power_units FROM mtrseltab
                          INTO :pr_pwr_u WHERE aircraft = :ac_name;
            if (sqlca.sqlcode) {
               printf("Power units %s", sqlca.sqlerrm.sqlerrmc);
               exit(4);}
            pr_pwr_u.arr[pr_pwr_u.len] = '\0';
            }

         misslabl.arr[misslabl.len] = '\0';

         EXEC SQL SELECT sortie_size FROM missions INTO :ac_in_form
                                  WHERE mission = :misslabl;
         if (sqlca.sqlcode) {
            printf("Sortie size %s", sqlca.sqlerrm.sqlerrmc);
            exit(4);}

         EXEC SQL SELECT alt, alt_ref, pwr, spd
                        FROM mtr_flight_param
                        INTO :altlevel, :altlev, :ac_power, :ac_speed
                       WHERE activity = :activity
                         AND seq = 1;
         if (sqlca.sqlcode) {
            printf("Flight parameters %s", sqlca.sqlerrm.sqlerrmc);
            if (sqlca.sqlcode == SQL_EOF) {  /* For checking only ! */
                altlevel = 300;
                strcpy(altlev,"? *");
                ac_power = 100.00;
                ac_speed = 450;
                }
            else  exit(4);
            }

      fprintf(rpt,"\n\n\nTable 2:  Description of Flight Operations on %s \
by Month", srcid.arr);
      fprintf(rpt,"\n\n\nMISSION  %s    (%d AIRCRAFT/FORMATION)\n\n",
                misslabl.arr,ac_in_form);
      fprintf(rpt,"AIRCRAFT:  %s                    POWER:  %7.2lf %s\n",
                ac_name.arr, ac_power, pr_pwr_u.arr);
      fprintf(rpt,"ALTITUDE:  %6d %s                SPEED:  %7.2lf kts\n\n",
                altlevel, altlev, ac_speed);
      fprintf(rpt,"                              OPERATIONS\n");
      fprintf(rpt,"MONTH                DAY                    NIGHT\n");
      fprintf(rpt,"-----------------------------------------------\n");

      EXEC SQL OPEN LINDA2;
      if (sqlca.sqlcode) {
         printf("Cursor 2: %s", sqlca.sqlerrm.sqlerrmc);
         exit (4); }

      EXEC SQL FETCH LINDA2 INTO :day_ops, :night_ops;
      for (i=0; i < 12; i++) {
         fprintf(rpt," %s                %4d                    %4d\n",
              month[i], day_ops[i], night_ops[i]);}
      EXEC SQL CLOSE LINDA2;
      }
   EXEC SQL CLOSE LINDA1;
}

table3(rpt)
```

```
FILE  *rpt;
{
int     i, j;
double  temp1, temp2, add, log10();

fprintf(rpt,"\fSUMMARY OF PREDICTED NOISE EXPOSURE\n\n");
fprintf(rpt,"Noise exposure produced by aircraft operations may be \
specified in a variety\n");
fprintf(rpt,"of units.  The noise of low altitude high speed flights on \
Military Training\n");
fprintf(rpt,"Routes is specified for current purposes by a cumulative noise \
metric called\n");
fprintf(rpt,"the onset rate adjusted monthly day-night average, A-weighted \
sound level,\n");
fprintf(rpt,"abbreviated Ldnmr.  As described by Plotkin et al. (1987), \
this metric is\n");
fprintf(rpt,"based on an integration period equal to the calendar month \
with the highest\n");
fprintf(rpt,"number of operations.\n\n\n");

comparemtr.arr[0] = '\0';

EXEC SQL OPEN LINDA3;
if (sqlca.sqlcode) {
   printf("Cursor 3: %s", sqlca.sqlerrm.sqlerrmc);
   exit (4);
   }

fprintf(rpt,"Table 3:  Summary of Maximum Noise Exposure Produced by Flight \
Operations\n\n\n");
fprintf(rpt,"   MFR    Mission   Aircraft   Month   Distance        Ldnmr\n\n");
fprintf(rpt,"-------------------------------------------------------------\
\n");

for (j=0;;j++) {
   EXEC SQL FETCH LINDA3 INTO :srcid, :misslabl, :ac_name, :activity;
   if (sqlca.sqlcode == SQL_EOF) break;
   if (sqlca.sqlcode) {
      printf("Fetch 3: %s", sqlca.sqlerrm.sqlerrmc);
      exit(4); }

   srcid.arr[srcid.len] = '\0';
   if (strncmp(comparemtr.arr, srcid.arr, srcid.len)) {
      strncpy(comparemtr.arr, srcid.arr, srcid.len);
      fprintf(rpt, "\n%-9s", srcid.arr);
      comparemis.arr[0] = '\0';   /* insure that these fail for new a/c */
      compareac.arr[0]  = '\0';
      }
   else fprintf(rpt, "\n         ");

   misslabl.arr[misslabl.len] = '\0';
   if (strncmp(comparemis.arr, misslabl.arr, misslabl.len)) {
      strncpy(comparemis.arr, misslabl.arr, misslabl.len);
      comparemis.arr[comparemis.len] = '\0';
      compareac.arr[0]  = '\0';

      EXEC SQL SELECT sortie_size FROM missions INTO :ac_in_form
                              WHERE mission = :misslabl;
      if (sqlca.sqlcode) {
         printf("Sortie size %s", sqlca.sqlerrm.sqlerrmc);
         exit(4);}

      fprintf(rpt, "%-9s", misslabl.arr);
      }
   else fprintf(rpt, "         ");

   ac_name.arr[ac_name.len] = '\0';
   if (strncmp(compareac.arr, ac_name.arr, ac_name.len)) {
      strncpy(compareac.arr, ac_name.arr, ac_name.len);
      compareac.arr[ac_name.len] = '\0';
      fprintf(rpt, "  %-9s", ac_name.arr);}
   else fprintf(rpt, "\n         ");

   EXEC SQL OPEN LINDA2;
   if (sqlca.sqlcode) {
      printf("Cursor 2: activity %d\n  %s", activity, sqlca.sqlerrm.sqlerrmc);
      exit (4); }

   EXEC SQL FETCH LINDA2 INTO :day_ops, :night_ops;
   if (sqlca.sqlcode) {
      printf("Fetch 2: %s", sqlca.sqlerrm.sqlerrmc);
      exit (4); }
```

```
EXEC SQL CLOSE LINDA2;

add = 0;
index = 0;
temp2 = 300.0;

for (i=0; i < 12; i++)  {
   temp1 = (double) (day_ops[i] + 10 * night_ops[i]);
   if (temp1 > add) {
      if (temp1 < temp2) temp2 = temp1 ;
      add = temp1;
      index = i;
      }
   }
if ((add-temp2) < 0.05) index = 12;
month[j] = index;

EXEC SQL SELECT sortie_size
                FROM missions
                INTO :ac_in_form
               WHERE mission = :misslab1;
if (sqlca.sqlcode) {
   printf("Sortie size %s", sqlca.sqlerrm.sqlerrmc);
   exit(4);}

if (add > 0) add = 10.0 * log10(add * (double) ac_in_form) - 64.1;
else add = -1000.00;

  /*  This is a bit sloppy we should subtract 10 LOG10 of
      the number of seconds in the month instead of 64.1, but ..... */

sprintf(workspace.arr, "SELECT ACT%04d FROM mtr_exp_tab", activity);
workspace.len = strlen(workspace.arr);
EXEC SQL PREPARE STMT FROM :workspace;
if (sqlca.sqlcode) {
   printf("Prepare: %s", sqlca.sqlerrm.sqlerrmc);
   exit(16);}
EXEC SQL DECLARE D_CURS CURSOR FOR STMT;
EXEC SQL OPEN D_CURS;
if (sqlca.sqlcode) {
   printf("Dyn Open: %s", sqlca.sqlerrm.sqlerrmc);
   exit(16);}

EXEC SQL FETCH D_CURS INTO :exptab;
if (sqlca.sqlcode) {
   printf("Exposure table: %s", sqlca.sqlerrm.sqlerrmc);
   if ((sqlca.sqlcode == SQL_EOF) ||
      (sqlca.sqlcode == NULL_FETCHED))      /* For checking only ! */
      for (i = 0; i < 27; i++) exptab[i] = 0.0;
    else  exit(4);
    }
EXEC SQL CLOSE D_CURS;

if (add > -500.0) {
   ldmmr1 = exptab[0] + add;
   aleval[j][0] = exptab[0];
}
else {
   ldmmr1 = (double) 0;
   aleval[j][0] = 0;
}

if (add > -500.0)  {
   ldmmr2 = exptab[16] + add;
   aleval[j][1] = exptab[16];
}
else {
   ldmmr2 = (double) 0;
   aleval[j][1] = 0;
}

if (add > -500.0)  {
   ldmmr3 = exptab[19] + add;
   aleval[j][2] = exptab[19];
}
else {
   ldmmr3 = (double) 0;
   aleval[j][2] = 0;
}

if (add > -500.0)  {
   ldmmr4 = 0.5*(exptab[20]+exptab[21])+add;
```

```
        aleval[j][3] = 0.5*(exptab[20]+exptab[21]);
      }
      else {
        ldnmr4 = (double) 0;
        aleval[j][3] = 0;
      }

      if (add > -500.0)  {
        ldnmr5 = exptab[22] + add;
        aleval[j][4] = exptab[22];
      }
      else {
        ldnmr5 = (double) 0;
        aleval[j][4] = 0;
      }

      ldnmr[j][0] = ldnmr1;
      ldnmr[j][1] = ldnmr2;
      ldnmr[j][2] = ldnmr3;
      ldnmr[j][3] = ldnmr4;
      ldnmr[j][4] = ldnmr5;
      fprintf(rpt,"  %-3s    0.0 miles        ", month[index]);
      if (ldnmr1 >= 45.0) fprintf(rpt,"%6.11f", ldnmr1);
                    else fprintf(rpt,"   -");

      fprintf(rpt,"\n\t\t                        0.5 miles        ");
      if (ldnmr2 >= 45.0) fprintf(rpt,"%6.11f", ldnmr2);
                    else fprintf(rpt,"   -");

      fprintf(rpt,"\n\t\t                        1.0 miles        ");
      if (ldnmr3 >= 45.0) fprintf(rpt,"%6.11f", ldnmr3);
                    else fprintf(rpt,"   -");

      fprintf(rpt,"\n\t\t                        1.5 miles        ");
      if (ldnmr4 >= 45.0) fprintf(rpt,"%6.11f", ldnmr4);
                    else fprintf(rpt,"   -");

      fprintf(rpt,"\n\t\t                        2.0 miles        ");
      if (ldnmr5 >= 45.0) fprintf(rpt,"%6.11f", ldnmr5);
                    else fprintf(rpt,"   -");
    }
EXEC SQL CLOSE LINDA3;
}


fillrtavals(i)
int i;
{
   rta_vals1[1][0] = annoyance (ldnmr[i][0], &(rta_vals1[1][1]));
   rta_vals2[1][0] = annoyance (ldnmr[i][1], &(rta_vals2[1][1]));
   rta_vals3[1][0] = annoyance (ldnmr[i][2], &(rta_vals3[1][1]));
   rta_vals4[1][0] = annoyance (ldnmr[i][3], &(rta_vals4[1][1]));
   rta_vals5[1][0] = annoyance (ldnmr[i][4], &(rta_vals5[1][1]));

   rta_vals1[2][0] = hearing_damage (aleval[i][0], &(rta_vals1[2][1]));
   rta_vals2[2][0] = hearing_damage (aleval[i][1], &(rta_vals2[2][1]));
   rta_vals3[2][0] = hearing_damage (aleval[i][2], &(rta_vals3[2][1]));
   rta_vals4[2][0] = hearing_damage (aleval[i][3], &(rta_vals4[2][1]));
   rta_vals5[2][0] = hearing_damage (aleval[i][4], &(rta_vals5[2][1]));

   rta_vals1[3][0] = sleep_interference (aleval[i][0], &(rta_vals1[3][1]), i);
   rta_vals2[3][0] = sleep_interference (aleval[i][1], &(rta_vals2[3][1]), i);
   rta_vals3[3][0] = sleep_interference (aleval[i][2], &(rta_vals3[3][1]), i);
   rta_vals4[3][0] = sleep_interference (aleval[i][3], &(rta_vals4[3][1]), i);
   rta_vals5[3][0] = sleep_interference (aleval[i][4], &(rta_vals5[3][1]), i);

   rta_vals1[4][0] = sen_land_use (&(rta_vals1[4][1]));
   rta_vals2[4][0] = sen_land_use (&(rta_vals2[4][1]));
   rta_vals3[4][0] = sen_land_use (&(rta_vals3[4][1]));
   rta_vals4[4][0] = sen_land_use (&(rta_vals4[4][1]));
   rta_vals5[4][0] = sen_land_use (&(rta_vals5[4][1]));

   rta_vals1[5][0] = endangered_species_reproduction (&(rta_vals1[5][1]));
   rta_vals2[5][0] = endangered_species_reproduction (&(rta_vals2[5][1]));
   rta_vals3[5][0] = endangered_species_reproduction (&(rta_vals3[5][1]));
   rta_vals4[5][0] = endangered_species_reproduction (&(rta_vals4[5][1]));
   rta_vals5[5][0] = endangered_species_reproduction (&(rta_vals5[5][1]));

   rta_vals1[6][0] = livestock_dmg (&(rta_vals1[6][1]));
   rta_vals2[6][0] = livestock_dmg (&(rta_vals2[6][1]));
   rta_vals3[6][0] = livestock_dmg (&(rta_vals3[6][1]));
   rta_vals4[6][0] = livestock_dmg (&(rta_vals4[6][1]));
```

```
    rta_vals5[6][0] = livestock_damg (&(rta_vals5[6][1]));

    rta_vals1[7][0] = speech_interference (&(rta_vals1[7][1]));
    rta_vals2[7][0] = speech_interference (&(rta_vals2[7][1]));
    rta_vals3[7][0] = speech_interference (&(rta_vals3[7][1]));
    rta_vals4[7][0] = speech_interference (&(rta_vals4[7][1]));
    rta_vals5[7][0] = speech_interference (&(rta_vals5[7][1]));

    rta_vals1[8][0] = glass_breakage (&(rta_vals1[8][1]));
    rta_vals2[8][0] = glass_breakage (&(rta_vals2[8][1]));
    rta_vals3[8][0] = glass_breakage (&(rta_vals3[8][1]));
    rta_vals4[8][0] = glass_breakage (&(rta_vals4[8][1]));
    rta_vals5[8][0] = glass_breakage (&(rta_vals5[8][1]));

    rta_vals1[9][0] = effects_comparison (&(rta_vals1[9][1]));
    rta_vals2[9][0] = effects_comparison (&(rta_vals2[9][1]));
    rta_vals3[9][0] = effects_comparison (&(rta_vals3[9][1]));
    rta_vals4[9][0] = effects_comparison (&(rta_vals4[9][1]));
    rta_vals5[9][0] = effects_comparison (&(rta_vals5[9][1]));
}

landuse(rpt,i)
FILE  *rpt;
int   i;
{
   fprintf(rpt,"\fDESCRIPTION OF LAND USE COMPATIBILITY\n\n");
   fprintf(rpt,"MTR:  %s    MISSION:  %s    AIRCRAFT:  %s   MONTH:  %s\n\n",
           srcid.arr, misslabl.arr, ac_name.arr, month[mnth[i]]);
   fprintf(rpt,"Land uses compatible with the noise exposure produced by the\
 flight\n");
   fprintf(rpt,"operations associated with the proposed action, as specified\
 in the Joint\n");
   fprintf(rpt,"Services Land Use Planning Manual, are as noted below under \
worst case\n");
   fprintf(rpt,"assumptions.  These land use interpretations are for the MTR \
segment and\n");
   fprintf(rpt,"month producing the highest noise exposure.\n");

   fprintf(rpt,"\n   0.0 miles from the MTR Centerline: \n\n");
   rta_vals1[0][0] = habitability (1, ldnmr[i][0], &(rta_vals1[0][1]),rpt);
   fprintf(rpt,"\n   0.5 miles from the MTR Centerline: \n\n");
   rta_vals2[0][0] = habitability (2, ldnmr[i][1], &(rta_vals2[0][1]),rpt);
   fprintf(rpt,"\n   1.0 miles from the MTR Centerline: \n\n");
   rta_vals3[0][0] = habitability (3, ldnmr[i][2], &(rta_vals3[0][1]),rpt);
   fprintf(rpt,"\n   1.5 miles from the MTR Centerline: \n\n");
   rta_vals4[0][0] = habitability (2, ldnmr[i][3], &(rta_vals2[0][1]),rpt);
   fprintf(rpt,"\n   2.0 miles from the MTR Centerline: \n\n");
   rta_vals5[0][0] = habitability (3, ldnmr[i][4], &(rta_vals3[0][1]),rpt);
}

inconsequential(rpt,i)
FILE  *rpt;
int   i;
{
   fprintf(rpt, "\fDESCRIPTION OF INCONSEQUENTIAL NOISE EFFECTS\n\n");
   fprintf(rpt,"MTR:  %s    MISSION:  %s    AIRCRAFT:  %s   MONTH:  %s\n\n",
           srcid.arr, misslabl.arr, ac_name.arr, month[mnth[i]]);
   fprintf(rpt, "     The following effects of noise exposure produced by the \
flight\n");
   fprintf(rpt, "operations associated with the proposed action on people, \
structures, or\n");
   fprintf(rpt, "animals were determined to be inconsequential in the current \
environmental\n");
   fprintf(rpt, "assessment:\n\n");
   check(rpt,0);
}

minor(rpt,i, which)
FILE  *rpt;
int   i, which;
{
   fprintf(rpt, "\fDESCRIPTION OF NOISE EFFECTS OF MINOR IMPORTANCE\n\n");
   fprintf(rpt,"MTR:  %s    MISSION:  %s    AIRCRAFT:  %s   MONTH:  %s\n\n",
           srcid.arr, misslabl.arr, ac_name.arr, month[mnth[i]]);
   fprintf(rpt, "     The following effects of noise exposure produced by the \
flight\n");
   fprintf(rpt, "operations associated with the proposed action on people, \
structures, or\n");
   fprintf(rpt, "animals were determined to be of minor importance in the \
current \n");
   fprintf(rpt, "environmental assessment:\n\n");
   check(rpt,i);
```

```
    if (which == 1) {
       if (annoy_flag == 1) {
          bpl_annoyance(rpt);
          annoy_flag = -2;
       }
       if (hearing_flag == 1) {
          bpl_hearing(rpt);
          hearing_flag = -2;
       }
       if (sleep_flag == 1) {
          bpl_sleep(rpt);
          sleep_flag = -2;
       }
       if (livestock_flag == 1) {
          bpl_livestck(rpt);
          livestock_flag = -2;
       }
       if (speech_flag == 1) {
          bpl_speech(rpt);
          speech_flag = -2;
       }
       if (glass_flag == 1) {
          bpl_glass(rpt);
          glass_flag = -2;
       }
    }
 }

 considerable(rpt,i,which)
 FILE  *rpt;
 int   i, which;
 {
   fprintf(rpt, "\fDESCRIPTION OF NOISE EFFECTS OF CONSIDERABLE IMPORTANCE\n\n");
    fprintf(rpt,"MTR:  %s    MISSION:  %s    AIRCRAFT:  %s   MONTH:  %s\n\n",
            srcid.arr, misslabl.arr, ac_name.arr, month[mnth[i]]);
    fprintf(rpt, "     The following effects of noise exposure produced by the \
flight\n");
    fprintf(rpt, "operations associated with the proposed action on people, \
structures, or\n");
    fprintf(rpt, "animals were determined to be of considerable importance in \
the current \n");
    fprintf(rpt, "environmental assessment:\n\n");
    check(rpt,2);
    if (which == 1) {
       if (annoy_flag == 2) {
          bpl_annoyance(rpt);
          annoy_flag = -2;
       }
       if (hearing_flag == 2) {
          bpl_hearing(rpt);
          hearing_flag = -2;
       }
       if (sleep_flag == 2) {
          bpl_sleep(rpt);
          sleep_flag = -2;
       }
       if (livestock_flag == 2) {
          bpl_livestck(rpt);
          livestock_flag = -2;
       }
       if (speech_flag == 2) {
          bpl_speech(rpt);
          speech_flag = -2;
       }
       if (glass_flag == 2) {
          bpl_glass(rpt);
          glass_flag = -2;
       }
    }
 }

 notconsidered(rpt,i)
 FILE  *rpt;
 int   i;
 {
    fprintf(rpt, "\fDESCRIPTION OF EFFECTS NOT CONSIDERED IN CURRENT \
ENVIRONMENTAL ASSESSMENT\n\n");
    fprintf(rpt,"MTR:  %s    MISSION:  %s    AIRCRAFT:  %s   MONTH:  %s\n\n",
            srcid.arr, misslabl.arr, ac_name.arr, month[mnth[i]]);
   fprintf(rpt, "     The following potential noise effects were not considered\
 in the \n");
   fprintf(rpt, "present analyses: \n\n");
```

```
    check(rpt,-1);
    fprintf(rpt, "\n\nReasons that these potential effects were not considered\
  included\n");
    fprintf(rpt, "insufficient information for evaluation, insufficient \
 precision of \n");
    fprintf(rpt, "estimation of noise exposure, and lack of generally accepted \
 means of \n");
    fprintf(rpt, "producing quantitative estimates of magnitudes of potential \
 effects.\n");
 }

 references(rpt)
 FILE  *rpt;
 {
    fprintf(rpt,"\fREFERENCES\n\n");
    fprintf(rpt,"Busnel, R.  1978.  \"Introduction,\" in J. L. Fletcher and \
 R. G. Busnel (eds),\n");
    fprintf(rpt,"Effects of Noise on Wildlife.  Academic Press, New York.\n\n");
    fprintf(rpt,"Cottereau, P. 1972.  Sonic Boom exposure effects:  effects on\
  animals.\n");
    fprintf(rpt,"Journal of Sound Vibration 20 (4):531-534.\n\n");
    fprintf(rpt,"Environmental Protection Agency.  1980.  Guidelines for noise \
 impact\n");
    fprintf(rpt,"analysis.  Office of Air, Noise, and Radiation, United States \
 Environmental\n");
    fprintf(rpt,"Protection Agency (USEPA).\n\n");
    fprintf(rpt,"Environmental Protection Agency.  1974. Information on Levels \
 of\n");
    fprintf(rpt,"environmental noise requisite to protect public health and \
 welfare with an\n");
    fprintf(rpt,"adequate margin of safety. EPA 550/9 74 004.\n\n");
    fprintf(rpt,"Fletcher, J. L., and Busnel, R. G., eds.  1978.  Effects of \
 noise\n");
    fprintf(rpt,"on wildlife. Academic Press.\n\n");
    fprintf(rpt,"Harshey, R. L., and Higgins, T. H., eds.  1973. Statistical \
 prediction model\n");
    fprintf(rpt,"for glass breakage from nominal sonic booms.  Federal \
 Aviation\n");
    fprintf(rpt,"Administration Report FAA RD 73-79.\n\n");
    fprintf(rpt,"Hinshaw, W. R.; Ball, W. B.; Ladson, T. A.; McNail, H. C. H.; \
 and Taylor, J.\n");
    fprintf(rpt,"P. 1970. An annotated bibliography on animal response to sonic\
  booms and\n");
    fprintf(rpt,"other loud sounds. Washington, D. C.\n\n");
    fprintf(rpt,"International Civil Aviation Organization (ICAO). 1970. Sonic \
 Boom Effects\n");
    fprintf(rpt,"on the Animal Kingdom.  Sonic Boom Panel, Montreal, 12 21 \
 October 1970.\n\n");
    fprintf(rpt,"Newman, J. and K. P. Beattie.  1985. Aviation Noise Effects. \
 FAA\n");
    fprintf(rpt,"NE 85 2.  Federal Aviation Administration, Noise Abatement \
 Branch,\n");
    fprintf(rpt,"Washington, D. C.\n\n");
    fprintf(rpt,"Nixon, C. W.; Hille, H. K.; Sommer, H. C.; and Guild, H. \
 1968.  \"Sonic Booms\n");
    fprintf(rpt,"resulting from extremely low altitude supersonic flight:  \
 measurements and\n");
    fprintf(rpt,"observations on houses, livestock and people.\"  Aerospace \
 Medical Research\n");
    fprintf(rpt,"Laboratories, Wright-Patterson Air Force Base, Report AMRL TR \
 68 52.\n\n");
 fprintf(rpt,"Plotkin, Kenneth J.; Sutherland, Louis C.; and Molino, John A.\n");
 fprintf(rpt,"1987. \"Environmental NOise Assessment for Military Aircraft\n");
    fprintf(rpt,"Training Routes. Volume 2: Recommended Noise Metric.\"  \
    Aerospace\n");
    fprintf(rpt,"Medical Research Laboritories, Wright-Patterson Air Force \
    Base,\n");
    fprintf(rpt,"Report AMRL-TR-87-001.\n\n");
    fprintf(rpt,"Shotton, L. R.  1982. \"Response of Wildlife and Farm Animals \
 to Low Level\n");
    fprintf(rpt,"Military Jet Overflight.\"  The Reporter II (6):161-164.\f");
 }

 bpl_livestck(rpt)
 FILE  *rpt;
 {

    fprintf(rpt,"\n\nLIVESTOCK DAMAGE\n\n");
    fprintf(rpt,"The U.S. Environmental Protection Agency (EPA) has reviewed \
 the literature\n");
    fprintf(rpt,"on noise effects in domestic animals (Dufour 1980).  In \
 general, there is an\n");
```

```
    fprintf(rpt,"overall trend for domestic animals to adapt to intermittent \
 (aircraft or\n");
    fprintf(rpt,"aircraft-like) noise under 120 dB (decibels).  Busnel (1978)\
 reviewed\n");
    fprintf(rpt,"effects around large airports and found no evidence to \
 indicate noise-\n");
    fprintf(rpt,"related adverse effects.\n\n");
    fprintf(rpt,"Negative behavioral effects from exposure to sonic booms are\
 rare among\n");
    fprintf(rpt,"domestic animals such as horses, cattle, sheep and poultry \
 (Cottereau 1972;\n");
    fprintf(rpt,"Fletcher & Busnel 1978; Hinshaw et al. 1970; Nixon et al. \
 1968;\n");
    fprintf(rpt,"International Civil Aviation Organization [ICAO] 1970).  Large\
 farm animals\n");
    fprintf(rpt,"may respond with spontaneous activity (i.e. galloping, \
 bellowing, jumping).\n");
    fprintf(rpt,"Poultry show mild reactions to the booms in most cases, but in\
 less than tenl\n");
    fprintf(rpt,"percent of the cases chickens reacted with crowding, cowering,\
 or noise.\n");
    fprintf(rpt,"There was reported to be no significant effect on egg \
 production, milk\n");
    fprintf(rpt,"production, or food consumption.  Pigs, both in the open and \
 in shelter,\n");
    fprintf(rpt,"show a tendency to be quiet (ICAO 1970).  Observations show \
 greater\n");
    fprintf(rpt,"responses resulting from low-level subsonic flights, \
 motorcycles, paper\n");
    fprintf(rpt,"blown by the wind and other startling stimuli (ICAO 1970).  \
 There appears to\n");
    fprintf(rpt,"be no report of panic, injury or negative effects upon \
 reproductive success\n");
    fprintf(rpt,"(Fletcher & Busnel 1978).\n\n");

 }

 bpl_hearing(rpt)
 FILE  *rpt;
 {

    fprintf(rpt,"\n\nHEARING DAMAGE RISK\n\n");
    fprintf(rpt,"Hearing loss can be either temporary or permanent.  A \
 noise-induced\n");
    fprintf(rpt,"temporary threshold shift is a temporary loss of hearing \
 experienced after a\n");
    fprintf(rpt,"relatively short exposure to excessive noise.  A noise-induced\
 threshold\n");
    fprintf(rpt,"shift means that the detection level of sound has been \
 increased.  Recovery\n");
    fprintf(rpt,"is fairly rapid after cessation of the noise.  A noise-induced\
 permanent\n");
    fprintf(rpt,"threshold shift is an irreversible loss of hearing caused by \
 prolonged\n");
    fprintf(rpt,"exposure to excessive noise.  This loss is essentially \
 indistinguishable\n");
    fprintf(rpt,"from the normal hearing loss associated with aging.  Permanent\
 hearing loss\n");
    fprintf(rpt,"is generally associated with destruction of the hair cells of \
 the inner ear.\n");
    fprintf(rpt,"Based on EPA (Environmental Protection Agency) criteria, \
 hearing loss is not\n");
    fprintf(rpt,"expected for people living within noise contours below DNL \
 levels of 75 dB\n");
    fprintf(rpt,"(decibels).  Further, as stated in the EPA \"Levels Document,\"\
 changes in\n");
    fprintf(rpt,"hearing levels of <5 dB are generally not considered noticeable\
 or\n");
    fprintf(rpt,"significant (EPA 1974).\n\n");

 }

 bpl_speech(rpt)
 FILE  *rpt;
 {

    fprintf(rpt,"\n\nSPEECH INTERFERENCE\n\n");
    fprintf(rpt,"Speech interference associated with aircraft noise is a \
 primary source of\n");
    fprintf(rpt,"annoyance to individuals on the ground.  The disruption of \
 leisure\n");
    fprintf(rpt,"activities (such as listening to the radio, television, music,\
```

```
and\n");
   fprintf(rpt,"conversation) gives rise to frustration and irritation.  \
Quality speech\n");
   fprintf(rpt,"communication is obviously also important in the classroom, \
office, and\n");
   fprintf(rpt,"industrial settings.  Researchers have found that aircraft \
noise of 75 dB\n");
   fprintf(rpt,"(decibels) annoyed the highest percentage of the population \
when it\n");
   fprintf(rpt,"interfered with the television sound.  Eighty percent of the \
list of\n");
   fprintf(rpt,"annoyances for the surveyed population was flickering of the \
television\n");
   fprintf(rpt,"picture and interference with casual conversation by aircraft \
noise (Newman\n");
   fprintf(rpt,"& Beattie 1985).\n\n");

}

bpl_annoyance(rpt)
FILE  *rpt;
{

   fprintf(rpt,"\n\nHUMAN ANNOYANCE\n\n");
   fprintf(rpt,"Noise annoyance is defined by the U.S. Environmental \
Protection Agency (EPA)\n");
   fprintf(rpt,"as any negative subjective reaction to noise on the part of an\
individual or\n");
   fprintf(rpt,"group (EPA 1978).  \"Except in the case of speech interference,\
however, the\n");
   fprintf(rpt,"degree of interference is hard to specify and difficult to \
relate to the\n");
   fprintf(rpt,"level of noise exposure\" (EPA 1978).  \"Aircraft noise may \
. . . be found\n");
   fprintf(rpt,"annoying because it may startle people, cause houses to shake,\
or elicit\n");
   fprintf(rpt,"fear of a crash\" (EPA 1978).\n\n");

}

bpl_glass(rpt)
FILE  *rpt;
{

   fprintf(rpt,"\n\nSTRUCTURAL DAMAGE\n\n");
   fprintf(rpt,"By far, the largest percentage of sonic boom damage claims \
stem from broken\n");
   fprintf(rpt,"or cracked glass.  All of the tests conducted in the United \
States have\n");
   fprintf(rpt,"confirmed that glass damage is the most prevalent damage caused\
by sonic\n");
   fprintf(rpt,"booms (Hershey & Higgins 1973).  Because the microstructure of\
glass is\n");
   fprintf(rpt,"amorphous rather than crystalline, the practical design \
strength of glass\n");
   fprintf(rpt,"is dependent on the surface scratch condition.  Glass that has\
been\n");
   fprintf(rpt,"sandblasted, scratched, or nicked will not exhibit the same \
strength as\n");
   fprintf(rpt,"a properly installed relatively new pane of glass.\n\n");

}

bpl_sleep(rpt)
FILE  *rpt;
{

   fprintf(rpt,"\n\nSLEEP INTERFERENCE\n\n");
   fprintf(rpt,"Sleep interference is one of the factors contributing to \
aircraft noise\n");
   fprintf(rpt,"annoyance.  Airport nighttime restrictions have been employed \
to minimize\n");
   fprintf(rpt,"this annoyance.  In the case of nighttime operations, an \
exterior maximum\n");
   fprintf(rpt,"sound level (AIm) of 72 dB (decibels) is identified as an \
acceptable sleep\n");
   fprintf(rpt,"interference condition for a windows-closed condition.  This \
corresponds to\n");
   fprintf(rpt,"an interior AIm of about 55 dB.\n\n");
   fprintf(rpt,"To provide a basis for estimation of the number of people who \
could be\n");
   fprintf(rpt,"awakened by a specific noise event, data developed by \
```

```
 Goldstein and Lukas\n");
    fprintf(rpt,"(1980) were used to develop a relationship between the SEL \
 value and the\n");
    fprintf(rpt,"percent of exposed persons who would be awakened by the noise \
 event.  These\n");
    fprintf(rpt,"data indicated that the percent awakened by a specific \
 interior noise level\n");
    fprintf(rpt,"can be expressed by the following equation:\n\n");
    fprintf(rpt,"            Percent Awakened = 1.1(ASEL) - 49.5,\n");
   fprintf(rpt,"where ASEL = the interior A-weighted sound exposure level.\n\n");
    fprintf(rpt,"Since noise must penetrate the home to disturb sleep, interior\
  noise levels\n");
    fprintf(rpt,"will be lower than outside levels due to the absorption of \
 sound energy\n");
    fprintf(rpt,"attenuation by the structure.  The amount of attenuation \
 provided by the\n");
    fprintf(rpt,"building is dependent on the type of construction and whether \
 windows are\n");
    fprintf(rpt,"open or closed.  The Environmental Protection Agency recommends\
  attenuation\n");
    fprintf(rpt,"factors of 17 dB (decibels) for summertime (windows open) \
 residential\n");
    fprintf(rpt,"conditions and 27 dB for wintertime (windows closed) \
 conditions.\n");
    fprintf(rpt,"Incorporating the attenuation factors into the above equation \
 gives the\n");
    fprintf(rpt,"following relationships for the percent awakened under \
 summertime and\n");
    fprintf(rpt,"wintertime conditions:\n\n");
    fprintf(rpt,"            Percent Awakened (summer) = 1.1(ASEL - 17) - 49.5\n");
    fprintf(rpt,"                                      = 1.1(ASEL) - 68.2\n\n");
    fprintf(rpt,"            Percent Awakened (winter) = 1.1(ASEL - 27) - 49.5\n");
    fprintf(rpt,"                                      = 1.1(ASEL) - 79.2\n\n");
 }

 check(rpt, which)
 FILE  *rpt;
 int   which;
 {

    fprintf(rpt,"\n   0.0 miles from the MFR Centerline: \n\n");
    if (rta_vals1[1][1] == which) {
       fprintf(rpt, "Prevalence of Annoyance Among Population\n");
       annoy_flag = which;
    }
    if (rta_vals1[2][1] == which) {
       fprintf(rpt, "Hearing Damage Risk to Residential Population\n");
       hearing_flag = which;
    }
    if (rta_vals1[3][1] == which) {
       fprintf(rpt, "Sleep Interference of Residential Population\n");
       sleep_flag = which;
    }
    if (rta_vals1[5][1] == which) {
       fprintf(rpt, "Reproductive Success or Population Size of an Endangered \
 Species\n");
       species_flag = which;
    }
    if (rta_vals1[6][1] == which) {
       fprintf(rpt, "Economic Damage to Livestock\n");
       livestock_flag = which;
    }
    if (rta_vals1[7][1] == which) {
       fprintf(rpt, "Speech Interference of Residential Population\n");
       speech_flag = which;
    }
    if (rta_vals1[8][1] == which) {
       fprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
       glass_flag = which;
    }
    if (rta_vals1[9][1] == which) {
       fprintf(rpt, "Effects Comparison Module\n");
       compare_flag = which;
    }
    if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
        livestock_flag != which && speech_flag != which && glass_flag != which &&
        compare_flag != which && species_flag != which)
        fprintf(rpt,"There were no effects of noise in this category at this \
 distance\n");

    fprintf(rpt,"\n   0.5 miles from the MFR Centerline: \n\n");
    if (rta_vals2[1][1] == which) {
```

```
        fprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
      }
      if (rta_vals2[2][1] == which) {
        fprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
      }
      if (rta_vals2[3][1] == which) {
        fprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
      }
      if (rta_vals2[5][1] == which) {
        fprintf(rpt, "Reproductive Success or Population Size of an Endangered \
 Species\n");
        species_flag = which;
      }
      if (rta_vals2[6][1] == which) {
        fprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
      }
      if (rta_vals2[7][1] == which) {
        fprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
      }
      if (rta_vals2[8][1] == which) {
        fprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
      }
      if (rta_vals2[9][1] == which) {
        fprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
      }
      if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
         livestock_flag != which && speech_flag != which && glass_flag != which &&
         compare_flag != which && species_flag != which)
         fprintf(rpt,"There were no effects of noise in this category at this \
 distance\n");

      fprintf(rpt,"\n   1.0 miles from the MTR Centerline: \n\n");
      if (rta_vals3[1][1] == which) {
        fprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
      }
      if (rta_vals3[2][1] == which) {
        fprintf(rpt, "Hearing Damage Risk to Residential Population\n");
        hearing_flag = which;
      }
      if (rta_vals3[3][1] == which) {
        fprintf(rpt, "Sleep Interference of Residential Population\n");
        sleep_flag = which;
      }
      if (rta_vals3[5][1] == which) {
        fprintf(rpt, "Reproductive Success or Population Size of an Endangered \
 Species\n");
        species_flag = which;
      }
      if (rta_vals3[6][1] == which) {
        fprintf(rpt, "Economic Damage to Livestock\n");
        livestock_flag = which;
      }
      if (rta_vals3[7][1] == which) {
        fprintf(rpt, "Speech Interference of Residential Population\n");
        speech_flag = which;
      }
      if (rta_vals3[8][1] == which) {
        fprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
        glass_flag = which;
      }
      if (rta_vals3[9][1] == which) {
        fprintf(rpt, "Effects Comparison Module\n");
        compare_flag = which;
      }
      if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
         livestock_flag != which && speech_flag != which && glass_flag != which &&
         compare_flag != which && species_flag != which)
         fprintf(rpt,"There were no effects of noise in this category at this \
 distance\n");

      fprintf(rpt,"\n   1.5 miles from the MTR Centerline: \n\n");
      if (rta_vals4[1][1] == which) {
        fprintf(rpt, "Prevalence of Annoyance Among Population\n");
        annoy_flag = which;
```

```
      }
      if (rta_vals4[2][1] == which) {
         fprintf(rpt, "Hearing Damage Risk to Residential Population\n");
         hearing_flag = which;
      }
      if (rta_vals4[3][1] == which) {
         fprintf(rpt, "Sleep Interference of Residential Population\n");
         sleep_flag = which;
      }
      if (rta_vals4[5][1] == which) {
         fprintf(rpt, "Reproductive Success or Population Size of an Endangered \
 Species\n");
         species_flag = which;
      }
      if (rta_vals4[6][1] == which) {
         fprintf(rpt, "Economic Damage to Livestock\n");
         livestock_flag = which;
      }
      if (rta_vals4[7][1] == which) {
         fprintf(rpt, "Speech Interference of Residential Population\n");
         speech_flag = which;
      }
      if (rta_vals4[8][1] == which) {
         fprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
         glass_flag = which;
      }
      if (rta_vals4[9][1] == which) {
         fprintf(rpt, "Effects Comparison Module\n");
         compare_flag = which;
      }
      if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
          livestock_flag != which && speech_flag != which && glass_flag != which &&
          compare_flag != which && species_flag != which)
          fprintf(rpt,"There were no effects of noise in this category at this \
 distance\n");

      fprintf(rpt,"\n   2.0 miles from the MTR Centerline: \n\n");
      if (rta_vals5[1][1] == which) {
         fprintf(rpt, "Prevalence of Annoyance Among Population\n");
         annoy_flag = which;
      }
      if (rta_vals5[2][1] == which) {
         fprintf(rpt, "Hearing Damage Risk to Residential Population\n");
         hearing_flag = which;
      }
      if (rta_vals5[3][1] == which) {
         fprintf(rpt, "Sleep Interference of Residential Population\n");
         sleep_flag = which;
      }
      if (rta_vals5[5][1] == which) {
         fprintf(rpt, "Reproductive Success or Population Size of an Endangered \
 Species\n");
         species_flag = which;
      }
      if (rta_vals5[6][1] == which) {
         fprintf(rpt, "Economic Damage to Livestock\n");
         livestock_flag = which;
      }
      if (rta_vals5[7][1] == which) {
         fprintf(rpt, "Speech Interference of Residential Population\n");
         speech_flag = which;
      }
      if (rta_vals5[8][1] == which) {
         fprintf(rpt, "Glass Breakage Claims Among Residential Population\n");
         glass_flag = which;
      }
      if (rta_vals5[9][1] == which) {
         fprintf(rpt, "Effects Comparison Module\n");
         compare_flag = which;
      }
      if (annoy_flag != which && hearing_flag != which && sleep_flag != which &&
          livestock_flag != which && speech_flag != which && glass_flag != which &&
          compare_flag != which && species_flag != which)
          fprintf(rpt,"There were no effects of noise in this category at this \
distance\n");

 }
/*********************************************************************/
/*                                                                   */
/*      rptasan.pc  --  ASAN Main Program (Temporary driver for the  */
/*                      report generator                             */
/*                                                                   */
```

```
/*************************************************************************/

#include <stdio.h>                  /*  The usual stuff, of course */
#include <process.h>                /*  Header for calls to MS-DOS */
#include <string.h>                 /*  String manipulation header */
#include <time.h>

EXEC SQL BEGIN DECLARE SECTION;

EXEC SQL INCLUDE hostvars.h;
EXEC SQL INCLUDE fmharris.h;

EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;             /*  SQL Communication Area      */
#define SQLCA_STORAGE_CLASS
#include "asan.h"                   /*  Standard ASAN Header file   */
                                    /*  In asan.pc it must be here  */
                                    /*  since we need to initialize */
                                    /*  character arrays!           */
char  targv[15];

main(argc, argv)
int   argc;
char  *argv[];
{
int i, j, myOname();
static char *legal_notice[11] =
{"\n\n\n\n\t\t\tRESTRICTED RIGHTS LEGEND\n\n",
"\tUse, duplication, or disclosure is subject to restrictions\n",
"\t      as set forth in subdivision (b)(3)(ii) of the\n",
"\t   Rights in Technical Data and Computer Software Clause\n",
"\t\t    at 52.227-7013 of the DOD FAR Supplement.\n\n",
"\t\t\tBBN LABORATORIES INCORPORATED\n",
"\t\t\t      10 MOULTON STREET\n",
"\t\t\t     Cambridge, MA 02238\n",
"\t\t\t        617-873-3000\n\n\n",
"      User Interface Copyright (C) 1985,  BBN Laboratories Incorporated\n",
"\t\t\t     All Rights Reserved"};

extern char *malloc();
void expOmsg();

FILE *fopen();

if (argc == 1) {
   printf("\nUsage: RPTASAN assessment_name printfile_name");
   printf("\n   The printfile_name is optional: ");
   printf("if unspecified printer [prn] is assumed.\n\n");
   exit(16);
   }
printf("%c[2J",27);
for (i = 0; i < 11; i++) printf("%s",legal_notice[i]);

printf("\n\n\n\n\t   Please tap the space bar to continue, \"CTRL-C\" \
to abort.\n\n");
do {
    i = getch();
    } while (i != ' ');

printf("ASAN Report Module starting.....");
strcpy(workspace.arr, argv[1]);
strcat(workspace.arr,"/");
strcat(workspace.arr,univpwd);
workspace.len = strlen(workspace.arr);


EXEC SQL CONNECT :workspace ;
if (sqlca.sqlcode) { EXECMSG;
      printf("\nLogon failed\n%s",sqlca.sqlerrm.sqlerrmc); exit(4);}

EXEC SQL SELECT myid, myprivs, myname
                FROM sys.v$mypuser
                INTO :userno, :O_auth, :my_name;
my_name.arr[my_name.len] = '\0';

strcpy(ASSESSMENT.name, my_name.arr);        /*  Store results in global */
ASSESSMENT.id = userno;                      /*  ASSESSMENT structure    */
EXEC SQL SELECT description FROM table_of_contents INTO :workspace
        WHERE idnumber = :userno;
if (sqlca.sqlcode) {
   printf("\nLogon: %s", sqlca.sqlerrm.sqlerrmc);}
```

```
else {
   j = (sizeof ASSESSMENT.desc) -1;
   if (workspace.len >= j) {
      for(i = 0; i < j; i++) ASSESSMENT.desc[i] = workspace.arr[i];
      ASSESSMENT.desc[j] = '\0'; }
   else  strcpy(ASSESSMENT.desc, workspace.arr);
   }

if (argc > 2)
   strcpy(targv,argv[2]);
else
   strcpy(targv,"pra");

Uinit();
printf("%c[24;1H",27);

exit(2);
}

callrpt()
{
   report(targv);
}

callrpt3()
{
   report3(targv,0);
}

callrpt4()
{
   report3(targv,1);
}
```

B.1 C Language Source Code

```
/*   ASAM Effects Modules                                                  */
/*                                                                         */
/*   Created:  January 8, 1987                                             */

/*   The return or success codes returned by the procedures in this        */
/*   module have the following meanings:                                   */
/*         -2 = execution is not possible because the module is            */
/*              not implemented yet                                        */
/*         -1 = execution is not possible because the available            */
/*              input is incomplete or improper                            */
/*          0 = the precision of the estimate is satisfactory              */
/*          1 = the module executed, but the precision of the              */
/*              estimate is unsatisfactory                                 */

/*   The severity of effect codes returned by the procedures in this       */
/*   module have the following meanings:                                   */
/*         -1 = effect not considered in the current analysis              */
/*          0 = magnitude of predicted effect is inconsequential           */
/*          1 = magnitude of predicted effect is of minor importance       */
/*          2 = magnitude of predicted effect is of considerable           */
/*              importance                                                 */

#include "stdio.h"

int  table[20][5] = {0, 0, 0, 30, 25, 0, 0, 35, 30, 25, 0, 0, 35, 30,
                     25, 0, 0, 0, 30, 25, 40, 35, 30, 25, 1, 0, 0, 30,
                     25, 1, 0, 0, 0, 30, 25, 0, 0, 30, 25, 1, 0, 0, 0,
                     0, 0, 0, 0, 30, 25, 1, 35, 30, 1, 1, 1, 0, 35, 30,
                     25, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
                     1, 0, 30, 25, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
                     0, 0, 0, 1, 1, 1, 1, 1, 1, 1};

char table2[20][80] =  { "Family housing", "Bachelor housing",
                     "Transient lodging - hotels, motels",
                     "Classrooms, libraries, churches",
                     "Offices and administration buildings, military",
                     "Offices - business and professional",
```

```
                    "Hospitals, medical facilities, nursing homes",
                    "Dental clinic, medical dispensaries",
                    "Outdoor music shells",
                    "Retail stores, restaurants, banks, movie theaters",
                    "Flight line operations, maintenance and training",
                    "Industrial, manufacturing and laboritories",
                    "Outdoor sports arenas, outdoor spectator sports",
                    "Playgrounds, active sport recreational areas",
                    "Neighborhood parks", "Gymnasiums, indoor pools",
                    "Outdoor - frequent speech communication",
                    "Outdoor - unfrequent speech communication",
                    "Livestock farming, animal breeding",
                    "Agricultural (except livestock)" };


habitability (which, ldn, effectcode,rpt)
double   ldn;
int      *effectcode, which;
FILE     *rpt;

/*  The above tables and below look-up computations were derived from
    the Joint Services Noise Planning Manual as referenced in Table 6.3
    of Theodore J. Schultz's book, Community Noise Rating, Second Edition
    (1982).                                                              */

{
   int   i, index;
   char  name[15];

   if (ldn > 65 && ldn <= 69) {
      index = 4;
      *effectcode = 1;
   }
   else if (ldn <= 74) {
      index = 3;
      *effectcode = 2;
   }
   else if (ldn <= 79) {
      index = 2;
      *effectcode = 2;
   }
   else if (ldn <= 84) {
      index = 1;
      *effectcode = 2;
   }
   else if (ldn <=89) {
      index = 0;
      *effectcode = 2;
   }

   if (ldn < 65)  {
      fprintf(rpt, "   The following are compatible land uses:  \n");
      for (i = 0; i < 20; i++)
      fprintf(rpt,"        %s \n",table2[i]);
      *effectcode = 0;
   }
   else if (ldn > 89)  {
      fprintf(rpt, "   The following are incompatible land uses:  \n");
      for (i = 0; i < 20; i++)
         fprintf(rpt,"        %s \n",table2[i]);
      *effectcode = 2;
   }
   else {
      fprintf(rpt, "   The following are compatible land uses:  \n");
      for (i = 0; i < 20; i++)
         if (table[i][index] == 1)
            fprintf(rpt,"        %s \n",table2[i]);
      fprintf(rpt,
         "\n   The following land uses are compatible with interior noise\n");
      fprintf(rpt,"   reduction noted in parenthases:\n");
      for (i = 0; i < 20; i++)
         if (table[i][index] != 1 && table[i][index] != 0)
            fprintf(rpt,"        %s(%d) \n",table2[i],table[i][index]);
      fprintf(rpt, "\n   The following are incompatible land uses:  \n");
      for (i = 0; i < 20; i++)
         if (table[i][index] == 0)
            fprintf(rpt,"        %s \n",table2[i]);
   }
   return(0);
}
```

```
annoyance (ldn, affectcode)
double  ldn;
int     *affectcode;

/*  The below calculation of percent of the population that will be highly
    annoyed by a sound of level Ldn was copied from Theodore J. Schultz's
    article, "Synthesis of social surveys on noise annoyance", published in
    the Journal of the Acoustical Society of America, Vol.64, No. 2, August
    1978, pp 377-405.                                                      */

{
   double  parha, pow();

   if (ldn < 40 || ldn > 85) {
      *affectcode = -1;
      return(-1);
   }
   else {
      parha = 0.8553 * ldn - 0.0401 * pow(ldn,2.) + 0.00047 * pow(ldn,3.);
      if (parha < 5)
         *affectcode = 0;
      else if (parha <= 10)
         *affectcode = 1;
      else if (parha <= 20)
         *affectcode = 2;
      else
         *affectcode = 2;
      return(0);
   }
}


hearing_damage (aleval, affectcode)
double  aleval;
int     *affectcode;

/*  The calculation of aa was derived from William Burns' book, Noise
    and Man, Second Edition (1973), page 238.  The dB loss values were
    read from the 50% curve on the bottom graph on page 239 of the same
    book                                                                   */

{
   int  aa, temp;        /* aa = A-weighted noise immission level        */

   aa = (int) aleval + 10; /* 10 is derived from 10*log(duration of sound
                              exposure in years) which is assumed to be 5
                              years here                                   */
   temp = aa % 5;
                         /* round to the nearest 5 dBs                   */
   if (temp <= 2.5)
      aa -= temp;
   else
      aa += (5 - temp);

/*  aa of 85 -> 50% of the population would experience a 1.2dB hearing loss */
/*        90 ->  2.3dB loss,   95 ->  4.5dB loss,  100 ->  8.7dB loss      */
/*       105 -> 14.1dB loss,  110 -> 22.3dB loss,  115 -> 31.2dB loss      */
/*       120 -> 40.0dB loss,  125 -> 43.1dB loss,  130 -> 43.1dB loss      */

   if (aa <= 95)
      *affectcode = 0;
   else if (aa <= 110)
      *affectcode = 1;
   else
      *affectcode = 2;
   return(0);
}


sleep_interference (aleval, affectcode,i)
double  aleval;
int     *affectcode, i;

{
   double  par_awakened;

   if (i > 4 && i < 10)   /* SUMMER */
      par_awakened = 1.1 * aleval - 68.2;
   else                   /* WINTER */
      par_awakened = 1.1 * aleval - 79.2;

   if (par_awakened <= 5.0)
```

```
        *effectcode = 0;
    else if (per_awakened <= 10.0)
        *effectcode = 1;
    else
        *effectcode = 2;
    return(0);
}


sen_land_use (effectcode)
int  *effectcode;

{
    *effectcode = -1;
    return(-2);
}


endangered_species_reproduction (effectcode)
int  *effectcode;

{
    *effectcode = -1;
    return(-2);
}


livestock_damg (effectcode)
int  *effectcode;

{
    *effectcode = -1;
    return(-2);
}


speech_interference (effectcode)
int  *effectcode;

{
    *effectcode = -1;
    return(-2);
}


glass_breakage (effectcode)
int  *effectcode;

{
    *effectcode = -1;
    return(-2);
}


effects_comparison (effectcode)
int  *effectcode;

{
    *effectcode = -1;
    return(-2);
}
/*************************************************************************
 *                                                                       *
 *                cu.c -- ASAN GRASS color utilities.                    *
 *                                                                       *
 *         THIS CODE ASSUMES 256-COLOR LUT AND 2 OVERLAY PLANES.         *
 *                                                                       *
 *************************************************************************/

#include "grass.h"
#include "gglobs.h"


init_colors ()
  {
    int j;

    /* ??? ( C_CURSOR , 255 , 255 , 255 ) */

    InitCI( C_BEG      ,   0 ,   0 ,   0 );

    InitCI( C_SCR_FRAME, 255 , 255 , 255 );
```

```
    InitCI( C_WIN_FRAME, 255 , 255 , 255 );
    InitCI( C_LBG_FRAME, 255 , 255 , 255 );
    InitCI( C_LBG_TEXT , 255 , 255 , 255 );

    InitCI( C_BLACK    ,   1 ,   1 ,   1 );
    InitCI( C_RED      , 255 ,   0 ,   0 );
    InitCI( C_GREEN    ,   0 , 255 ,   0 );
    InitCI( C_BLUE     ,   0 ,   0 , 255 );
    InitCI( C_YELLOW   , 255 , 255 ,   0 );
    InitCI( C_MAGENTA  , 255 ,   0 , 255 );
    InitCI( C_CYAN     ,   0 , 255 , 255 );
    InitCI( C_WHITE    , 255 , 255 , 255 );

    /* for DMA elevations, use a smooth palette at low intensity */

    ig_palet( C_1ST_DMA , NUM_DMA_C , 0.25 , 0.50 );

    /* for categories, use a repeating smooth palette at high intensity */

    for ( j = C_1ST_CAT ; j <= MAX_C - C_CAT_REPEAT + 1 ; j += C_CAT_REPEAT )
        ig_palet( j , C_CAT_REPEAT , 0.50 , 0.50 );
    if ( j < MAX_C )
        ig_palet( j , MAX_C - j + 1 , 0.50 , 0.50 );

    next_cat_pv = C_1ST_CAT;    /* initialize pixval to use for category */
  }


ig_palet ( fpv , nclrs , i , s )    /* load smooth palette in Imagraph LUT */
  int    fpv;                          /* first LUT location to load */
  int    nclrs;                        /* how many LUT locations to load */
  double i, s;                         /* intensity, saturation */
  {
    double hinc, h, r, g, b;
    int    pv;

    if ( ( fpv < 0 ) || ( nclrs < 1 ) || ( fpv + nclrs - 1 > MAX_C ) )
    {
        SLOUTP("?ig_palet: bad parameter");
        return( -1 );
    }

    hinc = 360.0 / (double) nclrs;
    h = 0;

    for ( pv = fpv ; pv < fpv + nclrs ; pv++ )
    {
        hisrgb( h , i , s , &r , &g , &b );
        InitCI( pv, dround(r * 255.0), dround(g * 255.0), dround(b * 255.0) );
        h += hinc;
    }
  }


/*
 *  hisrgb() -- converts (h,i,s) color to (r,g,b) intensities
 *
 *  call:       hisrgb( h , i , s , &r , &g , &b );
 *
 *  calling args (normalized before use):
 *   0.0 <= h <=360.0  (red = 120.0, green = 240.0, blue = 0.0)
 *   0.0 <= i <= 1.0     0.0 <= s <= 1.0
 *  returned args:
 *   0.0 <= r <= 1.0     0.0 <= g <= 1.0     0.0 <= b <= 1.0
 */

hisrgb ( h , i , s , r , g , b )
  double h, i, s, *r, *g, *b;
  {
    double m, m2;

    /* normalize calling args */

    while ( h < 0.0 )    h += 360.0;
    while ( h > 360.0 )  h -= 360.0;
    while ( i < 0.0 )    i += 1.0;
    while ( i > 1.0 )    i -= 1.0;
    while ( s < 0.0 )    s += 1.0;
    while ( s > 1.0 )    s -= 1.0;

    if ( i > 0.5 ) m = i + s - ( i * s );
    else           m = i * ( 1.0 + s );
```

```
    m2 = ( 2.0 * i ) - m;

    if      ( h < 60.0 )  *r = m2 + ( m - m2 ) * ( h / 60.0 );
    else if ( h < 180.0 ) *r = m;
    else if ( h < 240.0 ) *r = m2 + ( m - m2 ) * ( ( 240.0 - h ) / 60.0 );
    else                  *r = m2;

    if      ( h < 120.0 ) *g = m2;
    else if ( h < 180.0 ) *g = m2 + ( m - m2 ) * ( ( h - 120.0 ) / 60.0 );
    else if ( h < 300.0 ) *g = m;
    else                  *g = m2 + ( m - m2 ) * ( ( 360.0 - h ) / 60.0 );

    if      ( h < 60.0 )  *b = m;
    else if ( h < 120.0 ) *b = m2 + ( m - m2 ) * ( ( 120.0 - h ) / 60.0 );
    else if ( h < 240.0 ) *b = m2;
    else if ( h < 300.0 ) *b = m2 + ( m - m2 ) * ( ( h - 240.0 ) / 60.0 );
    else                  *b = m;                          /* questionable */
  }


/* end of cm.c */
/*********************************************************************
 *                                                                   *
 *     dasan.c -- dummy asan mainline to test map screen features    *
 *                                                                   *
 *********************************************************************/

 #define DB(str)

/*
 *  needs datum for where's my finger on map.
 *
 *  show.coordinates of specified location on map.
 */

#include <stdio.h>
#include "asantype.h"
#include "asan.h"

                /* DEFINED IN GRAPHICS CODE TO BE SUPPLIED BY FMH: */

double shwdist;
extern char dlnam[][12];

                /* ASSUMED TO BE DEFINED ELSEWHERE IN ASAN USER CODE: */

COORDINATE cmt;
COORDINATE show;
ASANHEADR ASSESSMENT;
char idptofint[6];
int shwareaLDN;
int shwareaLDN1;
int shwareaPSF;
int shwareaPSF1;

extern int ENA_DEBUG_FEATURES;  /* for debug */


main ()
 {
    Ginit();                    /* initialize graphics */

    DB( strcpy( dlnam[0] , "FirstMap" ); )
    DB( strcpy( dlnam[7] , "JustaMap" ); )
    DB( strcpy( dlnam[14] , "Last_Map" ); )

    ENA_DEBUG_FEATURES = 1;     /* enable U debugging features */
    Uinit();                    /* disappear into U */
 }


dummy ()
  {
    ELOUTP("*** inside dummy() ***");
  }

lat2dec ( cmt )
  int cmt;
  {
    ELOUTP("*** inside lat2dec() ***");
  }
```

```
lon2dec ( ent )
  int ent;
  {
    SLOUTP("*** inside lon2dec() ***");
  }

ASANcomm ( arr )
  unsigned char *arr;
  {
    SLOUTP("*** inside ASANcomm() ***");
  }


/* end of dasan.c */
/*********************************************************************
 *                                                                   *
 *              dc.c  -- ASAN GRASS call file handlers.              *
 *                                                                   *
 *********************************************************************/

#define PUTRASTOK  0            /* = 1 if okay to use PutRast */

#define DB(str)

#include "grass.h"
#include "gglobs.h"

#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>

extern char     DMA_elev_being_drawn;

static char     fnam[80];
static int      fh;
static int      nbytes;
static int      scrrow;
static char     rbuf[MAXCOLS];
static char     wbuf[MAXCOLS];


d_call ()
  {
    int r, c;
    int cc, dsw;  /* drawing-method-test only */

    set_display_params();               /* set up window & viewport */

    sprintf( fnam, "%s\\%s\\%s\\call\\%s",
      gisdbase, mapset, location, layer2add);

    if ( ( fh = open( fnam , O_RDONLY | O_BINARY ) ) == -1 )
    {
        SLOUTP( "?d_call: open %s failed" , fnam );
        return( -1 );
    }

    scrrow = 0;

#  if PUTRASTOK
                        /* color is wired in call file */
#  else
                /* major kludge alert */
#  endif /* PUTRASTOK */

    clip_to_window();

    for ( r = 0 ; r < W_nrows ; r++ )
    {
        if ( ( nbytes = read( fh , rbuf , W_ncols ) ) != W_ncols )
        {
            DCERR( "%s out of data at row %d" , fnam , r );
            return( -1 );
        }

#       if PUTRASTOK

        if ( DMA_elev_being_drawn )             /* fiddle with colors */
            for ( cc = 0 ; cc < W_ncols ; cc++ )
              if ( rbuf[cc] != 0 )
                rbuf[cc] = ( (int) ( ( (double) rbuf[cc] / 17. ) + 1. ) );
```

```
        PutRast( W_l , W_t - scrrow , W_l + W_ncols , zbuf );

#       else

        for ( cc = 0 ; cc < W_ncols ; cc++ )
            if ( zbuf[cc] != 0 )
            {
                if ( DMA_elev_being_drawn )     /* fiddle with colors */
                    SetFG( (int) ( ( (double) zbuf[cc] / 17. ) + 1. ) );
                else
                    SetFG( zbuf[cc] );

                aLine( W_l + cc , W_t - scrrow ,
                   W_l + cc + 1 , W_t - scrrow + 1 , 0x40 );
            }

#       endif /* PUTRASTOK */

        scrrow++;
    }

# ifdef DONT_CARE
    if ( ( nbytes = read( fh , zbuf , W_ncols ) ) > 0 )
    {
        DCERR( "%s has data left over (%d bytes)" , fnam , nbytes );
        return( -1 );
    }
# endif /* DONT_CARE */

    close( fh );
    clip_to_screen();
    return( 0 );
  }


screen2call ()
  {
    ELOUTP( "screen2call() is not yet available" );
  }


static DCERR ( a , b , c , d , e , f , g )
  {
    char msgbuf[80];
    char tmpbuf[80];

    sprintf( tmpbuf , a , b , c , d , e , f , g );
    sprintf( msgbuf , "%d_call: " );
    strcat( msgbuf , tmpbuf );
    ELOUTP( msgbuf );
    close( fh );
    clip_to_screen();
  }


/* end of dc.c */
/*********************************************************************
 *                                                                   *
 *     dd.c -- ASAN GRASS code to draw a "digit" file on the screen. *
 *                                                                   *
 *********************************************************************/

#define DB(str)

#include "grass.h"
#include "gglobs.h"

#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>

static char     fnam[80];
static FILE     *dfp;
static char     zbuf[100];
static int      fline;
static char     b_type;                                 /* block type */
static int      count;


d_digit ()
  {
```

```
int i, j, z, y;
int last_sig;    /* index to latest non-whitespace char (-1=none) */

set_display_params();                /* set up window & viewport */

sprintf( fnam, "%s\\%s\\%s\\digit\\%s",
  gisdbase, mapset, location, layer2add);

if ( ( dfp = fopen( fnam , "r" ) ) == NULL )
{
    ELOUTP( "?d_digit: fopen %s failed" , fnam );
    return( -1 );
}

fline = 0;

/* skip the 14 lines of unwanted header stuff */

for ( i = 0 ; i < 14 ; i++ )
{
    fline++;
    if ( fgets( rbuf , 100 , dfp ) == NULL )
    {
        DDERR( "fgets error" );
        return( -1 );
    }
}

clip_to_window();

/*
 *  **** KLUDGE ALERT:
 *
 *  We know that current digit files contain exactly one category each,
 *  and that next_cat_pv has just been incremented in build_legend(), so
 *  the following color selection code is MUCH simpler than it ought to be.
 *
 *  Color/legend stuff will be VERY screwed up if any of the subsequent
 *  error returns is taken.
 */

SetFG( next_cat_pv - 1 );

/* process the digit file one block at a time */

while ( ( b_type = fgetc( dfp ) ) != EOF )
{
DB(ELOUTP("dd: 1st char of block = %c = %o",b_type,b_type);)
    switch( b_type )
    {
        case 'A':                                    /* polyline */
                fline++;
                if ( fgetc( dfp ) != ':' )
                {
                    DDERR( "bad char follows block type" );
                    return( -1 );
                }
DB(ELOUTP("dd A: got ':'");)
                if ( fscanf( dfp , "%d" , &count ) < 1 )
                {
                    DDERR( "bad count or early EOF" );
                    return( -1 );
                }
DB(ELOUTP("dd A: got count=%d",count);)
                fline++;
                if ( fscanf( dfp , " %lf %lf " , &dx1 , &dy1 ) < 2 )
                {
                    DDERR( "bad 1st vertex or early EOF" );
                    return( -1 );
                }
DB(printf("dd A: got 1st vertex %lf %lf\n",dx1,dy1);)
                for ( i = 0 ; i < count - 1 ; i++ )
                {
                    fline++;
                    if ( fscanf( dfp , " %lf %lf ", &dx2, &dy2) < 2 )
                    {
                        DDERR( "bad data or early EOF" );
                        return( -1 );
                    }
DB(printf("dd A: got next vertex %lf %lf, call d_line()\n",dx2,dy2);)
                    d_line();
                    dx1 = dx2;
```

```
                        dy1 = dy2;
                    }
DB(SLOUTP("dd A: done");)
                    break;

            case 'T':                                   /* text */
                    fline++;
                    if ( fgetc( dfp ) != ':' )
                    {
                        DDERR( "bad char follows block type" );
                        return( -1 );
                    }
DB(SLOUTP("dd T: got ':'");)
                    if ( fgets( xbuf , 100 , dfp ) == NULL )
                    {
                        DDERR( "bad string or early EOF" );
                        return( -1 );
                    }
                    else
                    {
                        i = 0;  j = 0;  last_sig = -1;

                        while ( xbuf[i] != '\n' )
                        {
                            if ( ( xbuf[i] == ' ' ) || ( xbuf[i] == '\t' ) )
                                if ( last_sig == -1 )   /* toss leading white */
                                    i++;
                                else                    /* keep embedded white */
                                    txt[j++] = xbuf[i++];
                            else if ( xbuf[i] == '\0' ) /* embedded null */
                                break;
                            else                        /* regular old char */
                            {
                                last_sig = j;
                                txt[j++] = xbuf[i++];
                            }
                        }
                        txt[last_sig+1] = '\0';       /* toss trailing white */
                    }
DB(SLOUTP("dd T: got '%s'",txt);)
                    fline++;
                    if ( fscanf( dfp , " %lf %lf " , &dx1 , &dy1 ) < 2 )
                    {
                        DDERR( "bad location or early EOF" );
                        return( -1 );
                    }
DB(printf("dd T: got loc %lf %lf\n",dx1,dy1);)
                    fline++;
                    if ( fscanf( dfp, " %lf %lf %lf %lf %d %c ",
                      &height, &width, &rotation, &slant, &font, &just) < 6 )
                    {
                        DDERR( "bad data or early EOF" );
                        return( -1 );
                    }
DB(SLOUTP("dd T: got parameters, call d_text()");)
                    d_text( UTM_COORDS );
DB(SLOUTP("dd A: done");)
                    break;

            default:
                    fline++;
                    DDERR( "bad block type" );
                    return( -1 );
        }
    }

    fclose( dfp );
    clip_to_screen();
    return( 0 );
  }


static DDERR ( a , b , c , d , e , f , g )
  {
    char msgbuf[80];
    char tmpbuf[80];

    sprintf( tmpbuf , a , b , c , d , e , f , g );
    sprintf( msgbuf , "%dc %s (%d): " , fnam , fline );
    strcat( msgbuf , tmpbuf );
    SLOUTP( msgbuf );
    fclose( dfp );
```

```
    clip_to_screen();
  }


/* end of dd.c */
/*********************************************************************
 *                                                                   *
 *                draw.c -- code to draw actual graphics             *
 *                                                                   *
 *********************************************************************/

#define DB(str)

#include "grass.h"
#include "gglobs.h"

#define XPIX(xx)        ( W_l + dround( ( xx - V_west ) / UTM_per_pixel ) )
#define YPIX(yy)        ( W_b + dround( ( yy - V_south ) / UTM_per_pixel ) )

extern int      xfont, yfont;           /* Imagraph text params */
extern int      chardepth, charwidth;

static int      t_size = 1;             /* text zoom factor (1-16) */
static int      xpix, ypix;


d_line ()                       /* draw a line */
  {
    aline( XPIX( dx1 ) , YPIX( dy1 ) , XPIX( dx2 ) , YPIX( dy2 ) , 0x40 );
  }


d_text ( ctype )                /* draw text */
  char ctype;                   /* SCREEN_COORDS or UTM_COORDS */
  {
    int i, xf, yf;

    if ( ctype == UTM_COORDS )
    {
        xpix = XPIX( dx1 );
        ypix = YPIX( dy1 );
    }
    else  /* SCREEN_COORDS */
    {
        xpix = ix1;
        ypix = iy1;
    }

    t_size = 1;

    if ( font == 'R' )
        xf = xpix - ( strlen( txt ) * charwidth );
    else if ( font == 'C' )
        xf = xpix + ( ( strlen( txt ) * charwidth ) / 2 );
    else        /* assume font == 'L' */
        xf = xpix;

    yf = ypix - ( chardepth / 2 );      /* supplied Y is vertical center */

    for ( i = 0 ; i < strlen( txt ) ; i++ )
    {
        xfont = xf;
        yfont = yf;
DB(SLOUTP("gputc: xfont=%d yfont=%d char=%c=0%o",xfont,yfont,txt[i],txt[i]);)
        if ((txt[i] == ' ') || (txt[i] == '\t'))    /* space or tab -> space */
            xf += charwidth;
        else if ( txt[i] < 040 )        /* ignore ctrl chars altogether */
            ;
        else                            /* send everything else to screen */
        {
            gputc( txt[i] , t_size , 0 );
            xf += charwidth;
        }
    }

    SetPatn( -0 );                      /* restore solid pattern */
  }


/* end of draw.c */
/*********************************************************************
 *                                                                   *
```

```
 *          gglobs.c -- globals and parameters for ASAN GRASS.          *
 *                                                                      *
 *        ***** WARNING:  THIS FILE MUST MATCH gglobs.h  *****          *
 *                                                                      *
 ************************************************************************/

#include "grass.h"

char    dlnam[MAX_DLAYERS][12]; /* displayed layer names */
int     num_dlayers;            /* how many layers now displayed */
LLINE   leg[MAX_CATS];          /* legend/cat data structure */
int     num_llines;             /* how many legend/cat data entries now */
int     next_cat_pv;            /* next pixval to use for category */

char    *mapset;                /* current mapset ("ASAN") */
char    *location;              /* current location name ("Salls") */
char    *gisdbase;              /* current map data base dir ("\grass\maps") */
char    vpname[12];             /* current viewport name ("C"=coarse, etc.) */
char    winname[12];            /* current window name */
char    layer2add[12];          /* name of layer to add to display */
char    layer2del[12];          /* name of layer to remove from display */
char    layersavename[12];      /* name to use for layer being saved */
char    laytype;                /* layer is: 'C'= cell, 'D'= digit, 'L'= DLG */

char    window_frame_drawn;     /* YES if window frame drawn, else NO */
char    legend_drawn;           /* YES if legend drawn, else NO */
char    DMA_elev_drawn;         /* YES if DMA elevations are drawn, else NO */

double  dx1, dx2, dy1, dy2;     /* some vertices */
int     ix1, ix2, iy1, iy2;     /* some vertices */
char    txt[100];               /* data needed to draw text */
double  height, width, rotation, slant;
int     font;
char    just;

int     W_t = SCR_T;            /* current window, in screen coords */
int     W_b = SCR_B;
int     W_l = SCR_L;
int     W_r = SCR_R;
int     W_nrows;                /* how many rows */
int     W_ncols;                /* how many columns */

int     V_proj;                 /* current viewport, in UTMs */
int     V_zone;
double  V_north;
double  V_south;
double  V_west;
double  V_east;
double  V_ns_res;
double  V_ew_res;
int     V_format;

double  UTM_per_pixel;


/* end of gglobs.c */
/************************************************************************
 *                                                                      *
 *              grass.c  -- ASAN graphics main routines.                *
 *                                                                      *
 ************************************************************************/

#define DB(str)

#include "grass.h"
#include "gglobs.h"

char            DMA_elev_being_drawn;

static char dflt_mapset[]
                = { 'a','s','a','n','\0' };
static char dflt_location[] =
                { 'S','a','l','l','s','\0' };
static char dflt_gisdbase[] =
                { '\\','g','r','a','s','s','\\','m','a','p','s','\0' };
static char dflt_winname[] =
                {'w','i','n','d','o','w','\0'};

static char     fnam[80];
static FILE     *fp;
static int      after_Uinit;
static char     del_layer_in_progress;          /* KLUDGE ALERT */
```

```
Uinit ()                        /* call me first */
  {
    mapset = (char *) dflt_mapset;      /* used to be environment vars ... */
    location = (char *) dflt_location;
    gisdbase = (char *) dflt_gisdbase;
    strcpy( winname , dflt_winname );

    begin_IG_graphics();
/**     load_default_palette();   **/
    InitFont();

    init_colors();

    after_Uinit = NO;
    clear_screen();
    after_Uinit = YES;
    DMA_elev_being_drawn = NO;
  }


new_view ( vstr )               /* change viewport */
  char *vstr;
  {

    if ( strcmp( vstr , vpname ) == 0 )         /* ignore if no change */
        return;

    if ( ( strcmp( vstr , "Salls.C" ) == 0 ) ||
         ( strcmp( vstr , "Ajo.M"   ) == 0 ) ||
         ( strcmp( vstr , "Salls.M" ) == 0 ) ||
         ( strcmp( vstr , "Ajo.F"   ) == 0 ) ||
         ( strcmp( vstr , "Salls.F" ) == 0 ) )
    {
        strcpy( vpname , vstr );
        NEWVALS();
        clear_screen();
    }
    else
        SLOUTP( "? unknown view %s" , vstr );
  }


clear_screen ()          /* erase screen & clear parameters */
  {
    int i;

    erase_graphics_display();

    for ( i = 0 ; i < MAX_DLAYERS ; i++ )    /* no layers displayed */
        strcpy( dlnam[i] , "" );
    num_dlayers = 0;

    if ( after_Uinit )          /* NEWVALS before Uinit -> bizarre crashes */
        NEWVALS();
  }


erase_graphics_display ()       /* just erase the screen */
  {
    clip_to_screen();
    cls( C_BKG );               /* clear screen to background color */
    SetFG( C_SCR_FRAME );       /* draw screen frame */
    aRect( SCR_L + 1 , SCR_B + 1 , SCR_R - 1 , SCR_T - 1 , 0x40 );

    window_frame_drawn = NO;    /* window frame is gone */
    legend_drawn = NO;          /* legend is gone */
    DMA_elev_drawn = NO;        /* DMA elevations are gone */
  }


add_layer ()                    /* add a layer to display */
  {
    int i;

    if ( strcmp( vpname , "" ) == 0 )           /* if no active view, give up */
    {
        SLOUTP("choose a view first");
        strcpy( layer2add , "" );
        NEWVALS();
        return( -1 );
    }
```

```
    for ( i = 0 ; i < strlen( layer2add ) ; i++ ) /* assure name is uppercase */
        if ( ( layer2add[i] >= 'a' ) && ( layer2add[i] <= 'z' ) )
            layer2add[i] -= 040;

    if ( strcmp( layer2add , "DMA_ELEV" ) == 0 )  /* special case */
    {
        draw_dma_elev();
        return( 0 );
    }

    /* see if it exists */

    laytype = ' ';
    sprintf( fnam, "%s\\%s\\%s\\digit\\%s",              /* digit file? */
      gisdbase, mapset, location, layer2add);
    if ( ( fp = fopen( fnam , "r" ) ) != NULL )
        laytype = 'D';
    if ( laytype == ' ' )                                 /* no, cell file? */
    {
        sprintf( fnam, "%s\\%s\\%s\\cell\\%s",
          gisdbase, mapset, location, layer2add);
        if ( ( fp = fopen( fnam , "r" ) ) != NULL )
            laytype = 'C';
    }
    if ( laytype == ' ' )                                 /* no, doesn't exist */
    {
        SLOUTP("?add_layer: no such layer %s", layer2add);
        strcpy( layer2add , "" );
        NEWVALS();
        return( -1 );
    }
    fclose( fp );

   if ( ! del_layer_in_progress )
   {
    if ( num_dlayers >= MAX_DLAYERS )   /* add it to displayed-layer list */
    {
        SLOUTP("?add_layer: screen list is full");
        strcpy( layer2add , "" );
        NEWVALS();
        return( -1 );
    }
    strcpy( dlnam[num_dlayers] , layer2add );
    num_dlayers++;
   }

    build_legend();                              /* add it to legend */
    if ( legend_drawn )
    {
        hide_legend();
        show_legend();
    }

    SLOUT( "Drawing %s" , layer2add );

    if ( laytype == 'D' )                        /* draw it */
        d_digit();
    else if ( laytype == 'C' )
        d_cell();

    strcpy( layer2add , "" );
    NEWVALS();
  }


del_layer ()            /* remove a layer from display */
  {
    int i, idx;
    char save_id;

    for ( i = 0 ; i < strlen( layer2del ) ; i++ ) /* assure name is uppercase */
        if ( ( layer2del[i] >= 'a' ) && ( layer2del[i] <= 'z' ) )
            layer2del[i] -= 040;

    if ( strcmp( layer2add , "DMA_ELEV" ) == 0 )  /* special case */
        DMA_elev_drawn = NO;

    /* first remove it from displayed-layer list */

    idx = -1;                                         /* find it in list */
    for ( i = 0 ; i < num_dlayers ; i++ )
        if ( strcmp( dlnam[i] , layer2del ) == 0 )
```

```
            idx = i;
        if ( idx == -1 )
        {
            SLOUTP("layer %s is not on display", layer2del);
            strcpy( layer2del , "" );
            NEWVALS();
            return( -1 );
        }

        if ( idx < num_dlayers - 1 )                /* if it wasn't last in list */
        {
            for ( i = idx ; i < num_dlayers - 1 ; i++ )     /* all move up */
                strcpy( dlnam[idx] , dlnam[idx+1] );
            strcpy( dlnam[num_dlayers-1] , "" );            /* erase last */
        }

        num_dlayers--;

        NEWVALS();                                          /* ... gratuitous */

        /*
         * Actually delete the layer from the graphics display.
         *
         * For now, brute force: erase the graphics screen and
         * repaint all layers but the deleted one.
         */

        save_ld = legend_drawn;     /* remember legend_drawn */

        erase_graphics_display();

        del_layer_in_progress = YES;

        for ( i = 0 ; i < num_dlayers ; i++ )
        {
            strcpy( layer2add , dlnam[i] );
            if ( add_layer() == -1 )
            {
                SLOUTP("?del_layer: redraw error on %s" , layer2add );
                strcpy( layer2del , "" );
                NEWVALS();
                del_layer_in_progress = NO;
                return( -1 );
            }
        }

        legend_drawn = save_ld;             /* recall legend_drawn */
        if ( legend_drawn )                 /* & redraw it if necessary */
        {
            hide_legend();
            show_legend();
        }

        del_layer_in_progress = NO;

        strcpy( layer2del , "" );
        NEWVALS();
    }


store_screen ()
    {
        /* if screen is blank, give up */

        if ( num_dlayers == 0 )
        {
            SLOUTP("?store_screen: screen is empty -- nothing to save");
            strcpy( layersavename , "" );
            NEWVALS();
            return( -1 );
        }

        /* if layer with this name already exists, give up */

        sprintf( fnam, "%s\\%s\\%s\\digit\\%s",
          gisdbase, mapset, location, layersavename);
        if ( ( fp = fopen( fnam , "r" ) ) != NULL )
        {
            SLOUTP("?store_screen: layer %s already exists", layersavename);
            strcpy( layersavename , "" );
            NEWVALS();
            return( -1 );
```

```
    }
    sprintf( fnam, "%s\\%s\\%s\\cell\\%s",
      gisdbase, mapset, location, layersavename);
    if ( ( fp = fopen( fnam , "r" ) ) != NULL )
    {
        SLOUTP("?store_screen: layer %s already exists", layersavename);
        strcpy( layersavename , "" );
        RENVALS();
        return( -1 );
    }

    laytype = 'C';

    screen2cell();
    add_to_layertxt();

    strcpy( layersavename , "" );
    RENVALS();
  }


show_dist ()
  {
    SLOUTP( "This feature is not yet available" );
  }


show_coords ()
  {
    SLOUTP( "This feature is not yet available" );
  }


edit_colors ()
  {
    SLOUTP( "This feature is not yet available" );
  }


clip_to_window ()
  {
    SetADR( W_l , W_b , W_r , W_t );
  }


clip_to_screen ()
  {
    SetADR( SCR_L , SCR_B , SCR_R , SCR_T );
  }


draw_dma_elev ()                /* special branch of add_layer() */
  {
    int i;

    if ( ! del_layer_in_progress )
    {
     if ( num_dlayers >= MAX_DLAYERS )   /* add it to displayed-layer list */
     {
         SLOUTP("?add_layer: screen list is full");
         strcpy( layer2add , "" );
         RENVALS();
         return( -1 );
     }
     strcpy( dlnam[num_dlayers] , layer2add );
     num_dlayers++;
    }

    build_dma_elev_legend();                    /* special legend too */

    DMA_elev_being_drawn = YES;

    if ( strcmp( vpname , "Salls.C" ) == 0 )
    {
        strcpy( layer2add , "AJO_E.C" );
        SLOUT( "Drawing AJO_E.C (DMA_ELEV part 1 of 5)" );
        d_cell();
        strcpy( layer2add , "AJO_W.C" );
        SLOUT( "Drawing AJO_W.C (DMA_ELEV part 2 of 5)" );
        d_cell();
        strcpy( layer2add , "LUKEV.C" );
        SLOUT( "Drawing LUKEV.C (DMA_ELEV part 3 of 5)" );
```

```
            d_call();
            strcpy( layer2add , "NOGAL.C" );
            SLOUT( "Drawing NOGAL.C (DMA_ELEV part 4 of 5)" );
            d_call();
            strcpy( layer2add , "TUCSN.C" );
            SLOUT( "Drawing TUCSN.C (DMA_ELEV part 5 of 5)" );
            d_call();
        }
        else if ( strcmp( vpname , "Ajo.M" ) == 0 )
        {
            strcpy( layer2add , "AJO_E.MA" );
            SLOUT( "Drawing AJO_E.MA (DMA_ELEV part 1 of 1)" );
            d_call();
        }
        else if ( strcmp( vpname , "Salls.M" ) == 0 )
        {
            strcpy( layer2add , "AJO_E.MS" );
            SLOUT( "Drawing AJO_E.MS (DMA_ELEV part 1 of 4)" );
            d_call();
            strcpy( layer2add , "LUKEV.MS" );
            SLOUT( "Drawing LUKEV.MS (DMA_ELEV part 1 of 4)" );
            d_call();
            strcpy( layer2add , "NOGAL.MS" );
            SLOUT( "Drawing NOGAL.MS (DMA_ELEV part 1 of 4)" );
            d_call();
            strcpy( layer2add , "TUCSN.MS" );
            SLOUT( "Drawing TUCSN.MS (DMA_ELEV part 1 of 4)" );
            d_call();
        }
        else if ( strcmp( vpname , "Ajo.F" ) == 0 )
        {
            strcpy( layer2add , "AJO_E.FA" );
            SLOUT( "Drawing AJO_E.FA (DMA_ELEV part 1 of 1)" );
            d_call();
        }
        else if ( strcmp( vpname , "Salls.F" ) == 0 )
        {
            strcpy( layer2add , "NOGAL.FS" );
            SLOUT( "Drawing NOGAL.FS (DMA_ELEV part 1 of 1)" );
            d_call();
        }
        else
        {
            SLOUTP( "? no DMA elevation data for this view" );
            DMA_elev_being_drawn = NO;
            DMA_elev_drawn = NO;
            return( -1 );
        }

        strcpy( layer2add , "" );
        NEWVALS();

        DMA_elev_being_drawn = NO;
        DMA_elev_drawn = YES;
    }


drround ( num )          /* double-precision-to-integer rounding routine */
    double num;
    {
      if ( num > 0.0 ) return (int) (num + 0.499);
      else             return (int) (num - 0.499);
    }


/* end of grass.c */
/*************************************************************************
 *                                                                       *
 *      ig.c -- device driver functions for Imagraph AGC hardware.       *
 *                                                                       *
 *************************************************************************/

/*      #define  GRAPHICS_DEBUGGING  1          */

#include "stdio.h"
#include "acrtc.h"
#include "imagraph.h"

int     GDBA = 0, GDBP = 0;             /* debug switches */

/* Imagraph hardware/software stuff */
```

```
int     max_intensity;          /* max color intensity for this hardware */
char    *Palette;               /* intensities of 1st 16 colors */
char    *model;                 /* handy pointer to IG model ident */
int filpat[16] = {-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0,-0};
                                /* fill pattern mask */

/* Stuff defined in Imagraph libraries */

extern int      FourBit;        /* 4 => 4-bit, 0 => 8-bit */
extern IMAstru  *pIMA;
extern IMAstru  IMAmodel[];
extern SCRNDEF  BaseSC;
extern SCRNDEF  UpperSC;
extern SCRNDEF  LowerSC;

/* functions */

void    SCalc();
IMAstru *IMAopen();
char    *palalloc();
char    *getenv();


void begin_IG_graphics ()
  {
        IMAstru *board;
        int     xm, ym;

#ifdef GRAPHICS_DEBUGGING

        GDBA = 0;  GDBP = 0;
        printf("\nannounce every hardware call? (1=yes,0=no): ");
        scanf("%d", &GDBA);
        if ( GDBA )
        {
            printf("pauses after hardware call announcements? (1=yes,0=no): ");
            scanf("%d", &GDBP);
        }
#endif /* GRAPHICS_DEBUGGING */

        model = getenv("IMAMODEL");
        if ( ! ( board = IMAopen(model) ) )
        {
            debugout("\nIMAopen failed: Imagraph hardware not found\n");
            exit(0);
        }
        tweaker();                      /* let's tweak hardware  */
        xm = board->xmaxpix;
        ym = board->ymaxpix;
        UpperSC.maxx = 0;               /* don't use upper screen */
        UpperSC.clipx = 0;
        UpperSC.maxy = 0;
        UpperSC.clipy = 0;
        LowerSC.maxx = 0;               /* don't use lower screen */
        LowerSC.clipx = 0;
        LowerSC.maxy = 0;
        LowerSC.clipy = 0;
        BaseSC.maxx = xm + 1;           /* base screen for graphics */
        BaseSC.clipx = xm;
        BaseSC.maxy = ym + 1;
        BaseSC.clipy = ym;
        SCalc(RESET_SCREEN, &BaseSC);   /* reset frame buffer pointer */
        SCalc(BASE_SCREEN, &BaseSC);
        IG_hdw_init();                  /* initialize hardware */
        SetOrg( BaseSC.o1, BaseSC.o2);  /* coord system on base screen */
        max_intensity = (FourBit) ? 15 : 255;  /* max color intensity value */
        Palette = palalloc();           /* allocate palette image memory */
        load_default_palette();         /* load default palette */
/***     ClrScrn( &BaseSC , 0 ); ***/   /* clear graphics screen to BLACK */
        SetFG( 7 );                     /* default foreground is WHITE */
        SetBG( 0 );                     /* default background is BLACK */
        SetPatn( -0 );                  /* default pattern is solid */
  }


void end_IG_graphics ()
  {
        IG_hdw_release();               /* clean up hardware */
        free(Palette);                  /* clean up heap */
  }
```

```
static IG_hdw_init ()            /* init hardware */
  {
    IMAinit();                          /* initialize the hardware */
    SetADR( 0 , 0 , pIMA->xmaxpix , pIMA->ymaxpix );
    initCI( 0 , 0 , 0 , 0 );
    itxtmode( 1 );                      /* get us into graphics for SGA */

    return(0);
  }


static IG_hdw_release ()                /* close hardware */
  {
    itxtmode( 0 );                      /* get us out of graphics for SGA */

    return(0);
  }


load_default_palette ()
  {
    make_default_palette();
    putlut( Palette );

    return(0);
  }


static make_default_palette ()
  {
        int mi, mi34, mi2;

        mi = (FourBit) ? 15 : 255;
        mi34 = (mi >> 2) * 3;        /* 75% intensity */
        mi2 = mi >> 1;               /* 50% intensity */
        pinitCI(Palette, 0, 0, 0, 0);
        pinitCI(Palette, 1, mi, 0, 0);
        pinitCI(Palette, 2, mi, 0, mi);
        pinitCI(Palette, 3, 0, 0, mi);
        pinitCI(Palette, 4, 0, mi, mi);
        pinitCI(Palette, 5, 0, mi, 0);
        pinitCI(Palette, 6, mi, mi, 0);
        pinitCI(Palette, 7, mi, mi, mi);
        pinitCI(Palette, 8, mi2, mi2, mi2);
        pinitCI(Palette, 9, mi34, 0, 0);
        pinitCI(Palette, 10, mi, 0, mi34);
        pinitCI(Palette, 11, 0, 0, mi34);
        pinitCI(Palette, 12, 0, mi, mi34);
        pinitCI(Palette, 13, 0, mi34, 0);
        pinitCI(Palette, 14, mi34, 0, mi34);
        pinitCI(Palette, 15, mi34, mi34, mi34);

        return(0);
  }


/* end of ig.c */
/*********************************************************************
 *                                                                   *
 *       leg.c -- ASAN GRASS legend and category file handing.       *
 *                                                                   *
 *********************************************************************/

#define DB(str)

#include "grass.h"
#include "gglobs.h"

#define  LEG_FRAME_SPACE        ( charwidth / 4 )
#define  LEG_PV_WIDTH           ( charwidth )
#define  LEG_PV_HEIGHT          ( chardepth - 2 )
#define  LEG_PVTXT_SPACE        ( charwidth / 2 )

extern int      chardepth, charwidth;

static int      leg_frame_top;
static int      leg_frame_bot;
static int      leg_frame_left;
static int      leg_frame_right;
static int      leg_frame_height;
static int      leg_frame_width;
```

```
static char     fnam[80];
static FILE     *fp;
static int      count;
static int      fline;
static char     junk[100];


build_legend ()         /* build legend data structure */
  {
    int dlidx, i, j;

    num_llines = 0;

    for ( dlidx = 0 ; dlidx < num_dlayers ; dlidx++ )
    {
        sprintf( fnam, "%s\\%s\\%s\\cats\\%s",
          gisdbase, mapset, location, dlnam[dlidx]);
        if ( ( fp = fopen( fnam , "r" ) ) == NULL )
        {
            ELOUTP("%build_leg: fopen %s failed", fnam);
            return( -1 );
        }

        fline = 1;

        if ( fscanf( fp , " %d %s " , &count ) < 2 )     /* read 1st line */
        {
            ELERR( "bad count or early EOF" );
            return( -1 );
        }

        for ( i = 0 ; i < 2 ; i++ )                      /* skip next 2 lines */
        {
            fline++;
            if ( fgets( junk , 100 , fp ) == NULL )
            {
                ELERR( "fgets error" );
                return( -1 );
            }
        }

        for ( i = 0 ; i < count ; i++ )                  /* the rest are cats */
        {
            if ( num_llines >= MAX_CATS )                /* prevent overflow */
            {
                ELERR( "legend data space full" );
                return( -1 );
            }

            fline++;                                     /* store string */
            if ( fgets( leg[num_llines].str , 100 , fp ) == NULL )
            {
                ELERR( "bad data or early EOF" );
                return( -1 );
            }

            /* replace <ret> in string with <null> */

            for ( j = 0 ; j < strlen( leg[num_llines].str ) ; j++ )
                if ( leg[num_llines].str[j] == '\n' )
                    leg[num_llines].str[j] = '\0';

            leg[num_llines].pv = next_cat_pv;            /* store pirval */

            next_cat_pv++;
            num_llines++;
        }

        fclose( fp );
    }

DB( printf("count=%d num_llines=%d:\n", count, num_llines);     )
DB( for ( i = 0 ; i < num_llines ; i++ )          )
DB(   printf( "%s\n" , leg[i].str );              )
DB( getch();                                      )
  }


static ELERR ( a , b , c , d , e , f , g )
  {
    char msgbuf[80];
    char tmpbuf[80];
```

```
        sprintf( tmpbuf , a , b , c , d , e , f , g );
        sprintf( msgbuf , "?build_leg %s (%d): " , fnam , fline );
        strcat( msgbuf , tmpbuf );
        SLOUTP( msgbuf );
        fclose( fp );
      }


    show_legend ()          /* draw legend on graphics screen */
      {
        int i, j, wid, y;

        if ( legend_drawn )                  /* if legend is visible, erase it */
            hide_legend();

        leg_frame_top = W_b - 4;
        leg_frame_left = 5;

        wid = 0;                             /* find widest legend line string */
        for ( i = 0 ; i < num_llines ; i++ )
            if ( ( j = strlen( leg[i].str ) ) > wid )
                wid = j;

        leg_frame_height = ( num_llines * chardepth ) + ( LEG_FRAME_SPACE * 2 );

        leg_frame_width = LEG_PV_WIDTH + LEG_PV2TEXT_SPACE
                        + ( wid * charwidth ) + ( LEG_FRAME_SPACE * 2 );

        leg_frame_right = leg_frame_left + leg_frame_width;

        if ( ( leg_frame_bot = leg_frame_top - leg_frame_height ) < 0 )
        {
            SLOUTP("?legend frame below screen bottom");
            leg_frame_bot = 1;
        }

        SetFG( C_LEG_FRAME );                                /* draw frame */

        aRect( leg_frame_left , leg_frame_bot ,
               leg_frame_right , leg_frame_top , 0x40 );

        y = leg_frame_top - LEG_FRAME_SPACE - chardepth;     /* draw legend lines */

        for ( i = 0 ; i < num_llines ; i++ )
        {
            if ( y < 0 )
            {
                SLOUTP("?legend text below screen bottom");
                legend_drawn = YES;
                return( -1 );
            }

            SetFG( leg[i].pv );                              /* draw color block */

            aFRect( leg_frame_left + LEG_FRAME_SPACE , y ,
              leg_frame_left + LEG_FRAME_SPACE + LEG_PV_WIDTH , y + LEG_PV_HEIGHT ,
              0x40 );

            strcpy( txt , leg[i].str );                      /* draw string */
            ix1 = leg_frame_left +LEG_FRAME_SPACE + LEG_PV_WIDTH + LEG_PV2TEXT_SPACE;
            iy1 = y + ( chardepth / 2 );

            setFG( C_LEG_TEXT );
            d_text( SCREEN_COORDS );

            y -= chardepth;
        }

        if ( DMA_elev_drawn )
            show_DMA_elev_legend();

        legend_drawn = YES;
      }


    hide_legend ()          /* erase legend from graphics screen */
      {
        if ( legend_drawn )
        {
            SetFG( C_BKG );                  /* IS THIS THE CORRECT CMD?? */
            aFrect( leg_frame_left , leg_frame_bot ,
                    leg_frame_right , leg_frame_top , 0x40 );
```

```
        if ( DMA_elev_drawn )
         hide_DMA_elev_legend();
      }

      legend_drawn = NO;
    }


  build_dma_elev_legend ()          /* build special legend for DMA elevations */
    {
                                          /* ADD ME LATER */
    }


  show_DMA_elev_legend ()           /* draw DMA elevation legend */
    {
                                          /* ADD ME LATER */
    }


  hide_DMA_elev_legend ()           /* erase DMA elevation legend */
    {
                                          /* ADD ME LATER */
    }


  /* end of leg.c */

  /*********************************************************************
   *                                                                   *
   *  lt.c -- ASAN GRASS map layer list/description text file management. *
   *                                                                   *
   *        NOTE:  Name  "layers.txt"  is wired into this code.        *
   *                                                                   *
   *********************************************************************/

  #include "grass.h"
  #include "gglobs.h"

  static char     fnam[80];
  static FILE     *fp;
  static char     nambuf[12];


  add_to_layertxt ()                /* add new layer name to the file */
    {
      int i;

      if ( ( fp = fopen( "layers.txt" , "r" ) ) == NULL ) /* first find it */
      {
          SLOUTP("?add21: fopen[r] layers.txt failed");
          return( -1 );
      }

      fclose( fp );

      if ( ( fp = fopen( "layers.txt" , "a" ) ) == NULL ) /* now really open it */
      {
          SLOUTP("?add21: fopen[a] layers.txt failed");
          return( -1 );
      }

      /* assure layer name is uppercase */

      for ( i = 0 ; i < strlen( layersavename ) ; i++ )
          if ( ( layersavename[i] >= 'a' ) && ( layersavename[i] <= 'z' ) )
              nambuf[i] = layersavename[i] - 040;
          else
              nambuf[i] = layersavename[i];
      nambuf[++i] = '\0';

      fprintf( fp , "%s\n" , nambuf );

      fclose( fp );
    }


  rem_from_layertxt ()              /* remove a layer name from the file */
    {
    }
```

```
/* end of lt.c */
/* ADD BETTER FIELD CHECKING.  SAVE WIN & VP, CHECK AGAINST NEW ONES */

/*********************************************************************
 *                                                                   *
 *   vw.c -- read winddow & viewport files, set up display parameters *
 *                                                                   *
 *              FILE FORMATS ARE HARDWIRED INTO THIS CODE,           *
 *               AND ORDER OF ITEMS IN FILES IS CRUCIAL.             *
 *                                                                   *
 *********************************************************************/

#define DB(str)

#include "grass.h"
#include "gglobs.h"

#include <fcntl.h>
#include <sys\types.h>
#include <sys\stat.h>
#include <io.h>

static char     fnam[80];
static char     junk[20];
static FILE     *fp;


set_display_params ()                    /* set up display parameters */
  {
    double h_temp, v_temp;

    /* read the window file */

    sprintf( fnam, "%s\\%s", gisdbase, winname);

    if ( ( fp = fopen( fnam , "r" ) ) == NULL )
    {
        SLOUTP("?vw: fopen window %s failed", winname);
        return( -1 );
    }

    fscanf( fp , "%s %d" , junk , &W_l );
    fscanf( fp , "%s %d" , junk , &W_r );
    fscanf( fp , "%s %d" , junk , &W_b );
    fscanf( fp , "%s %d" , junk , &W_t );

    fclose( fp);

    /* Find number of rows & columns in screen window */

    W_arows = W_t - W_b + 1;
    W_acols = W_r - W_l + 1;

DB(SLOUTP("vw: W_t=%d W_b=%d W_l=%d W_r=%d W_arows=%d W_acols=%d",
                W_t,W_b,W_l,W_r,W_arows,W_acols);)

    if ( ! window_frame_drawn )
    {
        SetFG( C_WIN_FRAME );
        aRect( W_l - 1 , W_b - 1 , W_r + 1 , W_t + 1 , 0x40 );
        window_frame_drawn = YES;
    }

    /*  read the viewport file */

    sprintf( fnam, "%s\\%s\\%s\\vwport\\%s",
      gisdbase, mapset, location, vpname);

    if ( ( fp = fopen( fnam , "r" ) ) == NULL )
    {
        SLOUTP("?vw: fopen vwport %s failed", vpname);
        return( -1 );
    }

    fscanf( fp , "%s %d"     , junk ,          &V_proj );
    fscanf( fp , "%s %d"     , junk ,          &V_zone );
    fscanf( fp , "%s %lf"    , junk ,          &V_north );
    fscanf( fp , "%s %lf"    , junk ,          &V_south );
    fscanf( fp , "%s %lf"    , junk ,          &V_west );
    fscanf( fp , "%s %lf"    , junk ,          &V_east );
    fscanf( fp , "%s %s %lf" , junk , junk , &V_ns_res );
    fscanf( fp , "%s %s %lf" , junk , junk , &V_ew_res );
```

```
    fscanf( fp , "%s %d"     , junk ,        &V_format );

    fclose( fp);

    /*
     *  Find resolution (UTMs per pixel).
     *
     *  To assure that maps fit in the window in both directions,
     *  compute it twice (using horizontal and vertical dimensions
     *  and store whichever is numerically larger.
     */

    v_temp = ( V_north - V_south ) / (double)( W_t - W_b );
    h_temp = ( V_east - V_west ) / (double)( W_r - W_l );

    if ( v_temp >= h_temp )
        UTM_per_pixel = v_temp;
    else
        UTM_per_pixel = h_temp;

DB(SLOUTP("vw: UTM_per_pixel=%lf",UTM_per_pixel);)

    return( 0 );
  }


/* end of vw.c */
```