OMEGA SYSTEM PERFORMANCE ASSESSMENT AND COVERAGE EVALUATION (PACE) WORKSTATION DESIGN AND IMPLEMENTATION

George R. Desrochers

TASC
55 Walkers Brook Drive
Reading, Massachusetts 01867

February 1991
Technical Information Memorandum

Volume II

Document is available to the U.S. public through the
National Technical Information Service
Springfield, Virginia 22161

Prepared for:

U.S. DEPARTMENT OF TRANSPORTATION
UNITED STATES COAST GUARD
OMEGA Navigation System Center
Alexandria, Virginia 22310-3998

TECHNICAL REPORT DOCUMENTATION PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
CG-ONSCEN-02.2-91		

4. TITLE AND SUBTITLE

Omega System Performance Assessment and Coverage Evaluation (PACE) Workstation Design and Implementation

Volume II

5. REPORT DATE

15 February 1991

6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)

George R. Desrochers

8. PERFORMING ORGANIZATION REPORT NO.

TIM-5834-2-2

9. PERFORMING ORGANIZATION NAME AND ADDRESS

TASC
55 Walkers Brook Drive
Reading, Massachusetts 01867

10. WORK UNIT NO. (TRAIS)

DTCG23-89-C-20008
Task Order No. 89-0001

11. CONTRACT OR GRANT NO.

Technical Information

12. SPONSORING AGENCY NAME AND ADDRESS

U.S. Department of Transportation
U.S. Coast Guard Omega Navigation System Center
7323 Telegraph Road
Alexandria, VA 22310

13. TYPE OF REPORT AND PERIOD COVERED

Technical Information Manual

14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

16. ABSTRACT

The Performance Assessment and Coverage Evaluation (PACE) workstation is a microcomputer-based tool to assist Omega Navigation System Center personnel in answering questions regarding Omega system performance. The workstation implements a probabilistic model of Omega system availability for user-selected coverage and other criteria. This memorandum provides information on the design and implementation of the PACE workstation software.

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Microcomputer Omega Signal Coverage Omega Navigation System	

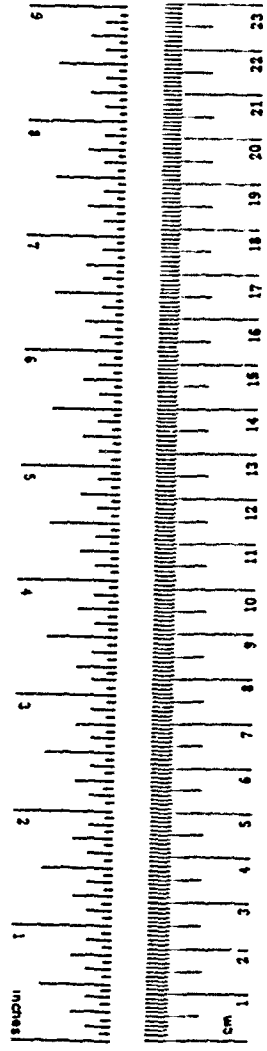
19 SECURITY CLASSIF. *(Of This Report)*	20. SECURITY CLASSIF. *(Of This Page)*	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified		

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

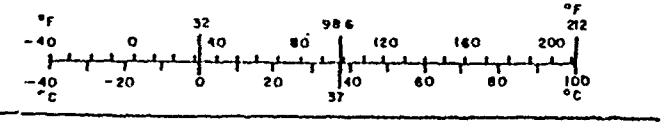
Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
	AREA			
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
	MASS (weight)			
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
	VOLUME			
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
	TEMPERATURE (exact)			
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.75, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
	LENGTH			
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
	AREA			
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
	MASS (weight)			
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
	VOLUME			
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
	TEMPERATURE (exact)			
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



PREFACE

This document contains Appendices B through H of the Omega System Performance Assessment and Coverage Evaluation (PACE) Workstation Design and Implementation report. The appendices provide code listings of the PACE software. Discussions about the listings contained herein are provided in Section 5 of Volume I.

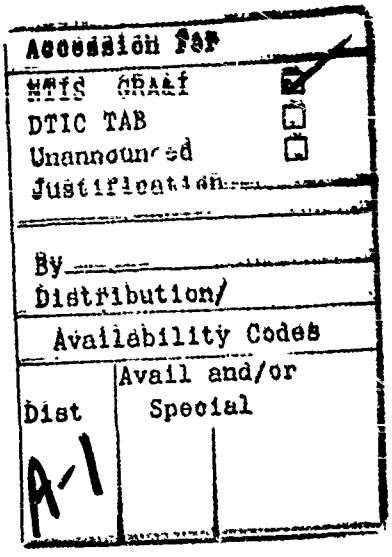


TABLE OF CONTENTS

TABLE OF CONTENTS

APPENDIX B
RBCONV2 LISTINGS

This appendix contains the Pascal code listings for the range/bearing to cell format database conversion program. A discussion of this program is provided in Section 5.1.

File Name: RBCONV2.SRC

The RBCONV2 program uses the following source program files as well as other PACE units:

RBCONV2.PAS
CONV_DB.PAS
DOM_MODE.PAS
GET_WGTS.PAS
ID_NOISE.PAS
LOADDEF.PAS
LOAD_RB.PAS
LOG10.PAS
PACE_RB.PAS
PHASE.PAS
POWER.PAS
QKSORT.PAS
SNRLONG.PAS
SNRSHORT.PAS
X_ANGLE.PAS

File Name: RBCONV2.PAS

```
(*************************************************************************
*                                                                       *
*        PROGRAM NAME - Omega Performance Assessment                    *
*                                and                                    *
*                          Coverage Evaluation                          *
*                                (PACE)                                 *
*                             Workstation                               *
*                                                                       *
*           UNIT NAME - RBCONV2 Program                                 *
*                                                                       *
*************************************************************************
*                                                                       *
*        This program was prepared by                                   *
*                                                                       *
*              The Analytic Science Corporation (TASC)                  *
*                    55 Walkers Brook Drive                             *
*                Reading, Massachusetts 01867                           *
*                                                                       *
*        PACE has been developed to run on a IBM PC/AT or compatible    *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main      *
*        memory and an EGA or compatible graphics adapter and color     *
*        monitor. This work was performed under contract number         *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for      *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.   *
*                                                                       *
*************************************************************************
*                                                                       *
*        PURPOSE                                                        *
*           Program to convert the range/bearing database to the cell   *
*           format database used by PACE.                               *
*                                                                       *
*                                                                       *
*************************************************************************)
PROGRAM RB_CONVERT;

USES Crt, Dos;

{
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

     Convert Range-Bearing Data to Matrix form

   Ver 0  05/06/88  KATench     Convert a single file as defined by constants
   Mod 1     11/88  DPFrank     Read and process data from reformatted Range/
                                 Bearing files
   Ver 2  05/30/90  KATench     Version for PACE files
   Ver 3     10/90  EMAustvold  Updated version for PACE files
                                 To generate "real" data for each station
                                 Added updates from PBMorris
```

```
File Name: RBCONV2.PAS


     The procedure is to generate a large temporary file which is
     organized as:        Öá      Öá      Öá
                   Station* Month* Hour* Cell
                          âá      âá      âá
     ( The loading and interpolating of Range/Bearing data is best this way)

     The last step is to access this file and convert it to one
     organized as:     Öá        Öá      Öá
                   Cell* Station* Month* Hour
                      âá        âá      âá
     ( The most efficient way for PACE to access it )

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

}
CONST
   noisepath     = '\pace\rbconv2\noise\';
   rbpath        = 'D:\';
   outpath       = '\pace\rbconv2\pacedb\';
   dbname        = '102COVDB.4M';
   freq          =  102;
   SNR_THRESHOLD = 799;   (* 99.9 * 8 = 799.2 which rounds to 799  *)
                          (* 8 is the resolution of 1/8 of a dB    *)
                          (* 99.9 represents the maximum amplitude *)
                          (*      that our signals will reach      *)
                          (* used in the computation of the Short  *)
                          (* and Long path SNR computation         *)

{$I CONV_DB.PAS }
{$I PACE_RB.PAS }

TYPE
   atom_ptr      = ^rng_atom;
   NeighborsType = array[1..4]         of atom_ptr;
                                  { pointers to nearest neighbor atoms }
   WType         = array[1..4]         of Real;
                                  { weights for the nearest neighbors }
   CovCells      = array[1..444]       of CovCell;
   NoiseType     = array[1..444,1..24] of Integer;

VAR
   i,j,k,
   icell,
   imonth,
   DMflag,
   tmpMLIMR,
   MLIMR           : integer;
   Neighbors       : NeighborsType;
```

File Name: RBCONV2.PAS

```
    W                : WType;
    Delta_Bearing,
    Delta_Range      : Real;
    The_Cell         : Cell_Defn_ptr;
    Temp_RB,
    Trail_RB         : RB_ptr;
    Sum              : Real;
    CovSum           : integer;
    fcell            : file of Cell_Record;
    cellfname        : string[40];
    fINTP            : text;
    fCOV             : text;
    ftemp            : file of CovCells;
    Cell             : CovCells;
    ffinal           : file of coviorec_4m;
    d                : coviorec_4m;
    fLOG             : text;
    Noise            : NoiseType;
    TEMPFILENAME     : string;
    gmtstr           : string;

{$I QKSORT.PAS}
{$I LD_NOISE.PAS }
{$I POWER.PAS}
{$I LOG10.PAS}
{$I LOADDEF.PAS }
{$I LOAD_RB.PAS }
{$I GET_WGTS.PAS }
{$I SNRSHORT.PAS }
{$I SNRLONG.PAS }
{$I PHASE.PAS }
{$I X_ANGLE.PAS }
{$I DOM_MODE.PAS }



BEGIN
   Gridsize := 10;
   RB_List := nil;
   Cell_Defn_List := nil;
   Assign(fLOG,'RB_CONV.LOG');
   Rewrite(fLOG);


   Assign(ftemp, outpath+'Temp.$$$'); Rewrite(ftemp);
   IF IOResult <> 0 Then
   BEGIN
      writeln;
      writeln(chr(7), '/// Error opening OUTPUT file ///');
```

```
      Halt(1);
   END;

   { Loop on stations }
   for selected_Station := 1 to 8 do      { Loop on stations }
   begin
     TextColor(7+selected_Station);
     Load_Cell_Definition( Selected_Station,
                           GridSize,
                           Cell_Defn_List,
                           Numb_Cells);
     for iMonth := 1 to 4 do              { Loop on months }
     begin
       Month := (iMonth-1)*3 + 2;           { to get 2,5,8 and 11 }
       load_noise_data(Month,Noise);
       for GMT := 1 to 24 do              { Loop on hours }
       BEGIN
         {$I-}
         Reset(fLOG);
         {$I+}
         If IOResult = 0 Then
         Begin
            Close(fLOG);
            Append(fLOG);
         End
         Else
            Rewrite(fLOG);
         Load_Range_Bearing_Data( Selected_Station,
                                  Month,
                                  GMT,
                                  Freq,
                                  RB_List);
         The_Cell := Cell_Defn_List;
         CovSum := 0;
         Writeln;


         FOR icell := 1 to numb_cells DO          { Loop on Cells }
         BEGIN
           IF (icell MOD 10) = 0 THEN
             Write('#');

           GetWeights(Neighbors,W,The_Cell,Temp_RB,Trail_RB);

           ComputePhase(Neighbors,W,Cell,icell);

           MLIMR := ComputeDom_Mode(Neighbors,W,Cell,icell);

           ComputeSNRShort(Neighbors,W,Cell,Noise[icell,GMT],icell,MLIMR);
```

File Name: RBCONV2.PAS

```
        ComputeSNRLong(Neighbors,W,Cell,Noise[icell,GMT],icell,M1IMR);

        ComputeX_Angle(The_Cell,Cell,icell,Selected_Station,Month,GMT);


        {Output SNR Information by Cell}

(* ***************************

        if ((icell = 1) or ((icell MOD 50) = 0)) AND
           ((GMT = 1) or (GMT = 12) or (GMT = 24)) then
        begin
          if (icell = 1) then
          begin
            STR(GMT,gmtstr);
            TEMPFILENAME := 'INTP' + gmtstr + '.OUT';
            Assign(fINTP,TEMPFILENAME);
            Rewrite(fINTP);
          end;
          if (icell > 1) then
            writeln(fINTP,chr($C));      { page eject }

          Writeln(fINTP,'Station = ',Selected_Station);
          Writeln(fINTP,'Month = ',Month);
          Writeln(fINTP,'GMT = ',GMT);
          Writeln(fINTP,'                              lat     lon       ');
          Writeln(fINTP,'                       cell   cntr    cntr      1st    2nd',
                        '                          Phsdev  X Angle (DM = 0 if mode 1 dominant
          Writeln(fINTP,'Cell Range(Mm),   Bearing, deg ,   deg ,     Rad,   Rad',
                        ', sp-SNR(dB), lp-SNR, (cecs),  (degs), DM, M1IMR');
          Writeln(fINTP);
        END;

        if ((GMT = 1) or (GMT = 12) or (GMT = 24)) then
        begin
          tmpM1IMR := (Get_Sigma(Cell[icell].SNR_S) SHL 4) OR
                      Get_Sigma(Cell[icell].SNR_L);

          if ((Cell[icell].Phase AND $80) = 0) then
            DMflag := 0
          else
            DMflag := 1;

          Writeln(fINTP, icell:3,
                         The_Cell^.Range/100                        :8:3,
                         The_Cell^.Bearing/10                       :13:1,
                         (The_Cell^.LatCenterinRadians*57.29578)    :8:2,
                         (The_Cell^.LonCenterinRadians*57.29578)    :8:2,
```

File Name: RBCONV2.PAS

```
                                Temp_RB^.RB.Bearing                 :9,
                                Trail_RB^.RB.Bearing                :7,
                                Get_SNR(Cell[icell].SNR_S)/8        :8:1,
                                Get_SNR(Cell[icell].SNR_L)/8        :11:1,
                                (Cell[icell].Phase AND $7F)         :6,
                                Cell[icell].X_Ang                   :9,
                                DMflag                              :6,
                                tmpMLIMR/10                         :7:1);
            end;

********************************************* *)

          The_Cell := The_cell^.next;

        END;  {FOR icell}
        write(ftemp, Cell);          { put the batch of cells into a file }
        writeln;
(*
        if (GMT = 1) or (GMT = 12) or (GMT = 24) then
          close(fINTP);
*)
      END;  { FOR GMT }
    END;  { FOR imonth }
  END;  {FOR selected_Station }

  Close(ftemp);

  Assign(ffinal, outpath+dbname); Rewrite(ffinal);

  { Open and massage the temporary file into the output format }
  Writeln('Converting the temporary file . . .');
    FOR icell := 1 to Numb_Cells DO BEGIN
       Write(iCell:4);
       Reset(ftemp);
       FOR selected_Station := 1 to 8 DO BEGIN
          FOR imonth := 1 to 4 DO BEGIN
             FOR GMT := 1 to 24 DO BEGIN
                Read(ftemp, Cell);
                d[imonth,GMT] := Cell[icell];
             END; {GMT}
          END; {imonth}
          Write(ffinal, d);
       END;  {Station}
       Close(ftemp);                 { so we can reopen it at the beginning }
    END;  {icell}
    Close(ffinal);
END.
```

File Name: CONV_DB.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                     55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Collection of cell definition variables and structures that  *
*           are used in converting from the range/bearing database to    *
*           the cell format database.                                    *
*                                                                        *
*************************************************************************}
CONST
   MAXATOMS                      = 200;
   digits : array[0..9] of char = ('0','1','2','3','4','5','6','7','8','9');
   MAXSORTLENGTH                 = 1000;
   NUMSTANDARD_RADIALS           = 36;

TYPE
   rng_atom = RECORD
         Range       : Integer; { megameters }
         SPM1Amp     : Integer; { Short-path Mode 1 Amplitude (dB) }
         SPM1Phase   : Integer; { Short-path Mode 1 Phase (cecs) }
         SPMSumAmp   : Integer; { Short-path Mode Sum Amplitude (dB) }
         SPMSumPhase : Integer; { Short-path Mode Sum Phase (cecs) }
         LPMSumAmp   : Integer; { Long-path Mode Sum Amplitude (dB) }
      END;
```

File Name: CONV_DB.PAS

```
   RB_Record = RECORD
         Bearing :  Integer;                          { deg*10 }
         X_Angle :  Integer;                          { crossing angle }
         a       : array[0..MAXATOMS] of rng_atom;    { the range atoms }
      END;

   RB_ptr = ^RB_Hold;
   RB_Hold = RECORD                        { for linked list storage in memory }
         next     : RB_ptr;
         NumAtoms : Integer;               { Actual # of atoms with data }
         RB       : RB_Record;
      END;

   RB = RECORD
          Range   : Integer;               { in 10 km steps }
                                           { note: (10 km * 100) = 1 megameter }
          Bearing : integer;               { degrees * 10 }
        END;
   Cell_Defn = RECORD
         Lat1,
         Lat2    : Integer;                { S,N extents in deg*10 }
         Lon1,
         Lon2    : Integer;                { W,E extents in deg*10 }
         xlate   : array[1..8] of RB;      { range/bearing of cell center }
                                           { relative to each transmitter }
         Weight  : Integer;                { weight for computing coverage}
      END;

   Cell_defn_ptr = ^Cell_defn_hold;
   Cell_defn_hold = RECORD                 { structure for memory linked lists }
         next                : Cell_defn_ptr;
         Lat1,Lat2           : Integer;          { S,N extents in deg*10 }
         Lon1,Lon2           : Integer;          { W,E extents in deg*10 }
         LatCenterInRadians,                     { center of cell - in radians }
         LonCenterInRadians  : real;             { EAST > 0, WEST < 0 }
         Range               : Integer;          { in 10 km steps }
                                           { note: (10 km * 100) = 1 megameter }
         Bearing             : real;             { deg * 10 }
         X1, X2              : Integer;          { Correspond to Lon1, Lon2 }
         Y1, Y2              : Integer;          { Correspond to Lat1, Lat2 }
         Weight              : Integer;          { weight for computing coverage}
      END;

   Cell_Record = RECORD                    { For matrix (gridded) data }
         SNR_Short : Integer;              { short-path SNR in dB*10 }
         SNR_Long  : Integer;              { long-path SNR in dB*10 }
         Phase_dev : Byte;                 { Phase deviation in cec }
         Dom_Mode  : Byte;                 { Number of the Dominant mode }
         X_Angle   : Integer;              { Crossing Angle in deg*10 }
```

```
        Coverage  : Integer;
      END;

   SortList = array[1..MAXSORTLENGTH] of integer; {Sort Buffer for
                                                   Qksort Procedure}

VAR
   RB_List: RB_Ptr;
   Cell_Defn_List: Cell_Defn_ptr;
   Cells: array[1..1800] of Cell_record;
   Numb_Cells: Integer;

   Selected_Station:Integer;      { 1=A, 2=B, etc. }
   GridSize: Integer;             { basic cell size }
   Month: Integer;                { Month number: 1 = Jan, 2=Feb, etc. }
   GMT: Integer;                  { GMT hour 12 = 1200z }

   SNRthreshold: real;          { SNR threshold for coverage computation }
   PhaseDevThreshold: real;     { Phase-Deviation threshold for coverage computation}
```

File Name: DOM_MODE.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*          UNIT NAME - Part of the RBCONV2 Program                       *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the code for computing the mode 1         *
*           dominance margin and the Mode 1 dominance flag.              *
*                                                                        *
*                                                                        *
**************************************************************************}

function ComputeDom_Mode(     Neighbors : NeighborsType;
                              W         : WIype;
                          var Cell      : CovCells;
                              icell     : integer ): integer;
{
      {
      ************************************************************************
      *                                                                      *
      *         PURPOSE                                                      *
      *            As part of the range/bearing to cell format database      *
      *            conversion, compute the mode 1 dominance margin and flag. *
      *            The mode 1 flag is set if mode 1 DOES NOT dominate the    *
      *            interfering mode signals by at least 6 dB (as hardcoded   *
      *            below).                                                   *
      ************************************************************************
```

```
File Name: DOM_MODE.PAS

      }
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

   ver  0    11/90  EMAustvold

   We will compute the ratio between Model and the Interfering Modes
   and store that in Model_to_InterferingModeRatio. This ratio is then
   compared to the DominantModeThreshold which will allow us
   to set the Dominant Mode flag.

   ( 1 bit flag = 1 if Mode 1 is NOT dominant
                = 0 if Mode 1 IS dominant      )

   if (Model_to_InterferingModeRatio < DominantModeThreshold) then
     Set the flag

   The Dominant Mode flag is the highest bit in the Phase Deviation byte.

   Again, algorithms are courtesy of PBMorris.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
}

const
  DominantModeThreshold = 6.0;  (* in dB *)

var
  Sum                          : Real;
  i                            : Integer;
  Model_to_InterferingModeRatio : real;
  DeltaX,
  DeltaPhi                     : Real;

begin
  Sum := 0;
  for i := 1 to 4 do
  begin
    {*****************
    Note: DeltaPhi is divided by 100 because we need the result of
          SPM1Phase - SPMSumPhase to be in cycles, not centicycles.
     *****************}

    DeltaPhi := (Neighbors[i]^.SPM1Phase - Neighbors[i]^.SPMSumPhase)/100;

    {*****************
    Note: DeltaX is divided by 10 because we need the result of
          SPM1Amp - SPMSumAmp to be in dB, not 10*dB.
     *****************}
```

File Name: DOM_MODE.PAS

```
    DeltaX := (Neighbors[i]^.SPMIAmp - Neighbors[i]^.SPMSumAmp)/10;

    Sum := Sum + (DeltaX - (10 * LOG10( (1 + Power(10,DeltaX/10.0) -
          (2 * POWER(10,DeltaX/20.0) * cos(2*pi*DeltaPhi) ))))) * W[i];
  end;

  Model_to_InterferingModeRatio := Sum;

  (* clip the value of the dominant mode between 0 and 25 dB *)

  if Model_to_InterferingModeRatio > 25.0 then
    Model_to_InterferingModeRatio := 25.0;

  if Model_to_InterferingModeRatio < 0.0 then
    Model_to_InterferingModeRatio := 0.0;

  if (Model_to_InterferingModeRatio < DominantModeThreshold) then
    Cell[icell].Phase := Cell[icell].Phase OR $80;  (* mode 1 not dominant *)

                (* note: Dominant mode flag = 1 if NOT dominant mode *)
                (*       and = 0 if dominant mode *)

  ComputeDom_Mode := round(Model_to_InterferingModeRatio * 10);

                (* Model_to_InterferingModeRatio is multiplied by 10 here *)
                (* so that we may insert this value into the database as  *)
                (* a number between 0 and 250. We can fit this value into *)
                (* 1 byte (range 0..255) and we can then give ourselves   *)
                (* resolution of 1/10 of a dB. *)
end;
```

File Name: GET_WGTS.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Routine to compute the relative weights of the database      *
*           values that are to be interpolated to the cell center.       *
*                                                                        *
*                                                                        *
**************************************************************************}

procedure GetWeights(VAR Neighbors        : NeighborsType;
                     VAR W                : WType;
                         The_Cell         : Cell_defn_ptr;
                     VAR Temp_RB,Trail_RB : RB_Ptr);

     {
     ************************************************************************
     *                                                                      *
     *        PURPOSE                                                       *
     *           Find the appropriate weights to use to perform the four    *
     *           corner (bilinear) interpolation to get a value for the cell*
     *           center.                                                    *
     ************************************************************************
     }
{
```

File Name: GET_WGTS.PAS

$$$

ver	0	05/88	KATench
ver	1	11/90	EMAustvold

1 : Search through the range/bearing list for 2 bearings, one which is just greater than the bearing from the station to the cell center and one which is just less than the bearing from the station to the cell center.

2 : Compute the weights of each bearing. (higher weight means the bearing is closer to the actual bearing of the station to the cell center)

3 : Search through the 2 radials which bound the cell center and find 2 range points which now bound the cell center.

4 : Compute the weights of each range point as it compares in distance from the cell center. (higher weight means that the point is closest to the cell center)

Note: range bearing data is assumed to be sorted into ascending order

$$$

```
)
var
  i : integer;
  sum : real;

begin
  Temp_RB := RB_List;
  WHILE (Temp_RB <> nil) AND
        (Temp_RB^.RB.Bearing < The_Cell^.Bearing/10) DO
  BEGIN
    Trail_RB := Temp_RB;
    Temp_RB := Temp_RB^.next;
  END;

  { get delta Bearing  and compute bearing part of weights }

  IF Temp_RB = nil THEN
  BEGIN      { wrap around? }
    Temp_RB := RB_List;
    Delta_Bearing := ((Temp_RB^.RB.Bearing + 360) - Trail_RB^.RB.Bearing);
    W[3] := 1-(((Temp_RB^.RB.Bearing+360)-The_Cell^.Bearing/10)/Delta_Bearing);
  END
  ELSE
  BEGIN
```

```
    Delta_Bearing := (Temp_RB^.RB.Bearing) - (Trail_RB^.RB.Bearing);
    W[3] := 1-((Temp_RB^.RB.Bearing - The_Cell^.Bearing/10)/Delta_Bearing);
  END;
  W[1] := 1-((The_Cell^.Bearing/10 - Trail_RB^.RB.Bearing)/Delta_Bearing);
  W[2] := W[1];
  W[4] := W[3];

{ search each radial for the range atoms which are closest }

{
 12/23/88  Note: This code DOES NOT correctly deal with the case
                 where the cell range is less than the minimum atom
                 range on either the Trail or the Temp radial which
                 bracket the cell center.  At the present, this prob-
                 lem is not corrected.  However the "Near-Field" test
                 given below eliminates this problem for the Range/Bear
                 ing data used for the Hawaii coverage effort.
}


(*

Note: In the following code, Cell^.Range is only divided by 10 so that it
      will be in the same units as the range which is read in from the
      range bearing file


*)



  i := 1;    (* START THE COUNT AT 1 BECAUSE THE 0th RECORD IS MEANINGLESS *)
             (* BECAUSE IT DOES NOT CONTAIN ANY VALID DATA. *)
  WHILE (i <= Trail_RB^.NumAtoms) AND
        (Trail_RB^.Rb.a[i].Range > 0) AND
        (Trail_RB^.RB.a[i].Range < (The_Cell^.Range/10)) DO
  BEGIN
    Inc(i);
  END;
  IF (i > Trail_RB^.NumAtoms) OR (Trail_RB^.RB.a[i].Range < 0) THEN
    Dec(i);                          { went too far. back up }

  Neighbors[2] := Addr( Trail_RB^.RB.a[i]);

  IF i > 1 THEN
  BEGIN
    Neighbors[1] := Addr( Trail_RB^.RB.a[i-1]);
    Delta_Range := Neighbors[2]^.Range - Neighbors[1]^.Range;
    W[1] := W[1] * (Neighbors[2]^.Range - The_Cell^.Range/10)/Delta_Range;
```

```
    W[2] := W[2] * (The_Cell^.Range/10 - Neighbors[1]^.Range)/Delta_Range;
  END
  ELSE
  { Cell range identical to first range point;
    use the first range point and set the second weight to zero}
  BEGIN
    Neighbors[1] := Addr( Trail_RB^.RB.a[i]);
    W[2] := 0;
  END;

  i := 1;
  WHILE (i <= Trail_RB^.NumAtoms) AND
        (Temp_RB^.RB.a[i].Range > 0) AND
        (Temp_RB^.RB.a[i].Range < The_Cell^.Range/10) DO
    Inc(i);

  IF (i > Trail_RB^.NumAtoms) OR (Temp_RB^.RB.a[i].Range < 0) THEN
    Dec(i);                          { went too far. back up }
  Neighbors[4] := Addr( Temp_RB^.RB.a[i]);

  IF i > 1 THEN
  BEGIN
    Neighbors[3] := Addr( Temp_RB^.RB.a[i-1]);
    Delta_Range := Neighbors[4]^.Range - Neighbors[3]^.Range;
    W[3] := W[3] * (Neighbors[4]^.Range - The_Cell^.Range/10)/Delta_Range;
    W[4] := W[4] * (The_Cell^.Range/10 - Neighbors[3]^.Range)/Delta_Range;
  END
  ELSE
  { Cell range identical to first range point;
    use the first range point and set the second weight to zero}
  BEGIN
    Neighbors[3] := Addr( Temp_RB^.RB.a[i]);
    W[4] := 0;
  END;
(***********************************************
  sum := 0;
  for i := 1 to 4 do
  BEGIN
    WRITELN(fLOG,'W[',i:1,'] = ',W[i]);
    SUM := sum + W[i];
  END;
  writeln(fLOG,'W1 + W2 + W3 + W4 = ',sum);
***********************************************)
END;
```

File Name: LD_NOISE.PAS

```
(*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the utility procedures necessary          *
*           to load the cell-format noise data that is used to           *
*           construct the short and long-path signal to noise ratio      *
*           (SNR) values for the PACE database.                          *
*************************************************************************)

procedure load_noise_data(Month : Integer; var Noise: NoiseType);

      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This routine reads the noise data for a particular month   *
      *           from the various noise database files.                     *
      ************************************************************************
      }

{
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

   ver 0  11/90  EMAustvold
```

File Name: LD_NOISE.PAS

```
This procedure reads in noise data which is organized by
cell (1..Numb_Cells) and by hour (GMT 1..24).  The data is stored in
a record Noise[CellNum,HourNum] and is used to compute the Signal
to Noise ratio for both the Short Path Amplitude and the Long Path
Amplitude.

Note: the value which is read in is the (Noise * 10).
      (ie. real noise value -> 23.9 = 239 <- read in from file as this)

At the present time, only the noise files for February, May, August,
and November are used to calculate noise. The reason for this is so that
we may interpolate the remaining 8 months from these 4 months as directed
by PBMorris and RRGupta.

Note: This will have to be changed in the future so that each month
      and hour in the final database will use the corresponding noise data
      for that month. Only the Signal Amplitude will be interpolated, not
      the noise data. This change will probably take place in the program
      MAKE12M.PAS.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
}


var
  NoiseFileName       : string;     (* the Noise file name                *)
  InRec               : string[30]; (* record read in from the Noise file *)
  fNoise              : text;       (* Noise file                         *)
  i,j,k,                            (* index for reading past header      *)
  Code,                             (* error code returned from val funct. *)
  CellNum,                          (* Cell number                        *)
  HourNum,                          (* Hour number                        *)
  IntTemp1            : integer;    (* temporary conversion number        *)
  freqstr             : string;

begin
  str(freq:3,freqstr);

  case Month of
     2: NoiseFileName := Noisepath + 'N_FEB.' + freqstr;(* FEBRUARY NOISE *)
     5: NoiseFileName := Noisepath + 'N_MAY.' + freqstr;(* MAY      NOISE *)
     8: NoiseFileName := Noisepath + 'N_AUG.' + freqstr;(* AUGUST   NOISE *)
    11: NoiseFileName := Noisepath + 'N_NOV.' + freqstr;(* NOVEMBER NOISE *)
  else
    NoiseFileName := Noisepath + 'N_FEB.' + freqstr;    (* FEBRUARY NOISE *)
  end; (* case stmt *)

  {$I-}
```

File Name: LD_NOISE.PAS

```
  Assign(fNoise, NoiseFileName);
  Reset(fNoise);
  {$I+}

  if (IOResult = 0) then                          (* no file error *)
  begin
    writeln;
    Writeln('Reading NOISE file ',NoiseFileName,' . . .');
    Write('%');
    for CellNum := 1 to Numb_Cells do
    begin
      if CellNum = 1 then                (* if processing a new file then *)
      begin                              (* read past the 25 lines of header *)
        for k := 1 to 25 do
          readln(fNoise, Inrec);
      end;

      if ((CellNum mod 10) = 0) then
        Write('~');

      for HourNum := 1 to 24 do
      begin
        Readln(fNoise, Inrec);
        Val(Copy(Inrec,13,4), IntTemp1, Code);
        if (Code = 0) then                    (* no error in string conversion *)
          Noise[CellNum,HourNum] := IntTemp1
        else
        begin
          writeln(fLOG,'***************');
          writeln(fLOG,'Error reading Noise file: ',NoiseFileName);
          writeln(fLOG,'CellNum = ',CellNum:4);
          writeln(fLOG,'HourNum = ',HourNum:4);
          writeln(fLOG,'Value read in from file = ^',Inrec,'^');
          writeln(fLOG,'Noise has been assigned the value of zero.');
          writeln(fLOG,'***************');
          Noise[CellNum,HourNum] := 0;
        end;
      end;
    end;
    Close(fNoise);
(*
    assign(fnoise,'noise.tmp');
    rewrite(fnoise);
    for i := 1 to Numb_cells do
    begin
      for j := 1 to 24 do
        write(fnoise,Noise[i,j]:5);
      writeln(fnoise);
    end;
```

```
    Close(fNoise);
*)

  end
  else      (* error in opening file *)
  begin
    Writeln(chr(7),'/// Noise file "',NoiseFileName,'" not found ///');
    Halt(1);
  end;
end;
```

File Name: LOADDEF.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the routines that are used by the         *
*           range/bearing to cell format database conversion code to     *
*           define the 10 by 10 degree cell grid.                        *
*                                                                        *
*************************************************************************}
(************************************************************************)

procedure GenerateCellCenterInRadians( var Cell : Cell_Defn_Ptr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           For a given cell, find the lat/lon of the cell center.     *
      *                                                                      *
      ************************************************************************
      }

  (*
    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

     ver 0  11/02/90   EMAustvold
```

File Name: LOADDEF.PAS

```
     This procedure should use the North, South, East, and West extents
     which are supplied by the Cell which is passed in. The extents
     are in (degrees * 10) and should be used to create the cell center
     in radians with (East > 0), (West < 0), (North > 0) and (South < 0).

     example of algorithm :

        lat1   lat2    lat center
         20     30         25
         10     20         15
          0     10          5
        -10      0         -5
        -20    -10        -15

    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
  *)
  const
    radiansperdegree = 0.0174533;
  var
    latcenter,
    loncenter   : real;

begin
  latcenter := (Cell^.lat1 + abs(Cell^.lat2 - Cell^.lat1)/2)/10;
  loncenter := (Cell^.lon1 + abs(Cell^.lon2 - Cell^.lon1)/2)/10;

  Cell^.LonCenterInRadians := loncenter * radiansperdegree;

  Cell^.LatCenterInRadians := latcenter * radiansperdegree;

end;

(*****************************************************************************)

PROCEDURE Load_Cell_Definition(     station,
                                    cellsize       : Integer;
                               VAR Cell_Defn_List : Cell_defn_ptr;
                               VAR Numb_Cells     : Integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Load the 10 by 10 degree cell definitions into the            *
      *           data structure used by RBCONV2. Similar to a routine          *
      *           that is contained in the load_def.pas program file.           *
      ***************************************************************************
      }
(
```

```
File Name: LOADDEF.PAS

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

   Load cell definition for indicated station and cellsize into memory as a
   list attached to the pointer Cell_Defn_List.  Cellsize is the basic size
   (size at the equator) in degrees (e.g., 5, 10).
   Numb_Cells is set to the number of cells read in.

   The format of the file name is:    GRID_nn.DEF
      Where nn is the basic cell size in degrees

   Ver 0     05/88  KATench
   Ver 1     05/90  KATench   Added Cell_Defn_List Disposal (for P_RBConv)
   Ver 2     11/90  EMAustvold

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
)
VAR
   Cdefnfname: string[40];
   Last_Cell, Temp_cell: Cell_defn_Ptr;( cell definition list )
   Temp_Cell_Defn: Cell_defn;          ( for reading in the definition )
   fCdefn: file of Cell_Defn;
   i: Integer;

BEGIN
   ( Clear out the old list if need be )
   Last_Cell :=  Cell_Defn_List;
   WHILE Last_Cell <> nil DO BEGIN
      Temp_Cell := Last_Cell^.next;
      Dispose(Last_Cell);
      Last_Cell := Temp_Cell;
   END;

   ( figure out the name of the file to use )
   Cdefnfname:='GRID_'+DIGITS[Cellsize div 10]+DIGITS[Cellsize mod 10]+'.DEF';
   ( Read in and store the cell definition file )
   Writeln;
   Writeln('Loading Cell Definition . . .');
   {$i-}
   Assign(fCdefn, Cdefnfname); Reset(fCdefn);
   {$I+}
   IF IOResult = 0 THEN BEGIN

      Read(fCdefn, Temp_cell_defn);
      New( Cell_Defn_List );

      Cell_Defn_List^.lat1 := Temp_cell_defn.lat1;
      Cell_Defn_List^.lat2 := Temp_cell_defn.lat2;
      Cell_Defn_List^.lon1 := Temp_cell_defn.lon1;
      Cell_Defn_List^.lon2 := Temp_cell_defn.lon2;
```

```
      Cell_Defn_List^.range := Temp_cell_defn.xlate[Station].range;
      Cell_Defn_List^.bearing := Temp_cell_defn.xlate[Station].bearing;
      GenerateCellCenterInRadians(Cell_Defn_List);
      Last_Cell := Cell_Defn_List;
      Numb_Cells := 1;
      WHILE NOT EOF(fCdefn) DO BEGIN
         Inc(Numb_Cells);
         New( Temp_Cell );
         Read(fCdefn, Temp_cell_defn);      { get a new definition and store away }
         Temp_Cell^.lat1 := Temp_cell_defn.lat1;
         Temp_Cell^.lat2 := Temp_cell_defn.lat2;
         Temp_Cell^.lon1 := Temp_cell_defn.lon1;
         Temp_Cell^.lon2 := Temp_cell_defn.lon2;
         Temp_Cell^.range := Temp_cell_defn.xlate[Station].range;
         Temp_Cell^.bearing := Temp_cell_defn.xlate[Station].bearing;
         GenerateCellCenterInRadians(Temp_Cell);
         Last_cell^.next := Temp_cell;    { link new one into the list }
         Last_cell := Temp_cell;
        END;
      Last_cell^.next := nil;             { mark the last link as the end }
      Close(fCdefn);

  {print the list to the screen to see if it's in memory OK }
(*
      Temp_Cell := Cell_Defn_List;
      i := 0;
      REPEAT
         Inc(i);
         WITH Temp_Cell^ DO BEGIN
            Write(i:4,'  Lats ',(Lat1/10):7:1, (Lat2/10):7:1);
            Write('  Lons ',(lon1/10):7:1, (Lon2/10):7:1);
            Writeln('  !  Station R/B: ',(Range/100):6:2, Bearing:7:1);
         END;
         Temp_Cell := Temp_Cell^.next;
      UNTIL Temp_Cell = nil;
*)
   END
   ELSE BEGIN
      Writeln(chr(7),'/// Cell Definition File "',Cdefnfname,'" not found ///');
      Cell_Defn_List := nil;
      Numb_cells := 0;
      Halt(1);
   END;

END;
```

File Name: LOAD_RB.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains utility code to read and parse the        *
*           radial format signal data needed by the range/bearing to     *
*           cell format database conversion program.                     *
*                                                                        *
*************************************************************************}

PROCEDURE Load_Range_Bearing_Data( Station, Month, GMT,
                                   Freq: Integer;
                                  VAR RB_List:RB_Ptr);
     {
     ************************************************************************
     *                                                                      *
     *        PURPOSE                                                       *
     *           Construct the appropriate range/bearing data file name,    *
     *           read in the range/bearing data and parse the input data    *
     *           to obtain the desired signal quantities.                   *
     ************************************************************************
     }
{
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
File Name: LOAD_RB.PAS

     Ver 1    05/88   KATench
     ver 2    11/88   DPFrank     to process reformatted Range/Bearing file
     ver 3    05/90   KATench     added RB_List disposal and hardwired sta C
                                  & added file name creation from input params
     ver 4    10/90   EMAustvold  Remove hardwired station C.
                                  This version should hardwire the month of
                                  February.
     ver 5    11/90   EMAustvold  Change to read new data input format
                                  Hardwired NonStandardRadials

new format :
line 1 - 6 is header information

 STA=A MON=Feb GMT=01 S_PATH BEAR=000 FREQ=10.2

 DIS  SPM1 SPM1 SPMS SPMS LPMS
      AMP  PHS  AMP  PHS  AMP
 *10  *10       *10       *10
 MM   DB   CEC  DB   CEC  DB
   0  999    0  999    0 -999

lines 7 - number of radials contains data in the columns

cols (01 - 04)  - distance along radial in tenths of a MegaMeter
                  (ie 1 = 0.1, 2 = 0.2,... )
cols (05 - 09)  - Short Path Mode 1 Amplitude (dB)
cols (10 - 14)  - Short Path Mode 1 Phase (cecs)
cols (15 - 19)  - Short Path Mode Sum Amplitude (dB)
cols (20 - 24)  - Short Path Mode Sum Phase (cecs)
cols (25 - 29)  - Long Path Mode Sum Amplitude (cecs)

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
}

const
  NumberOfNonStandardRadials = 14;

type
  NonStandardRadialType =
                array[1..8,1..NumberOfNonStandardRadials] of integer;
                (* 8 stations x 14 possible NonStandardRadials *)
const
  NumberRecordsPerRadial = 201;       (* records numbered 0 - 200 *)
  StepsizeAlongRadial    = 0.10;      (* one tenth of a megameter *)

  NonStandardRadials : NonStandardRadialType =
              (* A *)((  45, 225, 105, 285, 115, 295, 000,
                        000, 000, 000, 000, 000, 000, 000  ),
              (* B *) (  17, 197, 156, 336, 165, 345, 175,
```

File Name: LOAD_RB.PAS

```
                        355, 178, 358, 000, 000, 000, 000 );
                (* C *) (   5, 185,  25, 205,  27, 207, 152,
                        332, 163, 343, 167, 347, 000, 000 ),
                (* D *) (   8, 188,  34, 214,  44, 224, 165,
                        345, 000, 000, 000, 000, 000, 000 ),
                (* E *) (  13, 193,  23, 203, 148, 328, 155,
                        335, 167, 347, 174, 354, 000, 000 ),
                (* F *) (   5, 185,  16, 196,  22, 202, 152,
                        332, 175, 355, 178, 358, 000, 000 ),
                (* G *) (   5, 185,  18, 198, 165, 345, 173,
                        353, 000, 000, 000, 000, 000, 000 ),
                (* H *) (   2, 182,   5, 185,  26, 206, 165,
                        345, 172, 352, 175, 355, 000, 000 ) );

VAR
   Last_RB,
   Temp_RB      : RB_Ptr;  ( station range-bearing data list )
   fRB          : text;
   fDIAG        : text;
   RBfname      : string[40];
   i,j,k        : integer;
   Inrec        : string [80];
   RBFileName   : string [40];
   RealTemp     : real;
   IntTemp1     : integer;
   Code         : integer;
   Radials      : SortList;
   NumberRadials: integer;
   freqstr,
   MonthName    : String[3];
   stationstr   : string[1];

BEGIN
  { Dispose of old list if need be }
  Last_RB := RB_List;
  WHILE Last_RB <> nil DO BEGIN
    Temp_RB := Last_RB^.next;
    Dispose(Last_RB);
    Last_RB := Temp_RB;
  END;

  {
    Generate file name of the form smmmtt.fff
    where:
       s   = sta        (A,B,C,D,E,F,G,H)
       mmm = month      (FEB,MAY,AUG,NOV)  ( 4 months we're concerned with.)
       tt  = GMT        (01..24)
       fff = frequency  (102 or 136)
   }
```

File Name: LOAD_RB.PAS

```
  case Month of
     1 : MonthName := 'JAN';
     2 : MonthName := 'FEB';
     3 : MonthName := 'MAR';
     4 : MonthName := 'APR';
     5 : MonthName := 'MAY';
     6 : MonthName := 'JUN';
     7 : MonthName := 'JUL';
     8 : MonthName := 'AUG';
     9 : MonthName := 'SEP';
    10 : MonthName := 'OCT';
    11 : MonthName := 'NOV';
    12 : MonthName := 'DEC';
  else
    MonthName := 'XXX';
  end; (* case stmt *)

   str(freq:3,freqstr );
   stationstr := chr(ord(digits[station])+16);

   RBFileName := rbpath +
                 stationstr +
                 MonthName +
                 digits[GMT DIV 10] +
                 digits[GMT MOD 10] +
                 '.' +
                 freqstr;

 {$I-}
 Assign(fRB, RBFileName);
 Reset(fRB);
 {$I+}
 if (IOResult = 0) then       (* no error *)
 begin
   writeln;
   Writeln('Reading Range/Bearing file ',RBFileName,' . . .');

   {Assign Bearings for "Standard" Radials}
   For i := 1 to NUMSTANDARD_RADIALS Do     (* 36 standard radials *)
     Radials[i] := 10*(i-1);                (* 0..350, every 10 degrees *)

   {*********************************************************}
   {Input Bearings for "Additional" Radials                   }
   {*********************************************************}

   NumberRadials := NUMSTANDARD_RADIALS;
   i := 1;
   While (i <= NumberOfNonStandardRadials ) and
         (NonStandardRadials[Station,i] <> 0) Do
```

File Name: LOAD_RB.PAS

```
    BEGIN
      Radials[NUMSTANDARD_RADIALS+i] := NonStandardRadials[Station,i];
      Inc(i);
    END;

    NumberRadials := NUMSTANDARD_RADIALS + (i-1);

                                (Sort Radials if Necessary)

    If (NumberRadials > NUMSTANDARD_RADIALS) Then
      Qksort(Radials, 1, NumberRadials);

    {*******************************}
    {Process Range/Bearing Records:}
    {*******************************}

    Write('@');

    for j := 1 to NumberRadials do
    begin
      Write('.');
      New( Temp_RB );
      Temp_RB^.NumAtoms := NumberRecordsPerRadial;
      If j = 1 Then
      BEGIN
        RB_List := Temp_RB;
        Last_RB := RB_List;
      END
      Else
      BEGIN
        Last_RB^.next := Temp_RB;
        Last_RB := Temp_RB;
      END;

      Temp_RB^.RB.Bearing := Radials[j];

      For i := 0 to (NumberRecordsPerRadial-1)Do
      BEGIN
        if i = 0 then              (* if processing a new radial then *)
        begin                      (* read past the 6 lines of header *)
          for k := 1 to 6 do
            readln(fRB, Inrec);
        end;

        Readln(fRB, Inrec);

        Val(Copy(Inrec,1,4), IntTemp1, Code);
        if (Code = 0) then
          Temp_RB^.RB.a[i].Range := IntTemp1
```

```
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Range values');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Range has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].Range := 0;
        end;

        Val(Copy(Inrec,6,4), IntTemp1, Code);   (Short-path Mode 1 Amp)
        if (Code = 0) then
          Temp_RB^.RB.a[i].SPM1Amp := IntTemp1
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Short Path Mode 1 Amplitude');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Short Path Mode 1 Amplitude has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].SPM1Amp := 0;
        end;

        Val(Copy(Inrec,11,4), IntTemp1, Code);  (Short-path Mode 1 Phase)
        if (Code = 0) then
          Temp_RB^.RB.a[i].SPM1Phase := IntTemp1
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Short Path Mode 1 Phase');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Short Path Mode 1 Phase has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].SPM1Phase := 0;
```

```
        end;

        Val(Copy(Inrec,16,4), IntTempl, Code);  {Short-path Modesum Amp}
        if (Code = 0) then
          Temp_RB^.RB.a[i].SPMSumAmp := IntTempl
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Short Path Mode Sum Amplitude');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTempl:4,'^');
          writeln(flog,'Short Path Mode Sum Amplitude has been assigned the value of zero.')
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].SPMSumAmp := 0;
        end;

        Val(Copy(Inrec,21,4), IntTempl, Code);  {Short-path Modesum Phase}
        if (Code = 0) then
          Temp_RB^.RB.a[i].SPMSumPhase := IntTempl
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Short Path Mode Sum Phase');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTempl:4,'^');
          writeln(flog,'Short Path Mode Sum Phase has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].SPMSumPhase := 0;
        end;

        Val(Copy(Inrec,26,4), IntTempl, Code);  {Long-path Modesum Amp}
        if (Code = 0) then
          Temp_RB^.RB.a[i].LPMSumAmp := IntTempl
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Long Path Mode Sum Amplitude');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
```

File Name: LOAD_RB.PAS

```
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Long Path Mode Sum Amplitude has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].LPMSumAmp := 0;
        end;


      END;
    END;
    Last_RB^.next := nil;
    Close(fRB);


(*********************)
{
    Assign(fDIAG, 'DIAG.OUT');
    Rewrite(fDIAG);
    Temp_RB := RB_List;
    writeln;
    Writeln(fDIAG, Temp_RB^.NumAtoms);
    While (Temp_RB<>nil) Do
    BEGIN
      Writeln('@', Temp_RB^.RB.Bearing:5);
      Writeln(fDIAG,Temp_RB^.RB.Bearing:5);
      For i := 0 to (Temp_RB^.NumAtoms-1) Do
        Writeln(fDIAG,i:5,Temp_RB^.RB.a[i].Range:5,
                          Temp_RB^.RB.a[i].SPM1Amp:5,
                          Temp_RB^.RB.a[i].SPM1Phase:5,
                          Temp_RB^.RB.a[i].SPMSumAmp:5,
                          Temp_RB^.RB.a[i].SPMSumPhase:5,
                          Temp_RB^.RB.a[i].LPMSumAmp:5);
        Temp_RB := Temp_RB^.next;

    END;
    Close(fDIAG);
}
(*********************)


  end             { Range/Bearing Processing }
  else            { error reading file }
  begin
    Writeln(chr(7),'/// Range/Bearing file "',RBFileName,'" not found ///');
    Halt(1);
  end;
end;
```

File Name: LOG10.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Utility routine to perform base 10 log.                      *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************}

Function Log10(Value:Real) : Real;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Return the log base 10 value of the input number.          *
      *                                                                      *
      ************************************************************************
      }

{Function to Compute the Base 10 Logarithm of "Value"}

BEGIN
  Log10 := Ln(Value)/Ln(10.0);
  END;
```

File Name: PACE_RB.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*          UNIT NAME - Part of the RBCONV2 and MAKE12M Programs          *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*          This file contains utility routines and definitions for       *
*          the PACE 10 by 10 degree cell database. THESE ROUTINES ARE    *
*          DUPLICATES OF THOSE CONTAINED IN THE DATAUTIL.PAS PROGRAM     *
*          FILE -- THEY ARE REPEATED HERE FOR DEVELOPMENT PURPOSES OF    *
*          THE RBCONV2 AND MAKE12M PROGRAMS.                             *
**************************************************************************}
{ ************************************************************************
  *
  *        PACE cell database Data Definitions and utility routines
  *
  *        ver 0     05/90    KATench
  *        ver 1     11/90    EMAustvold
  *
  ************************************************************************}

TYPE
   { cell coverage information for 1 sta/month/hr}
   CovCell = RECORD
      SNR_S: Integer;
           { Short-path SNR (db*8) w/ Sigma_S in uppermost nybble (db*8) }
      SNR_L: Integer;
```

```
            { Long-path SNR (db*8) w/ nothing in uppermost nybble }
      Phase: Byte;
            { Phase Deviation with mode 1 flag in top bit (0=Mode 1) }
      X_Ang: Byte;
            { Crossing Angle }
   END;

   { record for disk I/O - 4 months, 24 hours of coverage information }
   { Disk file is organized as 8 stations worth of coviorec for each of 444 cells }
   coviorec_4m = array[1..4,1..24] of CovCell;
   coviorec_12m = array[1..12,1..24] of CovCell;  { 12 month version }

   { Noise data information for 1 month/hr }
   NoiseCell = RECORD
      Sigma_N: Byte;     { Noise standard deviation (db*8) }
      Noise: Byte;       { Noise value (db*8) }
   END;

   { record for disk I/O - 12 months, 24 hours of noise information }
   { Disk file is organized as 444 cells worth of noiseiorec }
   noiseiorec = array[1..12,1..24] of CovCell;


{ Routines to pack and unpack the fields of CovCell }

FUNCTION Pack_SNR(Sigma, SNR:Integer): Integer;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Pack the sigma value into top nibble of the SNR word in the*
      *           cell format signal database. The sigma value is actually   *
      *           half of the mode 1 dominance margin byte value             *
      *           as documented in the PACE final report. The variable called*
      *           sigma is shifted left by 12 bits and ORed into the SNR     *
      *           value.                                                     *
      ************************************************************************
      }
{ Pack the sigma value into top nybble of SNR }
BEGIN
   Pack_SNR := ((Sigma(*-12*)) SHL 12) OR ( SNR AND $FFF);  {12 = 1.5*8 db}
END;

FUNCTION Get_SNR(SNR_in: Integer): Integer;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Remove the lower three nibbles of information from a word  *
```

```
      *          length variable with a quantity packed into the upper      *
      *          nibble (the upper four bits). Sign extend by filling in    *
      *          the upper nibble with ones if the upper bit of the third   *
      *          nibble (bit # 12 is set).                                  *
      ***************************************************************************
      }
{ remove the top nybble and sign extend to make a full integer }
BEGIN
   SNR_in := SNR_in AND $FFF;
   IF (SNR_in AND $800) <> 0 THEN      { Need to extend sign bit ? }
      Get_SNR := SNR_in OR $F000
   ELSE
      Get_SNR := SNR_in;
END;

FUNCTION Get_Sigma(SNR_in: Integer): Integer;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Remove the value stored in the upper nibble of a word value*
      *          by masking off the lower 12 bits and then shifting the     *
      *          result right by 12.                                            *
      ***************************************************************************
      }
{ Extract sigma from top nybble }
BEGIN
   Get_Sigma := ((SNR_in AND $F000) SHR 12)(*+12*) (i.e., 1.5*8);
END;
```

File Name: PHASE.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the code necessary to compute the phase   *
*           difference for the database signals.                         *
*                                                                        *
*                                                                        *
*************************************************************************}

procedure ComputePhase(    Neighbors : NeighborsType;
                           W         : WIype;
                       var Cell      : CovCells;
                           icell     : integer );

      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Compute the phase deviation for the cell center using the  *
      *           neighboring points on the closest radials.                 *
      *                                                                      *
      ************************************************************************
      }
{
```

```
File Name: PHASE.PAS

 Algorithm to compute the phase
   deviation is courtesy of PBMorris. Thanks Pete.

}
var
  Sum,diff : Real;
  i   : Integer;

begin
  Sum := 0;
  FOR i := 1 to 4 DO
  begin
    diff := ABS(Neighbors[i]^.SPMSumPhase -
                        Neighbors[i]^.SPM1Phase) MOD 100;
    if (diff > 50) then
       diff := 100.0 - diff;

    Sum := Sum + (diff * W[i]);
  end;

  Cell[icell].Phase := round(Sum);
end;
```

File Name: POWER.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Utility routines to compute an exponent.                     *
*                                                                        *
*                                                                        *
*                                                                        *
*************************************************************************}

Function Power(Base, Exponent: Real) : real;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Function to Compute "Base**Exponent"                       *
      *                                                                      *
      ************************************************************************
      }

BEGIN
  Power := Exp( Exponent*Ln(Base) );
END;
```

File Name: QKSORT.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Provides a gereral sorting routine.                          *
*                                                                        *
*                                                                        *
*                                                                        *
**************************************************************************}

{ Modified version of Borland QSORT program}
{$R-}
{$S-}

procedure qksort(var a: SortList; Lo,Hi: integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           QUICKSORT sorts elements in the array A with indices between*
      *           LO and HI (both inclusive). Note that the QUICKSORT proce- *
      *           dure provides only an "interface" to the program. The actual*
      *           processing takes place in the SORT procedure, which executes*
      *           itself recursively.                                           *
      *                                                                         *
```

File Name: QKSORT.PAS

```
      *                                                                      *
      ************************************************************************
      }


procedure sort(l,r: integer);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Sort is a recursive sorting procedure that sorts a list    *
      *           between the values l and r.                                *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
var
  i,j,x,y: integer;
begin
  i:=l; j:=r; x:=a[(l+r) DIV 2];
  repeat
    while a[i]<x do i:=i+1;
    while x<a[j] do j:=j-1;
    if i<=j then
    begin
      y:=a[i]; a[i]:=a[j]; a[j]:=y;
      i:=i+1; j:=j-1;
    end;
  until i>j;
  if l<j then sort(l,j);
  if i<r then sort(i,r);
end;

begin {qksort};
  sort(Lo,Hi);
end;
```

File Name: SNRLONG.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                             (PACE)                                     *
*                          Workstation                                   *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the code to compute the long-path         *
*           SNR value for the cell database from the range/bearing       *
*           datatbase.                                                   *
*                                                                        *
*************************************************************************}

procedure ComputeSNRLong(    Neighbors : NeighborsType;
                             W         : WType;
                         var Cell      : CovCells;
                             Noise,
                             icell,
                             MLIMR       : integer );

     {
     *************************************************************************
     *                                                                       *
     *        PURPOSE                                                        *
     *           This procedure performs the four corner interpolaton of the*
     *           radial data to obtain the long-path SNR at the cell center.*
     *************************************************************************
     }
```

File Name: SNRLONG.PAS

```
(*
    ver 01  11/90  EMAustvold

 1. Find the long-path signal amplitude for each cell by computing
    the weighted average of the signal amplitudes for the closest
    neighboring points on the adjacent radials.

 2. After converting, subtract the noise for the cell.
    (note: the noise and the signal amplitude are scaled by a factor
           of 10, ie 999 = 99.9)

 3. After the noise is subtracted out, the result is scaled by 8 so that
    we may store the value in increments of 1/8 dB.

 4. Pack the SNR into the lowest 1 1/2 bytes of the SNR_L variable of
    the dataset. (note: SNR_L is an integer (2 bytes). We use the highest
                        nybble of SNR_L to store the lowest nybble of
                        the Model_to_InterferingModeRatio  (MLIMR) )

*)

var
  Sum : Real;
  i   : Integer;

begin
  Sum := 0;
  FOR i := 1 to 4 Do
  begin
    Sum := Sum + (Neighbors[i]^.LPMSumAmp/10) * W[i];
  end;

  Cell[icell].SNR_L := Round( (Sum - (Noise/10)) * 8 );

  if Cell[icell].SNR_L > SNR_THRESHOLD then
    Cell[icell].SNR_L := SNR_THRESHOLD;

  if Cell[icell].SNR_L < -SNR_THRESHOLD then
    Cell[icell].SNR_L := -SNR_THRESHOLD;

    (* put the lower nybble of MLIMR in the upper nybble of the *)
    (* long path snr *)

  Cell[icell].SNR_L := Pack_SNR((MLIMR AND $0F), Cell[icell].SNR_L);

end;
```

File Name: SNRSHORT.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                  and                                   *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the code that computes the short-path     *
*           SNR values for the cell format database from the range/      *
*           bearing database.                                            *
*                                                                        *
**************************************************************************}
procedure ComputeSNRShort(    Neighbors : NeighborsType;
                              W         : WType;
                          var Cell      : CovCells;
                              Noise,
                              icell,
                              MTIMR       : integer );

      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Routine to get the cell center short-path SNR by performing*
      *           the four corner interpolation of the radial values.        *
      *************************************************************************
      }
(*
```

```
File Name: SNRSHORT.PAS

    ver 01  11/90  EMAustvold

 1. Find the short-path signal amplitude for each cell by computing
    the weighted average of the signal amplitudes for the closest
    neighboring points on the adjacent radials.

 2. After converting, subtract the noise for the cell.

    (note: the noise and the signal amplitude are scaled
           by a factor of 10, ie 999 = 99.9)

 3. After the noise is subtracted out, the result is scaled by 8 so that
    we may store the value in increments of 1/8 dB.

 4. Pack the SNR into the lowest 1 1/2 bytes of the SNR_S variable of
    the dataset. (note: SNR_S is an integer (2 bytes). We use the highest
                        nybble of SNR_S to store the highest nybble of
                        the Model_to_InterferingModeRatio  (M1IMR) )

*)

var
  Sum : Real;
  i   : Integer;

begin
  Sum := 0;
  for i := 1 to 4 do
  begin
    Sum := Sum + (Neighbors[i]^.SPMSumAmp/10) * W[i]
  end;

  Cell[icell].SNR_S := Round( (Sum - (Noise/10)) * 8 );

  if Cell[icell].SNR_S > SNR_THRESHOLD then
    Cell[icell].SNR_S := SNR_THRESHOLD;

  if Cell[icell].SNR_S < -SNR_THRESHOLD then
    Cell[icell].SNR_S := -SNR_THRESHOLD;

    (* put the upper nybble of the M1IMR into the upper nybble of the *)
    (* short path SNR *)

  Cell[icell].SNR_S := Pack_SNR(((M1IMR AND $F0) SHR 4),Cell[icell].SNR_S);

end;
```

File Name: X_ANGLE.PAS

```
{*****************************************************************************
*                                                                            *
*        PROGRAM NAME - Omega Performance Assessment                         *
*                                and                                         *
*                          Coverage Evaluation                               *
*                              (PACE)                                        *
*                            Workstation                                     *
*                                                                            *
*           UNIT NAME - Part of the RBCONV2 Program                          *
*                                                                            *
*****************************************************************************
*                                                                            *
*        This program was prepared by                                        *
*                                                                            *
*              The Analytic Science Corporation (TASC)                       *
*                    55 Walkers Brook Drive                                  *
*                Reading, Massachusetts 01867                                *
*                                                                            *
*        PACE has been developed to run on a IBM PC/AT or compatible         *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main           *
*        memory and an EGA or compatible graphics adapter and color          *
*        monitor. This work was performed under contract number              *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for           *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.        *
*                                                                            *
*****************************************************************************
*                                                                            *
*        PURPOSE                                                             *
*           This file contains the code necessary to compute the path        *
*           /terminator crossing angle.                                      *
*                                                                            *
*                                                                            *
*****************************************************************************}

procedure ComputeX_Angle(     The_Cell     : Cell_Defn_Ptr;
                          var Cell         : CovCells;
                              iCell,
                              StationNum,
                              Month,
                              Hour         : integer );
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *           Computes the path/terminator crossing angle for the cell      *
     *           center for the time of interest.                              *
     ***************************************************************************
     }
{
```

```
File Name: X_ANGLE.PAS

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

    ver   0   11/90   EMAustvold
    ver   1   11/90   EMAustvold   added 1 line of code to do conversion
                                   of the latitude to geocentric coordinates

    This procedure computes the angle between the bearing of the signal
    which passes through the cell and the day/night terminator. The internal
    procedure TIME provides the necessary vectors to compute the day/night
    terminator.

    Algorithms courtesy of PEMorris. Aren't they wonderful?


$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
)

type

  StationVec = array[1..11,1..3] of real;

  SunVecType = Record
                 S1 : real;
                 S2 : Array[1..20] of real;
                 S3 : Array[1..20] of real;
               end;

const

  DegreesPerRadian = 57.29578;

    (*

       StationUnitVectors is the unit vectors of all stations.

    *)
  StationUnitVectors : StationVec =
                       ((  0.9155132 ,  0.3917598 ,   0.09143179),
                        (  0.1091006 ,  0.9768618 ,  -0.1839513 ),
                        (  0.3628323 , -0.8629653 ,  -0.3516303 ),
                        (  0.7214421 , -0.100388  ,  -0.6851595 ),
                        ( -0.3558544 ,  0.5321513 ,   0.7682336 ),
                        ( -0.6802296 ,  0.3075657 ,  -0.6653505 ),
                        ( -0.6196952 , -0.6577405 ,   0.4282004 ),
                        (  0.5654711 , -0.5241008 ,   0.6368367 ),
                        (  0.68521564,  0.18745213,  -0.70380482),
                        (  0.18445728,  0.46688533,  -0.86486624),
                        (  0.0       ,  0.0       ,   0.0       ));
```

```
File Name: X_ANGLE.PAS


var
  i                 : Integer;
  SunVector         : SunVecType;

  CosXAng,
  SinXAng,
  CosTransmitter,
  CosReceiver,
  CosTheta,
  SinTheta          : real;

  T_R_CrossProduct,
  ReceiverVector    : array [1..3] of real;

(*************************************************************************)

  procedure time(     Year,Month,Day,Hour,Minute: Integer;
                  Var SunVector : SunVecType);
      (
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Utility routine that calculates the sun vector for a given *
      *           set of input conditions.                                   *
      ************************************************************************
      )
  const
    NumDaysInYear    = 365;
    SecondsInFullDay = 86400;
    MinutesInFullDay = 1440;
    Factor1:real     = 0.017202791;
    Factor2:real     = 4.8821549;
    Factor3:real     = 0.01720197;
    Factor4:real     = 6.2367035;
    Factor5:real     = 0.03344403;
    Factor6:real     = 0.00034955625;
    Factor7:real     = 0.91745322;
    Factor8:real     = 0.39784368;

(*

  NOTE: In this local procedure pay particular attention to the conversion
        between reals and integers!!!!!!!!!!!!!! VERY IMPORTANT!!!!


*)

  var
```

File Name: X_ANGLE.PAS

```
    YearsSince68,
    DaysSince68,
    i                   : integer;

    RI,
    NEN,
    NumSecondsInDay,
    SUNTL,
    SunLon,
    SSPVC2,
    Past,
    EquationOfTime,
    RightAscension,
    EM,
    ElevationAngle,
    GMT,
    TimeofDay           : real;

  begin
    YearsSince68 := Year - 68;
    DaysSince68 :=
           (NumDaysInYear * YearsSince68) + ((YearsSince68+3) DIV 4) +
           ( ((YearsSince68+4) DIV 4)-((YearsSince68+3) DIV 4) - 3 ) *
           ((Month+9) DIV 12) + 31 * (Month-1)-((Month+7) DIV 12)-
           ((Month+5) DIV 12)-((Month+2) DIV 12)-
           (Month DIV 12) + Day - 1;

(*
   AS PER THE ABOVE NOTE, CONVERSION BETWEEN INTEGERS AND REALS IS
   EXTREMELY IMPORTANT IN THE NEXT LINE OF CODE!!!!!
*)

    NumSecondsinDay := Minute*60.0 + Hour*3600.0;

    TimeofDay := DaysSince68 + NumSecondsinDay/SecondsInFullDay;
    GMT := TimeofDay - DaysSince68;
    ElevationAngle := TimeofDay * Factor1 + Factor2;
    NEN := trunc(ElevationAngle / (2*pi));
    ElevationAngle := ElevationAngle - NEN*(2*pi);
    EM := TimeofDay * Factor3 + Factor4;
    SUNTL := ElevationAngle + Factor5 * sin(EM) + Factor6 * sin(2*EM);

{    RightAscension := arctan2(Factor7*sin(SUNTL),cos(SUNTL));}

    RightAscension := arctan(Factor7*sin(SUNTL)/cos(SUNTL));
    if (cos(SUNTL) < 0.0) then
      RightAscension := RightAscension + pi;
    if (RightAscension < 0.0) then
      RightAscension := RightAscension + (2*pi);
```

File Name: X_ANGLE.PAS

```
    EquationOfTime := ElevationAngle - RightAscension;
    SunLon := pi * (1 - (2*GMT)) - EquationOfTime;
    SunVector.S1 := Factor8 * sin(SUNTL);
    SSPVC2 := sqrt(1 - (SunVector.S1)*(SunVector.S1));
{  took out the loop cause we only use the first result anyway
    for i := 1 to 20 do
    begin
      RI := 6 * (i-1);
      Past := SunLon + 2*pi*RI/MinutesInFullDay;
}
      Past := SunLon;
      SunVector.S2[1] := SSPVC2 * cos(Past);
      SunVector.S3[1] := SSPVC2 * sin(Past);

{

    end;
}
  end;

(*************************************************************************)
begin

  time( 90 (*Year*),Month, 15 (*Day*),Hour, 0 (*Minute*),SunVector);

  (* conversion of the Latitude to GeoCentric coordinates *)

{  The_Cell^.LatCenterInRadians := arctan2(0.99330562 *
       sin(The_Cell^.LatCenterInRadians),cos(The_Cell^.LatCenterInRadians));}

  The_Cell^.LatCenterInRadians := arctan(0.99330562 *
       sin(The_Cell^.LatCenterInRadians)/cos(The_Cell^.LatCenterInRadians));

  ReceiverVector[1] := sin(The_Cell^.LatCenterInRadians);
  ReceiverVector[2] := cos(The_Cell^.LatCenterInRadians) *
                       cos(The_Cell^.LonCenterInRadians);
  ReceiverVector[3] := cos(The_Cell^.LatCenterInRadians) *
                       sin(The_Cell^.LonCenterInRadians);

  CosTheta := (StationUnitVectors[StationNum,1] * ReceiverVector[1]) +
              (StationUnitVectors[StationNum,2] * ReceiverVector[2]) +
              (StationUnitVectors[StationNum,3] * ReceiverVector[3]);

  SinTheta := sqrt(1-CosTheta*CosTheta);

  T_R_CrossProduct[1] := (StationUnitVectors[StationNum,2] * ReceiverVector[3]) -
                         (StationUnitVectors[StationNum,3] * ReceiverVector[2]);

  T_R_CrossProduct[2] := (StationUnitVectors[StationNum,3] * ReceiverVector[1]) -
```

```
                        (StationUnitVectors[StationNum,1] * ReceiverVector[3]);

  T_R_CrossProduct[3] := (StationUnitVectors[StationNum,1] * ReceiverVector[2]) -
                         (StationUnitVectors[StationNum,2] * ReceiverVector[1]);

{ took the following out trying to match PBM's algorithm
  GRD 11/19/90

  for i := 1 to 3 do
    T_R_CrossProduct[i] := T_R_CrossProduct[i] / SinTheta;
}

{ added / sintheta to the following sum to match PBM's algorithm }
  CosXAng := ((T_R_CrossProduct[1] * SunVector.S1   ) +
              (T_R_CrossProduct[2] * SunVector.S2[1]) +
              (T_R_CrossProduct[3] * SunVector.S3[1])  ) / SinTheta;

  SinXAng := sqrt(1 - CosXAng*CosXAng);

  CosTransmitter := (StationUnitVectors[StationNum,1] * SunVector.S1) +
                    (StationUnitVectors[StationNum,2] * SunVector.S2[1]) +
                    (StationUnitVectors[StationNum,3] * SunVector.S3[1]);

  CosReceiver := (ReceiverVector[1] * SunVector.S1) +
                 (ReceiverVector[2] * SunVector.S2[1]) +
                 (ReceiverVector[3] * SunVector.S3[1]);

  if ((CosTransmitter * CosReceiver) > 0) then
    Cell[icell].X_Ang := 90
  else
{    Cell[icell].X_Ang :=
               round(DegreesPerRadian * arctan2(abs(SinXAng),abs(CosXAng)));}
    Cell[icell].X_Ang :=
               round(DegreesPerRadian * arctan(abs(SinXAng)/abs(CosXAng)));
(*
   if icell=263 then begin
      writeln(flog,'sinxang = ',sinxang,' cosxang = ',cosxang,' arctan = ',
      arctan(abs(SinXAng)/abs(CosXAng)),' xang = ',cell[icell].x_ang);
      writeln(flog,'sunvectors are : ',sunvector.s1,' ',sunvector.s2[1],' ',sunvector.s3[1
   end;
*)
end;
```

APPENDIX C
SPLITDB LISTINGS

This appendix contains the Pascal code listings for the four-month database delivery preparation program. A discussion of this program is provided in Section 5.2.

File Name: SPLITDB.SRC

The SPLITDB program uses the following source program files as well as other PACE units:

SPLITDB.PAS

File Name: SPLITDB.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - SPLITDB Program                                  *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                     55 Walkers Brook Drive                             *
*                  Reading, Massachusetts 01867                          *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This program splits the four-month database files (one each*
*           for 10.2 and 13.6 kHz) into equal halves so that they can    *
*           fit on 1.2 MByte delivery floppies (4 total). This program   *
*           is as part of the PACE delivery preparation process.         *
*************************************************************************}
Program break_4m_db_in_half;
USES DOS;

CONST
  Blocksize = 10240;

VAR
   i,size: longint;
   Fin,Ffirsthalf,Fsecondhalf: file;
   out1,out2: string;
   Dir: dirstr;
   Name: Namestr;
   Ext: Extstr;
   Buffer:array[1..blocksize] of byte;

begin
```

File Name: SPLITDB.PAS

```
   IF ParamCount = 1 THEN
   begin
      Assign(Fin,Paramstr(1));
      Reset(Fin,1);
      size := filesize(Fin);
      Close(Fin);
      Fsplit(Paramstr(1),Dir,Name,Ext);
      Assign(FFirsthalf,Dir+Name+'.111');
      Assign(FSecondhalf,Dir+Name+'.222');
      Reset(Fin,1);
      Rewrite(FFirstHalf,1);
      Rewrite(FSecondHalf,1);
      FOR i := 1 TO round(size/blocksize/2) DO
      begin
         blockread(Fin,Buffer,blocksize);
         blockwrite(FFirstHalf,buffer,blocksize);
      end;
      FOR i := round(size/blocksize/2)+1 TO trunc(size/blocksize) DO
      begin
         blockread(Fin,Buffer,blocksize);
         blockwrite(FSecondHalf,buffer,blocksize);
      end;
      for i := 1 to round(size - blocksize*trunc(size/blocksize)) DO
      begin
         blockread(Fin,Buffer,1);
         blockwrite(FSecondHalf,buffer,1);
      end;
      Close(Fin);
      Close(FFirstHalf);
      Close(FSecondHalf);
   end
   ELSE
   begin
      writeln;
      writeln('Usage: splitdb infile');
   end;
end.
```

APPENDIX D
INSTALL LISTINGS

This appendix contains the Pascal code listings for the PACE installation program. A discussion of this program is provided in Section 5.7.

File Name: INSTALL.SRC

The INSTALL program uses the following source program files as well as other PACE units:

INSTALL.PAS

File Name: INSTALL.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - INSTALL Program                                  *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Perform the installation and setup of the PACE workstation   *
*           code on the user's machine. Includes initiating the          *
*           four-month to 12-month database conversion program.          *
*                                                                        *
*************************************************************************}
{$M 4096,0,16384}
program Install;

uses  dos,crt;

const
  PaceInitFileName = 'INITPACE.TXT';
  ENTER   = 13;
  ESC     = 27;
  F01     = 59+127;
  F02     = 60+127;
  F03     = 61+127;
  F04     = 62+127;
  F05     = 63+127;
  F06     = 64+127;
  F07     = 65+127;
```

File Name: INSTALL.PAS

```
  F08     = 66+127;
  F09     = 67+127;
  F10     = 68+127;

  PacePathLabel                = 'Pace Path :                 ';
  ArchivePathLabel             = 'Archive Path :              ';
  TemporaryArchivePathLabel    = 'Temporary Archive Path :    ';
  QrPathLabel                  = 'Qr Path :                   ';
  WeightsPathLabel             = 'Weights Path :              ';
  CovDBPathLabel               = 'Coverage Database Path :    ';
  HelpPathLabel                = 'Help Path :                 ';

  QuitLabel                    = 'Quit                        ';
  PaceDriveLabel               = 'Pace Drive :                ';
  ArchiveDriveLabel            = 'Archive Drive :             ';
  QrDriveLabel                 = 'Qr Drive :                  ';
  WeightsDriveLabel            = 'Weights Drive :             ';
  CovDBDriveLabel              = 'Coverage Database Drive :   ';
  HelpDriveLabel               = 'Help Drive :                ';
  ContinueLabel                = ' Continue Installation      ';

  DefaultPaceDrive               = 'C:';
  DefaultPaceDirectory           = '\PACE';
  DefaultArchiveDrive            = DefaultPaceDrive;
  DefaultArchiveDirectory        = '\ARCHIVE';
  DefaultQrDrive                 = DefaultPaceDrive;
  DefaultQrDirectory             = '\QR';
  DefaultWeightsDrive            = DefaultPaceDrive;
  DefaultWeightsDirectory        = '\WEIGHTS';
  DefaultCovDBDrive              = DefaultPaceDrive;
  DefaultCovDBDirectory          = '\COVDB';
  DefaultHelpDrive               = DefaultPaceDrive;
  DefaultHelpDirectory           = '\HELP';


var
  tmpDrive,
  tmpDirectory,
  PaceDrive,
  PaceDirectory,
  ArchiveDrive,
  ArchiveDirectory,
  TemporaryArchiveDrive,
  TemporaryArchiveDirectory,
  QrDrive,
  QrDirectory,
  WeightsDrive,
  WeightsDirectory,
  CovDBDrive,
```

File Name: INSTALL.PAS

```
  CovDBDirectory,
  HelpDrive,
  HelpDirectory     : string;
  key               : char;
  continue,
  done              : boolean;
  NewPacePath       : string; (*PathStr; *)
  dir               : DirStr;
  name              : NameStr;
  ext               : ExtStr;
  keyvalue,
  number,
  i                 : integer;

(*************************************************************************)

procedure GetKeyInput(var key:char);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Retrieves user key input for the installation selections.     *
      *                                                                         *
      ***************************************************************************
      }
begin
  repeat
  until keypressed;
  key := readkey;

  if key = #0 then
  begin
    (* special key has been pressed, must read secondary code *)
    key := chr(ord(ReadKey) + 127);
  end;
end;

(*************************************************************************)

procedure TestKeyInput(var k : char);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility routine to double check user's intention.             *
      *                                                                         *
      ***************************************************************************
      }
begin
```

```
  writeln('    Is this Correct?');
  writeln;
  writeln(' <Y> ..... CONTINUE    ');
  writeln(' <N> ..... TRY AGAIN   ');
  writeln(' <ESC> or <Q> ... QUIT        ');
  writeln;
  repeat
  until keypressed;
  k := readkey;
end;

(*****************************************************************************)

procedure WriteHeader;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility routine to display the PACE initialization header.    *
      *                                                                         *
      ***************************************************************************
      }

begin
  textbackground(blue);
  textcolor(white);
  clrscr;
  writeln ('*** PACE INSTALLATION ***');
  writeln;
end;

(*****************************************************************************)

procedure WriteMainScreen;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility routine to display the PACE initialization menu.      *
      *                                                                         *
      ***************************************************************************
      }

begin
  writeln('Select an Option (F1 - F10)');
  writeln('');
  writeln;
  writeln('////////////////////////////////////////////////////////////////');
  write(' '); textcolor(black);  textbackground(cyan);  write('F1');
```

File Name: INSTALL.PAS

```
  textcolor(white);textbackground(blue);
  write(' ');
  write(' Modify '+PacePathLabel);
  textcolor(yellow);
  writeln(PaceDrive+PaceDirectory);
  textcolor(white);textbackground(blue);
  write(' '); textcolor(black); textbackground(cyan); write('F2');
  textcolor(white);textbackground(blue);
  write(' ');
  write(' Modify '+ArchivePathLabel);
  textcolor(yellow);
  writeln(ArchiveDrive+ArchiveDirectory);
  textcolor(white);textbackground(blue);
  write(' '); textcolor(black); textbackground(cyan); write('F3');
  textcolor(white);textbackground(blue);
  write(' ');
  write(' Modify '+QrPathLabel);
  textcolor(yellow);
  writeln(QrDrive+QrDirectory);
  textcolor(white);textbackground(blue);
  write(' '); textcolor(black); textbackground(cyan); write('F4');
  textcolor(white);textbackground(blue);
  write(' ');
  write(' Modify '+WeightsPathLabel);
  textcolor(yellow);
  writeln(WeightsDrive+WeightsDirectory);

  textcolor(white);textbackground(blue);
  write(' '); textcolor(black); textbackground(cyan); write('F5');
  textcolor(white);textbackground(blue);
  write(' ');
  write(' Modify '+CovDBPathLabel);
  textcolor(yellow);
  writeln(CovDBDrive+CovDBDirectory);

  textcolor(white);textbackground(blue);
  write(' '); textcolor(black); textbackground(cyan); write('F6');
  textcolor(white);textbackground(blue);
  write(' ');
  write(' Modify '+HelpPathLabel);
  textcolor(yellow);
  writeln(HelpDrive+HelpDirectory);

  textcolor(white);textbackground(blue);
  writeln;
  write(' '); textcolor(black); textbackground(cyan); write('F9');
  textcolor(white);textbackground(blue);
  write(' ');
  write(ContinueLabel); writeln;
```

```
  write(' ');  textcolor(black);  textbackground(cyan);  write('F10');
  textcolor(white);textbackground(blue);
  write(' ');
  write(QuitLabel); writeln;
  writeln('////////////////////////////////////////////////////////////////');
end;

(*************************************************************************)

procedure ChangePath(DriveLabel,PathLabel,Drive,Directory : string);
      {
      ***************************************************************************
      *                                                                         *
      *       PURPOSE:                                                          *
      *          Utility routine to change the path that the PACE executable*
      *          will reside in.                                                *
      ***************************************************************************
      }

var
  alldone : boolean;

begin
  repeat
    alldone := true;
    clrscr;
    WriteHeader;
    writeln;
    writeln(PathLabel+Drive+Directory);
    writeln;
    write('Enter New Drive and Path : ');
    readln(NewPacePath);
    writeln;
    for i := 1 to length(NewPacePath) do
      NewPacePath[i] := upcase(NewPacePath[i]);
    fsplit(NewPacePath, dir,name,ext);
    tmpDrive     := copy(dir,1,2);
    tmpDirectory := '';
    if (tmpDrive <> '') then
    begin
      tmpDirectory := copy(dir,3,length(dir))+name + ext;
      writeln('New ' + DriveLabel+ tmpDrive);
      writeln('New ' + PathLabel + tmpDirectory);
    end
    else
    begin
      textbackground(red);
      textcolor(white);
      writeln;
```

```
      writeln('InCorrect Drive Selected... Please Try Again');
      writeln;
      textcolor(white);textbackground(blue);
      alldone := false;
    end;
    writeln;
    writeln('Press any key to continue...');
    GetKeyInput(key);
    if (ord(key) = ESC) then
      alldone := true;
  until alldone;
end;

(*****************************************************************************)

procedure CreateDirectory(dirpath:string);
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Utility routine to create the necessary individual               *
      *           directories.                                                     *
      ******************************************************************************
      }

var
  dirinfo : searchrec;
  temp    : string;
  NumSubDirectories,
  count,
  i,
  SubDirCount : integer;

begin
  temp := dirpath;
  i := 0;
  NumSubdirectories := 0;
  for i := 1 to length(dirpath) do
  begin
    temp[i] := ' ';
    if dirpath[i] = '\' then
      inc(NumSubdirectories);
  end;

  for SubDirCount := 2 to NumSubDirectories+1 do
  begin
    count := 0;
    i := 0;
    while (i < length(dirpath)) and (count <> SubDirCount) do
```

```
    begin
      inc(i);
      temp[i] := dirpath[i];
      if dirpath[i+1] = '\' then
        inc(count);
    end;
      findfirst(temp,Directory,dirinfo);
      if (DosError = 3) or (DosError = 18) then
        mkdir(temp);
  end;
end;

(*****************************************************************************)

procedure MakeDirectories;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Utility routine to create the selected  PACE directory           *
      *           structure.                                                       *
      ******************************************************************************
      }
var
  dirinfo : searchrec;
begin
  findfirst(PaceDrive+PaceDirectory,Directory,dirinfo);
  if (DosError = 3) or (DosError = 18) then
    createDirectory(PaceDrive+PaceDirectory);

  findfirst(ArchiveDrive+ArchiveDirectory,Directory,dirinfo);
  if (DosError = 3) or (DosError = 18)  then
    createDirectory(ArchiveDrive+ArchiveDirectory);

  findfirst(QrDrive+QrDirectory,Directory,dirinfo);
  if (DosError = 3) or (DosError = 18)  then
    createDirectory(QrDrive+QrDirectory);

  findfirst(WeightsDrive+WeightsDirectory,Directory,dirinfo);
  if (DosError = 3) or (DosError = 18)  then
    createDirectory(WeightsDrive+WeightsDirectory);

  findfirst(CovDBDrive+CovDBDirectory,Directory,dirinfo);
  if (DosError = 3) or (DosError = 18)  then
    createDirectory(CovDBDrive+CovDBDirectory);

  findfirst(HelpDrive+HelpDirectory,Directory,dirinfo);
  if (DosError = 3) or (DosError = 18)  then
    createDirectory(HelpDrive+HelpDirectory);
```

```
File Name: INSTALL.PAS

end;

(*****************************************************************************)

procedure CheckAvailableDiskSpace;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Routine to ensure that there is enough disk space to          *
      *           install PACE. Checks all proposed PACE directories.           *
      ***************************************************************************
      }
CONST
   PaceDriveSize = 800000;
   ArchiveDriveSize = 650000;
   QRDriveSize = 30000;
   WeightsDriveSize = 1000;
   CovDBDriveSize = 15000000;
   HelpDriveSize = 90000;

VAR
   i:integer;
   requiredspace:array[1..26] of longint;

   function drivenum(driveletter:string):integer;
   begin
      drivenum := ord(upcase(driveletter[1]))-$40;
   end;

   function drivelet(drivenumber:integer):string;
   begin
      drivelet := chr(drivenumber+$40) + ':';
   end;

begin
   (* first mark all selected drives *)
   FOR i := 0 to 25 DO
      requiredspace[i] := 0;

   requiredspace[drivenum(PaceDrive)]       :=
      requiredspace[drivenum(PaceDrive)] + PaceDriveSize;
   requiredspace[drivenum(ArchiveDrive)]    :=
      requiredspace[drivenum(ArchiveDrive)] + ArchiveDriveSize;
   requiredspace[drivenum(QRDrive)]         :=
      requiredspace[drivenum(QRDrive)] + QRDriveSize;
   requiredspace[drivenum(WeightsDrive)]    :=
      requiredspace[drivenum(WeightsDrive)] + WeightsDriveSize;
   requiredspace[drivenum(CovDBDrive)]      :=
```

```
      requiredspace[drivenum(CovDBDrive)] + CovDBDriveSize;
   requiredspace[drivenum(HelpDrive)]       :=
      requiredspace[drivenum(HelpDrive)] + HelpDriveSize;

   (* check the space on the specified drives *)
   FOR i := 0 to 25 DO
      IF requiredspace[i] > 0 THEN
         IF (requiredspace[i] > diskfree(i)) THEN
         begin
            clrscr;
            writeln;
            writeln('Unable to continue PACE installation due to insufficient disk space');
            writeln('on Drive ',drivelet(i),'.');
            writeln;
            writeln('For the selected configuration, at least ',requiredspace[i],' bytes');
            writeln('of free disk space must be available on Drive ',drivelet(i),'.');
            writeln;
            writeln('Please free the necessary disk space or reassign the drives');
            writeln('and rerun the PACE installation program.');
            halt;
         end;
end;

(*****************************************************************************)

(*****************************************************************************)

function check_disk_label(expected_label:string):boolean;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Utility to ensure that the user has inserted the correct         *
      *           PACE delivery diskette.                                          *
      ******************************************************************************
      }
VAR
   res:searchrec;
begin
   findfirst('*.*',VolumeID,res);
   IF res.name = expected_label THEN
      check_disk_label := true
   ELSE
      check_disk_label := false;
end;

(*****************************************************************************)

procedure CopyFiles;
```

File Name: INSTALL.PAS

```
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Utility to copy the PACE files from the delivery diskettes *
      *           to the user's hard disk. Also initiates database build.    *
      ********************************************************************************
      }
VAR
   correct_disk:boolean;
begin

  correct_disk := check_disk_label('PACE1');
  while NOT correct_disk DO
  begin
     clrscr;
     WriteHeader;
     writeln('Insert PACE Disk Labeled : Disk 1 of 5');
     writeln;
     writeln('Press any key to continue...');
     GetKeyInput(key);
     correct_disk := check_disk_label('PACE1');
  end;
  clrscr;
  WriteHeader;
  writeln;
  writeln('Copying PACE executable code...');
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY \EXE\*.* '+PaceDrive+PaceDirectory);
  swapvectors;

  writeln;
  writeln('Copying PACE Reliability files');
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY \QR\*.* '+QrDrive+QrDirectory);
  swapvectors;

  writeln;
  writeln('Copying PACE Weights files');
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY \WEIGHTS\*.* '+WeightsDrive+WeightsDirectory);
  swapvectors;

  writeln;
  writeln('Copying PACE Coverage Database files');
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY \COVDB\*.* '+CovDBDrive+CovDBDirectory);
  swapvectors;
```

File Name: INSTALL.PAS

```
  writeln;
  writeln('Copying PACE Help files');
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY \HELP\*.* '+HelpDrive+HelpDirectory);
  swapvectors;

  correct_disk := FALSE;
  while NOT correct_disk DO
  begin
     clrscr;
     WriteHeader;
     writeln('Insert PACE Disk Labeled : Disk 2 of 5');
     writeln;
     writeln('Press any key to continue...');
     GetKeyInput(key);
     correct_disk := check_disk_label('PACE2');
  end;
  writeln('Copying PACE Coverage Database files');
  writeln;
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY *.* '+CovDBDrive+CovDBDirectory);
  swapvectors;

  correct_disk := FALSE;
  while NOT correct_disk DO
  begin
     clrscr;
     WriteHeader;
     writeln('Insert PACE Disk Labeled : Disk 3 of 5');
     writeln;
     writeln('Press any key to continue...');
     GetKeyInput(key);
     correct_disk := check_disk_label('PACE3');
  end;
  writeln('Copying PACE Coverage Database files');
  writeln;
  swapvectors;
  exec(Getenv('COMSPEC'),'/C COPY *.* '+CovDBDrive+CovDBDirectory);
  swapvectors;

  correct_disk := FALSE;
  while NOT correct_disk DO
  begin
     clrscr;
     WriteHeader;
     writeln('Insert PACE Disk Labeled : Disk 4 of 5');
     writeln;
     writeln('Press any key to continue...');
     GetKeyInput(key);
```

```
    correct_disk := check_disk_label('PACE4');
 end;
 writeln('Copying PACE Coverage Database files');
 writeln;
 swapvectors;
 exec(Getenv('COMSPEC'),'/C COPY *.* '+CovDBDrive+CovDBDirectory);
 swapvectors;

 correct_disk := FALSE;
 while NOT correct_disk DO
 begin
    clrscr;
    WriteHeader;
    writeln('Insert PACE Disk Labeled : Disk 5 of 5');
    writeln;
    writeln('Press any key to continue...');
    GetKeyInput(key);
    correct_disk := check_disk_label('PACE5');
 end;
 writeln('Copying PACE Coverage Database files');
 writeln;
 swapvectors;
 exec(Getenv('COMSPEC'),'/C COPY *.* '+CovDBDrive+CovDBDirectory);
 swapvectors;

 clrscr;
 WriteHeader;
 writeln;
 writeln;
 writeln('Combining the partial PACE Coverage Database files');
 writeln;

 ChDir(CovDBDrive+CovDBDirectory);

 swapvectors;
 exec(Getenv('COMSPEC'),'/C COPY /b 102covdb.111 + 102covdb.222 102covdb.4m');
 exec(Getenv('COMSPEC'),'/C DEL 102covdb.111');
 exec(Getenv('COMSPEC'),'/C DEL 102covdb.222');
 exec(Getenv('COMSPEC'),'/C COPY /b 136covdb.111 + 136covdb.222 136covdb.4m');
 exec(Getenv('COMSPEC'),'/C DEL 136covdb.111');
 exec(Getenv('COMSPEC'),'/C DEL 136covdb.222');
 swapvectors;

 clrscr;
 WriteHeader;
 writeln;
 writeln;
 writeln('Making the PACE 12 month Coverage Database files');
 writeln('This process may take up to one hour .....');
```

```
  writeln;
  swapvectors;
  exec('Make12m.exe','');
  exec(Getenv('COMSPEC'),'/C DEL 102covdb.4m');
  exec(Getenv('COMSPEC'),'/C DEL 136covdb.4m');
  swapvectors;
  ChDir(PaceDrive+PaceDirectory);
end;

(*************************************************************************)

procedure CreateInitFile;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Utility to create the text file that contains the PACE     *
      *           environment setup information used at PACE startup.        *
      ************************************************************************
      }

var
  f : text;

{ ********** Main INSTALL program code *********** }
begin
  assign(f,PaceDrive+PaceDirectory+'\'+PaceInitFileName);
  rewrite(f);
  writeln(f,PaceDrive);
  writeln(f,PaceDirectory);
  writeln(f,ArchiveDrive);
  writeln(f,ArchiveDirectory);
  writeln(f,QrDrive);
  writeln(f,QrDirectory);
  writeln(f,WeightsDrive);
  writeln(f,WeightsDirectory);
  writeln(f,CovDBDrive);
  writeln(f,CovDBDirectory);
  writeln(f,HelpDrive);
  writeln(f,HelpDirectory);
  writeln(f,'DEFAULT.QR');       (* default qr file *)
  writeln(f,'DEFAULT.WGT');      (* default weights file *)
  writeln(f,'102COVDB.12M');     (* default coverage database name *)
  writeln(f,'136COVDB.12M');     (* default coverage database name *)
  writeln(f,'CONTEXT.HLP');      (* associates a context to a help file *)
  writeln(f,'HELP.FLS');         (* lists all help files in order  *)
  close(f);
end;
```

File Name: INSTALL.PAS

```
(*****************************************************************************)

begin
  done := false;
  PaceDrive                 :=   DefaultPaceDrive;
  PaceDirectory             :=   DefaultPaceDirectory;
  ArchiveDrive              :=   DefaultArchiveDrive;
  ArchiveDirectory          :=   PaceDirectory+DefaultArchiveDirectory;
  QrDrive                   :=   DefaultQrDrive;
  QrDirectory               :=   PaceDirectory+DefaultQrDirectory;
  WeightsDrive              :=   DefaultWeightsDrive;
  WeightsDirectory          :=   PaceDirectory+DefaultWeightsDirectory;
  CovDBDrive                :=   DefaultCovDBDrive;
  CovDBDirectory            :=   PaceDirectory+DefaultCovDBDirectory;
  HelpDrive                 :=   DefaultHelpDrive;
  HelpDirectory             :=   PaceDirectory+DefaultHelpDirectory;

  while not done do
  begin

    WriteHeader;
    WriteMainScreen;
    GetKeyInput(key);

    if upcase(key) = 'Q' then
      keyvalue := ESC
    else keyvalue := ord(key);

    case (keyvalue) of
      F01   : begin
                ChangePath(PaceDriveLabel,PacePathLabel,PaceDrive,PaceDirectory);
                if (ord(key) <> ESC) then begin
                  PaceDrive        := tmpDrive;
                  PaceDirectory    := tmpDirectory;
                  ArchiveDrive     := tmpDrive;
                  ArchiveDirectory := PaceDirectory + DefaultArchiveDirectory;
                  TemporaryArchiveDrive := tmpDrive;
                  TemporaryArchiveDirectory := PaceDirectory + DefaultArchiveDirector
                  QrDrive          := tmpDrive;
                  QrDirectory      := PaceDirectory + DefaultQrDirectory;
                  WeightsDrive     := tmpDrive;
                  WeightsDirectory := PaceDirectory + DefaultWeightsDirectory;
                  CovDBDrive       := tmpDrive;
                  CovDBDirectory   := PaceDirectory + DefaultCovDBDirectory;
                  HelpDrive        := tmpDrive;
                  HelpDirectory    := PaceDirectory + DefaultHelpDirectory;
                end;
                done := false;
              end;
```

File Name: INSTALL.PAS

```
      F02  : begin
               ChangePath(ArchiveDriveLabel,ArchivePathLabel,ArchiveDrive,ArchiveDirectory)
               if (ord(key)) <> ESC then begin
                 ArchiveDrive     := tmpDrive;
                 ArchiveDirectory := tmpDirectory;
               end;
               done := false;
             end;

      F03  : begin
               ChangePath(QrDriveLabel,QrPathLabel,QrDrive,QrDirectory);
               if (ord(key)) <> ESC then begin
                 QrDrive     := tmpDrive;
                 QrDirectory := tmpDirectory;
               end;
               done := false;
             end;

      F04  : begin
               ChangePath(WeightsDriveLabel,WeightsPathLabel,
                          WeightsDrive,WeightsDirectory);
               if (ord(key)) <> ESC then begin
                 WeightsDrive     := tmpDrive;
                 WeightsDirectory := tmpDirectory;
               end;
               done := false;
             end;

      F05  : begin
               ChangePath(CovDBDriveLabel,CovDBPathLabel,
                          CovDBDrive,CovDBDirectory);
               if (ord(key)) <> ESC then begin
                 CovDBDrive     := tmpDrive;
                 CovDBDirectory := tmpDirectory;
               end;
               done := false;
             end;

      F06  : begin
               ChangePath(HelpDriveLabel,HelpPathLabel,
                          HelpDrive,HelpDirectory);
               if (ord(key)) <> ESC then begin
                 HelpDrive     := tmpDrive;
                 HelpDirectory := tmpDirectory;
               end;
               done := false;
             end;

      F07  : begin
```

```
            end;
      F08   : begin
            end;
      F09   : begin
                CheckAvailableDiskSpace;
                MakeDirectories;
                CopyFiles;
                CreateInitFile;
                textbackground(black);
                clrscr;
                writeln('Exit Initialization Program');
                done := true;
              end;

      ESC,
      F10   : begin
                textbackground(black);
                clrscr;
                writeln('Exit Initialization Program');
                done := true;
              end;
    end; (* case *)
  end;
end.
```

APPENDIX E
MAKE12M LISTINGS

This appendix contains the Pascal code listings for the four-month to 12-month database conversion program. A discussion of this program is provided in Section 5.3.

File Name: MAKE12M.SRC

The MAKE12M program uses the following source program files as well as other PACE units:

MAKE12M.PAS
PACE_RB.PAS

File Name: MAKE12M.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - MAKE12M Program                                  *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This program handles the interpolation of the four-month     *
*           signal database to the 12-month format. Both the 10.2 kHz    *
*           and the 13.6 kHz database halves are interpolated.           *
*                                                                        *
*************************************************************************}

{$R+}
PROGRAM Make_12_month_db;
Uses Crt,DOS;
{
     Convert the four month databases to 12 month databases

     ver 0   6/5/90   KATench
}

   FUNCTION Interpolate( Major, Minor: Integer): Integer;
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Utility function to perform 1/3-2/3 interpolation of two    *
```

File Name: MAKE12M.PAS

```
      *                    input values                                    *
      ********************************************************************
      }
   { Do the two-thirds/one-third interpolation in integer math }
   BEGIN
      Interpolate := (Major + Major + Minor) DIV 3;
   END;

TYPE
   interptype = RECORD
      major, minor: Byte;              { indices for 2/3 and 1/3 weighting }
   END;

CONST
   Twirl:string[4] = '|/-\';
   NUM_DBS = 2;
   infile: array[1..NUM_DBS] of string[20] = ('102COVDB.4M','136COVDB.4M');
   outfile: array[1..NUM_DBS] of string[20] = ('102COVDB.12M','136COVDB.12M');
   { pointers from 1..12 to the 1..4 organization of months
     w/ interpolation weighting indicated }
   Interp: array[1..12] of interptype = (
       (major:1; minor:4), (major:1; minor:1), (major:1; minor:2),
       (major:2; minor:1), (major:2; minor:2), (major:2; minor:3),
       (major:3; minor:2), (major:3; minor:3), (major:3; minor:4),
       (major:4; minor:3), (major:4; minor:4), (major:4; minor:1));

{$R-}
{$I PACE_RB.PAS }
{$R+}

VAR
   fin: file of coviorec_4m;           { File variables }
   fout: file of coviorec_12m;
   inbuffer: coviorec_4m;              { data buffers }
   outbuffer: coviorec_12m;
   iDB, i, icell, imonth, itime: Integer;     { indices/counters }
   SNR_Short, SNR_Long, mode_1_diff_hi,mode_1_diff_lo (*Sigma*), Phase_dev, X_angle: arra
   Dcm_Flag: Byte;
   x, y: Integer;

BEGIN
   FOR iDB := 1 to NUM_DBS DO BEGIN
      {$I-}
      Assign(fin, infile[iDB]); Reset(fin);
      {$I+}
      IF IOResult <> 0 THEN BEGIN
         Writeln(chr(7),'/// Error opening input DB file "',infile[iDB],'" ///');
         Halt(1);
      END;
```

```
      Assign(fout, outfile[iDB]); Rewrite(fout);

      Writeln('Processing DB file # ',iDB,' . . .');
      x := WhereX;
      y := WhereY;

      { Loop through the cells - icell is simply a counter for diagnostics }
      icell := 0;
      WHILE NOT EOF(fin) DO BEGIN
         inc(icell);     { actually counts stations and cells (8*444) }
         GotoXY(x,y); Write(Twirl[(icell MOD 4)+1]);
         {$I-}
         Read(fin, inbuffer);
         {$I+}
         IF IOResult <> 0 THEN BEGIN
            Writeln(chr(7),'/// Error reading cell # ',((icell-1) DIV 8)+1,
               ' - Station # ',((icell-1) MOD 8)+1,' ///');
            Writeln('/// of DB file "',infile[iDB],'" ///');
            Halt(1);
         END;

         { Loop through the months moving or interpolating data }
         FOR imonth := 1 to 12 DO BEGIN
            { Interpolation required ? }
            IF interp[imonth].major = interp[imonth].minor THEN BEGIN
               FOR itime := 1 to 24 DO BEGIN
                  outbuffer[imonth,itime] := inbuffer[interp[imonth].major,itime];
               END;
            END
            ELSE BEGIN
               { We'll have to interpolate }
               FOR itime := 1 to 24 DO BEGIN
                  { Unload the quantities into variables }
                  SNR_Short[1] := Get_SNR(inbuffer[interp[imonth].major,itime].SNR_S);
                  SNR_Short[2] := Get_SNR(inbuffer[interp[imonth].minor,itime].SNR_S);
                  SNR_Long[1] := Get_SNR(inbuffer[interp[imonth].major,itime].SNR_L);
                  SNR_Long[2] := Get_SNR(inbuffer[interp[imonth].minor,itime].SNR_L);
                  mode_1_diff_hi[1] := Get_SIGMA(inbuffer[interp[imonth].major,itime].SNR_S)
                  mode_1_diff_hi[2] := Get_SIGMA(inbuffer[interp[imonth].minor,itime].SNR_S)
                  mode_1_diff_lo[1] := Get_SIGMA(inbuffer[interp[imonth].major,itime].SNR_L)
                  mode_1_diff_lo[2] := Get_SIGMA(inbuffer[interp[imonth].minor,itime].SNR_L)
                  Phase_Dev[1] := inbuffer[interp[imonth].major,itime].Phase AND $7F;
                  Phase_Dev[2] := inbuffer[interp[imonth].minor,itime].Phase AND $7F;
                  Dom_flag := inbuffer[interp[imonth].major,itime].Phase AND $80;
                  X_Angle[1] := inbuffer[interp[imonth].major,itime].X_Ang;
                  X_Angle[2] := inbuffer[interp[imonth].minor,itime].X_Ang;

                  { Now interpolate and load data into outbuffer }
                  outbuffer[imonth,itime].SNR_S := Pack_SNR(
```

```
                    Interpolate(mode_1_diff_hi[1], mode_1_diff_hi[2]),
                    Interpolate(SNR_Short[1], SNR_Short[2]));
              outbuffer[imonth,itime].SNR_L := Pack_SNR(
                    Interpolate(mode_1_diff_lo[1], mode_1_diff_lo[2]),
                    Interpolate(SNR_Long[1], SNR_Long[2]));
              outbuffer[imonth,itime].Phase :=
                 (Interpolate(Phase_Dev[1], Phase_Dev[2])AND $7F) OR Dom_Flag;
              outbuffer[imonth,itime].X_Ang :=
                    Interpolate(X_Angle[1], X_Angle[2]);
           END; {FOR itime}
        END;  {ELSE}
     END; {FOR imonth}

     {$I-}
     Write(fout, outbuffer);
     {$I+}
     IF IOResult <> 0 THEN BEGIN
        Writeln(chr(7),'/// Error WRITING cell # ',((icell-1) DIV 8)+1,
           ' - Station # ',((icell-1) MOD 8)+1,' ///');
        Writeln('/// of DB file "',outfile[iDB],'" ///');
        Halt(1);
     END;
  END;  {WHILE}

  GotoXY(x,y); Write(' '); GotoXY(x,y);
  Close(fin);
  Close(fout);
  exec(Getenv('COMSPEC'),'/C DEL '+infile[iDB]);
END;  {FOR iDB}

NoSound; Sound(1000); Delay(100); Sound(500); Delay(100); NoSound;

END.
```

File Name: PACE_RB.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*          UNIT NAME - Part of the RBCONV2 and MAKE12M Programs          *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains utility routines and definitions for      *
*           the PACE 10 by 10 degree cell database. THESE ROUTINES ARE   *
*           DUPLICATES OF THOSE CONTAINED IN THE DATAUTIL.PAS PROGRAM    *
*           FILE -- THEY ARE REPEATED HERE FOR DEVELOPMENT PURPOSES OF   *
*           THE RBCONV2 AND MAKE12M PROGRAMS.                            *
*************************************************************************}
{ ************************************************************************
  *
  *        PACE cell database Data Definitions and utility routines
  *
  *        ver 0     05/90    KATench
  *        ver 1     11/90    EMAustvold
  *
  ************************************************************************}

TYPE
   { cell coverage information for 1 sta/month/hr}
   CovCell = RECORD
      SNR_S: Integer;
            { Short-path SNR (db*8) w/ Sigma_S in uppermost nybble (db*8) }
      SNR_L: Integer;
```

```
File Name: PACE_RB.PAS

            { Long-path SNR (db*8) w/ nothing in uppermost nybble }
        Phase: Byte;
            { Phase Deviation with mode 1 flag in top bit (0=Mode 1) }
        X_Ang: Byte;
            { Crossing Angle }
    END;

    { record for disk I/O - 4 months, 24 hours of coverage information }
    { Disk file is organized as 8 stations worth of coviorec for each of 444 cells }
    coviorec_4m = array[1..4,1..24] of CovCell;
    coviorec_12m = array[1..12,1..24] of CovCell;  { 12 month version }

    { Noise data information for 1 month/hr }
    NoiseCell = RECORD
        Sigma_N: Byte;      { Noise standard deviation (db*8) }
        Noise: Byte;        { Noise value (db*8) }
    END;

    { record for disk I/O - 12 months, 24 hours of noise information }
    { Disk file is organized as 444 cells worth of noiseiorec }
    noiseiorec = array[1..12,1..24] of CovCell;


{ Routines to pack and unpack the fields of CovCell }

FUNCTION Pack_SNR(Sigma, SNR:Integer): Integer;
        {
        ************************************************************************
        *                                                                      *
        *         PURPOSE                                                      *
        *            Pack the sigma value into top nibble of the SNR word in the*
        *            cell format signal database. The sigma value is actually  *
        *            half of the mode 1 dominance margin byte value            *
        *            as documented in the PACE final report. The variable called*
        *            sigma is shifted left by 12 bits and ORed into the SNR    *
        *            value.                                                    *
        ************************************************************************
        }
{ Pack the sigma value into top nybble of SNR }
BEGIN
    Pack_SNR := ((Sigma(*-12*)) SHL 12) OR ( SNR AND $FFF);  {12 = 1.5*8 db}
END;

FUNCTION Get_SNR(SNR_in: Integer): Integer;
        {
        ************************************************************************
        *                                                                      *
        *         PURPOSE                                                      *
        *            Remove the lower three nibbles of information from a word *
```

```
      *          length variable with a quantity packed into the upper      *
      *          nibble (the upper four bits). Sign extend by filling in    *
      *          the upper nibble with ones if the upper bit of the third   *
      *          nibble (bit # 12 is set).                                  *
      ***************************************************************************
      }
{ remove the top nybble and sign extend to make a full integer }
BEGIN
   SNR_in := SNR_in AND $FFF;
   IF (SNR_in AND $800) <> 0 THEN      { Need to extend sign bit ? }
      Get_SNR := SNR_in OR $F000
   ELSE
      Get_SNR := SNR_in;
END;

FUNCTION Get_Sigma(SNR_in: Integer): Integer;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Remove the value stored in the upper nibble of a word value*
      *          by masking off the lower 12 bits and then shifting the     *
      *          result right by 12.                                            *
      ***************************************************************************
      }
{ Extract sigma from top nybble }
BEGIN
   Get_Sigma := ((SNR_in AND $F000) SHR 12)(*+12*) (i.e., 1.5*8);
END;
```

APPENDIX F
PACE LISTINGS

This appendix contains the Pascal code listings for the PACE workstation program. A discussion of this program is provided in Section 5.6.

File Name: PACE.SRC

The PACE program uses the following source program files as well as other PACE units:

CELLUTIL.PAS
CELLWIN.PAS
CHECKMEM.PAS
CONMAN.PAS
CONT1.PAS
CONTROLS.PAS
COVGRID.PAS
CURSROBJ.PAS
DATAUTIL.PAS
DEFS.PAS
ERRLOG.PAS
HELPMENU.PAS
LOAD_DEF.PAS
LOAD_RB.PAS
MEMOAREA.PAS
PACE.PAS
PACEINIT.PAS
PACEOBJS.PAS
PROCS.PAS
PROCS2.PAS
PROCS3.PAS
SAVEMENU.PAS
STATBARS.PAS
TASCLOGO.PAS
WORLDMAP.PAS

File Name: CELLUTIL.PAS

```
Unit cellutil; { 10 degree cell definition/generation utility }
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - CELLUTIL                                         *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*             Routines for defining and intializing the 444 cell         *
*             grid.                                                      *
*                                                                        *
*************************************************************************}

Interface

uses errlog;

CONST
   MAXATOMS = 314;
   digits: array[0..9] of char = ('0','1','2','3','4','5','6','7','8','9');

TYPE
   rng_atom = RECORD
         Range: Integer;              { Units of 10 km }
         SNR_Short: Integer;          { Short-path SNR in dB*10 }
         SNR_Long: Integer;           { Long-path SNR in dB*10 }
         Phase_dev: Byte;             { Phase deviation in cec }
         Dom_Mode: Byte;              { Number of the dominant mode }
```

File Name: CELLUTIL.PAS

```
      END;
   RB_Record = RECORD
         Bearing: Integer;              { deg*10 }
         X_Angle: Integer;              { terminator crossing angle in deg*10 }
         a: array[1..MAXATOMS] of rng_atom;   { the range atoms }
      END;

   RB_ptr = ^RB_Hold;
   RB_Hold = RECORD                     { for linked list storage in memory }
         next: RB_ptr;
         RB: RB_Record;
      END;

   Cell_Record = RECORD                 { For matrix (gridded) data }
         SNR_Short: Integer;            { short-path SNR in dB*10 }
         SNR_Long: Integer;             { long-path SNR in dB*10 }
         Phase_dev: Byte;               { Phase deviation in cec }
         Dom_Mode: Byte;                { Number of the Dominant mode }
         X_Angle: Integer;              { Crossing Angle abs(100*cos(X_ang)) }
         Coverage: Integer;             { coverage value for cell }
      END;
      Cellarray = array[1..1800] of Cell_record;
      cellarrayptr = ^Cellarray;
   RB = RECORD
           Range: Integer;          { in 10 km steps }
           Bearing: Integer;        { deg * 10 }
        END;

   Cell_Defn = RECORD
         Lat1, Lat2: Integer;           { S,N extents in deg*10 }
         Lon1, Lon2: Integer;           { W,E extents in deg*10 }
         xlate: array[1..8] of RB;      { range/bearing of cell center }
         Weight: Integer;               { weight for computing coverage }
      END;                              { relative to each transmitter }

   Cell_defn_ptr = ^Cell_defn_hold;

   Cell_defn_hold = RECORD              { structure for memory linked lists }
         next: Cell_defn_ptr;
         Lat1, Lat2: Integer;           { S,N extents in deg*10 }
         Lon1, Lon2: Integer;           { W,E extents in deg*10 }
         Range: Integer;                { in 10 km steps }
         Bearing: Integer;              { deg * 10 }
         X1, X2: Integer;               { Correspond to Lon1, Lon2 }
         Y1, Y2: Integer;               { Correspond to Lat1, Lat2 }
         Weight: Integer;               { weight for computing coverage }
      END;
```

File Name: CELLUTIL.PAS

```
VAR
   Cell_Lst        : Cell_defn_ptr;
   NCells          : Integer;

Procedure InitCellGrid;

Implementation

{$I load_def.pas}

Procedure InitCellGrid;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Initialize the cell 444 cells with their respective        *
      *           latitude/longitude coordinates for each cell edge.         *
      *           Cell latitudes and longitudes are scaled by 10 and given   *
      *           in degrees. The cells are stored in a linked list pointed  *
      *           to by Cell_lst.                                            *
      ************************************************************************
      }

VAR
  cellsize        : Integer;
  tmplst          : Cell_defn_ptr;
  i               : Integer;

begin
  cellsize := 10;
  Load_Cell_Definition(cellsize, Cell_Lst, NCells);
  tmplst := Cell_Lst;
  for i := 1 to NCells do
  begin
      tmplst^.Lat1    := tmplst^.Lat1*10;
      tmplst^.Lon1    := tmplst^.Lon1*10;
      tmplst^.Lat2    := tmplst^.Lat2*10;
      tmplst^.Lon2    := tmplst^.Lon2*10;
      tmplst := tmplst^.next;
   end;
end;

begin
   { no initialization section }
end.
```

File Name: CELLWIN.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of PACEOBJS unit                            *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*               The Analytic Science Corporation (TASC)                  *
*                     55 Walkers Brook Drive                             *
*                  Reading, Massachusetts 01867                          *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Routines for displaying and removing from the display        *
*           the summary cell, detailed cell, and difference query        *
*           display windows. Also contains facilities for obtaining the  *
*           appropriate information from the databases for display       *
**************************************************************************}

Function GetGDOP(CellNum,Coverage:longint): string;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Get the GDOP value for the particular cell/coverage        *
      *           combination from the GDOP database.                        *
      ************************************************************************
      }
VAR
   GDOPFile : file of byte;
   i,GDOPVal,numbercovering : byte;
   tempgdop : single;
   s : string;
```

File Name: CELLWIN.PAS

```
begin
   GDOPVal := ReadGdop(cellnum,coverage);

   { count the number of stations covering the cell }
   numbercovering := 0;
   FOR i := 0 TO 7 DO
      IF coverage AND ($01 SHL i) > 0 THEN
         inc(numbercovering);
   IF numbercovering >= 3 THEN
   begin
      IF GDOPVal = 255 THEN
         GetGDOP := '> 25'
      ELSE
      begin
         tempgdop := GDOPVal/10;
         str(tempgdop:2:1,s);
         GetGDOP := s;
      end;
   end
   ELSE
      GetGDOP := 'N/A';
end;

constructor CellPopWindow.Init(InitX,InitY,InitWidth,InitHeight : integer;
                               InitColor,InitBorderColor,InitTextColor,
                               InitHiliteColor,InitSelectedColor: word;
                               InitActionProc:CellActionProcedure;
                               InitShowProc:CellShowProcedure;
                               InitCellColor:word;Initeptr:emenuptr);
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *           Object initialization code for the summary cell query         *
     *           window.                                                       *
     ***************************************************************************
     }
var
  mon,hr: integer;

begin
  menu.Init(InitX,InitY,InitWidth,InitHeight,InitColor,
            InitBorderColor,2,WindowShadowWidth-2);
  Pac := 0;
  Weight := 0;
  Month   := 1;
  Hour    := 1;
  eptr    := Initeptr;
  xoffset    := 37;
```

```
  yoffset   := 27;
  cellwidth  := 10;
  cellheight := 10;
  TextColor  := InitTextColor;
  SelectedColor := InitSelectedColor;

  for mon := 1 to 12 do
    for hr := 1 to 24 do begin
      SubCells[mon,hr].Init(InitX+xoffset+((mon-1)*cellwidth),
                            InitY+yoffset+((hr-1)*cellheight),
                            cellwidth,cellheight,0,0,(mon*hr),
                            InitCellColor,InitCellColor,
                            InitBorderColor,InitHiliteColor,
                            InitActionProc,InitShowProc);
    end;
end;

(*****************************************************************************)

procedure CellPopWindow.Show;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Object method for showing the summary cell query window.         *
      *                                                                            *
      ******************************************************************************
      }

var
  i,xx,yy,mon,hr,xi,yi,tw,th : integer;
  CellNumStr,PacStr,WtStr : String;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  Window.Show;
  for mon := 1 to 12 do
    for hr := 1 to 24 do
    begin
      IF eptr^.MonthSelectors[mon].selected AND
         eptr^.HourSelectors[hr].selected THEN
      begin
         subcells[mon,hr].aboveColor := lightblue;
         subcells[mon,hr].belowcolor := lightred;
      end
      ELSE
      begin
         subcells[mon,hr].aboveColor := blue;
```

```
            subcells[mon,hr].belowcolor := red;
          end;
          SubCells[mon,hr].Show;
        end;


      setcolor(TextColor);

      settextstyle(smallfont,horizdir,4);
      settextjustify(lefttext,toptext);

      th := textheight('1')+2;
      tw := textwidth('1');
      xi := xoffset+textwidth('1')-2;
      yi := yoffset-th-1;

      (* get the cell number and display it in the window *)

      str(CellNum,CellNumStr);
      CellNumStr := '#' + CellNumStr  ;
      outtextxy(X+5,Y+yi,CellNumStr);

      Pac := HilitedCellPtr^.Pac;
      str(Pac:7:6,PacStr);
      PacStr := 'P   ' + PacStr;

      if Pac < eptr^.Psa.value.value THEN
      begin
         setfillstyle(solidfill,red);
         bar(x+4,y+3,x+5+textwidth(PacStr)+1,y+3+textheight(PacStr)+2);
      end;

      outtextxy(X+5,Y+3,PacStr);
      outtextxy(X+5+tw,Y+5,'AC');

      Weight := HilitedCellPtr^.Weight_or_Coverage;
      str(Weight:1,WtStr);
      WtStr := 'Weight '+WtStr;
      outtextxy(X+width-textwidth(WtStr)-5,Y+3,WtStr);

    { write out the month selections }

      outtextxy(X+xi+ 0*tw,Y+yi,'J');
      outtextxy(X+xi+ 2*tw,Y+yi,'F');
      outtextxy(X+xi+ 4*tw,Y+yi,'M');
      outtextxy(X+xi+ 6*tw,Y+yi,'A');
      outtextxy(X+xi+ 8*tw,Y+yi,'M');
      outtextxy(X+xi+10*tw,Y+yi,'J');
      outtextxy(X+xi+12*tw,Y+yi,'J');
```

File Name: CELLWIN.PAS

```
  outtextxy(X+xi+14*tw,Y+yi,'A');
  outtextxy(X+xi+16*tw,Y+yi,'S');
  outtextxy(X+xi+18*tw,Y+yi,'O');
  outtextxy(X+xi+20*tw,Y+yi,'N');
  outtextxy(X+xi+22*tw,Y+yi,'D');

  setfillstyle(solidfill,Selectedcolor);

  for i := 1 to 12 do
    IF eptr^.MonthSelectors[i].selected THEN bar(X-2+xi+(i-1)*2*tw,
                                                 Y+yi,
                                                 X+2+xi+(i-1)*2*tw+tw,
                                                 Y+yi+th-1);

  setcolor(TextColor);

  IF eptr^.MonthSelectors[ 1].selected THEN outtextxy(X+xi+ 0*tw,Y+yi,'J');
  IF eptr^.MonthSelectors[ 2].selected THEN outtextxy(X+xi+ 2*tw,Y+yi,'F');
  IF eptr^.MonthSelectors[ 3].selected THEN outtextxy(X+xi+ 4*tw,Y+yi,'M');
  IF eptr^.MonthSelectors[ 4].selected THEN outtextxy(X+xi+ 6*tw,Y+yi,'A');
  IF eptr^.MonthSelectors[ 5].selected THEN outtextxy(X+xi+ 8*tw,Y+yi,'M');
  IF eptr^.MonthSelectors[ 6].selected THEN outtextxy(X+xi+10*tw,Y+yi,'J');
  IF eptr^.MonthSelectors[ 7].selected THEN outtextxy(X+xi+12*tw,Y+yi,'J');
  IF eptr^.MonthSelectors[ 8].selected THEN outtextxy(X+xi+14*tw,Y+yi,'A');
  IF eptr^.MonthSelectors[ 9].selected THEN outtextxy(X+xi+16*tw,Y+yi,'S');
  IF eptr^.MonthSelectors[10].selected THEN outtextxy(X+xi+18*tw,Y+yi,'O');
  IF eptr^.MonthSelectors[11].selected THEN outtextxy(X+xi+20*tw,Y+yi,'N');
  IF eptr^.MonthSelectors[12].selected THEN outtextxy(X+xi+22*tw,Y+yi,'D');

  { now write the hour choices }
  tw := textwidth('11')+1;
  xi := (xoffset-21)+textwidth('1');
  yi := yoffset;

  setcolor(TextColor);

  outtextxy(X+xi,Y+yi+ 0*th,'01');
  outtextxy(X+xi,Y+yi+ 1*th,'02');
  outtextxy(X+xi,Y+yi+ 2*th,'03');
  outtextxy(X+xi,Y+yi+ 3*th,'04');
  outtextxy(X+xi,Y+yi+ 4*th,'05');
  outtextxy(X+xi,Y+yi+ 5*th,'06');
  outtextxy(X+xi,Y+yi+ 6*th,'07');
  outtextxy(X+xi,Y+yi+ 7*th,'08');
  outtextxy(X+xi,Y+yi+ 8*th,'09');
  outtextxy(X+xi,Y+yi+ 9*th,'10');
  outtextxy(X+xi,Y+yi+10*th,'11');
  outtextxy(X+xi,Y+yi+11*th,'12');
  outtextxy(X+xi,Y+yi+12*th,'13');
```

```
  outtextxy(X+xi,Y+yi+13*th,'14');
  outtextxy(X+xi,Y+yi+14*th,'15');
  outtextxy(X+xi,Y+yi+15*th,'16');
  outtextxy(X+xi,Y+yi+16*th,'17');
  outtextxy(X+xi,Y+yi+17*th,'18');
  outtextxy(X+xi,Y+yi+18*th,'19');
  outtextxy(X+xi,Y+yi+19*th,'20');
  outtextxy(X+xi,Y+yi+20*th,'21');
  outtextxy(X+xi,Y+yi+21*th,'22');
  outtextxy(X+xi,Y+yi+22*th,'23');
  outtextxy(X+xi,Y+yi+23*th,'24');

  setfillstyle(solidfill,selectedcolor);
  for i := 1 to 24 do
    IF eptr^.HourSelectors[i].selected THEN bar(X+xi-2,
                                                Y+1+yi+(i-1)*th,
                                                X+xi+tw,
                                                Y+yi+(i-1)*th+th);

  setcolor(TextColor);

  IF eptr^.HourSelectors[ 1].selected THEN outtextxy(X+xi,Y+yi+ 0*th,'01');
  IF eptr^.HourSelectors[ 2].selected THEN outtextxy(X+xi,Y+yi+ 1*th,'02');
  IF eptr^.HourSelectors[ 3].selected THEN outtextxy(X+xi,Y+yi+ 2*th,'03');
  IF eptr^.HourSelectors[ 4].selected THEN outtextxy(X+xi,Y+yi+ 3*th,'04');
  IF eptr^.HourSelectors[ 5].selected THEN outtextxy(X+xi,Y+yi+ 4*th,'05');
  IF eptr^.HourSelectors[ 6].selected THEN outtextxy(X+xi,Y+yi+ 5*th,'06');
  IF eptr^.HourSelectors[ 7].selected THEN outtextxy(X+xi,Y+yi+ 6*th,'07');
  IF eptr^.HourSelectors[ 8].selected THEN outtextxy(X+xi,Y+yi+ 7*th,'08');
  IF eptr^.HourSelectors[ 9].selected THEN outtextxy(X+xi,Y+yi+ 8*th,'09');
  IF eptr^.HourSelectors[10].selected THEN outtextxy(X+xi,Y+yi+ 9*th,'10');
  IF eptr^.HourSelectors[11].selected THEN outtextxy(X+xi,Y+yi+10*th,'11');
  IF eptr^.HourSelectors[12].selected THEN outtextxy(X+xi,Y+yi+11*th,'12');
  IF eptr^.HourSelectors[13].selected THEN outtextxy(X+xi,Y+yi+12*th,'13');
  IF eptr^.HourSelectors[14].selected THEN outtextxy(X+xi,Y+yi+13*th,'14');
  IF eptr^.HourSelectors[15].selected THEN outtextxy(X+xi,Y+yi+14*th,'15');
  IF eptr^.HourSelectors[16].selected THEN outtextxy(X+xi,Y+yi+15*th,'16');
  IF eptr^.HourSelectors[17].selected THEN outtextxy(X+xi,Y+yi+16*th,'17');
  IF eptr^.HourSelectors[18].selected THEN outtextxy(X+xi,Y+yi+17*th,'18');
  IF eptr^.HourSelectors[19].selected THEN outtextxy(X+xi,Y+yi+18*th,'19');
  IF eptr^.HourSelectors[20].selected THEN outtextxy(X+xi,Y+yi+19*th,'20');
  IF eptr^.HourSelectors[21].selected THEN outtextxy(X+xi,Y+yi+20*th,'21');
  IF eptr^.HourSelectors[22].selected THEN outtextxy(X+xi,Y+yi+21*th,'22');
  IF eptr^.HourSelectors[23].selected THEN outtextxy(X+xi,Y+yi+22*th,'23');
  IF eptr^.HourSelectors[24].selected THEN outtextxy(X+xi,Y+yi+23*th,'24');

  settextstyle(defaultfont,horizdir,1);
  Show_Cursor(xx,yy);
end;
```

File Name: CELLWIN.PAS

```
(*****************************************************************************)

procedure CellPopWindow.Hilite(Xpos, Ypos:word);
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Dummy object method for hilighting the summary cell query        *
      *           window. Deliberately does nothing.                               *
      ******************************************************************************
      }
begin
end;

(*****************************************************************************)

function CellPopWindow.Action(Xpos, Ypos: word):boolean;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Object method for handling queries into the month/hour           *
      *           matrix of displayed PAT values.                                  *
      ******************************************************************************
      }

var
  i,j: integer;
  dummy    : boolean;

begin
  for i := 1 to 12 do begin
    for j := 1 to 24 do begin
      month := i;
      hour := j;
      SubCellPopUp^.HilitedCellPtr := @SubCells[i,j];
      if SubCells[i,j].Action(Xpos,Ypos) then
      begin
         exit; { drop out of the loops so that the
                 hilitedcellptr doesn't change }
      end;
    end;
  end;
end;

(*****************************************************************************)

procedure CellPopWindow.Cancel;
```

File Name: CELLWIN.PAS

```
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object method for removing the summary cell query window   *
      *                 from the screen.                                     *
      ************************************************************************
      }
begin
  menu.Cancel;
  HilitedCellPtr^.Cancel;
end;

(************************************************************************)

constructor SubCellPopWindow.Init(InitX,InitY,InitWidth,InitHeight : integer;
                                  InitColor,InitBorderColor,InitTextColor,
                                  InitSelectedColor: word;
                                  InitActionProc:ActionProcedure);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object initialization code for the detailed cell query     *
      *           window.                                                    *
      ************************************************************************
      }
var
  labelarray : namearray;
begin
  xoffset := 10;
  yoffset := 20;
  Menu.Init(InitX,InitY,InitWidth,InitHeight,InitColor,
            InitBorderColor,2,WindowShadowWidth-2);
  labelarray[1] := 'A'; labelarray[2] := 'B'; labelarray[3] := 'C';
  labelarray[4] := 'D'; labelarray[5] := 'E'; labelarray[6] := 'F';
  labelarray[7] := 'G'; labelarray[8] := 'H';
  Stations.Init(InitX+20,InitY+20,10,11,InitColor,lightblue,
                InitSelectedColor,InitBorderColor,white,InitBorderColor,'',
                8,0,labelarray,horizontal);

  MDU.Init(InitX+15,InitY+130,90,12,white,black,black,white,'Coverage',InitActionProc,'C');

  BottomMenu.Init(InitX,InitY+50,InitWidth,InitHeight+60,InitColor,
                  InitBorderColor,2,WindowShadowWidth-2);

  Pat  := 0;
  TextColor := InitTextColor;
  SelectedColor := InitSelectedColor;
```

```
File Name: CELLWIN.PAS

  MDU_shown := FALSE;
  BottomMenuShown := false;
end;

(*****************************************************************************)

procedure SubCellPopWindow.FillBottomOfWindow;
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           Object method that displays the cell database information      *
      *           whenever a station is selected from the detailed cell          *
      *           query window.                                                  *
      ****************************************************************************
      }

var
  xx,yy,xi,yi    : integer;
  StationLetter : String[1];
  FREQ102,
  FREQ136,
  SNR102,
  SNR136,
  SIRATIO102,
  SIRATIO136,
  DEV102,
  DEV136,
  ANG102,
  ANG136,
  STR102,
  STR136         : string[5];
  DM102,
  DM136          : string[3];
  StationInfo102,stationinfo136: stationcoverageinfoptr;
  DB102,DB136: DBFile;
  S102,S136,L102,L136,D102,D136,A102,A136: real;
  RelString: string;
  relarray: QRDatabaseSubType;
  relval: single;
  relcolor : word;
begin

  Get_CursorXY(xx,yy);
  Hide_Cursor;

  settextstyle(smallfont,horizdir,4);
  settextjustify(lefttext,toptext);
```

File Name: CELLWIN.PAS

```
  xi := 5;
  yi := 50;

  setcolor(textcolor);

  case Stations.Picked of
    0: StationLetter := '';
    1: StationLetter := 'A';
    2: StationLetter := 'B';
    3: StationLetter := 'C';
    4: StationLetter := 'D';
    5: StationLetter := 'E';
    6: StationLetter := 'F';
    7: StationLetter := 'G';
    8: StationLetter := 'H';
  end;



  if Stations.Picked = 0 then begin

    FREQ102      := '';
    FREQ136      := '';
    SNR102       := '';
    SNR136       := '';
    SIRATIO102   := '';
    SIRATIO136   := '';
    DM102        := '   ';
    DM136        := '   ';
    DEV102       := '';
    DEV136       := '';
    ANG102       := '';
    ANG136       := '';
  end
  else begin

    Assign(DB102,DATABASEPATH+'\'+DATABASE102);
    Reset(DB102);
    Assign(DB136,DATABASEPATH+'\'+DATABASE136);
    Reset(DB136);

    new(stationinfo102);
    new(stationinfo136);

    ReadFromDatabase(DB102,CellPopUp^.cellnum,stationinfo102);
    ReadFromDatabase(DB136,CellPopUp^.cellnum,stationinfo136);

    Close(DB102);
    Close(DB136);
```

File Name: CELLWIN.PAS

```
    S102 := Get_SNR(stationinfo102^[stations.picked,CellPopUp^.month,CellPopUp^.hour].SNR_S)
    S136 := Get_SNR(stationinfo136^[stations.picked,CellPopUp^.month,CellPopUp^.hour].SNR_S)

    IF CellPopUp^.eptr^.stationpower[stations.picked].on.selected THEN
    begin
       S102 := S102+CellPopUp^.eptr^.stationpower[stations.picked].sld.value.value;
       S136 := S136+CellPopUp^.eptr^.stationpower[stations.picked].sld.value.value;
       GetStationReliabilities(relarray,CellPopUp^.eptr^.QRFile);
       CASE CellPopUp^.eptr^.StationReliabilityModel.Picked OF
       SRMBEST:
          relval := relarray[CellPopup^.Month,stations.picked,Scheduled] +
                    relarray[CellPopup^.Month,stations.picked,UnScheduled];
       SRMNOM:
          relval := relarray[CellPopup^.Month,stations.picked,Scheduled] +
                    relarray[CellPopup^.Month,stations.picked,UnScheduled] +
                    relarray[CellPopup^.Month,stations.picked,Maintenance];
       SRMWRST:
          IF relarray[CellPopup^.Month,stations.picked,Maintenance] <> 0 THEN
             relval := 0
          ELSE
             relval := relarray[CellPopup^.Month,stations.picked,Scheduled] +
                       relarray[CellPopup^.Month,stations.picked,UnScheduled] +
                       relarray[CellPopup^.Month,stations.picked,Maintenance];
       end; { CASE }
       str(relval:6:5,RelString);
       relcolor := blue;
    end
    ELSE
    begin
       RelString := ' OFF';
       relcolor  := red;
    end;

    L102 := Get_SNR(stationinfo102^[stations.picked,CellPopUp^.month,CellPopUp^.hour].SNR_L)
    L136 := Get_SNR(stationinfo136^[stations.picked,CellPopUp^.month,CellPopUp^.hour].SNR_L)
    SIR( 10.2 :5:1,FREQ102);
    SIR( 13.6 :5:1,FREQ136);
    SIR(S102:5:1,  SNR102);
    SIR(S136:5:1,  SNR136);
    SIR(S102-L102:5:1,  SIRATIO102);
    SIR(S136-L136:5:1,  SIRATIO136);
    IF (stationinfo102^[stations.picked,CellPopUp^.month,CellPopUp^.hour].Phase AND $80) = 0
       DM102      := '  1'
    ELSE
       DM102      := '  X';
    IF (stationinfo136^[stations.picked,CellPopUp^.month,CellPopUp^.hour].Phase AND $80) = 0
       DM136      := '  1'
    ELSE
```

```
      DM136      := '  X';

    D102 := stationinfo102^[stations.picked,CellPopUp^.month,CellPopUp^.hour].Phase AND $7F;
    D136 := stationinfo136^[stations.picked,CellPopUp^.month,CellPopUp^.hour].Phase AND $7F;
    STR(D102:5:0,  DEV102);
    STR(D136:5:0,  DEV136);
    A102 := stationinfo102^[stations.picked,CellPopUp^.month,CellPopUp^.hour].X_Ang;
    A136 := stationinfo136^[stations.picked,CellPopUp^.month,CellPopUp^.hour].X_Ang;
    STR(A102:5:0,  ANG102);
    STR(A136:5:0,  ANG136);
    dispose(stationinfo102);
    dispose(stationinfo136);

    setfillstyle(solidfill,relcolor);
    bar(X+Width-textwidth('1234567')-7,Y+yi+5,X+Width-5,Y+yi+5+textheight('1')+2);
    setcolor(white);
    rectangle(X+Width-textwidth('1234567')-7,Y+yi+5,X+Width-5,Y+yi+5+textheight('1')+2);
    outtextxy(X+Width-textwidth('REL: 1234567')-5,Y+yi+5,'     '+RelString);
    setcolor(textcolor);
    outtextxy(X+Width-textwidth('REL: 1234567')-5,Y+yi+5,'REL: ');

    yi := yi + 21;

    outtextxy(X+xi,Y+55,'STATION: '+StationLetter);
    outtextxy(X+xi,Y+yi+ 0*(textheight('1')+3),' FREQ:');
    outtextxy(X+xi,Y+yi+ 1*(textheight('1')+3),' SNR: ');
    outtextxy(X+xi,Y+yi+ 2*(textheight('1')+3),' S/L: ');
    outtextxy(X+xi,Y+yi+ 3*(textheight('1')+3),' DM:  ');
    outtextxy(X+xi,Y+yi+ 4*(textheight('1')+3),' DEV: ');
    outtextxy(X+xi,Y+yi+ 5*(textheight('1')+3),' ANG: ');

    setfillstyle(solidfill,blue);
    setcolor(white);



    bar(X+xi+textwidth('123456789')-2,Y+55-1,
        X+xi+textwidth('1234567890')+2,Y+55+2+textheight('1'));
    rectangle(X+xi+textwidth('123456789')-2,Y+55-1,
              X+xi+textwidth('1234567890')+2,Y+55+3+textheight('1'));
    outtextxy(X+xi,Y+55,'         '+StationLetter);

    bar(X+xi+textwidth('1234567'),Y+yi-1+0*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+6*(textheight('1')+3));
    outtextxy(X+xi+1,Y+yi+ 0*(textheight('1')+3),'         '+FREQ102);
    outtextxy(X+xi+1,Y+yi+ 0*(textheight('1')+3),'                 '+FREQ136);

    if (S102 < Round(CellPopUp^.eptr^.SNR.value.value)) then
      setfillstyle(solidfill,red)
```

```
    else
      setfillstyle(solidfill,blue);
    bar(X+xi+textwidth('1234567'),Y+yi-1+1*(textheight('1')+3),
        X+xi+textwidth('123456789012345'),Y+yi-1+2*(textheight('1')+3));

    if (S136 < Round(CellPopUp^.eptr^.SNR.value.value)) then
      setfillstyle(solidfill,red)
    else
      setfillstyle(solidfill,blue);
    bar(X+xi+textwidth('123456789012345'),Y+yi-1+1*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+2*(textheight('1')+3));
    outtextxy(X+xi,Y+yi+ 1*(textheight('1')+3),'          '+SNR102+'   '+SNR136);

    if (S102-L102 < Round(CellPopUp^.eptr^.ShortLongRatio.value.value)) then
      setfillstyle(solidfill,red)
    else
      setfillstyle(solidfill,blue);
    bar(X+xi+textwidth('1234567'),Y+yi-1+2*(textheight('1')+3),
        X+xi+textwidth('123456789012345'),Y+yi-1+3*(textheight('1')+3));
    if (S136-L136 < Round(CellPopUp^.eptr^.ShortLongRatio.value.value)) then
      setfillstyle(solidfill,red)
    else
      setfillstyle(solidfill,blue);
    bar(X+xi+textwidth('123456789012345'),Y+yi-1+2*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+3*(textheight('1')+3));
    outtextxy(X+xi,Y+yi+ 2*(textheight('1')+3),'          '+SIRATIO102+'   '+SIRATIO136);

    IF (stationinfo102^[stations.picked,CellPopUp^.month,CellPopUp^.hour].Phase AND $80) = 0
       setfillstyle(solidfill,blue)
    ELSE
       setfillstyle(solidfill,red);
    bar(X+xi+textwidth('1234567'),Y+yi-1+3*(textheight('1')+3),
        X+xi+textwidth('123456789012345'),Y+yi-1+4*(textheight('1')+3));
    IF (stationinfo136^[stations.picked,CellPopUp^.month,CellPopUp^.hour].Phase AND $80) = 0
       setfillstyle(solidfill,blue)
    ELSE
       setfillstyle(solidfill,red);
    bar(X+xi+textwidth('123456789012345'),Y+yi-1+3*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+4*(textheight('1')+3));
    outtextxy(X+xi,Y+yi+ 3*(textheight('1')+3),'          '+DM102+'     '+DM136);

    if (D102 <= Round(CellPopUp^.eptr^.PhaseDev.value.value)) or
       (D102 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))  then
      setfillstyle(solidfill,blue)
    else
      setfillstyle(solidfill,red);
    bar(X+xi+textwidth('1234567'),Y+yi-1+4*(textheight('1')+3),
        X+xi+textwidth('123456789012345'),Y+yi-1+5*(textheight('1')+3));
    if (D136 <= Round(CellPopUp^.eptr^.PhaseDev.value.value)) or
```

```
      (D136 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))  then
     setfillstyle(solidfill,blue)
   else
     setfillstyle(solidfill,red);
   bar(X+xi+textwidth('123456789012345'),Y+yi-1+4*(textheight('1')+3),
       X+xi+textwidth('12345678901234567890123'),Y+yi-1+5*(textheight('1')+3));
   outtextxy(X+xi,Y+yi+ 4*(textheight('1')+3),'         '+DEV102+'   '+DEV136);

   if (A102 >= Round(CellPopUp^.eptr^.XAngle.value.value)) then
     setfillstyle(solidfill,blue)
   else
     setfillstyle(solidfill,red);
   bar(X+xi+textwidth('1234567'),Y+yi-1+5*(textheight('1')+3),
       X+xi+textwidth('123456789012345'),Y+yi-1+6*(textheight('1')+3));

   if (A136 >= Round(CellPopUp^.eptr^.XAngle.value.value)) then
     setfillstyle(solidfill,blue)
   else
     setfillstyle(solidfill,red);
   bar(X+xi+textwidth('123456789012345'),Y+yi-1+5*(textheight('1')+3),
       X+xi+textwidth('12345678901234567890123'),Y+yi-1+6*(textheight('1')+3));
   outtextxy(X+xi,Y+yi+ 5*(textheight('1')+3),'         '+ANG102+'   '+ANG136);

   (* draw the vertical and horizontal lines to enclose the data *)

   line(X+xi+textwidth('1234567'),Y+yi-1+0*(textheight('1')+3),
        X+xi+textwidth('1234567'),Y+yi-1+6*(textheight('1')+3));
   line(X+xi+textwidth('123456789012345'),Y+yi-1+0*(textheight('1')+3),
        X+xi+textwidth('123456789012345'),Y+yi-1+6*(textheight('1')+3));
   line(X+xi+textwidth('12345678901234567890123'),Y+yi-1+0*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+6*(textheight('1')+3));

   line(X+xi+textwidth('1234567'),Y+yi-1+0*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+0*(textheight('1')+3));
   line(X+xi+textwidth('1234567'),Y+yi-1+1*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+1*(textheight('1')+3));
   line(X+xi+textwidth('1234567'),Y+yi-1+2*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+2*(textheight('1')+3));
   line(X+xi+textwidth('1234567'),Y+yi-1+3*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+3*(textheight('1')+3));
   line(X+xi+textwidth('1234567'),Y+yi-1+4*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+4*(textheight('1')+3));
   line(X+xi+textwidth('1234567'),Y+yi-1+5*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+5*(textheight('1')+3));
   line(X+xi+textwidth('1234567'),Y+yi-1+6*(textheight('1')+3),
        X+xi+textwidth('12345678901234567890123'),Y+yi-1+6*(textheight('1')+3));

   settextstyle(defaultfont,horizdir,1);
```

File Name: CELLWIN.PAS

```
    { only show and activate the MDU button if a station is selected }
    IF NOT MDU_shown THEN BEGIN
       MDU_shown := TRUE;
       addtotablist('SubCellPopUp',@MDU);
    end;
    MDU.ChangeXY(X+ 30,Y+139);

    MDU.Show;
  end; { else clause of if picked = 0 }

  Show_Cursor(xx,yy);
end;

(*****************************************************************************)

procedure SubCellPopWindow.Show;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Object method to display the top part of the detailed cell    *
      *           query window.                                                 *
      ***************************************************************************
      }

var
  hour, month : integer;
  i,th,tw,xi,yi,xx,yy: integer;
  mon,hr,PatStr, GDOPStr, GDOPstr1 : string;
  gdpvalue,code:integer;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  Window.Show;
  setcolor(TextColor);
  settextstyle(smallfont,horizdir,4);

  th := textheight('1') + 2;
  tw := textwidth('1') + 2;
  xi := xoffset+textwidth('1')-2;
  yi := yoffset;

  settextstyle(defaultfont,horizdir,1);

  Stations.ChangeXY(X+xi,Y+yi);
  Stations.Picked := 0;
  for i := 1 to Stations.NumberButtons do begin
    Stations.ButtonArray[i].TurnOff;
```

```
    if (HilitedCellPtr^.Weight_Or_Coverage and ($80 SHR (i-1))) > 0 then begin
      Stations.ButtonArray[i].SelectedColor := SelectedColor;
      Stations.ButtonArray[i].Color := SelectedColor;
    end
    else begin
      Stations.ButtonArray[i].Color := Color;
      Stations.ButtonArray[i].SelectedColor := Color;
    end;
  end;

  Stations.Show;

  settextstyle(smallfont,horizdir,4);
  settextjustify(lefttext,toptext);

  Pat := HilitedCellPtr^.Pac;
  str(Pat:7:6,PatStr);
  PatStr := 'P   ' + PatStr;

  if Pat < cellPopup^.eptr^.Psa.value.value THEN
  begin
     setfillstyle(solidfill,red);
     bar(x+4,y+3,x+4+textwidth(PatStr)+1,y+3+textheight(PatStr)+2);
  end;

  outtextxy(X+5,Y+3,PatStr);
  outtextxy(X+5+tw,Y+5,'AT');

  GDOPStr1 := GetGdop(cellpopup^.CellNum,
                      HilitedCellPtr^.weight_or_coverage);

  GDOPStr := 'GDOP ' + GDOPStr1;

  GDPValue := ReadGdop(cellpopup^.CellNum,
                      HilitedCellPtr^.weight_or_coverage);

  if (GDOPStr1 = 'N/A') OR (GDOPStr1 = '> 25') THEN
  begin
     setfillstyle(solidfill,red);
     bar(X+width-textwidth(GDOPStr)-6,Y+3,
         X+width-6+1,y+3+textheight(GDOPStr)+2);
  end
  ELSE
  begin
     if (gdpvalue > cellpopup^.eptr^.GDOP.value.value*10) then
     begin
        setfillstyle(solidfill,red);
        bar(X+width-textwidth(GDOPStr)-6,Y+3,
            X+width-6+1,y+3+textheight(GDOPStr)+2);
```

```
    end;
  end;


  outtextxy(X+width-textwidth(GDOPStr)-5,Y+3,GDOPStr);

  str(CellPopUp^.hour,hr);
  if CellPopUp^.hour < 10 then
    hr:= '0' + hr + '00'
  else
    hr:= hr + '00';

  outtextxy(X+5,Y+35,MonthNames[CellPopup^.Month]+' '+hr);

  (*FillBottomOfWindow;*)

  settextstyle(defaultfont,horizdir,1);
  Show_Cursor(xx,yy);
end;

(*************************************************************************)

function SubCellPopWindow.Action(Xpos, Ypos: word):boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object method that handles the selection of a station for  *
      *           display of the database information on the bottom half of  *
      *           the detailed cell query window.                            *
      ************************************************************************
      }

var
  dummy : boolean;

begin
  Action := FALSE;
  if Stations.Action(Xpos,Ypos) then
  begin
    if not BottomMenuShown then
    begin
      BottomMenu.Show;
      BottomMenuShown := true;
    end;
    FillBottomOfWindow;
    Action := TRUE;
  end;
  IF stations.picked > 0 THEN
```

```
    IF MDU.Action(Xpos,Ypos) THEN
       Action := TRUE;
end;

(*****************************************************************************)

function SubCellPopWindow.KeyAction(Xpos,Ypos:word;key:integer):boolean;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Hotkey object handling method for the detailed cell query        *
      *           window.                                                          *
      ******************************************************************************
      }

begin
  Keyaction := FALSE;
  IF stations.picked > 0 THEN
     Keyaction := MDU.KeyAction(Xpos,Ypos,key);
end;

(*****************************************************************************)

procedure SubCellPopWindow.Hilite(Xpos,Ypos:word);
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Highlight routine that handles the highlighting of the           *
      *           coverage display command button ONLY when the lower half         *
      *           of the detailed cell query window is shown.                      *
      ******************************************************************************
      }

begin
  Stations.Hilite(Xpos,Ypos);
  IF stations.picked > 0 THEN
     MDU.Hilite(Xpos,Ypos);
end;

(*****************************************************************************)

procedure SubCellPopWindow.Cancel;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Method for removing the detailed cell query window from          *
```

```
      *           the display.                                                *
      ************************************************************************
      }

begin
  IF MDU_shown THEN begin
     MDU_shown := FALSE;
     deletefromtablist('SubCellPopUp',@MDU);
  end;
  if BottomMenuShown then
  begin
    BottomMenuShown := false;
    BottomMenu.Hide;
  end;
  menu.Cancel;
  HilitedCellPtr^.Cancel;
end;

(*************************************************************************)

constructor DiffCellPopWindow.Init(InitX,InitY,InitWidth,InitHeight : integer;
                                   InitColor,InitBorderColor,InitTextColor,
                                   InitHiliteColor,InitSelectedColor: word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object initialization for the difference display query     *
      *           window.                                                    *
      ************************************************************************
      }

begin
  menu.Init(InitX,InitY,InitWidth,InitHeight,InitColor,
            InitBorderColor,2,WindowShadowWidth-2);
  TextColor  := InitTextColor;
  SelectedColor := InitSelectedColor;
end;

(*************************************************************************)

procedure DiffCellPopWindow.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object display method for the difference display query     *
      *           window.                                                    *
      ************************************************************************
```

```
File Name: CELLWIN.PAS

      }

var
  i,xx,yy,mon,hr,xi,yi,tw,th : integer;
  LPacStr,RPacStr,PacStr,deltastr : String;
  lval,rval,denom,val : Real;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  Window.Show;
  setcolor(TextColor);
  settextjustify(lefttext,toptext);
  xi := 70;
  yi := 5;
  outtextxy(x+xi,y+yi,'Left  Right');

  lval  :=  leftcellgrid^.cellarray[HilitedCellPtr^.cellnumber].pac;
  rval :=  rightcellgrid^.cellarray[HilitedCellPtr^.cellnumber].pac;

  str(lval:5:4,LPacStr);
  str(rval:5:4,RPacStr);
  PacStr := 'P     ' + LPacStr + ' ' + RPacStr;
  yi := yi + 14;
  outtextxy(X+10,Y+yi,PacStr);
  tw := textwidth('P');
  outtextxy(X+10+tw,Y+yi+4,'AC');

  IF abs(lval - rval) < 0.00001 THEN
     val := 0
  ELSE
  CASE diffmenu^.diffmode.picked OF
    1: val := lval - rval;
    2:
       begin
          IF lval > rval THEN
             denom := lval
          ELSE
             denom := rval;
          IF denom <> 0 THEN
             val := (lval-rval)/denom
          ELSE
             val := 1;
       end;
    3:
       begin
          IF (1-lval) > (1-rval) THEN
             denom := 1 - lval
          ELSE
```

```
            denom := 1 - rval;
          IF (denom) <> 0 THEN
            val := ((1-lval)-(1-rval))/denom
          ELSE
            val := 1;
        end;
    end; { case }
    yi := yi + 20;
    str(val:5:4,deltastr);
    outtextxy(X+30,Y+yi,chr(127) + '      ' + deltastr);

    Show_Cursor(xx,yy);
end;

(*****************************************************************************)

procedure DiffCellPopWindow.Cancel;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method for removing the difference display query window       *
      *                  from the display.                                      *
      ***************************************************************************
      }
begin
  menu.Cancel;
  HilitedCellPtr^.Cancel;
end;
```

File Name: CHECKMEM.PAS

```
{*************************************************************************
*                                                                        *
*       PROGRAM NAME - Omega Performance Assessment                      *
*                              and                                       *
*                      Coverage Evaluation                               *
*                           (PACE)                                       *
*                        Workstation                                     *
*                                                                        *
*          UNIT NAME - Part of PACEOBJS unit                             *
*                                                                        *
*************************************************************************
*                                                                        *
*       This program was prepared by                                     *
*                                                                        *
*            The Analytic Science Corporation (TASC)                     *
*                  55 Walkers Brook Drive                                *
*               Reading, Massachusetts 01867                             *
*                                                                        *
*       PACE has been developed to run on a IBM PC/AT or compatible      *
*       under MS-DOS 3.3 or higher with a minimum of 640K of main        *
*       memory and an EGA or compatible graphics adapter and color       *
*       monitor. This work was performed under contract number           *
*       DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for        *
*       the Omega Navigation System Center (ONSCEN), Alexandria, VA.     *
*                                                                        *
*************************************************************************
*                                                                        *
*       PURPOSE                                                          *
*          Utility routine for reporting the amount of available         *
*          system memory. Invoked by ALT-M. NOT A DOCUMENTED PACE        *
*          FEATURE.                                                      *
*                                                                        *
*************************************************************************}


procedure CheckMemory(key : integer);
var
  ma        : longint;
  M_avail,
  T_avail   : string;

begin
  if (key = 177) then begin    (* alt m *)
    ma := memavail;
    str(ma,T_avail);
    ma := maxavail;
    str(ma,M_avail);
    messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Total Available Memory = ' + T_a
                              (GetMaxY DIV 2) - (textheight('1') DIV 2),
```

File Name: CHECKMEM.PAS

```
                                2,
                                length('Total Available Memory = ' + T_avail),
                                lightred,
                                white,
                                'Total Available Memory = ' + T_avail +
                                'Max. Contiguous Memory = ' + M_avail);
    messagewait;
    messagebox.hidemessage;
  end;
end;
```

File Name: CONMAN.PAS

```
UNIT ConMan; (context manager unit for object oriented controls)
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - CONMAN                                           *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*             Manages the context list which contains which windows      *
*             and objects are currently active and displayed. Also       *
*             handles a stack of displayed windows so that they can      *
*             be removed in the proper order. Maintains the list of      *
*             hotkeys. All user input passes throught the context        *
*             manager and is parsed out to the appropriate object        *
*             by either hotkey association or by user selected screen    *
*             location (i.e., the mouse coordinates ).                   *
*************************************************************************)

Interface
Uses Controls, CursrObj;

Procedure CreateContext(ContextName:shortstring);

Procedure AddToContext(ContextName:shortstring; Obj:ControlPtr);

Procedure AddToTabList(ContextName:shortstring; Obj:ControlPtr);

Procedure DeleteFromContext(ContextName:shortstring; Obj:ControlPtr);
```

File Name: CONMAN.PAS

```
Procedure DeleteFromTabList(ContextName:shortstring; Obj:ControlPtr);

Procedure PushContext(ContextName:shortstring);

Procedure PopContext;

Procedure SwitchContext(ContextName:shortstring);

Procedure ShowContext;

Procedure HideContext;

Procedure HiLiteContext(Xpos,Ypos:word);

Procedure ActionContext(Xpos,Ypos:word);

Procedure KeyActionContext(Xpos,Ypos:word; key:integer);

Function  TopContext:boolean;

Procedure CancelContext;

Function CurrentContext:shortstring;


Implementation

Type
   ctlptr = ^ctl;
   ctl = record
       next,prev: ctlptr;
       obj: Controlptr;
   end;

   ContextElementPtr = ^ContextElement;
   ContextElement = record
                       name:shortstring;
                       nextelement,prevelement:ContextElementPtr;
                       controllist:ctlptr;
   end;

   ContextPtr = ^Context;
   Context = record
           Element:contextelementptr;
           nextcontext:contextptr;
   end;

Var
```

File Name: CONMAN.PAS

```
   contextlist,tablist : Contextptr;
   contextpool,tabpool : ContextElementPtr;


Procedure CreateContext(ContextName:shortstring);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Associate a name with a context. A context is a collection *
      *           of objects that are active at the same time.               *
      ************************************************************************
      }
var
   chaser:ContextElementPtr;
begin
   If contextpool = NIL THEN
   begin { if no contexts exist, create the first new context and tab list }
      new(contextpool);
      contextpool^.name := contextname;
      contextpool^.nextelement := NIL;
      contextpool^.prevelement := NIL;
      contextpool^.controllist := NIL;

      new(tabpool);
      tabpool^.name := contextname;
      tabpool^.nextelement := NIL;
      tabpool^.prevelement := NIL;
      tabpool^.controllist := NIL;
   end
   ELSE
   begin { context and tab pool already exist, add to end of each }
      chaser := contextpool;
      While chaser^.nextelement <> NIL DO
        chaser := chaser^.nextelement;
      new(chaser^.nextelement);
      chaser^.nextelement^.name := contextname;
      chaser^.nextelement^.nextelement := NIL;
      chaser^.nextelement^.prevelement := chaser;
      chaser^.nextelement^.controllist := NIL;

      chaser := tabpool;
      While chaser^.nextelement <> NIL DO
        chaser := chaser^.nextelement;
      new(chaser^.nextelement);
      chaser^.nextelement^.name := contextname;
      chaser^.nextelement^.nextelement := NIL;
      chaser^.nextelement^.prevelement := chaser;
      chaser^.nextelement^.controllist := NIL;
```

```
   end;
end;

Procedure AddToContext(ContextName:shortstring; Obj:ControlPtr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Add a new object to an already created context (created    *
      *           with createcontext.                                        *
      ************************************************************************
      }
var
   chaser:Contextelementptr;
   objchaser: ctlptr;
begin
   chaser := contextpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   If chaser^.controllist = NIL THEN
   begin
      new(chaser^.controllist);
      chaser^.controllist^.next := NIL;
      chaser^.controllist^.prev := NIL;
      chaser^.controllist^.obj := Obj;
   end
   ELSE
   begin
      objchaser := chaser^.controllist;
      While objchaser^.next <> NIL DO
        objchaser := objchaser^.next;
      new(objchaser^.next);
      objchaser^.next^.next := NIL;
      objchaser^.next^.prev := objchaser;
      objchaser^.next^.obj := Obj;
   end;
end;

Procedure AddToTabList(ContextName:shortstring; Obj:ControlPtr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Add a new object to the list of objects that will be       *
      *           highlighted by repeatedly pressing the TAB or SHIFT-TAB key*
      ************************************************************************
      }

var
```

```
   chaser:Contextelementptr;
   objchaser: ctlptr;
begin { add object pointers to the tab list associated with the named context }
   chaser := tabpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   If chaser^.controllist = NIL THEN
   begin
      new(chaser^.controllist);
      chaser^.controllist^.next := NIL;
      chaser^.controllist^.prev := NIL;
      chaser^.controllist^.obj := Obj;
   end
   ELSE
   begin
      objchaser := chaser^.controllist;
      While objchaser^.next <> NIL DO
        objchaser := objchaser^.next;
      new(objchaser^.next);
      objchaser^.next^.next := NIL;
      objchaser^.next^.prev := objchaser;
      objchaser^.next^.obj := Obj;
   end;
end;

Procedure DeleteFromContext(ContextName:shortstring; Obj:ControlPtr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           remove an object from the list assocaited with a particular*
      *           context. It (the object) will no longer be active in this  *
      *           context.                                                      *
      ***************************************************************************
      }
var
   chaser:Contextelementptr;
   objchaser: ctlptr;
begin
   chaser := contextpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   IF chaser^.controllist <> NIL THEN
   begin
      objchaser := chaser^.controllist;
      While (objchaser^.next <> NIL) AND (objchaser^.obj <> Obj) DO
         objchaser := objchaser^.next;
      IF objchaser <> chaser^.controllist THEN
      begin
```

```
         objchaser^.prev^.next := objchaser^.next;
         objchaser^.next^.prev := objchaser^.prev;
      end
      ELSE
         chaser^.controllist := objchaser^.next;
      dispose(objchaser);
   end;
end;

Procedure DeleteFromTabList(ContextName:shortstring; Obj:ControlPtr);
      {
      ************************************************************************
      *                                                                      *
      *       PURPOSE                                                        *
      *          remove an object from the tab list associated with a        *
      *          particular context. It (the object) will no longer be       *
      *          accessible by tabbing through the list                      *
      ************************************************************************
      }
var
   chaser:Contextelementptr;
   objchaser: ctlptr;
begin
   chaser := Tabpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   IF chaser^.controllist <> NIL THEN
   begin
      objchaser := chaser^.controllist;
      While (objchaser^.next <> NIL) AND (objchaser^.obj <> Obj) DO
         objchaser := objchaser^.next;
      IF objchaser <> chaser^.controllist THEN
      begin
         objchaser^.prev^.next := objchaser^.next;
         objchaser^.next^.prev := objchaser^.prev;
      end
      ELSE
         chaser^.controllist := objchaser^.next;
      dispose(objchaser);
   end;
end;

Procedure PushContext(ContextName:shortstring);
      {
      ************************************************************************
      *                                                                      *
      *       PURPOSE                                                        *
      *          add a new context to the context stack. The previously      *
      *          displayed contexts are all pushed down one level            *
```

```
      ********************************************************************************
      }
var
   chaser:Contextelementptr;
   temp: Contextptr;
begin { push the context and tab list onto the stack }
   chaser := contextpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   new(temp);
   temp^.nextcontext := contextlist;
   temp^.element := chaser;
   contextlist := temp;

   chaser := tabpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   new(temp);
   temp^.nextcontext := tablist;
   temp^.element := chaser;
   tablist := temp;
end;

Procedure PopContext;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Remove a context from the context stack. The previously            *
      *           next context on the stack becomes the current context.             *
      ********************************************************************************
      }
var
   temp: Contextptr;
begin
   IF contextlist^.nextcontext <> NIL THEN
   begin
      temp := contextlist;
      contextlist := contextlist^.nextcontext;
      dispose(temp);

      temp := tablist;
      tablist := tablist^.nextcontext;
      dispose(temp);
   end;
end;

Procedure SwitchContext(ContextName:shortstring);
      {
```

File Name: CONMAN.PAS

```
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Switch to another context by making the desired one the    *
      *           current context and placing it on top of the context stack.*
      ************************************************************************
      }
var
   chaser:Contextelementptr;
   temp: Contextptr;
begin
   chaser := contextpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   contextlist^.element := chaser;

   chaser := tabpool;
   While (chaser^.nextelement <> NIL) AND (chaser^.name <> contextname) DO
      chaser := chaser^.nextelement;
   tablist^.element := chaser;
end;

Procedure ShowContext;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Show (display) all objects that are associated with the    *
      *           context that is on the top of the context stack. Each      *
      *           object's SHOW method in the context will be invoked.       *
      ************************************************************************
      }
var
   objchaser:ctlptr;
begin
   objchaser := contextlist^.element^.controllist;
   while objchaser <> NIL Do
   begin
      objchaser^.obj^.show;
      objchaser := objchaser^.next;
   end;
end;

Procedure HideContext;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Hide (undisplay) all objects that are associated with the  *
```

```
      *          context that is on the top of the context stack. Each      *
      *          object's HIDE method in the context will be invoked.       *
      ***************************************************************************
      }
var
   objchaser:ctlptr;
begin
   objchaser := contextlist^.element^.controllist;
   while objchaser <> NIL Do
   begin
      objchaser^.obj^.Hide;
      objchaser := objchaser^.next;
   end;
end;

Procedure HiLiteContext(Xpos,Ypos:word);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Highlight all objects that are associated with the             *
      *          context that is on the top of the context stack AND whose      *
      *          'hot' area encloses the input x/y coordinates. Each            *
      *          object's HILITE method in the context will be invoked.         *
      ***************************************************************************
      }
var
   objchaser:ctlptr;
begin
   objchaser := contextlist^.element^.controllist;
   while objchaser <> NIL Do
   begin
      objchaser^.obj^.hilite(Xpos,Ypos);
      objchaser := objchaser^.next;
   end;
end;

Procedure ActionContext(Xpos,Ypos:word);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Asks all objects that are associated with the                  *
      *          context that is on the top of the context stack AND whose      *
      *          'hot' area encloses the input x/y coordinates to perform       *
      *          the action that is associated with themselfs. Each             *
      *          object's ACTION method in the context will be invoked.         *
      ***************************************************************************
      }
```

```
File Name: CONMAN.PAS

var
   objchaser:ctlptr;
   dummy:boolean;
begin
   objchaser := contextlist^.element^.controllist;
   while objchaser <> NIL Do
   begin
      dummy := objchaser^.obj^.action(Xpos,Ypos);
      objchaser := objchaser^.next;
   end;
end;

Procedure KeyActionContext(Xpos,Ypos:word; key:integer);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Utility routine that handles all keyboard input for the    *
      *           on-screen controls. Parses out the key command to the      *
      *           appropriate control object by asking each object whether   *
      *           the input x/y coordinates are within their 'hot' areas.    *
      *           Also checks for the tab/shifttab and esc (i.e., cancel     *
      *           current context) keys and performs the appropriate action. *
      ************************************************************************
      }

var
   objchaser:ctlptr;
   dummy:boolean;
   X,Y: integer;
   xx,yy: integer;

begin
   CASE key OF { check for TAB and BackTab and any other keys }
   FWD_NEXT_OBJECT,BKWD_NEXT_OBJECT:
   begin
      objchaser := tablist^.element^.controllist;
      IF objchaser <> NIL THEN { make sure there is something in the tab list }
      begin
         Hide_Cursor;
         { put on the first element of the list in case there is nothing
           currently hilited or in case we get to the end of the list }
         xx := objchaser^.obj^.X+objchaser^.obj^.width DIV 2;
         yy := objchaser^.obj^.Y+objchaser^.obj^.Height DIV 2;
         while objchaser <> NIL Do
         begin { find the currently hilighted object ( if there is one ) }
(*
            IF (objchaser^.obj^.Hilited) then
*)
```

```
            IF (objchaser^.obj^.Hilited) OR (objchaser^.obj^.IsOn(Xpos,Ypos)) then
            CASE key OF
            FWD_NEXT_OBJECT:
            begin
               IF (objchaser^.next <> NIL) THEN
               begin { move the cursor to the next object }
                  xx := objchaser^.next^.obj^.X+
                        objchaser^.next^.obj^.width DIV 2;
                  yy := objchaser^.next^.obj^.Y+
                        objchaser^.next^.obj^.Height DIV 2;
                  objchaser := NIL; { to end the loop }
               end;
            end;
            BKWD_NEXT_OBJECT:
            begin
               IF (objchaser^.prev <> NIL) THEN
               begin { move the cursor to the next object }
                  xx := objchaser^.prev^.obj^.X+
                        objchaser^.prev^.obj^.width DIV 2;
                  yy := objchaser^.prev^.obj^.Y+
                        objchaser^.prev^.obj^.Height DIV 2;
                  objchaser := NIL; { to end the loop }
               end
               ELSE { at start of list, go to end }
               begin
                  while objchaser^.next <> NIL DO
                     objchaser := objchaser^.next;
                  xx := objchaser^.obj^.X+
                        objchaser^.obj^.width DIV 2;
                  yy := objchaser^.obj^.Y+
                        objchaser^.obj^.Height DIV 2;
               end;
               objchaser := NIL;
            end;
            end; { CASE key of }
            IF objchaser <> NIL THEN
               objchaser := objchaser^.next;
         end; { while objchaser <> NIL }
         Show_Cursor(xx,yy);
      end; { IF objchaser <> NIL }
   end;

   ELSE
   begin
      objchaser := contextlist^.element^.controllist;
      while objchaser <> NIL Do
      begin
         if (objchaser^.obj^.keyaction(Xpos,Ypos,key)) then
         begin
```

```
            get_cursorXY(X,Y);
            HiliteContext(X,Y);
            dummy := objchaser^.obj^.action(X,Y);
          end;
          objchaser := objchaser^.next;
       end;
    end;
    end; { CASE key }
end;

Function TopContext:boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Boolean function that tells you whether the context that   *
      *           you are interested in is the one on the top of the context *
      *           stack.                                                     *
      ************************************************************************
      }
begin
   IF contextlist^.nextcontext <> NIL THEN
      TopContext := FALSE
   ELSE
      TopContext := TRUE;
end;

Procedure CancelContext;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Informs all objects in the current context to cancel       *
      *           themselfs when a cancel the current context command is     *
      *           issued.                                                    *
      ************************************************************************
      }
var
   objchaser:ctlptr;
begin
   objchaser := contextlist^.element^.controllist;
   while objchaser <> NIL Do
   begin
      objchaser^.obj^.Cancel;
      objchaser := objchaser^.next;
   end;
end;

Function CurrentContext: shortstring;
```

File Name: CONMAN.PAS

```
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Returns the name of the current context (the one on the       *
      *           top of the context stack.                                     *
      ***************************************************************************
      }
begin
   CurrentContext := contextlist^.element^.name;
end;

Begin
   contextlist := NIL;
   contextpool := NIL;
   tablist := NIL;
   tabpool := NIL;
End.
```

File Name: CONT1.PAS

```
{*****************************************************************************
*                                                                           *
*        PROGRAM NAME - Omega Performance Assessment                        *
*                                and                                        *
*                          Coverage Evaluation                              *
*                              (PACE)                                       *
*                            Workstation                                    *
*                                                                           *
*           UNIT NAME - Part of the CONTROLS Unit                           *
*                                                                           *
*****************************************************************************
*                                                                           *
*        This program was prepared by                                       *
*                                                                           *
*              The Analytic Science Corporation (TASC)                      *
*                    55 Walkers Brook Drive                                 *
*                 Reading, Massachusetts 01867                              *
*                                                                           *
*        PACE has been developed to run on a IBM PC/AT or compatible        *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main          *
*        memory and an EGA or compatible graphics adapter and color         *
*        monitor. This work was performed under contract number             *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for          *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.       *
*                                                                           *
*****************************************************************************
*                                                                           *
*        PURPOSE                                                            *
*           Contains all of the object, constant and unit variable          *
*           declarations for the controls unit.                             *
*                                                                           *
*****************************************************************************}
const
   { vertical and horizontal bar contstants }
   logscale          =  1;
   linearscale       =  0;
   reverselogscale   = -1;
   H_Barwidth        = 95;
   H_Barheight       = 10;
   V_Barwidth        = 15;
   V_Barheight       = 129;
   Slidewidth        =  7;
   Slideheight       =  7;
   arrowwidth        = 10;
   arrowheight       = 10;
   valuewidth        = 60;
   H_SlideLength     = H_Barwidth - arrowwidth - arrowwidth - Slidewidth;
   V_SlideLength     = V_Barheight - arrowheight - arrowheight - Slideheight;
   shortstringlength = 16;
```

File Name: CONT1.PAS

```
   editstrlength  = 19;

   (* orientation for the mutually exclusive buttons *)
   (* used in Controls.pas for computing the slide length *)
   (*   if a generic slide bar is made, this wouldn't be necessary... *)

   Horizontal = true;
   Vertical   = false;

   (* default pixel width of the shadows on all of the windows *)

   WindowShadowWidth = 6;

type

  shortstring = string[shortstringlength];
  editstring  = string[editstrlength];

  Control = object
       X,Y,Width,Height: integer;
       Color:word;
       Hilited: Boolean;
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor:word);
       procedure Show; virtual;
       procedure Hide; virtual;
       function  Action(Xpos,Ypos:word): boolean; virtual;
       function  IsOn(Xpos,Ypos:word)  : boolean; virtual;
       function  IsHilited             : boolean; virtual;
       procedure SetHilite(Hilite_:boolean);      virtual;
       procedure HiLite(Xpos,Ypos:word);          virtual;
       procedure Cancel;                          virtual;
       function KeyAction(Xpos,Ypos:word; key:integer) : boolean; virtual;
       procedure ChangeXY(Xpos,Ypos:word);                        virtual;
  end;

  ControlPtr = ^Control;

  BorderedArea = object(Control)
     BorderColor: word;
     BorderWidth: integer;
     constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                  Initcolor,InitBorderColor:word;
                  InitBorderWidth:integer);
     procedure Show; virtual;
  end;


  WindowPtr = ^Window;
```

File Name: CONT1.PAS

```
  Window = object(BorderedArea)
(*          BitImg,AuxBitImg: ^byte; *)
          Visible      : Boolean;
          shadowsize   : integer;
          constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                           Initcolor,InitBorderColor:word;
                           InitBorderWidth,InitShadowSize:integer);
          procedure Show; virtual;
          procedure Hide; virtual;
  end;


  FixedControl = object(Control)
       HiLiteColor: word;
       ControlColor,HiliteControlColor: word;
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
       procedure HiLite(Xpos,Ypos:word); virtual;
       procedure Show; virtual;
       procedure Hide; virtual;
       function Action(Xpos,Ypos:word):boolean; virtual;
  end;


  Slider = object(FixedControl)
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
       procedure MoveTo(Xpos,Ypos:word); virtual;
       procedure Show; virtual;
       procedure HiLite(Xpos,Ypos:word); virtual;
  end;


  TextButton = object(FixedControl)
     Name       : shortstring;
     NameChanged : Boolean;
     HotKey     : char;
     constructor Init(InitX,InitY,InitWidth,InitHeight,
                      Initcolor,InitHiLiteColor,InitNameColor,
                      InitHiliteNameColor:word; InitName:shortstring;
                      InitHotKey:char);
     procedure   ChangeName(NewName:shortstring);                virtual;
     procedure   HiLite(Xpos,Ypos:word);                         virtual;
     procedure   Show;                                           virtual;
     function    KeyAction(Xpos,Ypos:word; key:integer):boolean; virtual;
  end;

  EditButton = object(FixedControl)
     editstringlength : integer;
```

File Name: CONT1.PAS

```
    Text_String    : string;
    Display_String : editstring;
    CursorPosition,
    textoffset     : integer;
    ClearString    : boolean;
    constructor Init(InitX,InitY,InitWidth,InitHeight,
                     Initcolor,InitHiLiteColor,InitNameColor,
                     InitHiliteNameColor:word;
                     InitText:string;InitCursorPosition,
                     InitTextOffSet:integer;
                     InitStringLength : integer;
                     InitClearString : boolean);
    function    Action(Xpos,Ypos:word) : boolean;      virtual;
    procedure   HiLite(Xpos,Ypos:word);                virtual;
    procedure   Show;                                  virtual;
    procedure   ReShow;                                virtual;
    procedure   DisplayText(NewText:editstring);       virtual;
    procedure   Change_String(NewText:string);         virtual;
  end;

  NumberButton = object(TextButton)
    Value: real (longint);
    int,frac:integer; { integer and fractional part of the button }
    constructor Init(InitX,InitY,InitWidth,InitHeight,
          Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor:word;
          InitValue:real;InitInt,InitFrac:integer);
    procedure Show; virtual;
    function Action(Xpos,Ypos:word):boolean; virtual;
  end;

  StatusButton = object(TextButton)
    Selected: boolean;
    SelectedColor,SelectedNameColor:word;
    constructor Init(InitX,InitY,InitWidth,InitHeight,
                Initcolor,InitHiLiteColor,InitSelectedColor,
                InitNameColor,InitHiliteNameColor,InitSelectedNameColor:word;
                InitName:shortstring);
    procedure Show; virtual;
    procedure Hilite(Xpos,Ypos:word); virtual;
    procedure TurnOn; virtual;
    procedure TurnOff; virtual;
  end;

  TextButton3D = object(TextButton)
    constructor Init(InitX,InitY,InitWidth,InitHeight,
                Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor:word;
                InitName:shortstring;InitHotKey:char);
    procedure Show; virtual;
    procedure Hilite(Xpos,Ypos:word); virtual;
```

File Name: CONT1.PAS

```
    procedure ChangeName(NewName:shortstring); virtual;
  end;

  EditButton3D = object(EditButton)
    constructor Init(InitX,InitY,InitWidth,InitHeight,
                     Initcolor,InitHiLiteColor,InitNameColor,
                     InitHiliteNameColor:word;
                     InitText:string;InitCursorPosition,
                     InitStringLength:integer;
                     InitClearString:boolean);
    procedure    Show;                          virtual;
    procedure    Hilite(Xpos,Ypos:word);        virtual;
  end;


  UpArrowButton = object(FixedControl)
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
    procedure Show; virtual;
    procedure Hilite(Xpos,Ypos:word); virtual;
  end;

  DownArrowButton = object(FixedControl)
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
    procedure Show; virtual;
    procedure Hilite(Xpos,Ypos:word);virtual;
  end;

  LeftArrowButton = object(FixedControl)
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
    procedure Show; virtual;
    procedure Hilite(Xpos,Ypos:word); virtual;
  end;

  RightArrowButton = object(FixedControl)
       constructor Init(InitX,InitY,InitWidth,InitHeight:integer;
                   Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
    procedure Show; virtual;
    procedure Hilite(Xpos,Ypos:word); virtual;
  end;

  HorizontalSlideBar = object(FixedControl)
    Name,Units: shortstring;
    HiLim,LoLim,Increment:real;
    AccelFactor:real;
    Accel:integer;
    Left:LeftArrowButton;
```

File Name: CONT1.PAS

```
    Right:RightArrowButton;
    SlideBar:Slider;
    Value:NumberButton;
    constructor Init(InitX,InitY,InitColor,InitHiliteColor,
                InitControlColor,InitHiliteControlColor:word;
                InitLoLim,InitHiLim,InitInc:real;
                InitAccel:integer;InitName,InitUnits:shortstring;
                InitInt,InitFrac:integer);
    procedure Hilite(Xpos,Ypos:word); virtual;
    procedure Show; virtual;
    function Action(Xpos,Ypos:word):boolean; virtual;
  end;

  VerticalSlideBar = object(FixedControl)
    Value,HiLim,LoLim,Increment    : real;
    AccelFactor                    : real;
    Accel                          : integer;
    Up                             : UpArrowButton;
    Down                           : DownArrowButton;
    SlideBar                       : Slider;
    constructor Init(InitX,InitY,InitColor,InitHiliteColor,
                     InitControlColor,InitHiliteControlColor:word;
                     InitLoLim,InitHiLim,InitInc:real;InitAccel:integer);
    procedure   Hilite(Xpos,Ypos:word);         virtual;
    procedure   Show;                           virtual;
    function    Action(Xpos,Ypos:word):boolean; virtual;
  end;

var
  WindowImageNumber : byte;     (* the image number currently being stored *)
  BitImg, AuxBitImg : ^byte;

  procedure GetImageFileName(var f:File; ImageNumber : byte);
```

File Name: CONTROLS.PAS

```
unit Controls;
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                             (PACE)                                     *
*                          Workstation                                   *
*                                                                        *
*           UNIT NAME - CONTROLS                                         *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*             This unit contains the object declarations and methods     *
*             for all of the basic on-screen control and display         *
*             objects.                                                   *
*                                                                        *
*************************************************************************}

interface

uses Graph, Crt, cursrobj, ErrLog;

{$I cont1.pas }

implementation

const
  CR = $0D;
  ESC = $1B;
  DEL = $08;
```

```
File Name: CONTROLS.PAS

{-------------------------------------------------------}
{      Control's method implementations:                }
{-------------------------------------------------------}

constructor Control.Init(InitX,InitY,InitWidth,InitHeight:integer;
           Initcolor:word);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Object definition for the 'grandaddy' object in the PACE      *
      *           object heirarchy. Has some basic attributes that are          *
      *           generic to all of the objects used in PACE.                   *
      ***************************************************************************
      }
begin
   X := InitX;
   Y := InitY;
   Width := InitWidth;
   Height := InitHeight;
   Color := Initcolor;
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure Control.Show;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to fill the area occupied by a generic on-screen       *
      *           control with the objects declared color.                      *
      ***************************************************************************
      }
begin
   setfillstyle(solidfill,color);
   bar(X,Y,X+Width,Y+Height);
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure Control.Hide;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to remove a generic object from the display.           *
      *           Declared here as a placeholder to be filled in by more        *
      *           specific object declarations.                                 *
```

```
      ********************************************************************
      }
begin
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure Control.Hilite(Xpos,Ypos:Word);
      {
      ********************************************************************
      *                                                                  *
      *        PURPOSE                                                   *
      *           Method to highlight a generic object on the display.   *
      *           Declared here as a placeholder to be filled in by more *
      *           specific object declarations.                          *
      ********************************************************************
      }
begin
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

function Control.Action(Xpos,Ypos:word):boolean;
      {
      ********************************************************************
      *                                                                  *
      *        PURPOSE                                                   *
      *           Method to activate generic object based on whether the user*
      *           is currently pointing at the object on the display.    *
      *           Declared here as a placeholder to be filled in by more *
      *           specific object declarations.                          *
      ********************************************************************
      }
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
      action := TRUE
   ELSE
      action := FALSE;
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

function Control.IsOn(Xpos,Ypos:word):boolean;
      {
      ********************************************************************
      *                                                                  *
      *        PURPOSE                                                   *
      *           Indicates whether the cursor is on the object.         *
      *                                                                  *
```

```
      ************************************************************************
      }
begin
   IsOn := Control.Action(Xpos,Ypos);
end;

(* >-------------------------------------------------------------------------< *)

function Control.IsHilited:boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Indicates whether the object is currently highlighted.     *
      *                                                                      *
      ************************************************************************
      }

begin
   IsHilited := Hilited;
end;

(* >-------------------------------------------------------------------------< *)

procedure Control.SetHilite(Hilite_:boolean);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Forces the object's highlight indicator on.                *
      *                                                                      *
      ************************************************************************
      }
begin
   Hilited := Hilite_;
end;

(* >-------------------------------------------------------------------------< *)

procedure Control.Cancel;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to deactivate generic object                        *
      *           Declared here as a placeholder to be filled in by more     *
      *           specific object declarations.                              *
      ************************************************************************
      }
```

File Name: CONTROLS.PAS

```
begin
end;

(* >< *)

function Control.KeyAction(Xpos,Ypos:word; key:integer):boolean;
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Method to activate generic object based on whether the user*
      *           is currently pointing at the object on the display.         *
      *           Declared here as a placeholder to be filled in by more      *
      *           specific object declarations.                               *
      *************************************************************************
      }
begin
   KeyAction := FALSE;
end;

(* >< *)

procedure Control.ChangeXY(Xpos,Ypos:word);
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Method to change the x/y/ screen location where the object *
      *           will be displayed.                                          *
      *************************************************************************
      }

begin
  X := Xpos;
  Y := Ypos;
end;

(* >< *)

constructor BorderedArea.Init(InitX,InitY,InitWidth,InitHeight:integer;
                              Initcolor,InitBorderColor:word;
                              InitBorderWidth:integer);
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Method to initialize a bordered area on the screen. Builds *
      *           on the control object.                                      *
      *************************************************************************
```

```
      }
begin
   Control.Init(InitX,InitY,InitWidth,InitHeight,Initcolor);
   BorderColor := InitBorderColor;
   BorderWidth := InitBorderWidth;
end;

(* >-------------------------------------------------------------------< *)

procedure BorderedArea.Show;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to display a bordered area on the screen. Uses         *
      *           the the control object's show.                                *
      ***************************************************************************
      }
var
   i,xx,yy:integer;
begin
   get_cursorXY(xx,yy);
   Hide_Cursor;
   Control.Show;
   setcolor(BorderColor);
   for i := 1 to borderwidth do
     rectangle(X+i,Y+i,X+Width-i,Y+height-i);
   show_cursor(xx,yy);
end;

(* >-------------------------------------------------------------------< *)

constructor Window.Init(InitX,InitY,InitWidth,InitHeight:integer;
                        Initcolor,InitBorderColor:word;
                        InitBorderWidth,InitShadowSize : integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to initialize a display window object. A window is     *
      *           a bordered object that saves what is underneath for later     *
      *           display.                                                      *
      ***************************************************************************
      }

begin
  shadowsize := InitShadowSize;
  borderedarea.Init(InitX,InitY,InitWidth,InitHeight,Initcolor,
                    InitBorderColor,InitBorderWidth);
```

File Name: CONTROLS.PAS

```
end;

(* >><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure GetImageFileName(var f:File; ImageNumber : byte);
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Utility routine used by window.show to store display data *
      *           so that the window can be removed and what was there before*
      *           will re-appear. Sequentially stores image files to disk for*
      *           later retrieval.                                          *
      ***********************************************************************
      }
var
  filename,
  ext : string;

begin
  str(ImageNumber, ext);

  if (ImageNumber < 10) then
    ext := '00' + ext;
  if (ImageNumber >=10) and (ImageNumber < 100) then
    ext := '0' + ext;

  filename := 'IMG.'+ ext;
  Assign(f,filename);

end;


(* >><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure Window.Show;
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Method that saves what will be overlaid by the window in  *
      *           question and then displays the window area (essentially a *
      *           bordered area. If necessary, breaks up the underlying area *
      *           into pieces that are small enough to hold in the remaining *
      *           memory before storing them to disk.                       *
      ***********************************************************************
      }

var
```

```
   Size,
   numwritten :word;
   st1,st2 : string;
   xx,yy,i:integer;
   f : file;

begin
   get_cursorxy(xx,yy);
   hide_cursor;

   Size := imagesize(X,Y,X+Width+shadowsize,Y+Height+shadowsize);
   { Imagesize returns 0 if > 64K }
{   IF imagesize(X,Y,X+Width+shadowsize,Y+Height+shadowsize) <> 0 THEN}
   IF (Size <> 0) AND (Size < $4000) THEN
   begin
      inc(WindowImageNumber);
      GetImageFileName(f,WindowImageNumber);
      if maxavail > Size then
      begin
         getmem(BitImg,Size);
         getimage(X,Y,X+Width+shadowsize,Y+Height+shadowsize,BitImg^);
         Rewrite(f,1);
         BlockWrite(f,BitImg^,Size,NumWritten);
         if (Size <> NumWritten) then
         begin
           str(Size,St1);
           str(NumWritten,St2);
           Log_Error('Size = '+st1+'   NumWritten = '+st2+'   IN Window.Show');
         end;
         freemem(BitImg,Size);
         Close(f);
      end
      else
      begin
        str(Size,St1);
        Log_Error('Not enough memory to store whole window image, size = '+st1);
      end;
   end
   else
   for i := 3 downto 0 do
   begin
      inc(WindowImageNumber);
      GetImageFileName(f,WindowImageNumber);
      Size := imagesize(X,Y+i*(Height+shadowsize)DIV 4,
              X+Width+shadowsize,Y+(i+1)*(Height+shadowsize)DIV 4);
      if maxavail > Size then
      begin
         getmem(AuxBitImg,Size);
         getimage(X,Y+i*(Height+shadowsize)DIV 4,
```

```
                X+Width+shadowsize,Y+(i+1)*(Height+shadowsize)DIV 4,AuxBitImg^);
        Rewrite(f,1);
        BlockWrite(f,AuxBitImg^,Size,NumWritten);
        if (Size <> NumWritten) then
        begin
          str(Size,St1);
          str(NumWritten,St2);
          Log_Error('Size = '+st1+'   NumWritten = '+st2+'   IN Window.Show');
        end;
        freemem(AuxBitImg,Size);
        close(f);
      end
      else
      begin
        str(Size,St1);
        Log_Error('Not enough memory to store half window image, size = '+st1);
      end;
   end;

(*
      inc(WindowImageNumber);
      GetImageFileName(f,WindowImageNumber);
      Size := imagesize(X,Y,X+Width+shadowsize,(Y+Height+shadowsize)DIV 2);
      if maxavail > Size then
      begin
         getmem(BitImg,Size);
         getimage(X,Y,X+Width+shadowsize,(Y+Height+shadowsize)DIV 2,BitImg^);
         Rewrite(f,1);
         BlockWrite(f,BitImg^,Size,NumWritten);
         if (Size <> NumWritten) then
         begin
           str(Size,St1);
           str(NumWritten,St2);
           Log_Error('Size = '+st1+'   NumWritten = '+st2+'   IN Window.Show');
         end;
         freemem(BitImg,Size);
         close(f);
      end
      else
      begin
        Log_Error('Not enough memory to store window image');
      end;
   end;
*)

   BorderedArea.show;

   if shadowsize > 0 then begin
     setfillstyle(solidfill,black);
```

```
      bar(X+Width+1,Y+shadowsize,X+Width+shadowsize,Y+Height+shadowsize);
      bar(X+shadowsize,Y+Height+1,X+Width,Y+Height+shadowsize);
   end;

   show_cursor(xx,yy);
   visible := TRUE;
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure Window.Hide;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method that removes the window display and replaces it with*
      *           what was there before.                                             *
      *           If necessary, breaks up the underlying area                        *
      *           into pieces that are small enough to hold in the remaining *
      *           memory before getting them from disk.                              *
      ********************************************************************************
      }

var
   st1,st2 : string;
   Size,Size2,
   NumRead : word;
   xx,yy,i:integer;
   f : File;

begin
   IF visible THEN
      BEGIN
         get_cursorxy(xx,yy);
         hide_cursor;
         Size := imagesize(X,Y,X+Width+shadowsize,Y+Height+shadowsize);
         { Imagesize returns 0 if > 64K }
{         IF imagesize(X,Y,X+Width+shadowsize,Y+Height+shadowsize) <> 0 THEN}
         IF (Size <> 0) AND (Size < $4000) THEN
         begin
            GetImageFileName(f,WindowImageNumber);
            if maxavail > Size then
            begin
               getmem(BitImg,Size);
               Reset(f,1);
               BlockRead(f,BitImg^,Size,NumRead);
               if (Size <> NumRead) then
               begin
                 str(Size,St1);
```

```
                str(NumRead,St2);
                Log_Error('Size = '+st1+'   NumRead = '+st2+'   IN Window.Hide');
              end;
              Close(f);
              putimage(X,Y,BitImg^,CopyPut);
              freemem(BitImg,Size);
              dec(WindowImageNumber);
            end
            else
            begin
              Log_Error('Not enough memory to store window image');
            end;
          end
          else
          for i := 0 to 3 do
          begin
(*
            GetImageFileName(f,WindowImageNumber);
            Size2 := imagesize(X,Y,X+Width+shadowsize,(Y+Height+shadowsize)DIV 2);
            if maxavail > Size2 then
            begin
              getmem(BitImg,Size2);
              Reset(f,1);
              BlockRead(f,BitImg^,Size2,NumRead);
              if (Size2 <> NumRead) then
              begin
                str(Size2,St1);
                str(NumRead,St2);
                Log_Error('Size2 = '+st1+'   NumRead = '+st2+'   IN Window.Hide');
              end;
              dec(WindowImageNumber);
              close(f);
              putimage(X,Y,BitImg^,CopyPut);
              freemem(BitImg,Size2);
            end
            else
            begin
              Log_Error('Not enough memory to store window image');
            end;
*)
            GetImageFileName(f,WindowImageNumber);
            Size := imagesize(X,Y+i*(Height+shadowsize)DIV 4,
                    X+Width+shadowsize,Y+(i+1)*(Height+shadowsize)DIV 4);
            if maxavail > Size then
            begin
              getmem(AuxBitImg,Size);
              Reset(f,1);
              BlockRead(f,AuxBitImg^,Size,NumRead);
              if (Size <> NumRead) then
```

```
              begin
                str(Size,St1);
                str(NumRead,St2);
                Log_Error('Size = '+st1+'   NumRead = '+st2+'   IN Window.Hide');
              end;
              dec(WindowImageNumber);
              putimage(X,Y+i*(Height+shadowsize)DIV 4,AuxBitImg^,CopyPut);
              freemem(AuxBitImg,Size);
              close(f);
            end
            else
            begin
              Log_Error('Not enough memory to store window image');
            end;
          end;

          show_cursor(xx,yy);
          visible := FALSE;
        END;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor FixedControl.Init(InitX,InitY,InitWidth,InitHeight:integer;
            Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Object declaration for a kind of control that has an action*
    *          associated with it (e.g., an on-screen control).            *
    ************************************************************************
    }
begin
   Control.Init(InitX,InitY,InitWidth,InitHeight,InitColor);
   Control.Show;
   HiliteColor := InitHiliteColor;
   ControlColor := InitControlColor;
   HiliteControlColor := InitHiliteControlColor;
   Hilited := FALSE;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure FixedControl.Show;
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
```

```
      *          Method for displaying a fixed object on the screen.          *
      ********************************************************************************
      }
Var
   xx,yy:integer;
begin
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure FixedControl.Hide;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method for removing a fixed object from the screen                 *
      ********************************************************************************
      }
begin
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure FixedControl.Hilite(Xpos,YPos:word);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method for highlighting a fixed object.                            *
      ********************************************************************************
      }
begin
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

function FixedControl.Action(Xpos,Ypos:word):boolean;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Action method for a fixed object.                                  *
      ********************************************************************************
      }
begin
   IF Hilited THEN
      Action := TRUE
   ELSE
      Action := FALSE;
```

```
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

constructor Slider.Init(InitX,InitY,InitWidth,InitHeight:integer;
          Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word)
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *          Object intialization for the little slider button on slide *
     *          type controls.                                                 *
     ***************************************************************************
     }
begin
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
          Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor);
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

Procedure Slider.Show;
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *          Object method for displaying little slider button on slide *
     *          type controls.                                                 *
     ***************************************************************************
     }
Var
   xx,yy:integer;
begin
   Get_CursorXY(xx,yy);
   hide_cursor;
   setfillstyle(solidfill,color);
   bar(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

procedure Slider.Hilite(Xpos,YPos:word);
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *          Object method for highlighting the little slider button    *
     *          on slide type controls.                                        *
```

File Name: CONTROLS.PAS

```
      ************************************************************************
      }
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   begin
      IF NOT Hilited THEN
      begin
         hide_cursor;
         Hilited := TRUE;
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
      end
   end
   ELSE
      IF Hilited THEN
      begin
         hilited := FALSE;
         Show;
      end;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

Procedure Slider.MoveTo(Xpos,Ypos:word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object method for moving the little slider button          *
      *           on slide type controls.                                    *
      ************************************************************************
      }
begin
   Show;
   setfillstyle(solidfill,controlcolor);
   bar(X,Y,X+Width,Y+Height);
   X := Xpos;
   Y := Ypos;
   IF Hilited THEN BEGIN
      setfillstyle(solidfill,hilitecolor);
      bar(X,Y,X+Width,Y+Height);
   END
   ELSE
      Show;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)
```

File Name: CONTROLS.PAS

```
constructor TextButton.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor:word;
            InitName:shortstring;InitHotKey:char);
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Object initialization for a command button that has a label*
      *           on it.                                                    *
      ***********************************************************************
      }
begin
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor);
   Name := InitName;
   NameChanged := FALSE;
   HotKey := InitHotKey;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

function HotKeyPosition(Name:shortstring; var key:char) : integer;
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Utility function for finding the location of the letter   *
      *           that corresponds to the hotkey for the text button.       *
      ***********************************************************************
      }
var
  position: integer;
  found   : boolean;

begin
  position := 1;
  repeat
    found := (key = Name[position]);
    if not(found) then
      position := position + 1;
  until (found) or (position > shortstringlength);

  if not(found) and (key <> chr(0)) then
  begin
    Log_Error ('HotKey character ('+key+') not found in TextButton Name ('+
            Name+')' );
    key := chr(0);
    position := 0;
  end;
```

```
  HotKeyPosition := position;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

procedure TextButton.Show;
      {
      **********************************************************************
      *                                                                    *
      *         PURPOSE                                                    *
      *            Method to display the text button on the screen.        *
      **********************************************************************
      }
Var
   xx,yy,x1,y1,x2,y2,pos,tmp:integer;
begin
   Get_CursorXY(xx,yy);
   hide_cursor;
   x2 := X+Width;
   y2 := Y+Height;
   if (hotkey = chr(0)) then begin
     setfillstyle(solidfill,color);
     bar(X,Y,X+Width,Y+Height);
     setcolor(Controlcolor);
     settextjustify(centertext,centertext);
     outtextxy(X+Width DIV 2,Y + Height DIV 2 + 1,name);
   end
   else begin
     x1 := x + (Width DIV 2 - TextWidth(Name) DIV 2);
     y1 := y + (Height DIV 2 - TextHeight(Name) DIV 2);
     setfillstyle(solidfill,color);
     bar(X,Y,X2,Y2);
     setcolor(Controlcolor);
     settextjustify(lefttext,toptext);
     outtextxy(X1, Y1, Name);
     pos := HotKeyPosition(Name,hotkey);
     tmp := textwidth(hotkey);
     line(x1 + tmp*(pos - 1)-1, y1+textheight(hotkey) {y2-4},
         x1 + tmp*pos-1, y1+textheight(hotkey) {y2-4});
    end;
    rectangle(X,Y,X+Width,Y+Height);
    show_cursor(xx,yy);
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

procedure TextButton.ChangeName(NewName:shortstring);
      {
      **********************************************************************
```

```
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to change the name of the text button               *
      ***************************************************************************
      }
begin
   IF Hilited THEN
      setcolor(HiliteColor)
   else
      setcolor(color);
   settextjustify(centertext,centertext);
   outtextxy(X+Width DIV 2,Y + Height DIV 2 + 1,name);
   Name := newname;
   IF Hilited THEN
      setcolor(HiliteControlColor)
   else
      setcolor(Controlcolor);
   settextjustify(centertext,centertext);
   outtextxy(X+Width DIV 2,Y + Height DIV 2 + 1,name);
   NameChanged := TRUE;
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

procedure TextButton.Hilite(Xpos,Ypos:word);
      {
      ***************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to highlight a text button when the cursor is on it.*
      ***************************************************************************
      }
var
   x1,y1,x2,y2,pos,tmp:integer;
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   BEGIN
     IF NOT Hilited THEN
     begin
       hide_cursor;
       Hilited := TRUE;
       x2 := X+Width;
       y2 := Y+Height;
       if (hotkey = chr(0)) then begin
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         setcolor(hiliteControlcolor);
         settextjustify(centertext,centertext);
         outtextxy(X+Width DIV 2,Y + Height DIV 2 + 1,name);
```

```
        end
        else begin
          x1 := x + (Width DIV 2 - TextWidth(Name) DIV 2);
          y1 := y + (Height DIV 2 - TextHeight(Name) DIV 2);
          setfillstyle(solidfill,hilitecolor);
          bar(X,Y,X2,Y2);
          setcolor(HiliteControlcolor);
          settextjustify(lefttext,toptext);
          outtextxy(X1, Y1, Name);
          pos := HotKeyPosition(Name,hotkey);
          tmp := textwidth(hotkey);
          line(x1 + tmp*(pos - 1)-1, y1+textheight(hotkey) {y2-4},
              x1 + tmp*pos-1, y1+textheight(hotKey) {y2-4});
         end;
         rectangle(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
      end;
   END
   ELSE
      IF HiLited THEN
      begin
         hilited := FALSE;
         Show;
      end;

end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

function TextButton.KeyAction(Xpos,Ypos:word; key:integer):boolean;
      {
      ***********************************************************************
      *                                                                     *
      *       PURPOSE                                                       *
      *          Method to invoke the associated action when the user       *
      *          selects the text button.                                   *
      ***********************************************************************
      }
var
  dummy,checkit : boolean;
  temp,
  lckey,
  uckey : integer;

begin
  KeyAction := false;
  if (HotKey <> chr(0)) then
  begin
    checkit := FALSE;
```

```
    if (key >= 65) and (key <= 90) then   { between a .. z }
    begin
      lckey := key + 32;
      uckey := key;
      checkit := TRUE;
    end
    else if (key >= 97) and (key <= 122) then    { between A .. Z }
      begin
        lckey := key;
        uckey := key - 32;
      checkit := TRUE;
      end;
    temp := ord(HotKey);
    if checkit AND (
       (temp = lckey) or
       (temp = uckey) or
       (temp = key))   then
    begin
      Hide_Cursor;
      Show_Cursor(X + width DIV 2, Y + height DIV 2);
      KeyAction := true;
    end
    else
      KeyAction := false;
  end;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor EditButton.Init(InitX,InitY,InitWidth,InitHeight,
                            Initcolor,InitHiLiteColor,InitNameColor,
                            InitHiliteNameColor:word;
                            InitText:string;InitCursorPosition,
                            InitTextOffSet:integer;
                            InitStringLength : integer;
                            InitClearString : boolean);
      {
      ***********************************************************************
      *                                                                     *
      *          PURPOSE                                                    *
      *             Object initialization for a text button that the user can *
      *             type into. Essentially a little one-line note pad.      *
      ***********************************************************************
      }
begin
   ClearString := InitClearString;
   editstringlength := InitStringLength;
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
          Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor);
```

```
   Text_String   := InitText;
   CursorPosition := InitCursorPosition;
   Display_String := copy(Text_String,1,editstringlength);
   textoffset     := InitTextOffSet;
end;

(* >-------------------------------------------------------------------------< *)

procedure EditButton.Show;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to show an edit button.                                *
      ***************************************************************************
      }
Var
   xx,yy,x1,y1:integer;
begin
   Get_CursorXY(xx,yy);
   hide_cursor;
   setfillstyle(solidfill,color);
   bar(X, Y, X+Width, Y+Height);
   setcolor(Controlcolor);
   settextjustify(lefttext,centertext);
   outtextxy(X+textoffset,Y + Height DIV 2 + 1,Display_String);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;

(* >-------------------------------------------------------------------------< *)

procedure EditButton.ReShow;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to reshow an edit button after the user has changed    *
      *           it.                                                           *
      ***************************************************************************
      }
Var
   xx,yy,x1,y1:integer;
begin
   Get_CursorXY(xx,yy);
   hide_cursor;
   if hilited then
   begin
     setfillstyle(solidfill,Hilitecolor);
```

```
      setcolor(HiliteControlcolor);
    end
    else
    begin
      setfillstyle(solidfill,color);
      setcolor(Controlcolor);
    end;
    bar(X, Y, X+Width, Y+Height);
    settextjustify(lefttext,centertext);
    outtextxy(X+textoffset,Y + Height DIV 2 + 1,Display_String);
    rectangle(X,Y,X+Width,Y+Height);
    show_cursor(xx,yy);
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure EditButton.Hilite(Xpos,Ypos:word);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method to highlight an edit button.                                *
      ********************************************************************************
      }
var
   x1,y1:integer;
begin
   IF (Xpos >= X)        AND
      (Xpos < X+Width)   AND
      (Ypos >= Y)        AND
      (Ypos < Y+Height)  THEN
   BEGIN
     if not(Hilited) then
     begin
       hide_cursor;
       Hilited := TRUE;
       setfillstyle(solidfill,hilitecolor);
       bar(X,Y,X+Width,Y+Height);
       setcolor(hiliteControlcolor);
       settextjustify(lefttext,centertext);
       outtextxy(X+textoffset,Y + Height DIV 2 + 1,Display_String);
       rectangle(X,Y,X+Width,Y+Height);
       show_cursor(Xpos,Ypos);
     end;
   END
   ELSE
      IF HiLited THEN
      begin
        hilited := FALSE;
```

```
        Show;
      end;

end;

(* >-------------------------------------------------------------------------< *)

procedure EditButton.DisplayText(NewText:editstring);

      {
      ***************************************************************************
      *                                                                         *
      *       PURPOSE                                                           *
      *          Method to write new text to an already displayed edit          *
      *          button.                                                        *
      ***************************************************************************
      }
var
  x1 : integer;

begin
   IF Hilited THEN
      setcolor(HiliteColor)
   else
      setcolor(color);
   settextjustify(lefttext,centertext);
   line (X+textoffset,Y+textheight(Display_String[1])+(Height div 2)-2,
         X+Width-textoffset,Y+textheight(Display_String[1])+(Height div 2)-2);
   outtextxy(X+textoffset,Y + Height DIV 2 + 1,Display_String);
   Display_String := NewText;
   IF Hilited THEN
      setcolor(HiliteControlColor)
   else
      setcolor(Controlcolor);
   settextjustify(lefttext,centertext);
   outtextxy(X+textoffset,Y + Height DIV 2 + 1,Display_String);

   x1 := X + textoffset + (CursorPosition * textwidth(Display_String[1]));
   line (x1-textwidth(Display_String[1]),
         Y+textheight(Display_String[1])-2+(Height div 2),
         x1,
         Y+textheight(Display_String[1])-2+(Height div 2));
end;

(* >-------------------------------------------------------------------------< *)

function EditButton.Action(Xpos,Ypos : word): boolean;
      {
      ***************************************************************************
```

File Name: CONTROLS.PAS

```
      *                                                                    *
      *        PURPOSE                                                     *
      *           Method to handle the user text input to the edit button  *
      **********************************************************************
      }

const
  DEL      = 210;
  HOME     = 198;
  END_     = 206;
  CTRL_END = 244;

var
  tmp : string[1];
  DStr            : editstring;
  tempstring      : string;
  key,
  button,
  maxchars,
  tempLength,
  Str_Start,
  Str_Stop    : Integer;

begin
  Action := FALSE;
  if FixedControl.Action(Xpos,Ypos) then begin
    Hide_Cursor;
    Action := true;
    if ClearString then
      tempstring := ''
    else
      tempstring := Text_String;
    maxchars := editstringlength;
    templength := length(tempstring);
    if (templength < editstringlength) then begin
      Str_Start := 1;
      CursorPosition := templength + 1;
    end
    else begin
      Str_Start := templength - editstringlength + 2;
      CursorPosition := editstringlength;
    end;
    Str_Stop := tempLength;

    repeat
      DStr := copy(tempstring,Str_Start,Str_Stop);
      DisplayText(DStr);

      key := 0;
```

```
      button := 0;

      while (key = 0) do begin
        get_key(key);
        button := get_buttons;
        if (button = left_button_pressed) then
          key := enter;
        if (button = right_button_pressed) then
          key := esc;
      end;

      case key of

        BACKSPACE : begin
                      if (CursorPosition > 1) then begin
                        delete(tempstring, Str_Start+CursorPosition-2, 1);
                        dec(CursorPosition);
                        dec(templength);
                      end
                      else begin
                        if (Str_Start > 1) then begin
                          dec(Str_Start);
                          delete(tempstring, Str_Start+CursorPosition-1, 1);
                          dec(templength);
                        end
                        else
                          beep;
                      end;
                    end;

        DEL       : begin
                      if (Str_Start + CursorPosition - 1 <= templength) then begin
                         delete(tempstring, Str_Start+CursorPosition-1, 1);
                        dec(templength);
                      end
                      else beep;
                    end;

        HOME      : begin
                      Str_Start := 1;
                      CursorPosition := 1;
                    end;

        END_      : begin
                      if (templength < maxchars) then begin
                        Str_Start := 1;
                        CursorPosition := templength + 1;
                      end
                      else begin
```

```
                    Str_Start := templength - maxchars + 2;
                    CursorPosition := maxchars;
                  end;
                end;

      CTRL_END   : begin
                    if (Str_Start + CursorPosition - 1 <= templength) then begin
                      delete(tempstring, Str_Start + CursorPosition - 1, 255);
                      templength := length(tempstring);
                    end;
                  end;

      LEFT_ARROW : begin
                    if (CursorPosition > 1) then
                      dec(CursorPosition)
                    else
                      if (Str_Start > 1) then
                        dec(Str_Start)
                      else beep;
                  end;

     RIGHT_ARROW : begin
                    if (CursorPosition < maxchars) and
                       ((Str_Start + CursorPosition - 1) <= templength) then
                      inc(CursorPosition)
                    else
                      if ((Str_Start + CursorPosition - 1) <= templength) then
                        inc(Str_Start)
                      else beep;
                  end;

    else begin
      if ((CursorPosition = 1) and (Str_Start = 1) and (chr(key) = ' ' )) then
        beep                    (* do not allow leading blanks in string... *)
      else begin
        if (chr(key) > ' ') and (chr(key) <= 'z') then begin
          tmp := upcase(chr(key));
          insert(tmp, tempstring, Str_Start + CursorPosition - 1);
          inc(templength);
          if (CursorPosition < maxchars) then
            inc(CursorPosition)
          else
            inc(Str_Start);
          if templength < maxchars then
            Str_Stop := templength
          else
            Str_Stop := Str_Start + maxchars - 1;
        end;
      end;
```

```
      end;
    end;                    (* of case statement *)
    until (( key = ENTER ) or ( key = ESC ));

    if (key = ENTER) then begin
      Text_String := tempstring;
      DStr := copy(Text_String,1,editstringlength);
      DisplayText(DStr);
    end
    else begin
      DStr := copy(Text_String,1,editstringlength);
      DisplayText(DStr);
    end;
    setcolor(HiliteColor);
    line (X+textoffset,Y+textheight(Display_String[1])+(Height div 2)-2,
         X+Width-textoffset,Y+textheight(Display_String[1])+(Height div 2)-2);
    Show_Cursor(Xpos,Ypos);
  end;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure EditButton.Change_String(NewText:string);
      {
      **************************************************************************
      *                                                                        *
      *        PURPOSE                                                         *
      *           Method to change the edit button's text wholesale            *
      **************************************************************************
      }
begin
  Text_String := NewText;
  Display_String := copy(Text_String,1,EditStringLength);
end;
(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor NumberButton.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor:word;
            InitValue:real;InitInt,InitFrac:integer);
      {
      **************************************************************************
      *                                                                        *
      *        PURPOSE                                                         *
      *           Object initializatio for a button that contains a number    *
      *           that can be changed by the user.                             *
      **************************************************************************
      }
var
   Valuest:shortstring;
```

File Name: CONTROLS.PAS

```
begin
   Int := InitInt;
   Frac := InitFrac;
   Str(InitValue:Int:Frac,Valuest);
   Value := Initvalue;
   TextButton.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor,
            Valuest,chr(0));
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure NumberButton.Show;
      {
      ****************************************************************************
      *                                                                          *
      *         PURPOSE                                                          *
      *            Method to display a number button.                            *
      ****************************************************************************
      }
var
   xx,yy:integer;
begin
   str(value:int:frac,name);
   TextButton.Show;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

function NumberButton.Action(Xpos,Ypos:word):boolean;
      {
      ****************************************************************************
      *                                                                          *
      *         PURPOSE                                                          *
      *            Method to handle user editing      number button.             *
      ****************************************************************************
      }
var
  tempvalue,newvalue:shortstring;
  button,num:integer;
  key:char;
begin
   IF TextButton.Action(Xpos,Ypos) THEN
   begin
      Action := TRUE;
      hide_cursor;
      tempvalue := Name;
      ChangeName('');
      newvalue := '';
```

```
      repeat
      begin
         IF keypressed THEN
         begin
            key := readkey;
            CASE key OF
               '0'..'9':
               begin
                  IF (textwidth(newvalue+key) < Width) THEN
                     newvalue := newvalue + key;
               end;
               '.':
               begin
                  IF POS('.',newvalue) = 0 THEN
                     newvalue := newvalue + key;
               end;
               '-':
               begin
                  IF (textwidth(newvalue+key) < Width) THEN
                     IF Length(newvalue) = 0 THEN
                        newvalue := newvalue + key;
               end;
               chr(DEL):
                begin
                  IF (length(newvalue) > 0) THEN
                     newvalue[0] := chr(length(newvalue) - 1);
               end;
            end; {case}
            ChangeName(newvalue);
         end;
         button := get_buttons;
      end;
      until(key = chr(CR)) OR (key = chr(ESC))
            OR (button = right_button_pressed) OR (button = left_button_pressed);
      IF ((key = chr(CR)) OR (button = left_button_pressed)) AND (length(newvalue) > 0) THEN
      begin
         val(newvalue,value,num);
         str(value:int:frac,newvalue);
         val(newvalue,value,num); { do it again to truncate the unwanted characters }
                                  { e.g., if the value is 0.25 and you specified a formattin
                                  { of 5:0, you want to get 0 for the value, not 0.25 }
{           ChangeText(newvalue);}
      end
      else
      begin
         val(tempvalue,value,num);
         str(value:int:frac,tempvalue);
         val(tempvalue,value,num);{ do it again to truncate the unwanted characters }
                                  { e.g., if the value is 0.25 and you specified a formattin
```

```
                                  { of 5:0, you want to get 0 for the value, not 0.25
{          ChangeText(tempvalue);}
      end;
      show_cursor(XPos,Ypos);
   end
   ELSE
      Action := FALSE;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

constructor StatusButton.Init(InitX,InitY,InitWidth,InitHeight,
                 Initcolor,InitHiLiteColor,InitSelectedColor,
                 InitNameColor,InitHiliteNameColor,InitSelectedNameColor:word;
                 InitName:shortstring);
      {
      *******************************************************************
      *                                                                 *
      *        PURPOSE                                                  *
      *           Object initialization for a kind of text button that has  *
      *           a remembered on/off state. Usefull for selection controls. *
      *******************************************************************
      }
begin
   TextButton.Init(InitX,InitY,InitWidth,InitHeight,
                 Initcolor,InitHiLiteColor,
                 InitNameColor,InitHiliteNameColor,InitName,' ');
   SelectedColor := InitSelectedColor;
   SelectedNameColor := InitSelectedNameColor;
   Selected := FALSE;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

procedure StatusButton.Show;
      {
      *******************************************************************
      *                                                                 *
      *        PURPOSE                                                  *
      *           Method for showing a status button.                   *
      *******************************************************************
      }
var
   xx,yy:integer;
begin
   get_cursorXY(xx,yy);
   Hide_cursor;
   IF Selected THEN
   begin
```

```
      setfillstyle(solidfill,Selectedcolor);
      setcolor(SelectedNamecolor);
   end
   ELSE
   begin
      setfillstyle(solidfill,color);
      setcolor(Controlcolor);
   end;

   bar(X,Y,X+Width,Y+Height);
   settextjustify(centertext,centertext);
   outtextxy(X+Width DIV 2,Y + Height DIV 2 + 1,name);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure StatusButton.HiLite(Xpos,Ypos:word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for highlighting a status button.                   *
      ************************************************************************
      }
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   BEGIN
{
      IF NameChanged THEN
      BEGIN
         Hide_Cursor;
         ChangeName(name);
         Show_Cursor(Xpos,Ypos);
         NameChanged := FALSE;
      END;
}
      IF NOT Hilited THEN
      begin
         hide_cursor;
         Hilited := TRUE;
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         setcolor(hiliteControlcolor);
         settextjustify(centertext,centertext);
         outtextxy(X+Width DIV 2,Y + Height DIV 2 + 1,name);
         rectangle(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
```

```
      end;
   END
   ELSE
      IF HiLited THEN
      begin
         hilited := FALSE;
         Show;
      end;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

procedure StatusButton.TurnOn;
      {
      **********************************************************************
      *                                                                    *
      *        PURPOSE                                                     *
      *           Method for turning the status button state on.           *
      **********************************************************************
      }
var
   xx,yy:integer;
begin
   selected := TRUE;
   Show;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

procedure StatusButton.TurnOff;
      {
      **********************************************************************
      *                                                                    *
      *        PURPOSE                                                     *
      *           Method for turning the status button state off.          *
      **********************************************************************
      }
var
   xx,yy:integer;
begin
   selected := FALSE;
   Show;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

constructor TextButton3D.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitNameColor,InitHiliteNameColor:word;
            InitName:shortstring;InitHotKey:char);
```

```
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for three dimensional looking text button.          *
      ************************************************************************
      }
var
   x2, y2, i: Integer;
begin
   TextButton.Init(InitX,InitY,InitWidth,InitHeight,
                   Initcolor,InitHiLiteColor,InitNameColor,
                   InitHiliteNameColor,InitName,InitHotKey);
end;

(* >-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-< *)

procedure TextButton3D.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for three dimensional looking text button.          *
      ************************************************************************
      }
Var
   i,x2,y2,xx,yy:integer;
begin
   Get_CursorXY(xx,yy);
   hide_cursor;
   x2 := X+Width;
   y2 := Y+Height;
   TextButton.Show;
   setcolor(color);
   rectangle(X,Y,X+Width,Y+Height);
   SetColor(lightgray);
   FOR i := 0 to 2 DO BEGIN
      Line(X+1+i, Y+i, x2-i, Y+i);
      Line(X+i,Y+1+i, X+i, y2-i);
   END;
   SetColor(darkgray);
   FOR i := 0 to 2 DO BEGIN
      Line(x2-1-i, y2-i, X+1+i,y2-i);
      Line(x2-i, y2-1-i, x2-i, Y+1+i);
   END;
   show_cursor(xx,yy);
end;

(* >-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-< *)
```

```
procedure TextButton3D.Hilite(Xpos,Ypos:word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for three dimensional looking text button.          *
      ************************************************************************
      }
var
   i,x2,y2:integer;
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   BEGIN
      if not Hilited then begin
        TextButton.HiLite(Xpos,Ypos);
        hide_cursor;
        setcolor(hilitecolor);
        rectangle(X,Y,X+Width,Y+Height);
        SetColor(darkgray);
        x2 := X+Width;
        y2 := Y+Height;
        FOR i := 0 to 2 DO BEGIN
           Line(X+1+i, Y+i, x2-i, Y+i);
           Line(X+i,Y+1+i, X+i, y2-i);
        END;
        SetColor(lightgray);
        FOR i := 0 to 2 DO BEGIN
          Line(x2-1-i, y2-i, X+1+i,y2-i);
          Line(x2-i, y2-1-i, x2-i, Y+1+i);
        END;
        show_cursor(Xpos,Ypos);
     end;
   END
   ELSE
      IF HiLited THEN
      begin
         hilited := FALSE;
         Show;
      end;
end;

(* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>< *)

constructor EditButton3D.Init(InitX,InitY,InitWidth,InitHeight,
                              Initcolor,InitHiLiteColor,InitNameColor,
                              InitHiliteNameColor:word;
                              InitText:string;InitCursorPosition,
                              InitStringLength:integer;
```

```
                                InitClearString:boolean);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for three dimensional looking edit button.          *
      ************************************************************************
      }
begin
   EditButton.Init(InitX,InitY,InitWidth,InitHeight,
                   Initcolor,InitHiLiteColor,InitNameColor,
                   InitHiliteNameColor,InitText,InitCursorPosition,4,
                   InitStringLength,InitClearString);
end;

(* >-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-< *)

procedure EditButton3D.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for three dimensional looking edit button.          *
      ************************************************************************
      }
Var
   i,x2,y2,xx,yy:integer;
begin
   Get_CursorXY(xx,yy);
   hide_cursor;
   x2 := X+Width;
   y2 := Y+Height;
   EditButton.Show;
   setcolor(color);
   rectangle(X,Y,X+Width,Y+Height);
   SetColor(lightgray);
   FOR i := 0 to 2 DO BEGIN
      Line(X+1+i, Y+i, x2-i, Y+i);
      Line(X+i,Y+1+i, X+i, y2-i);
   END;
   SetColor(darkgray);
   FOR i := 0 to 2 DO BEGIN
      Line(x2-1-i, y2-i, X+1+i,y2-i);
      Line(x2-i, y2-1-i, x2-i, Y+1+i);
   END;
   show_cursor(xx,yy);
end;

(* >-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-<>-< *)
```

File Name: CONTROLS.PAS

```
procedure EditButton3D.Hilite(Xpos,Ypos:word);
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Method for three dimensional looking edit button.        *
      ***********************************************************************
      }
var
   i,x2,y2:integer;
begin
   IF (Xpos >= X)       AND
      (Xpos < X+Width) AND
      (Ypos >= Y)       AND
      (Ypos < Y+Height) THEN
   BEGIN
      if not(Hilited) then begin
        EditButton.HiLite(Xpos,Ypos);
        hide_cursor;
        setcolor(hilitecolor);
        rectangle(X,Y,X+Width,Y+Height);
        SetColor(darkgray);
        x2 := X+Width;
        y2 := Y+Height;
        FOR i := 0 to 2 DO BEGIN
           Line(X+1+i, Y+i, x2-i, Y+i);
           Line(X+i,Y+1+i, X+i, y2-i);
        END;
        SetColor(lightgray);
        FOR i := 0 to 2 DO BEGIN
          Line(x2-1-i, y2-i, X+1+i,y2-i);
          Line(x2-i, y2-1-i, x2-i, Y+1+i);
        END;
        show_cursor(Xpos,Ypos);
     end;
   END
   ELSE
      IF HiLited THEN
      begin
         hilited := FALSE;
         Show;
      end;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure TextButton3D.ChangeName(NewName:shortstring);
      {
```

File Name: CONTROLS.PAS

```
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for three dimensional looking text button.          *
      ************************************************************************
      }
var
   x2,y2,i:integer;
begin
   Name := NewName;
   Show;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor UPArrowButton.Init(InitX,InitY,InitWidth,InitHeight:integer;
          Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object initialization for an up arrow control display.     *
      ************************************************************************
      }
begin
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
          Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor);
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure UpArrowButton.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for displaying an up arrow control.                 *
      ************************************************************************
      }
var
   xx,yy:integer;
begin
   get_cursorXY(xx,yy);
   Hide_cursor;
   setfillstyle(solidfill,color);
   bar(X,Y,X+Width,Y+Height);
   setcolor(ControlColor);
   line(X+Width DIV 2,Y+2,X+2,Y+Height-2);
   line(X+Width DIV 2,Y+2,X+Width-2,Y+Height-2);
```

```
   line(X+2,Y+Height-2,X+Width-2,Y+Height-2);
   setfillstyle(solidfill,Controlcolor);
   floodfill(X+Width DIV 2,Y+Height DIV 2,Controlcolor);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;

(* >><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure UpArrowButton.HiLite(Xpos,Ypos:Word);
      {
      *******************************************************************
      *                                                                 *
      *         PURPOSE                                                 *
      *            Method for highlighting an up arrow control.         *
      *******************************************************************
      }
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   begin
      IF NOT Hilited THEN
      begin
         hide_cursor;
         Hilited := TRUE;
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         setcolor(HiliteControlColor);
         line(X+Width DIV 2,Y+2,X+2,Y+Height-2);
         line(X+Width DIV 2,Y+2,X+Width-2,Y+Height-2);
         line(X+2,Y+Height-2,X+Width-2,Y+Height-2);
         setfillstyle(solidfill,HiliteControlColor);
         floodfill(X+Width DIV 2,Y+Height DIV 2,HiliteControlcolor);
         rectangle(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
      end
   end
   ELSE
      IF Hilited THEN
      begin
        HiLited := FALSE;
        Show;
      end;
end;

(* >><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor DownArrowButton.Init(InitX,InitY,InitWidth,InitHeight:integer;
            Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
      {
```

File Name: CONTROLS.PAS

```
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Object initialization for a down arrow control display.    *
      ************************************************************************
      }
begin
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
           Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor);
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

procedure DownArrowButton.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for down arrow control display.                     *
      ************************************************************************
      }
var
   xx,yy:integer;
begin
   get_cursorXY(xx,yy);
   Hide_cursor;
   setfillstyle(solidfill,color);
   bar(X,Y,X+Width,Y+Height);
   setcolor(ControlColor);
   line(X+Width DIV 2,Y+Height-2,X+2,Y+2);
   line(X+Width DIV 2,Y+Height-2,X+Width-2,Y+2);
   line(X+2,Y+2,X+Width-2,Y+2);
   setfillstyle(solidfill,Controlcolor);
   floodfill(X+Width DIV 2,Y+Height DIV 2,Controlcolor);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

procedure DownArrowButton.HiLite(Xpos,Ypos:Word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for down arrow control highlighting.                *
      ************************************************************************
      }
begin
```

```
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   begin
      IF NOT Hilited THEN
      begin
         hide_cursor;
         Hilited := TRUE;
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         setcolor(HiliteControlColor);
         line(X+Width DIV 2,Y+Height-2,X+2,Y+2);
         line(X+Width DIV 2,Y+Height-2,X+Width-2,Y+2);
         line(X+2,Y+2,X+Width-2,Y+2);
         setfillstyle(solidfill,HiliteControlColor);
         floodfill(X+Width DIV 2,Y+Height DIV 2,HiliteControlcolor);
         rectangle(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
      end
   end
   ELSE
      IF Hilited THEN
      begin
        HiLited := FALSE;
        Show;
      end;
end;

(* >-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-< *)

constructor LeftArrowButton.Init(InitX,InitY,InitWidth,InitHeight:integer;
            Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Initialization for left arrow control.                     *
      ************************************************************************
      }
begin
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor);
end;

(* >-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-<-< *)

procedure LeftArrowButton.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
```

```
      *           Method for left arrow control display.                    *
      ****************************************************************************
      }
var
   xx,yy:integer;
begin
   get_cursorXY(xx,yy);
   Hide_cursor;
   setfillstyle(solidfill,color);
   bar(X,Y,X+Width,Y+Height);
   setcolor(ControlColor);
   line(X+2,Y+Height DIV 2,X+Width-2,Y+2);
   line(X+2,Y+Height DIV 2,X+Width-2,Y+Height-2);
   line(X+Width-2,Y+2,X+Width-2,Y+Height-2);
   setfillstyle(solidfill,Controlcolor);
   floodfill(X+Width DIV 2,Y+Height DIV 2,Controlcolor);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure LeftArrowButton.HiLite(Xpos,Ypos:Word);
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           Method for left arrow control highlighting.                    *
      ****************************************************************************
      }
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   begin
      IF NOT Hilited THEN
      begin
         hide_cursor;
         Hilited := TRUE;
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         setcolor(HiliteControlColor);
         line(X+2,Y+Height DIV 2,X+Width-2,Y+2);
         line(X+2,Y+Height DIV 2,X+Width-2,Y+Height-2);
         line(X+Width-2,Y+2,X+Width-2,Y+Height-2);
         setfillstyle(solidfill,HiliteControlColor);
         floodfill(X+Width DIV 2,Y+Height DIV 2,HiliteControlcolor);
         rectangle(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
      end
   end
```

```
   ELSE
      IF Hilited THEN
      begin
        HiLited := FALSE;
        Show;
      end;
end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

constructor RightArrowButton.Init(InitX,InitY,InitWidth,InitHeight:integer;
            Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor:word)
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Initialization for right arrow control.                    *
      ************************************************************************
      }
begin
   FixedControl.Init(InitX,InitY,InitWidth,InitHeight,
            Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor);
end;

procedure RightArrowButton.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for right arrow control display.                    *
      ************************************************************************
      }
var
   xx,yy:integer;
begin
   get_cursorXY(xx,yy);
   Hide_cursor;
   setfillstyle(solidfill,color);
   bar(X,Y,X+Width,Y+Height);
   setcolor(ControlColor);
   line(X+width-2,Y+Height DIV 2,X+2,Y+2);
   line(X+width-2,Y+Height DIV 2,X+2,Y+Height-2);
   line(X+2,Y+2,X+2,Y+Height-2);
   setfillstyle(solidfill,Controlcolor);
   floodfill(X+Width DIV 2,Y+Height DIV 2,Controlcolor);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
end;
```

File Name: CONTROLS.PAS

```
(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

procedure RightArrowButton.HiLite(Xpos,Ypos:Word);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method for right arrow control highlighting.                  *
      ***************************************************************************
      }
begin
   IF (Xpos >= X) AND (Xpos < X+Width) AND (Ypos >= Y) AND (Ypos < Y+Height) THEN
   begin
      IF NOT Hilited THEN
      begin
         hide_cursor;
         Hilited := TRUE;
         setfillstyle(solidfill,hilitecolor);
         bar(X,Y,X+Width,Y+Height);
         setcolor(HiliteControlColor);
         line(X+width-2,Y+Height DIV 2,X+2,Y+2);
         line(X+width-2,Y+Height DIV 2,X+2,Y+Height-2);
         line(X+2,Y+2,X+2,Y+Height-2);
         setfillstyle(solidfill,HiliteControlColor);
         floodfill(X+Width DIV 2,Y+Height DIV 2,HiliteControlcolor);
         rectangle(X,Y,X+Width,Y+Height);
         show_cursor(Xpos,Ypos);
      end
   end
   ELSE
      IF Hilited THEN
      begin
        HiLited := FALSE;
        Show;
      end;
end;

(* >><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><<>><< *)

function compute_delta(VAR V,Hi,Lo:real;Acc:integer;
                       Accelf:real;orientation:boolean):integer;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Function for computing how far a slide bar slider control     *
      *           has travelled.                                                *
      ***************************************************************************
      }
```

File Name: CONTROLS.PAS

```
var
   del:integer;
   templength : integer;

begin
      if orientation = horizontal then
        templength := H_SlideLength
      else  (* vertical *)
        templength := V_SlideLength;

      IF v >= Hi THEN
      begin
         v := Hi;
         del := templength;
      end
      else
         IF V <= Lo THEN
         begin
            v := Lo;
            del := 0;
         end
         ELSE
            CASE Acc OF
               logscale:
                  del := round(Ln(v - Lo)/AccelF);
               reverselogscale:
                  del := round(templength - Ln(Hi-V)/AccelF);
               linearscale:
                  del := round((V-Lo)/AccelF);
            END; { CASE }
   IF del >= templength-2 THEN
      del := templength-2;
   compute_delta := del;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor HorizontalSlideBar.Init(InitX,InitY,InitColor,InitHiliteColor,
            InitControlColor,InitHiliteControlColor:word;
            InitLoLim,InitHiLim,InitInc:real;
            InitAccel:integer;InitName,Initunits:shortstring;InitInt,InitFrac:integer);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Initialization for a horizontal slide bar control.         *
      ************************************************************************
      }
var
```

```
   initvalue: real;
   delta:integer;
begin
   Name := InitName;
   Units := Initunits;
   HiLim := InitHiLim;
   LoLim := InitLoLim;
   Increment := InitInc;
   Accel := InitAccel;
   CASE Accel OF
      logscale:
         AccelFactor := Ln(HiLim-LoLim)/H_SlideLength;
      reverselogscale:
         AccelFactor := Ln(HiLim-LoLim)/H_SlideLength;
      linearscale:
         AccelFactor := (HiLim - LoLim)/H_SlideLength;
      else
      begin
         Accel := linearscale;
         AccelFactor := (HiLim - LoLim)/H_SlideLength;
      end;
   END; { CASE }
   initvalue := LoLim + (HiLim - LoLim)/2;
   FixedControl.Init(InitX,InitY,H_barwidth,H_barheight,
         Initcolor,InitHiLiteColor,InitControlColor,InitHiliteControlColor);
   Right.Init(X+Width-arrowwidth,Y,arrowwidth,H_barheight,
         color,HiLiteColor,ControlColor,HiliteControlColor);
   Left.Init(X,Y,arrowwidth,H_barheight,
         color,HiLiteColor,ControlColor,HiliteControlColor);
   SlideBar.Init(X+arrowwidth+1,Y+1,slidewidth,Height-2,
         {Hilitecolor,HiliteControlColor,Color,ControlColor);}
         ControlColor,HiLiteColor,Color,HiLiteControlColor);
   Value.Init(X+H_barwidth+5,Y,valueWidth,H_barheight,
         color,HiLiteColor,ControlColor,HiliteControlColor,initvalue,InitInt,InitFrac);
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure HorizontalSlideBar.Hilite(Xpos,Ypos:word);
      {
      ***********************************************************************
      *                                                                     *
      *       PURPOSE                                                       *
      *          Method for a horizontal slide bar control highlighting.   *
      ***********************************************************************
      }
begin
   Left.Hilite(Xpos,Ypos);
   Right.Hilite(Xpos,Ypos);
```

```
   SlideBar.Hilite(Xpos,Ypos);
   Value.Hilite(Xpos,Ypos);
end;

(* >< *)

procedure HorizontalSlideBar.Show;
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Method for a horizontal slide bar control display.        *
      ***********************************************************************
      }
var
   delta:integer;
   xx,yy:integer;
begin
   get_cursorxy(xx,yy);
   setfillstyle(solidfill,color);
   hide_cursor;
   bar(X,Y,X+Width,Y+Height);
   setcolor(ControlColor);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
   Right.Show;
   Left.Show;
   SlideBar.Show;
   Value.Show;
   delta := compute_delta(Value.value,HiLim,LoLim,Accel,Accelfactor,Horizontal);
   SlideBar.MoveTo(X+arrowwidth+1+delta,SlideBar.Y);
   get_cursorxy(xx,yy);
   hide_cursor;
   setcolor(controlcolor);
   settextjustify(righttext,centertext);
   outtextXY(X-5,Y+height DIV 2 + 1,Name);
   settextjustify(lefttext,centertext);
   outtextXY(value.X+valuewidth+5,Y+height DIV 2 + 1,units);
   show_cursor(xx,yy);
end;

(* >< *)

function HorizontalSlideBar.Action(Xpos,Ypos:word):boolean;
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           Method for a horizontal slide bar control action.         *
```

File Name: CONTROLS.PAS

```
      ************************************************************************
      }
var
  s:shortstring;
  oldx,xinc,newx,newy:integer;
  delta:integer;
begin

   Action := FALSE;

   IF Left.Action(Xpos,Ypos) THEN
   begin
      Value.value := Value.value - Increment;
      delta := compute_delta(Value.value,HiLim,LoLim,Accel,Accelfactor,Horizontal);
      str(Value.value:value.int:value.frac,s);
      Hide_Cursor;
      Value.ChangeName(s);
      SlideBar.MoveTo(X+arrowwidth+1+delta,SlideBar.Y);
      Show_Cursor(Xpos,Ypos);
      Action := TRUE;
   end;
   IF Right.Action(Xpos,Ypos) THEN
   begin
      Value.value := Value.value + Increment;
      delta := compute_delta(Value.value,HiLim,LoLim,Accel,Accelfactor,Horizontal);
      str(Value.value:value.int:value.frac,s);
      Hide_Cursor;
      Value.ChangeName(s);
      SlideBar.MoveTo(X+arrowwidth+1+delta,SlideBar.Y);
      Show_Cursor(Xpos,Ypos);
      Action := TRUE;
   end;
   if SlideBar.Action(Xpos,Ypos) THEN
   begin
      oldx := Xpos;
      WHILE Left_Button_Down DO
      begin
         Get_CursorXY(newx,newy);
         IF (newx <> oldx) OR (newy <> Ypos) THEN
         begin
            hide_cursor;
            xinc := newx-oldx;
            IF SlideBar.X + xinc > Right.X - SlideBar.Width - 1 THEN
            begin
               newx := oldx;
               xinc := 0;
            end;
            IF SlideBar.X + xinc < Left.X + Left.Width + 1 THEN
            begin
```

```
                newx := oldx;
                xinc := 0;
             end;
             IF xinc <> 0 THEN
             begin
                delta := SlideBar.X+xinc-X-arrowwidth-1;
                SlideBar.MoveTo(X+arrowwidth+1+delta,SlideBar.Y);
                IF (delta = 0) THEN
                   Value.value := LoLim
                ELSE
                   CASE Accel OF
                      logscale:
                         Value.value := exp(delta*AccelFactor) + LoLim;
                      reverselogscale:
                         Value.value := HiLim - exp((H_SlideLength-2-delta)*AccelFactor);
                      linearscale:
                         Value.value := delta*AccelFactor+LoLim;
                   END; { CASE }
                if delta >= H_SlideLength-2 THEN
                   Value.value := HiLim;
                IF (Value.value > HiLim) THEN
                   Value.value := HiLim;
                IF Value.value < LoLim THEN
                   Value.value := LoLim;
                str(Value.value:value.int:value.frac,s);
                Value.ChangeName(s);
             end;
             Show_Cursor(newx,Ypos);
          end;
          oldx := newx;
       end;
       Action := TRUE;
    end;
    IF Value.Action(Xpos,Ypos) THEN
    begin
       delta := compute_delta(Value.value,HiLim,LoLim,Accel,Accelfactor,Horizontal);
       str(Value.value:value.int:value.frac,s);
       Hide_Cursor;
       Value.ChangeName(s);
       SlideBar.MoveTo(X+arrowwidth+1+delta,SlideBar.Y);
       Show_Cursor(Xpos,Ypos);
       Action := TRUE;
    end;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

constructor VerticalSlideBar.Init(InitX,InitY,InitColor,InitHiliteColor,
                                  InitControlColor,
```

```
                                  InitHiliteControlColor:word;
                                  InitLoLim,InitHiLim,
                                  InitInc:real;InitAccel:integer);
     {
     *****************************************************************
     *                                                               *
     *       PURPOSE                                                 *
     *          Initialization for a vertical   slide bar control.   *
     *****************************************************************
     }
var
  delta:integer;

begin
  Increment := InitInc;
  Accel := InitAccel;
  LoLim := InitLoLim;
  HiLim := InitHiLim;


  FixedControl.Init(InitX, InitY, V_BarWidth, V_BarHeight,
                    Initcolor, InitHiLiteColor, InitControlColor,
                    InitHiliteControlColor);
  Up.Init(X, Y, V_BarWidth, arrowheight, color,
          HiLiteColor, ControlColor, HiliteControlColor);
  Down.Init(X, Y+Height-arrowheight, V_BarWidth, arrowheight,
           color,HiLiteColor,ControlColor,HiliteControlColor);
  SlideBar.Init(X+1, Y+arrowheight+1, Width-2, slideheight,
                ControlColor, HiLiteColor, Color, HiLiteControlColor);
end;

(* <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

procedure VerticalSlideBar.Hilite(Xpos,Ypos:word);
     {
     *****************************************************************
     *                                                               *
     *       PURPOSE                                                 *
     *          Method for a vertical slide bar control highlighting. *
     *****************************************************************
     }

begin
  Up.Hilite(Xpos,Ypos);
  Down.Hilite(Xpos,Ypos);
  SlideBar.Hilite(Xpos,Ypos);
end;

(* <><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)
```

File Name: CONTROLS.PAS

```
procedure VerticalSlideBar.Show;
      {
      *********************************************************************
      *                                                                   *
      *       PURPOSE                                                     *
      *          Method for a vertical slide bar control display.         *
      *********************************************************************
      }

var
   delta:integer;
   xx,yy:integer;

begin
   get_cursorxy(xx,yy);
   setfillstyle(solidfill,color);
   hide_cursor;
   bar(X,Y,X+Width,Y+Height);
   setcolor(ControlColor);
   rectangle(X,Y,X+Width,Y+Height);
   show_cursor(xx,yy);
   Up.Show;
   Down.Show;
   SlideBar.Show;
   delta := compute_delta(Value,HiLim,LoLim,Accel,Accelfactor,Vertical);
   SlideBar.MoveTo(SlideBar.X, Y+arrowheight+1+delta);
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)

function VerticalSlideBar.Action(Xpos,Ypos:word):boolean;
      {
      *********************************************************************
      *                                                                   *
      *       PURPOSE                                                     *
      *          Method for a vertical slide bar control action.          *
      *********************************************************************
      }

var
  dummy:boolean;
  s:shortstring;
  oldy,yinc,newx,newy:integer;
  delta:integer;

begin
   Action:=false;
```

File Name: CONTROLS.PAS

```
  CASE Accel OF
    logscale        : AccelFactor := Ln(HiLim-LoLim)/V_SlideLength;
    reverselogscale : AccelFactor := Ln(HiLim-LoLim)/V_SlideLength;
    linearscale     : AccelFactor := (HiLim - LoLim)/V_SlideLength;
    else begin
      Accel := linearscale;
      AccelFactor := (HiLim - LoLim)/V_SlideLength;
    end;
  END; { CASE }

  if Down.Action(Xpos,Ypos) then begin
    Action:=true;
      if Value < HiLim then
        Value := Value + increment;
      delta := compute_delta(Value,HiLim,LoLim,Accel,Accelfactor,Vertical);
      Hide_Cursor;
      SlideBar.MoveTo(SlideBar.X, Y+arrowheight+1+delta);
      Show_Cursor(Xpos,Ypos);
  end;

  if Up.Action(Xpos,Ypos) then begin
    Action:=true;
    if Value > LoLim then
      Value := Value - increment;
    delta := compute_delta(Value,HiLim,LoLim,Accel,Accelfactor,Vertical);
    Hide_Cursor;
    SlideBar.MoveTo(SlideBar.X, Y+arrowheight+1+delta);
    Show_Cursor(Xpos,Ypos);
  end;

  if SlideBar.Action(Xpos,Ypos) then begin
    Action:=true;
    oldy := Ypos;

    while Left_Button_Down do begin

      Get_CursorXY(newx,newy);

      if (newx <> Xpos) OR (newy <> oldy) then begin
        hide_cursor;
        yinc := newy - oldy;

        if SlideBar.Y + yinc > Down.Y - SlideBar.Height - 1 then begin
          newy := oldy;
          yinc := 0;
        end;

        if SlideBar.Y + yinc < Up.Y + Up.Height + 1 then begin
          newy := oldy;
```

```
          yinc := 0;
        end;

        if yinc <> 0 then begin

          delta := SlideBar.Y + yinc - Y - arrowheight - 1;
          Value :=  Trunc(HiLim * (delta/V_SlideLength));
          SlideBar.MoveTo(SlideBar.X, Y+arrowheight+1+delta);

          if delta >= V_SlideLength-2 then
            Value := HiLim;
          if (Value > HiLim) then
            Value := HiLim;
          if Value < LoLim THEN
            Value := LoLim;

        end;
        Show_Cursor(Xpos,newy);
      end;
      oldy := newy;
     end;
  end;
end;

(* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> *)

begin
  WindowImageNumber := 0;   (* no images stored at start of program *)
end.
```

File Name: COVGRID.PAS

```
(*****************************************************************************
*                                                                           *
*        PROGRAM NAME - Omega Performance Assessment                        *
*                                and                                        *
*                        Coverage Evaluation                                *
*                              (PACE)                                       *
*                           Workstation                                     *
*                                                                           *
*          UNIT NAME - Part of the PACEOBJS Unit                            *
*                                                                           *
*****************************************************************************
*                                                                           *
*        This program was prepared by                                       *
*                                                                           *
*              The Analytic Science Corporation (TASC)                      *
*                    55 Walkers Brook Drive                                 *
*                Reading, Massachusetts 01867                               *
*                                                                           *
*        PACE has been developed to run on a IBM PC/AT or compatible        *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main          *
*        memory and an EGA or compatible graphics adapter and color         *
*        monitor. This work was performed under contract number             *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for          *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.       *
*                                                                           *
*****************************************************************************
*                                                                           *
*        PURPOSE                                                            *
*          This file contains the method for showing the coverage           *
*          grid information using the coveragegrid object.                  *
*                                                                           *
*                                                                           *
*****************************************************************************)
  Procedure CoverageGrid.Show;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *          Method to fill in the pertinent information on the coverage*
    *          cell grid cells, and then to display them along with the   *
    *          appropriate user selected thresholds.                      *
    ***************************************************************************
    }
  VAR
    xx,yy     : integer;
    i: integer;
    DB102,DB136: TimeSpecificDBFile;
    info102,info136: Covcell;
    S102,S136,L102,L136,D102,D136,A102,A136: real;
```

```
      st: string;
{
      archivefile: TimeSpecificArchiveFile;
      archiveinfo: cellrecord;
      GDOP: byte;
}

   begin
{
      Assign(archivefile,ARCHIVEPATH + '\' + cellpopup^.archivename);
      Reset(archivefile);
}
      Assign(DB102,DATABASEPATH+'\'+DATABASE102);
      Reset(DB102);
      Assign(DB136,DATABASEPATH+'\'+DATABASE136);
      Reset(DB136);
      Get_CursorXY(xx,yy);
      Hide_Cursor;

      FOR i := 1 TO NCELLS DO
      begin
   { get the database data for each cell and color it appropriately.
     The coloring scheme is as follows:

     SNR - red
     S/L ratio - light red
     Modal - M in the cell center
     XAngle - X across the cell

     The various object variables associated with a cell type variable are
     used to hold flags that indicate the condition of the above threshold
     tests. These are related as follows:

     selfptr^.abovecolor - SNR or S/L color (red or light red or lightblue of OK)
     selfptr^.belowcolor - 0 if not modal, else modal
     selfptr^.weight or coverage - 0 if Xangle OK, else not OK
   }

         cellarray[i].abovecolor := lightblue;
   { read the data for the frequencies selected }
         CASE cellpopup^.eptr^.Freq.picked OF
            FRQ102:
            begin
               ReadTimeSpecificFromDatabase(DB102,i,subcellpopup^.stations.picked,
                                            cellpopup^.month,cellpopup^.hour,info102);
            end;
            FRQ136:
            begin
               ReadTimeSpecificFromDatabase(DB136,i,subcellpopup^.stations.picked,
```

```
                                    cellpopup^.month,cellpopup^.hour,info136);
            end;
            ELSE
            begin
               ReadTimeSpecificFromDatabase(DB102,i,subcellpopup^.stations.picked,
                                    cellpopup^.month,cellpopup^.hour,info102);
               ReadTimeSpecificFromDatabase(DB136,i,subcellpopup^.stations.picked,
                                    cellpopup^.month,cellpopup^.hour,info136);
            end;
         end; { CASE }

{
         ReadTimeSpecificFromArchive(archivefile,i,cellpopup^.month,
                                    cellpopup^.hour,archiveinfo);
         GDOP := ReadGdop(i,archiveinfo.coverage);
}
         S102 := Get_SNR(info102.SNR_S)/8;
         L102 := Get_SNR(info102.SNR_L)/8;
         D102 := info102.Phase AND $7F;
         A102 := info102.X_Ang;
         S136 := Get_SNR(info136.SNR_S)/8;
         L136 := Get_SNR(info136.SNR_L)/8;
         D136 := info136.Phase AND $7F;
         A136 := info136.X_Ang;

         IF (cellpopup^.eptr^.Freq.picked = FRQ102) THEN
         begin
            if (S102-L102 < Round(CellPopUp^.eptr^.ShortLongRatio.value.value)) then
               cellarray[i].abovecolor := lightred;
            if (S102 < Round(CellPopUp^.eptr^.SNR.value.value)) then
               cellarray[i].abovecolor := red;
            IF ((info102.Phase AND $80) = 0) AND
                ((D102 <= Round(CellPopUp^.eptr^.PhaseDev.value.value)) or
                 (D102 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))) THEN
               cellarray[i].belowcolor := 0
            ELSE
               cellarray[i].belowcolor := 255;
            if (A102 >= Round(CellPopUp^.eptr^.XAngle.value.value)) then
               cellarray[i].weight_or_coverage := 0
            ELSE
               cellarray[i].weight_or_coverage := 255;
         end;

         IF (cellpopup^.eptr^.Freq.picked = FRQ136) THEN
         begin
            if (S136-L136 < Round(CellPopUp^.eptr^.ShortLongRatio.value.value)) then
               cellarray[i].abovecolor := lightred;
            if (S136 < Round(CellPopUp^.eptr^.SNR.value.value)) then
               cellarray[i].abovecolor := red;
```

```
            IF ((info136.Phase AND $80) = 0) AND
                ((D136 <= Round(CellPopUp^.eptr^.PhaseDev.value.value)) or
                 (D136 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))) THEN
               cellarray[i].belowcolor := 0
            ELSE
               cellarray[i].belowcolor := 255;
            if (A136 >= Round(CellPopUp^.eptr^.XAngle.value.value)) then
               cellarray[i].weight_or_coverage := 0
            ELSE
               cellarray[i].weight_or_coverage := 255;
         end;

         IF (cellpopup^.eptr^.Freq.picked = FRQOR) THEN
         begin
            IF (
                (((info102.Phase AND $80) = 0)
               AND
                (
                 (D102 <= Round(CellPopUp^.eptr^.PhaseDev.value.value))
                 OR
                 (D102 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))
                ))
               OR
               (((info136.Phase AND $80) = 0)
               AND
                 (
                  (D136 <= Round(CellPopUp^.eptr^.PhaseDev.value.value))
                  OR
                  (D136 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))
                 ))
               ) THEN
               cellarray[i].belowcolor := 0
            ELSE
               cellarray[i].belowcolor := 255;
            if (A102 >= Round(CellPopUp^.eptr^.XAngle.value.value))
               OR (A136 >= Round(CellPopUp^.eptr^.XAngle.value.value)) then
               cellarray[i].weight_or_coverage := 0
            ELSE
               cellarray[i].weight_or_coverage := 255;
         end;

         IF (cellpopup^.eptr^.Freq.picked = FRQAND) THEN
         begin
            if (S102-L102 < Round(CellPopUp^.eptr^.ShortLongRatio.value.value))
               AND (S136-L136 < Round(CellPopUp^.eptr^.ShortLongRatio.value.value)) then
               cellarray[i].abovecolor := lightred;
            if (S102 < Round(CellPopUp^.eptr^.SNR.value.value))
               AND (S136 < Round(CellPopUp^.eptr^.SNR.value.value)) then
               cellarray[i].abovecolor := red;
```

File Name: COVGRID.PAS

```
            IF (
                ((info102.Phase AND $80) = 0)
               AND
                (
                 (D102 <= Round(CellPopUp^.eptr^.PhaseDev.value.value))
                 OR
                 (D102 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))
                )
               AND
                ((info136.Phase AND $80) = 0)
               AND
                (
                 (D136 <= Round(CellPopUp^.eptr^.PhaseDev.value.value))
                 OR
                 (D136 >= 100 - Round(CellPopUp^.eptr^.PhaseDev.value.value))
                )
               ) THEN
               cellarray[i].belowcolor := 0
            ELSE
               cellarray[i].belowcolor := 255;
            if (A102 >= Round(CellPopUp^.eptr^.XAngle.value.value))
               AND (A136 >= Round(CellPopUp^.eptr^.XAngle.value.value)) then
               cellarray[i].weight_or_coverage := 0
            ELSE
               cellarray[i].weight_or_coverage := 255;
         end;

{
         IF (GDOP > Round(CellPopUp^.eptr^.GDOP.value.value)*10) THEN
            cellarray[i].abovethreshold := TRUE
         ELSE
            cellarray[i].abovethreshold := FALSE;
}

         CellArray[i].Show;
      end;
      FOR i := 1 TO NCELLS DO
         if CellArray[i].Region then
            CellArray[i].ShowRegion;

      { hilite the cell that was picked at the start of all this }
      cellarray[CellPopUp^.HilitedCellPtr^.cellnumber].hilite(xx,yy);

      { now write some kind of legend at the bottom of the screen }
      setcolor(yellow);
      settextjustify(lefttext,toptext);
      str(Round(CellPopUp^.eptr^.SNR.value.value),st);
      outtextxy(20,getmaxy-20,'SNR < '+st+' = ');
      setfillstyle(solidfill,red);
```

File Name: COVGRID.PAS

```
      bar(20+textwidth('SNR < '+st+' = ')+5,getmaxy-20,
          20+textwidth('SNR < '+st+' = ')+25,getmaxy-20+textheight('1'));
      str(Round(CellPopUp^.eptr^.SHORTLONGRATIO.value.value),st);
      outtextxy(160,getmaxy-20,'S/L < '+st+' = ');
      setfillstyle(solidfill,lightred);
      bar(160+textwidth('S/L < '+st+' = ')+5,getmaxy-20,
          160+textwidth('S/L < '+st+' = ')+25,getmaxy-20+textheight('1'));
      str(Round(CellPopUp^.eptr^.XANGLE.value.value),st);
      outtextxy(300,getmaxy-20,'ANG < '+st+' = ');
      setcolor(black);
      line(300+textwidth('ANG < '+st+' = ')+5,getmaxy-20,
          300+textwidth('ANG < '+st+' = ')+25,getmaxy-20+textheight('1'));
      line(300+textwidth('ANG < '+st+' = ')+25,getmaxy-20,
          300+textwidth('ANG < '+st+' = ')+5,getmaxy-20+textheight('1'));
      setcolor(yellow);
      str(Round(CellPopUp^.eptr^.PhaseDev.value.value),st);
      outtextxy(440,getmaxy-20,'Modal (DEV > '+st+') = ');
      setcolor(white);
      outtextxy(440+textwidth('Modal (DEV > '+st+') = ')+5,getmaxy-20,'M');

{ added the frequency to the coverage display 12/18/90 GRD }
      CASE CellPopUp^.eptr^.Freq.picked OF
         FRQ102 : st := '10.2';
         FRQ136 : st := '13.6';
         FRQAND : st := '10.2 AND 13.6';
         FRQOR  : st := '10.2 OR 13.6';
      end; { case }

      IF cellpopup^.hour < 10 THEN
         Outtextxy(getmaxx DIV 2 -
            textwidth(MonthNames[CellPopUp^.month]+'   0'+HourNames[cellpopup^.hour] +
                      '00   Station:  '+'   Freq: 10.2 AND 13.6') DIV 2,getmaxy - 10,
            MonthNames[CellPopUp^.month]+'   0'+HourNames[cellpopup^.hour] +'00   Station: '
            +stationnames[subcellpopup^.stations.picked]+'   Freq: '+st)
      ELSE
         Outtextxy(getmaxx DIV 2 -
            textwidth(MonthNames[CellPopUp^.month]+'   '+HourNames[cellpopup^.hour] +
                      '00   Station:  '+'   Freq: 10.2 AND 13.6') DIV 2,getmaxy - 10,
            MonthNames[CellPopUp^.month]+'   '+HourNames[cellpopup^.hour] +'00   Station: '
            +stationnames[subcellpopup^.stations.picked]+'   Freq: '+st);
{
         Outtextxy(getmaxx DIV 2 -
            textwidth(MonthNames[CellPopUp^.month]+'   '+HourNames[cellpopup^.hour] +'00   S
            MonthNames[CellPopUp^.month]+'   '+HourNames[cellpopup^.hour] +'00   Station: '
            +stationnames[subcellpopup^.stations.picked]);
}

      Show_Cursor(xx,yy);
      Close(DB102);
```

```
      Close(DB136);
{
      Close(archivefile);
}
   end;
```

File Name: CURSROBJ.PAS

```
UNIT CursrObj;
{*****************************************************************************
*                                                                            *
*        PROGRAM NAME - Omega Performance Assessment                         *
*                                and                                         *
*                        Coverage Evaluation                                 *
*                            (PACE)                                          *
*                          Workstation                                       *
*                                                                            *
*           UNIT NAME - CURSROBJ                                             *
*                                                                            *
*****************************************************************************
*                                                                            *
*        This program was prepared by                                        *
*                                                                            *
*              The Analytic Science Corporation (TASC)                       *
*                    55 Walkers Brook Drive                                  *
*                Reading, Massachusetts 01867                                *
*                                                                            *
*        PACE has been developed to run on a IBM PC/AT or compatible         *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main           *
*        memory and an EGA or compatible graphics adapter and color          *
*        monitor. This work was performed under contract number              *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for           *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.        *
*                                                                            *
*****************************************************************************
*                                                                            *
*        PURPOSE                                                             *
*             Cursor handling and cursor-sensitive object                    *
*        management routines used to generate the "point and shoot"          *
*        type user interface.                                                *
*                                                                            *
*****************************************************************************}

INTERFACE
  USES
    Graph, Dos, Crt, errlog;

  CONST
    Click_on_Button = TRUE;          { provide an audible click when }
                                     { a mouse button is depressed ? }

  TYPE
    Cursor_kinds = (arrow, finger,wait);

  CONST

    ENTER = 13;                       { special keys }
```

File Name: CURSROBJ.PAS

```
      ESC = 27;
      BACKSPACE = 8;

      FKEY_1 = 186;                    { function key constants }
      FKEY_2 = 187;
      FKEY_3 = 188;
      FKEY_4 = 189;
      FKEY_5 = 190;
      FKEY_6 = 191;
      FKEY_7 = 192;
      FKEY_8 = 193;
      FKEY_9 = 194;
      FKEY_10 = 195;

      PGDN = 208;                      { help constants }
      PGUP = 200;

      LEFT_BUTTON_PRESSED = 1;         { mouse constants }
      RIGHT_BUTTON_PRESSED = 2;
      X_MIN_LIMIT = 0;                 { cursor boundaries}
      X_MAX_LIMIT = 649;
      Y_MIN_LIMIT = 0;                 { cursor boundaries}
      Y_MAX_LIMIT = 347;

      FWD_NEXT_OBJECT = 9;             { cursor control constants }
      BKWD_NEXT_OBJECT = 142;
      HEAD_OBJECT = 198;
      UP_ARROW = 199;
      LEFT_ARROW = 202;
      RIGHT_ARROW = 204;
      LAST_OBJECT = 206;
      DOWN_ARROW = 207;
      INSERT_KEY = 209;
      CTRL_LEFT_ARROW = 242;
      CTRL_RIGHT_ARROW = 243;
      CTRL_END_ARROW = 244;
      CTRL_PGDWN_ARROW = 245;
      CTRL_HOME_ARROW = 246;
      CTRL_PGUP_ARROW = 259;


   VAR
      _Cursor_set : SET OF UP_ARROW..CTRL_HOME_ARROW;
      _Active_page, _Display_Page: Word;

   PROCEDURE Beep;

   PROCEDURE Get_Key(VAR KEY_PRESSED : Integer);
   { get the key pressed value if there is one }
```

File Name: CURSROBJ.PAS

```
   PROCEDURE Set_Cursor_Shape(ckind: cursor_kinds);
   {    Set the cursor shape by passing kind flag }

   PROCEDURE GET_CURSORXY(VAR X_Coord : Integer; VAR Y_Coord : Integer);
   {    Get the current cursor coordinates }

   PROCEDURE Hide_Cursor;
   {    Hide the cursor - remove it from the display }

   PROCEDURE Show_Cursor(X_Coord : Integer; Y_Coord : Integer);
   {    Show the cursor at the indicated coordinates of the cursor "hot spot" }

   PROCEDURE Move_Cursor(_Key_Pressed: Integer; VAR Old_X_Coord, Old_Y_Coord : Integer);
   {    Move the cursor according to a the cursor movement key pressed }

   FUNCTION Left_Button_Down: Boolean;
   {    Indicates whether the left mouse button is depressed }

   FUNCTION Right_Button_Down: Boolean;
   {    Indicates whether the left mouse button is depressed }

   FUNCTION Get_Buttons : Integer;
   {    Retrieve the integer representation of the Mouse Button pushed }

   PROCEDURE Set_Cursor_Page(Page_num: Byte);
   {     Set the Mouse Cursor to display on the proper page }

IMPLEMENTATION
TYPE
   Curs = Array[1..134] of byte;
   Cursptr = ^curs;
   curarray = ARRAY[0..31] OF word;
CONST
   {  Some sample cursor shapes for multi-page cursor }
   { "Conventional" Arrowhead cursor }
   Arrow_CURSOR_M1: Curs = (
       15,  0, 15,  0, 63,255, 63,255, 63,255, 63,255, 31,255, 31,255, 31,255,
       31,255, 15,255, 15,255, 15,255, 15,255,  7,255,  7,255,  7,255,  7,255,
        3,255,  3,255,  3,255,  3,255,  1,255,  1,255,  1,255,  1,255,  0,255,
        0,255,  0,255,  0,255,  0,127,  0,127,  0,127,  0,127,  0, 63,  0, 63,
        0, 63,  0, 63,  0, 31,  0, 31,  0, 31,  0, 31,  1,255,  1,255,  1,255,
        1,255, 16,255, 16,255, 16,255, 16,255, 48,255, 48,255, 48,255, 48,255,
      248,127,248,127,248,127,248,127,248,127,248,127,248,127,248,127,252, 63,
      252, 63,252, 63,252, 63,  0,  0);
   Arrow_cursor_M2: Curs = (
       15,  0, 15,  0,  0,  0,  0,  0,  0,  0,  0,  0, 64,  0, 64,  0, 64,  0,
       64,  0, 96,  0, 96,  0, 96,  0, 96,  0,112,  0,112,  0,112,  0,112,  0,
      120,  0,120,  0,120,  0,120,  0,124,  0,124,  0,124,  0,124,  0,126,  0,
```

```
    126,  0,126,  0,126,  0,127,  0,127,  0,127,  0,127,  0,127,128,127,128,
    127,128,127,128,127,192,127,192,127,192,127,192,124,  0,124,  0,124,  0,
    124,  0, 70,  0, 70,  0, 70,  0, 70,  0,  6,  0,  6,  0,  6,  0,  6,  0,
      3,  0,  3,  0,  3,  0,  3,  0,  3,  0,  3,  0,  3,  0,  3,  0,  1,128,
      1,128,  1,128,  1,128,  0,  0);

  { pointing finger cursor }
  Finger_CURSOR_M1: Curs = (
     15,  0, 15,  0,243,255,243,255,243,255,243,255,225,255,225,255,225,255,
    225,255,225,255,225,255,225,255,225,255,225,255,225,255,225,255,225,255,
    225,255,225,255,225,255,225,255,224, 15,224, 15,224, 15,224, 15,224,  1,
    224,  1,224,  1,224,  1,224,  0,224,  0,224,  0,224,  0,128,  0,128,  0,
    128,  0,128,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  1,  0,  1,  0,  1,
    128,  1,128,  1,128,  1,128,  1,192,  3,192,  3,192,  3,192,  3,224,  7,
    224,  7,224,  7,224,  7,  0,  0);
  Finger_CURSOR_M2: Curs = (
     15,  0, 15,  0, 12,  0, 12,  0, 12,  0, 12,  0, 18,  0, 18,  0, 18,  0,
     18,  0, 18,  0, 18,  0, 18,  0, 18,  0, 18,  0, 18,  0, 18,  0, 18,  0,
     18,  0, 18,  0, 18,  0, 18,  0, 19,240, 19,240, 19,240, 19,240, 18, 78,
     18, 78, 18, 78, 18, 78, 18, 73, 18, 73, 18, 73, 18, 73,114,  9,114,  9,
    114,  9,114,  9,144,  1,144,  1,144,  1,144,  1,144,  1,144,  1,144,  1,
    144,  1,128,  1,128,  1,128,  1,128,  1,128,  2,128,  2,128,  2,128,  2,
     64,  2, 64,  2, 64,  2, 64,  2, 32,  4, 32,  4, 32,  4, 32,  4, 31,248,
     31,248, 31,248, 31,248,  0,  0);

  Wait_CURSOR_M1: Curs = (
     15,  0, 15,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
      0,  0,128,  1,128,  1,128,  1,128,  1,128,  1,128,  1,128,  1,128,  1,
    192,  3,192,  3,192,  3,192,  3,224,  7,224,  7,224,  7,224,  7,240, 15,
    240, 15,240, 15,240, 15,248, 31,248, 31,248, 31,248, 31,240, 15,240, 15,
    240, 15,240, 15,224,  7,224,  7,224,  7,224,  7,192,  3,192,  3,192,  3,
    192,  3,128,  1,128,  1,128,  1,128,  1,128,  1,128,  1,128,  1,128,  1,
      0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,255,255,
    255,255,255,255,255,255, 20,167);

  Wait_CURSOR_M2: Curs = (
     15,  0, 15,  0,  0,  0,  0,  0,  0,  0,  0,  0,127,254,127,254,127,254,
    127,254, 32,  4, 32,  4, 32,  4, 32,  4, 32,  4, 32,  4, 32,  4, 32,  4,
     19,200, 19,200, 19,200, 19,200, 15,240, 15,240, 15,240, 15,240,  7,224,
      7,224,  7,224,  7,224,  2, 64,  2, 64,  2, 64,  2, 64,  4, 32,  4, 32,
      4, 32,  4, 32,  9,144,  9,144,  9,144,  9,144, 19,200, 19,200, 19,200,
     19,200, 39,228, 39,228, 39,228, 39,228, 47,244, 47,244, 47,244, 47,244,
    127,254,127,254,127,254,127,254,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
      0,  0,  0,  0,  0,  0,  0,  0,206, 20);

    { Cursor forms for the mouse driver }
    { traditional arrowhead cursor }
    Arrow_CURSOR : cursarray = (
```

File Name: CURSRORJ.PAS

```
    $3FFF, $1FFF, $0FFF, $07FF, $03FF, $01FF, $00FF, $007F, $003F,
    $001F, $01FF, $10FF, $30FF, $F87F, $F87F, $FC3F,
    $0000, $4000, $6000, $7000, $7800, $7C00, $7E00, $7F00, $7F80,
    $7FC0, $7C00, $4600, $0600, $0300, $0300, $0180);

    { pointing finger cursor }
    finger_CURSOR : curarray = (
    $F3FF, $E1FF, $E1FF, $E1FF, $E1FF, $E00F, $E001, $E000, $8000,
    $0000, $0000, $0000, $0001, $8001, $C003, $E007,
    $0C00, $1200, $1200, $1200, $1200, $13F0, $124E, $1249, $7209,
    $9001, $9001, $8001, $8002, $4002, $2004, $1FF8);

    { Waiting cursor }
    Wait_CURSOR: curarray = (
    $0000, $0000, $8001, $8001, $C003, $E007, $F00F, $F81F, $F00F,
    $E007, $C003, $8001, $8001, $0000, $0000, $FFFF,
    $0000, $7FFE, $2004, $2004, $13C8, $0FF0, $07E0, $0240, $0420,
    $0990, $13C8, $27E4, $2FF4, $7FFE, $0000, $0000);

VAR
   _Saved_pixels : ^byte;              { mouse stuff }
   _Mouse_Enabled : Boolean;
   _X_Point, _Y_Point : Integer;
   _X_Save, _Y_Save: Integer;
   Cursor_m1, Cursor_m2: cursptr;
   Cursor_showflag: Integer;
   _Mouse_Bttn_Count: Byte;
   _Insert_Key_flag: Boolean;
   mstat, M_B_P: Integer;


PROCEDURE Beep;
{ ************************************************************************
  *                                                                      *
  *        PURPOSE                                                       *
  *           Generate One 1/10 Second Beep                              *
  *                                                                      *
  ************************************************************************}
BEGIN
   Sound(880);                        {make sound}
   Delay(100);                        {delay 1/10 second}
   NoSound;                           {stop sound}
END;

PROCEDURE Click;
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
```

File Name: CURSROBJ.PAS

```
      *          Make a clicking sound for use when a mouse button is pushed*
      *                                                                      *
      ************************************************************************
      }
   { Make a clicking sound  }
   VAR
      i : Integer;
   BEGIN
      FOR i := 1 TO 250 DO BEGIN
         Sound(2000+i);
      END;
      NoSound
   END;


   {library of MicroSoft mouse interface routines}

   PROCEDURE mshape(xhot, yhot : Integer; VAR cursor : curarray);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Set the mouse cursor shape.                                *
      *                                                                      *
      ************************************************************************
      }
         { xhot and yhot pixel designations for the cursor hot spot. cursor is
             the array
         containing the cursor shape.  curarray is array type defined as
         ARRAY[0..31] OF INTEGER }

      TYPE
         regset = RECORD
                     ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
                  END;
      VAR regpack : Registers;

      BEGIN
         WITH regpack DO
            BEGIN
               ax := 9;
               bx := xhot;
               cx := yhot;
               dx := Ofs(cursor[0]);
               es := Seg(cursor[0])
            END;
         Intr($33, regpack)
      END;
```

File Name: CURSROBJ.PAS

```
PROCEDURE mstatus(VAR mstat, nbuttons : Integer);
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           returns mouse status and resets mouse parameters.             *
   *           mstat=0 if mouse is not installed. -1 otherwise.              *
   *           nbuttons is number of buttons enabled, =2 for this program    *
   *                                                                         *
   *                                                                         *
   ***************************************************************************
   }

   TYPE
      regset = RECORD
                  ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
               END;
   VAR recpack : Registers;

   BEGIN
      recpack.ax := 0;
      Intr($33, recpack);
      WITH recpack DO
         BEGIN
            mstat := ax;
            nbuttons := bx
         END
   END;

PROCEDURE mshcw;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           show the mouse cursor at the current position                 *
   *                                                                         *
   ***************************************************************************
   }
      { show mouse cursor }

   TYPE
      regset = RECCRD
                  ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
               END;
   VAR recpack : Registers;

   BEGIN
      recpack.ax := 1;
      Intr($33, recpack)
```

```
    END;

PROCEDURE mhide;
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *           remove the mouse cursor from the screen                            *
    *                                                                              *
    ********************************************************************************
    }
       { hide mouse cursor }

    TYPE
       regset = RECORD
                   ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
                END;
    VAR recpack : Registers;

    BEGIN
       recpack.ax := 2;
       Intr($33, recpack)
    END;

PROCEDURE mpos(VAR mbt, mx, my : Integer);
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *           returns position of mouse, coordinates mx, my.                     *
    *              if mbt=1, left button was pressed                               *
    *              if mbt=2, right button was pressed                              *
    *              if mbt=3, both buttons were pressed                             *
    *                                                                              *
    ********************************************************************************
    }

    TYPE
       regset = RECORD
                   ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
                END;
    VAR recpack : Registers;

    BEGIN
       recpack.ax := 3;
       Intr($33, recpack);
       WITH recpack DO
          BEGIN
             mbt := bx;
```

```
            mx := cx;
            my := dx
         END
   END;

PROCEDURE mput(mx, my : Integer);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           put the mouse cursor at coordinates mx,my.                 *
   *           legal values must be suppled                               *
   *                                                                      *
   ************************************************************************
   }

   TYPE
      regset = RECORD
                  ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
               END;
   VAR recpack : Registers;

   BEGIN
      recpack.ax := 4;
      recpack.cx := mx;
      recpack.dx := my;
      Intr($33, recpack)
   END;

PROCEDURE mvlimit(minpos, maxpos : Integer);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           set min and max vertical limits for cursor positions       *
   *                                                                      *
   ************************************************************************
   }

   TYPE
      regset = RECORD
                  ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
               END;
   VAR recpack : Registers;

   BEGIN
      recpack.ax := 8;
      recpack.cx := minpos;
      recpack.dx := maxpos;
```

```
      Intr($33, recpack);
   END;

PROCEDURE mhlimit(minpos, maxpos : Integer);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *            set min and max vertical limits for cursor positions      *
   *                                                                      *
   ************************************************************************
   }

   TYPE
      regset = RECORD
                  ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
               END;
   VAR recpack : Registers;

   BEGIN
      recpack.ax := 7;
      recpack.cx := minpos;
      recpack.dx := maxpos;
      Intr($33, recpack)
   END;

PROCEDURE mtext(select, mask1, mask2 : Integer);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           select the text cursor.  select=0 gives software cursor,   *
   *            select=1 gives hardware cursor.                           *
   *            For select=0 specify screen mask and cursor mask in mask1,*
   *            mask2.  For select=1, specify mask1=0 and                 *
   *            mas2=no. scan lines in cursor                             *
   *                                                                      *
   ************************************************************************
   }

   TYPE
      regset = RECORD
                  ax, bx, cx, dx, bp, di, si, ds, es, flags : Integer
               END;
   VAR recpack : Registers;

   BEGIN
      recpack.ax := 10;
      recpack.cx := mask1;
```

```
        recpack.dx := mask2;
        Intr($33, recpack)
    END;

PROCEDURE Mouse_Intr(call_routine: pointer; mask: word);
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *    set up a user defined routine to be called whenever certain          *
    *    events occur                                                         *
    *    Mask defined which events cause interrupts:                          *
    *    Bit 0 - movement (position change)                                   *
    *        1 - left button pressed                                          *
    *        2 - left button released                                         *
    *        3 - right button pressed                                         *
    *        4 - right button released                                        *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }

VAR
   recpack : registers;

BEGIN
   recpack.ax := 12;
   recpack.cx :=  mask;
   recpack.dx := OFS(Call_routine^);
   recpack.es := SEG(call_routine^);

   Intr($33, recpack);
END;

PROCEDURE msetCRTpage(page:byte);
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *          Set the mouse CRT page (page to display on)                    *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }
VAR
   recpack : registers;

BEGIN
   recpack.ax := 29;
```

```
   recpack.bx := page;

   Intr($33, recpack);
END;

PROCEDURE mgetCRTpage(VAR page:byte);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Get the current mouse CRT page                                *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
VAR
   recpack : registers;

BEGIN
   recpack.ax := 30;

   Intr($33, recpack);

   recpack.bx := page;
END;



PROCEDURE Get_Key(VAR KEY_PRESSED : Integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Retrieve the integer value of the keyboard key pressed.       *
      *                                                                         *
      *        INPUTS                                                           *
      *           *None*                                                        *
      *                                                                         *
      *        OUTPUTS                                                          *
      *           Key_Pressed                                                   *
      *                                                                         *
      *        GLOBALS                                                          *
      *           _Demo_Flag                                                    *
      *                                                                         *
      *        SUBPROGRAMS CALLED                                               *
      *           Demo_Key                                                      *
      *                                                                         *
      *        REMARKS                                                          *
      *           *None*                                                        *
```

File Name: CURSROBJ.PAS

```
      *                                                                      *
      ************************************************************************
      }

   VAR
      ASCII_CODE : Char;

   BEGIN
(*       IF (_Demo_Flag) THEN
          Key_Pressed := Demo_key
      ELSE *)
         IF (KeyPressed)
         THEN BEGIN
            ASCII_CODE := ReadKey;
            IF ord(ASCII_CODE) = 0
            THEN KEY_PRESSED := ord(ReadKey)+127  { CMEGA ACCESS INTERNAL CODE }
            ELSE
               KEY_PRESSED := ord(ASCII_CODE);
         END
         ELSE
            KEY_PRESSED := 0;
   END;


   PROCEDURE Set_Cursor_Shape(ckind: cursor_kinds);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           set the kind of cursor shape to display for the mouse      *
      *           pointer                                                    *
      *                                                                      *
      ************************************************************************
      }
   BEGIN
      CASE ckind OF
         arrow: BEGIN
            Cursor_m1 := @Arrow_Cursor_M1;
            Cursor_m2 := @Arrow_Cursor_M2;
            mshape( 0,0, Arrow_Cursor);
         END;
         finger: BEGIN
            Cursor_m1 := @finger_Cursor_M1;
            Cursor_m2 := @Finger_Cursor_M2;
            mshape( 0,0, Finger_Cursor);
         END;
         wait: BEGIN
            Cursor_m1 := @WAIT_Cursor_M1;
            Cursor_m2 := @WAIT_Cursor_M2;
```

```
            mshape(7,0, WAIT_Cursor);
         END;
      END;  {CASE}
   END;

PROCEDURE GET_CURSORXY(VAR X_Coord : Integer; VAR Y_Coord : Integer);
      {
      ********************************************************************************
      *  PURPOSE                                                                     *
      *    Get the current cursor coordinates                                        *
      *                                                                              *
      *  INPUTS                                                                      *
      *                                                                              *
      *  OUTPUTS                                                                     *
      * For non_mouse operations, return the values of the global variables          *
      * _X_Point and _Y_Point                                                        *
      *                                                                              *
      * GLOBALS                                                                      *
      *  _X_Point                                                                    *
      *  _Y_Point                                                                    *
      *  _Mouse_Enabled                                                              *
      *                                                                              *
      * SUBPROGRAMS CALLED                                                           *
      *  mpos                                                                        *
      ********************************************************************************
      }

   VAR
      X_Mouse_Coord, Y_Mouse_Coord, M_B_P : Integer;

   BEGIN
      IF (_Mouse_Enabled) THEN BEGIN
         MPOS(M_B_P, X_Mouse_Coord, Y_Mouse_Coord);
         X_Coord := X_Mouse_Coord;
         Y_Coord := Y_Mouse_Coord;
         _X_Point := X_Coord;
         _Y_Point := Y_Coord;
      END
      ELSE BEGIN
         X_Coord := _X_Point;
         Y_Coord := _Y_Point;
      END;                                {IF}
   END;

PROCEDURE Hide_Cursor;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
```

```
    *                  Hide the cursor - remove it from the display        *
    *          For non-mouse operation, this procedure restores the pixels *
    *           that were saved in the global buffer _Saved_Pixels at the  *
    *           last location  which was saved in _X_Point, _Y_Point.      *
    *                                                                      *
    *   *** Modified for multi-page operation ***                          *
    *                                                                      *
    *        INPUTS                                                        *
    *           *None*                                                     *
    *                                                                      *
    *        OUTPUTS                                                       *
    *           *None*                                                     *
    *                                                                      *
    *        GLOBALS                                                       *
    *           _X_point,_Y_point                                          *
    *                                                                      *
    *        CALLING ROUTINE                                               *
    *           Routines which hide the cursor                             *
    *                                                                      *
    *        SUBPROGRAMS CALLED                                            *
    *           mhide                                                      *
    *           putpic                                                     *
    *        REMARKS                                                       *
    *           *None*                                                     *
    *                                                                      *
    ************************************************************************
    }
  VAR
    saved : ViewPortType;

  BEGIN
    IF (_Mouse_Enabled)
    THEN BEGIN
      mhide
    END
    ELSE BEGIN
      Dec(Cursor_Showflag);
      IF Cursor_Showflag = -1 THEN BEGIN
        GetViewSettings(saved);
        SetViewPort(0, 0, 639, 349, clipoff);
        IF _Active_Page <> _Display_Page THEN SetActivePage(_Display_Page);
        PutImage(_X_Save, _Y_Save, _Saved_Pixels^, Normalput);
        IF _Active_Page <> _Display_Page THEN SetActivePage(_Active_Page);
        WITH saved DO SetViewPort(x1, y1, x2, y2, clip);
      END;
    END;
  END;
```

File Name: CURSROBJ.PAS

```
PROCEDURE Show_Cursor(X_Coord : Integer; Y_Coord : Integer);
     {
     ********************************************************************************
     *                                                                              *
     * PURPOSE                                                                      *
     *  Show the cursor at the indicated coordinates of the cursor "hot spot"*
     *  UPDATED FOR Turbo 4.0                                                       *
     *                                                                              *
     *  *** Modified for multi-page display ***                                     *
     *                                                                              *
     * INPUTS                                                                       *
     *   Current X and Y coordinates of cursor.                                     *
     *                                                                              *
     * OUTPUTS                                                                      *
     *   Cursor displayed at current X and Y coordinates of cursor.                 *
     *                                                                              *
     * GLOBALS                                                                      *
     *  For non-mouse operation the following global variables are set:             *
     *                                                                              *
     * _X_Point, _Y_Point   coordinates of current cursor hot spot                  *
     * _Saved_Pixels        copy of pixels which were under the cursor              *
     * (used by HIDE_CURSOR to restore screen area)                                 *
     *                                                                              *
     *                                                                              *
     * SUBPROGRAMS CALLED                                                           *
     *  GETIMAGE                                                                    *
     *  PUTIMAGE                                                                    *
     *  mput                                                                        *
     *  mshow                                                                       *
     ********************************************************************************
     }


   VAR
      saved : viewporttype;


   BEGIN
      _X_Point := X_Coord;              { set the global variables }
      _Y_Point := Y_Coord;
      { limit the excursion of the cursor if necessary }

      IF _X_Point > X_MAX_LIMIT
      THEN _X_Point := X_MAX_LIMIT;
      IF _X_Point < X_MIN_LIMIT
      THEN _X_Point := X_MIN_LIMIT;
      IF _Y_Point > Y_MAX_LIMIT
      THEN _Y_Point := Y_MAX_LIMIT;
```

```
      IF _Y_Point < Y_MIN_LIMIT
      THEN _Y_Point := Y_MIN_LIMIT;
      IF (_Mouse_Enabled)
      THEN BEGIN
         mput(_X_Point, _Y_Point);
         mshow;
      END
      ELSE BEGIN
         Inc(Cursor_Showflag);
         IF Cursor_Showflag = 0 THEN BEGIN
            { copy pixels in area at new X,Y to saved pixel buffer }
            GetViewSettings(saved);
            SetViewPort(0, 0, 639, 349, clipoff);

            IF _Active_Page <> _Display_Page THEN SetActivePage(_Display_Page);
            GetImage(_X_Point, _Y_Point, (_X_Point+15),
            (_Y_Point+15), _Saved_Pixels^);
            _X_Save := _X_Point;
            _Y_Save := _Y_Point;

            { put the two cursor masks over the screen }
            PutImage(_X_Point, _Y_Point, CURSOR_M1^, Andput);
            PutImage(_X_Point, _Y_Point, CURSOR_M2^, ORput);
            WITH saved DO SetViewPort(x1, y1, x2, y2, clip);
            IF _Active_Page <> _Display_Page THEN SetActivePage(_Active_Page);
         END;
      END;                                    {IF}
   END;


PROCEDURE Move_Cursor(_Key_Pressed: Integer; VAR Old_X_Coord, Old_Y_Coord : Integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Move the cursor according to a the cursor movement key        *
      *           pressed. Arrow keys move the cursor by 8 pixels, ctrl         *
      *           arrow keys move the cursor by one pixel.                      *
      *                                                                         *
      *                                                                         *
      *        INPUTS                                                           *
      *           *None*                                                        *
      *                                                                         *
      *        OUTPUTS                                                          *
      *           *None*                                                        *
      *                                                                         *
      *        GLOBALS                                                          *
      *           _Key_Pressed                                                  *
      *           HEAD_OBJECT                                                   *
```

```
      *          LAST_OBJECT                                                    *
      *          FWD_NEXT_OBJECT                                                *
      *          BKWD_NEXT_OBJECT                                               *
      *          UP_ARROW                                                       *
      *          RIGHT_ARROW                                                    *
      *          LEFT_ARROW                                                     *
      *          DOWN_ARROW                                                     *
      *                                                                         *
      *       CALLING ROUTINE                                                   *
      *          OMGMAIN                                                        *
      *                                                                         *
      *       SUBPROGRAMS CALLED                                                *
      *          CURSOR_TO_OBJECT                                               *
      *          BEEP                                                           *
      *          Hide_Cursor                                                    *
      *          Show_Cursor                                                    *
      *                                                                         *
      *       REMARKS                                                           *
      *          *None*                                                         *
      *                                                                         *
      ***************************************************************************
      }
   VAR
      X_Coord, Y_Coord : Integer;

   BEGIN
      X_Coord := Old_X_Coord;
      Y_Coord := Old_Y_Coord;
      CASE _Key_Pressed OF
         UP_ARROW : BEGIN                    (UP ARROW)
                       IF (_Insert_Key_Flag) THEN
                          BEGIN
                             Y_Coord := Old_Y_Coord-1;
                             X_Coord := Old_X_Coord
                          END
                       ELSE
                          BEGIN
                             Y_Coord := Old_Y_Coord-12;
                             X_Coord := Old_X_Coord;
                          END
                    END;
         RIGHT_ARROW : BEGIN                 (RIGHT ARROW)
                       IF (_Insert_Key_Flag) THEN
                          BEGIN
                             X_Coord := Old_X_Coord+1;
                             Y_Coord := Old_Y_Coord
                          END
                       ELSE
                          BEGIN
```

```
                              X_Coord := Old_X_Coord+12;
                              Y_Coord := Old_Y_Coord;
                           END
                     END;
        LEFT_ARROW : BEGIN                (LEFT ARROW)
                        IF (_Insert_Key_Flag) THEN
                           BEGIN
                              X_Coord := Old_X_Coord-1;
                              Y_Coord := Old_Y_Coord
                           END
                        ELSE
                           BEGIN
                              X_Coord := Old_X_Coord-12;
                              Y_Coord := Old_Y_Coord
                           END
                     END;
        DOWN_ARROW : BEGIN                (DOWN ARROW)
                        IF (_Insert_Key_Flag) THEN
                           BEGIN
                              Y_Coord := Old_Y_Coord+1;
                              X_Coord := Old_X_Coord
                           END
                        ELSE
                           BEGIN
                              Y_Coord := Old_Y_Coord+12;
                              X_Coord := Old_X_Coord
                           END
                     END;
        INSERT_KEY: _Insert_Key_Flag := NOT(_Insert_Key_Flag);
      ELSE BEGIN
         BEEP;
      END;
      END;                                (CASE)

      Hide_Cursor;

      Show_Cursor(X_Coord, Y_Coord);

      Old_X_Coord := X_Coord;
      Old_Y_Coord := Y_Coord;

   END;


FUNCTION Left_Button_Down: Boolean;

      {
      ***************************************************************
            Purpose
                  Reports whether the left mouse button is
```

```
                    depressed. Returns TRUE if it is down;
                    FALSE otherwise
      ***************************************************************
      }
VAR
      Initial_Mouse_Button_Pushed : Integer;
      X_Mouse_Coord, Y_Mouse_Coord: Integer;
BEGIN
      IF (_MOUSE_ENABLED) THEN BEGIN
         Initial_Mouse_Button_Pushed := 0;
         MPOS(Initial_Mouse_Button_Pushed, X_Mouse_Coord, Y_Mouse_Coord);
         IF (Initial_Mouse_Button_Pushed = Left_Button_Pressed) THEN
            Left_Button_Down := TRUE
         ELSE
            Left_Button_Down := FALSE;
         END
      ELSE
         Left_Button_Down := FALSE;
END;


FUNCTION Right_Button_Down: Boolean;
      {
      ***************************************************************
            Purpose
                  Reports whether the Right mouse button is
                  depressed. Returns TRUE if it is down,
                  FALSE otherwise
      ***************************************************************
      }
VAR
      Initial_Mouse_Button_Pushed : Integer;
      X_Mouse_Coord, Y_Mouse_Coord: Integer;
BEGIN
      IF (_MOUSE_ENABLED) THEN BEGIN
         Initial_Mouse_Button_Pushed := 0;
         MPOS(Initial_Mouse_Button_Pushed, X_Mouse_Coord, Y_Mouse_Coord);
         IF (Initial_Mouse_Button_Pushed = Right_Button_Pressed) THEN
            Right_Button_Down := TRUE
         ELSE
            Right_Button_Down := FALSE;
         END
      ELSE
         Right_Button_Down := FALSE;
END;


FUNCTION Get_Buttons : Integer;
      {
```

File Name: CURSROBJ.PAS

```
{*****************************************************************************
*                                                                            *
*        PURPOSE                                                             *
*           Retrieve the integer representation of the Mouse                 *
*           Button Pushed while "debouncing" the mouse buttons.              *
*           A constantly held button has no meaning, making the              *
*           buttton respond when it is released.                             *
*                                                                            *
*        INPUTS                                                              *
*           Initial_Mouse_Button_Pushed,Mouse_Button_Pushed                  *
*                                                                            *
*        OUTPUTS                                                             *
*           Get_Buttons                                                      *
*                                                                            *
*        GLOBALS                                                             *
*           MOUSE_ENABLED                                                    *
*           CLICK_ON_BUTTON                                                  *
*                                                                            *
*        CALLING ROUTINE                                                     *
*           Utility                                                          *
*                                                                            *
*        SUBPROGRAMS CALLED                                                  *
*           MPOS                                                             *
*                                                                            *
*        REMARKS                                                             *
*           *None*                                                           *
*                                                                            *
*****************************************************************************}

VAR
   Initial_Mouse_Button_Pushed, Mouse_Button_Pushed : Integer;
   X_Mouse_Coord, Y_Mouse_Coord, i : Integer;

BEGIN

   IF (_MOUSE_ENABLED) THEN BEGIN
      Initial_Mouse_Button_Pushed := 0;
      MPOS(Initial_Mouse_Button_Pushed, X_Mouse_Coord, Y_Mouse_Coord);
      IF (Initial_Mouse_Button_Pushed > 0) THEN BEGIN (*MOUSE BUTTON PUSHED*)
         CASE _Mouse_Bttn_Count OF
            0: _Mouse_Bttn_Count := 1;
            1: BEGIN
                  Inc(_Mouse_Bttn_Count);
                  i := 1;
                  WHILE (i < 10) AND (Initial_Mouse_Button_Pushed > 0) DO BEGIN
                     Inc(i);
                     Delay(50);
                     MPOS(Initial_Mouse_Button_Pushed, X_Mouse_Coord, Y_Mouse_Coord
```

```
                    END;
                  END;
               2: Delay(50);
            END;  {CASE}
            MPOS(Initial_Mouse_Button_Pushed, X_Mouse_Coord, Y_Mouse_Coord);
         END
         ELSE _Mouse_bttn_Count := 0;
         Get_Buttons := Initial_Mouse_Button_Pushed;

         IF CLICK_ON_BUTTON AND (Initial_Mouse_Button_Pushed <> 0) THEN
            Click;

      END                                  {IF THEN}
      ELSE
         Get_Buttons := 0;

   END;


   PROCEDURE Set_Cursor_Page(Page_num:Byte);
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           Set the Mouse Cursor to display on the proper page             *
      *                                                                          *
      *                                                                          *
      ****************************************************************************
      }
   BEGIN
      IF _Mouse_Enabled THEN
         msetCRTpage(Page_num);
   END;


{ Unit Initialization stuff }
BEGIN
   _Active_Page := 0;
   _Display_Page := 0;
   Set_Cursor_Page(_Display_page);

   _Cursor_set := [FWD_NEXT_OBJECT, BKWD_NEXT_OBJECT, HEAD_OBJECT, UP_ARROW,
   LEFT_ARROW, RIGHT_ARROW, LAST_OBJECT, DOWN_ARROW, INSERT_KEY,
   CTRL_LEFT_ARROW..CTRL_HOME_ARROW];

   Cursor_showflag := -1;
   _Insert_Key_Flag := FALSE;
   Set_Cursor_Shape(Arrow);
   GetMem(_Saved_Pixels,SizeOf(Curs) );
```

```
   { check on the availability of the mouse and initialize it for our us }
   mstatus(mstat, M_B_P);              {initialize mouse driver}
   IF (mstat = -1)
      THEN BEGIN
         _Mouse_Enabled := True;
         mvlimit(Y_MIN_LIMIT, (Y_MAX_LIMIT-2));
         mhlimit(X_MIN_LIMIT, (X_MAX_LIMIT-2));
      END
      ELSE _Mouse_Enabled := False;

    { Object initialization }
END.
```

File Name: DATAUTIL.PAS

```
Unit DATAUTIL;
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                            (PACE)                                      *
*                          Workstation                                   *
*                                                                        *
*          UNIT NAME - DATAUTIL                                          *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*          This unit contains the cell format and archive file           *
*          format declarations as well as a collection of routines       *
*          that are used to read data from the cell format signal        *
*          database as well as to read and write the archive files.      *
**************************************************************************}
Interface
Uses PaceInit;
{*************************************************************************
*                                                                        *
*          PACE cell database and Archive File Data Definitions and routines *
*                                                                        *
*       ver 0     5/29/90    KATench                                     *
*       ver 1     6/7/90     GRDesrochers  'Unitized' the routines       *
**************************************************************************}

TYPE
   { cell coverage information for 1 sta/month/hr}
   CovCell = RECORD
      SNR_S: Integer;   { Short-path SNR (db*8) w/ Sigma_S in uppermost nybble (db*8) }
      SNR_L: Integer;   { Long-path SNR (db*8) w/ nothing in uppermost nybble }
```

```
      Phase: Byte;       { Phase Deviation with mode 1 flag in top bit (0=Mode 1) }
      X_Ang: Byte;       { Crossing Angle }
   END;

   { record for disk I/O - 4 months, 24 hours of coverage information }
   { Disk file is organized as 8 stations worth of coviorec for each of 444 cells }
   coviorec_4m = array[1..4,1..24] of CovCell;
   coviorec_12m = array[1..12,1..24] of CovCell;  { 12 month version }

   StationCoverageinfoPtr = ^StationCoverageInfo;
   StationCoverageInfo = ARRAY[1..8,1..12,1..24] OF CovCell;

   DBFile = File of StationCoverageInfo;
   TimeSpecificDBFile = File of CovCell;

   { Noise data information for 1 month/hr }
   NoiseCell = RECORD
      Sigma_N: Byte;     { Noise standard deviation (db*8) }
      Noise: Byte;       { Noise value (db*8) }
   END;

   { record for disk I/O - 12 months, 24 hours of noise information }
   { Disk file is organized as 444 cells worth of noiseiorec }
   noiseiorec = array[1..12,1..24] of CovCell;

   CellRecord = record
                Pat: single;
                Coverage: byte;
   end;

   ArchivedCellType = Array[1..12,1..24] of CellRecord;
   TimeSpecificArchiveFile = File of cellrecord;
   ArchiveFile = File OF ArchivedCellType;

{ Routines to access the archived file info }
   procedure StoreCellToArchive(VAR AF: Archivefile; cellnumber: integer;
                                CellInfo:ArchivedCellType);
   procedure LoadCellFromArchive(VAR AF: Archivefile; cellnumber: integer;
                                VAR CellInfo:ArchivedCellType);
   procedure ReadTimeSpecificFromArchive(VAR AF: TimeSpecificArchiveFile; cellnumber,
                                         month,hour: longint;
                                         VAR archiveinfo: cellrecord);


{ Routines to pack and unpack the fields of CovCell }
FUNCTION Pack_SNR(Sigma, SNR:Integer): Integer;
FUNCTION Get_SNR(SNR_in:Integer ): Integer;
FUNCTION Get_Sigma(SNR_in: Integer): Integer;
Procedure ReadFromDatabase(VAR DB:DBFile;cellnum:integer;
```

```
File Name: DATAUTIL.PAS

                                    VAR covparams:stationcoverageinfoptr);
procedure ReadTimeSpecificFromDatabase(VAR DB:TimeSpecificDBFile;
                                       CellNum,station,month,hour:longint;
                                       VAR TimeSpecificInfo:CovCell);

Function ReadGdop(cellnum,coverage:longint): byte;

Implementation

FUNCTION Pack_SNR(Sigma, SNR:Integer): Integer;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Pack the sigma value into top nibble of the SNR word in the*
      *           cell format signal database. The sigma value is actually   *
      *           half of the mode 1 dominance margin byte value             *
      *           as documented in the PACE final report. The variable called*
      *           sigma is shifted left by 12 bits and ORed into the SNR     *
      *           value.                                                     *
      ********************************************************************************
      }
BEGIN
   Pack_SNR := ((Sigma-12) SHL 12) OR ( SNR AND $FFF);  (12 = 1.5*8 db)
END;

(SR-)
FUNCTION Get_SNR(SNR_in:Integer ): Integer;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Remove the lower three nibbles of information from a word  *
      *           length variable with a quantity packed into the upper      *
      *           nibble (the upper four bits). Sign extend by filling in    *
      *           the upper nibble with ones if the upper bit of the third   *
      *           nibble (bit # 12 is set).                                  *
      ********************************************************************************
      }
{ remove the top nybble and sign extend to make a full integer }
BEGIN
   SNR_in := SNR_in AND $FFF;
   IF (SNR_in AND $800) <> 0 THEN      { Need to extend sign bit ? }
      Get_SNR := SNR_in OR $F000
   ELSE
      Get_SNR := SNR_in;
END;

FUNCTION Get_Sigma(SNR_in: Integer): Integer;
```

File Name: DATAUTIL.PAS

```
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Remove the value stored in the upper nibble of a word value*
      *           by masking off the lower 12 bits and then shifting the     *
      *           result right by 12.                                        *
      ********************************************************************************
      }
( Extract sigma from top nybble )
BEGIN
   Get_Sigma := ((SNR_in AND $F000) SHR 12) + 12;  ( as in 1.5*8 = 12 )
END;
{$R+}

   Procedure ReadFromDatabase(VAR DB:DBFile;cellnum:integer;
                              VAR covparams:stationcoverageinfoptr);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           This procedure reads an entire cell's worth (over station, *
      *           month and hour) of signal information from the cell        *
      *           format signal database.                                    *
      ********************************************************************************
      }
   begin
      ( use some global unit variables of type file to hold the
        database file handles. Just access them here for effeciency
        sake, open and close them in compute_Psa.
        THIS PROCEDURE ASSUMES THAT THE DATABASE FILE IS ALREADY
        OPENED AS A FILE OF TYPE DBFILE !!!!!!!!!!!!!!!!! )


      seek(DB,cellnum-1);
      read(DB,covparams^);

   end;

procedure ReadTimeSpecificFromDatabase(VAR DB:TimeSpecificDBFile;
                                       CellNum,station,month,hour:longint;
                                       VAR TimeSpecificInfo:CovCell);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           This procedure reads a cell/station/month/time specific    *
      *           record of signal information from the cell                 *
      *           format signal database.                                    *
```

File Name: DATAUTIL.PAS

```
      ********************************************************************************
      }
VAR
  index: longint;

   begin
      { use some global unit variables of type file to hold the
        database file handles. Just access them here for effeciency
        sake, open and close them elsewhere.
        THIS PROCEDURE ASSUMES THAT THE DATABASE FILE IS ALREADY
        OPENED AS A FILE OF TYPE TIMESPECIFICDBFILE !!!!!!!!!!!!!!!!! }

      index := ((cellnum-1)*8*12*24) + ((station-1)*12*24) +
               ((month-1)*24) + (hour-1);
      seek(DB,index);
      read(DB,TimeSpecificInfo);

   end;

   procedure LoadCellFromArchive(VAR AF: Archivefile; cellnumber: integer;
                                 VAR CellInfo:ArchivedCellType);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *          Reads a cell's worth of archived scenario information from *
      *          an archive file.                                                    *
      ********************************************************************************
      }
   (****** NOTE: this procedure assumes that the archive file is already
                 opened!!!!!!!! ***********)
   VAR
      i,j:integer;
   begin
      { seek into the archive file past the comment and parameters
        area to the correct cell and month/hour location }
      { cell location 0 holds the comment and parameters area }
      Seek(AF,cellnumber);
      read(AF,CellInfo);
   end;

(********************************************************************************)

   procedure StoreCellToArchive(VAR AF: Archivefile; cellnumber: integer;
                                CellInfo:ArchivedCellType);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
```

```
      *          Writes a cell's worth of archived scenario information to  *
      *          an archive file.                                           *
      ***************************************************************************
      }
   begin
      { seek into the archive file past the comment and parameters
        area to the correct cell and month/hour location }
      { cell location 0 holds the comment and parameters area }
      Seek(AF,cellnumber);
      write(AF,CellInfo);
   end;

(***************************************************************************)

procedure ReadTimeSpecificFromArchive(VAR AF: TimeSpecificArchiveFile; cellnumber,
                                            month,hour: longint;
                                            VAR archiveinfo: cellrecord);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Reads a cell/month/time specific record of archived            *
      *          scenario information to from an archive file.                  *
      ***************************************************************************
      }
VAR
   index: longint;
begin
   index := ((cellnumber -1)*12*24) + ((month-1)*24) + (hour-1);
   seek(AF,index);
   read(AF,archiveinfo);
end;

(***************************************************************************)

Function ReadGdop(cellnum,coverage:longint): byte;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Reads the GDOP for the particular cell and coverage            *
      *          vector provided from the GDOP database file.                   *
      ***************************************************************************
      }
VAR
   index: longint;
   GDOPFile : file of byte;
   tempresult: byte;
begin
```

File Name: DATAUTIL.PAS

```
    index := ((cellnum-1)*256)+coverage;
    Assign(GDOPFile,PACEPATH+'\'+GDOPDATABASE);
    Reset(GDOPFile);
    { get to the appropriate cell/coverage byte of the file }
    seek(GDOPFile,index);
    read(GDOPFile,tempresult);
    close(GDOPFile);
    ReadGdop := tempresult;
end;

begin
end.
```

File Name: DEFS.PAS

```
(*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                              (PACE)                                    *
*                           Workstation                                  *
*                                                                        *
*           UNIT NAME - Part of the PACEOBJS Unit                        *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains a collection of the object definitions    *
*           whose implementation code is found in the various PACEOBJS   *
*           Unit files. Also included here are some of the global PACE   *
*           variables and constants that are used by the various         *
*           objects                                                      *
*************************************************************************)

(* include file for pacecbjs.pas    EMA  10/4/90 *)


const

(* the maximum list size, used in lmenu for the directory list *)
   maxlistsize = 10;

(* Global boolean flags which are used in displaying the *)
(* textstring for the EditButton *)
   ShowCursor    = TRUE;
   HideTheCursor = FALSE;

   MaxNumberMutualExclusiveButtons = 12;
```

File Name: DEFS.PAS

```
   MaxNumberPickButtons = 12;
   MaxNumMessageBoxLines = 10;
   FRQ102 = 1;
   FRQ136 = 2;
   FRQAND = 3;
   FRQOR  = 4;
   PSARAN = 1;
   PSADET = 2;
   DMOFF  = 1;
   DMON   = 2;
   SRMBEST = 1;
   SRMNOM  = 2;
   SRMWRST = 3;
   PSAMIN = 1;
   PSAMEAN = 2;
   PSAMAX = 3;

   MonthNames: ARRAY[1..12] OF shortstring =
               ('Jan','Feb','Mar','Apr','May','Jun',
               'Jul','Aug','Sep','Oct','Nov','Dec');
   MA : ARRAY[1..12] OF CHAR =
      ('J','F','M','A','M','J','J','A','S','O','N','D');
   HourNames: ARRAY[1..24] OF Shortstring =
              ('1','2','3','4','5','6','7','8','9','10',
              '11','12','13','14','15','16','17','18','19',
              '20','21','22','23','24');
   StationNames : array[1..8] OF shortstring =
      ('A','B','C','D','E','F','G','H');

   LEFTDEFAULTARCHIVE  = 'LEFTTEMP.@@@';
   RIGHTDEFAULTARCHIVE = 'RIGHTEMP.@@@';

type
(* for the Memo Area - 4 lines of text @ 50 characters per line *)
   MemoTextType = Array[1..4] of String[50];

   savedparamsrecordptr = ^savedparamsrecord;
   savedparamsrecord = record
      Psa,SNR,ShortLongRatio,XAngle,PhaseDev,GDOP: real;
      DMoff,DMon,SRMbest,SRMnom,SRMworst,Fr102,Fr136,FrAND,FrOr,
      {PsaDET,PsaRAN,}PsaMIN,PsaMEAN,PsaMAX:boolean;
      DMpicked,SRMpicked,Frpicked{,PsaCalpicked},PsaReppicked: integer;
      StationPower: ARRAY[1..8] OF real;
      StationPowerOn,StationPowerOff: ARRAY[1..8] OF Boolean;
      MonthSelectors: ARRAY[1..12] OF boolean;
      HourSelectors: ARRAY[1..24] OF Boolean;
      QRFile,WeightsFile: string;
   end;
```

```
File Name: DEFS.PAS

(********** NOTE: the following type definition for commentareatype MUST be
    consistent with the definition of the comment area in
    the QRDatabaseType record in QRUTIL.PAS. ***************)
   commentareatype = string;


    batchqueueentryptr = ^batchqueueentry;
    batchqueueentry = record ParameterStuff: SavedParamsRecord;
                             CommentArea:CommentAreaType;
                             filename: string;
                             next: batchqueueentryptr;
    end;
   Listptr = ^List;
   List = Record
            FName : string[12];
            Next  : Listptr;
          end;

   cellgridptr = ^cellgrid;
   CellPtr     = ^Cell;
   CellShowProcedure   = procedure(selfptr : CellPtr);
   CellGridCancelProc  = procedure(selfptr : CellPtr);
   ActionProcedure     = procedure;
   CellActionProcedure = procedure(selfptr : CellPtr; CellNumber: integer);
   PickMenuActionArray = array[1..MaxNumberPickButtons] of ActionProcedure;
   PickMenuNameArray = array[1..MaxNumberPickButtons] of shortstring;
   PickMenuHotkeyArray = array[1..MaxNumberPickButtons] of char;
   namearray = array[1..MaxNumberMutualExclusiveButtons] of shortstring;


   map = object(control)
      CenterX,
      CenterY         : integer;
      ZoomFactor      : real;
      MapBackGroundColor,
      MapOutlineColor : word;
      constructor Init(InitX, InitY, InitWidth, InitHeight : integer;
                       InitBackgroundColor, InitMapOutlineColor: word);
      procedure   Show; virtual;
      procedure   HiLite(Xpos,Ypos:word); virtual;
   end;

   { window and menu type declarations }
   menu = object(window) { menu type windows }
      procedure Cancel; virtual;
   end;


   linkedbutton = object(textbutton)
```

File Name: DEFS.PAS

```
      ActionProc:procedure;
      constructor Init(InitX,InitY,InitWidth,InitHeight,
                  InitColor,InitHilitecolor,
                  InitNameColor,InitHiliteNameColor:Integer;
                  InitName:shortstring;InitActionProc:ActionProcedure;InitHotKey:char);
      function action(XX,YY:word):boolean; virtual;
   end;

   linkedbutton3D = object(textbutton3d)
      ActionProc:procedure;
      constructor Init(InitX,InitY,InitWidth,InitHeight,
                  InitColor,InitHilitecolor,
                  InitNameColor,InitHiliteNameColor:Integer;
                  InitName:shortstring;InitActionProc:ActionProcedure;InitHotKey:char);
      function action(XX,YY:word):boolean; virtual;
   end;


   MutExclusiveN = object(Control)
      Picked:integer;
      ControlLabel: shortstring;
      NumberButtons:integer;
      Orientation : boolean;
      xoffset,yoffset : integer;
      ButtonArray: Array[1..MaxNumberMutualExclusiveButtons] OF StatusButton;
      constructor Init(InitX,InitY,InitWidth,InitHeight,
                  InitColor,InitHilitecolor,InitSelectedColor,
                  InitNameColor,InitHiliteNameColor,InitSelectedNameColor:Integer;
                  InitControlLabel:shortstring;InitNumberButtons,InitButtonSelected:integer;
                  ButtonNameList:namearray;InitOrientation:boolean);
      procedure HiLite(Xpos,Ypos:word); virtual;
      procedure Show; virtual;
      procedure ChangeXY(Xpos,Ypos:word); virtual;
      function Action(Xpos,Ypos:word):boolean; virtual;
   end;

   OnOffSlideControl = object(Control)
      On,Off:StatusButton;
      Sld:HorizontalSlideBar;
      constructor Init(InitX,InitY,InitColor,InitHiliteColor,InitSelectedColor,
                  InitControlColor,InitHiliteControlColor,InitSelectedControlColor:word;
                  InitLoLim,InitHiLim,InitInc:longint;
                  InitAccel:integer;InitName,InitUnits:shortstring);
      procedure HiLite(Xpos,Ypos:word); virtual;
      procedure Show; virtual;
      function Action(Xpos,Ypos:word):boolean; virtual;
   end;

   SelectButton = object(StatusButton)
```

```
      function Action(Xpos,Ypos:word):boolean; virtual;
   end;

   pickmenu = object(menu)
      buttonarray: array[1..MaxNumberPickButtons] of linkedbutton3D;
      NumberButtons: integer;
      constructor Init(Xpos,Ypos:word;InitNumberButtons:integer;
         InitNameList:PickMenuNameArray;InitActions:PickMenuActionArray;
         InitHotkeys:PickMenuHotkeyArray);
      procedure Show; virtual;
      procedure HiLite(Xpos,Ypos:word); virtual;
      function Action(Xpos,Ypos:word):boolean;virtual;
      function KeyAction(Xpos,Ypos:word;Key:integer):boolean; virtual;
   end;

   emenuptr = ^emenu;

   emenu = object(menu)
      { flags indicate what to do in process }
      recompute,rethreshold: boolean;
      QRFile,Weightsfile: string;
      Psa,SNR,ShortLongRatio,XAngle,PhaseDev,GDOP:HorizontalSlideBar;
      DM,{PsaCalculateMode,}StationReliabilityModel,
      PsaReportMode,Freq:MutExclusiveN;
      StationPower: ARRAY[1..8] OF OnOffSlideControl;
{      PwrA,PwrB,PwrC,PwrD,PwrE,PwrF,PwrG,PwrH:OnOffSlideControl; }
      MonthSelectors: ARRAY[1..12] OF SelectButton;
      HourSelectors: ARRAY[1..24] OF SelectButton;
      Reliability,Weights,Batch,Process:linkedbutton3d;
      constructor Init(Xpos,Ypos:word;Rlink,Wlink,Blink,Plink:ActionProcedure);
      procedure Show; virtual;
      procedure HiLite(Xpos,Ypos:word); virtual;
      function Action(Xpos,Ypos:word):boolean;virtual;
      function KeyAction(Xpos,Ypos:word;Key:integer):boolean; virtual;
      procedure Cancel; virtual;
   end;

   dmenuptr = ^dmenu;

   dmenu = object(menu)
      lothresh,hithresh:HorizontalSlideBar;
      DiffMode:MutExclusiveN;
      Accept:linkedbutton3d;
      constructor Init(Xpos,Ypos:word;Alink:ActionProcedure);
      procedure Show; virtual;
      procedure HiLite(Xpos,Ypos:word); virtual;
      function Action(Xpos,Ypos:word):boolean;virtual;
      function KeyAction(Xpos,Ypos:word;Key:integer):boolean; virtual;
   end;
```

File Name: DEFS.PAS

```
{ 10 degree cell definition object }


Cell = object(control)
    BitImg,AuxBitImg       : ^byte;
    Weight_or_Coverage     : byte;
    Pac                    : single;
    AboveColor,belowcolor  : integer;
    CellNumber             : integer;
    ActionProc             : CellActionProcedure;
    ShowProc               : CellShowProcedure;
    region, AboveThreshold : boolean;
    BorderColor,
    BorderHiliteColor      : word;
    lat,lon                : integer;
    Constructor Init(InitX,InitY,InitWidth,InitHeight,InitLat,InitLon,
                     InitCellNumber: integer;
                     InitAboveColor,InitBelowColor,InitBorderColor,
                     InitBorderHiliteColor: word;
                     InitActionProc : CellActionProcedure;
                     InitShowProc   : CellShowProcedure);
    Procedure   Show;                                    virtual;
    Procedure   ShowRegion;
    Procedure   HideRegion;
    Procedure   Hilite(Xpos,Ypos: word);                 virtual;
    Procedure   UnHilite;
    Function    Action(Xpos,Ypos: word) : boolean;       virtual;
    Function    IsAboveThreshold        : boolean;       virtual;
    Procedure   Cancel;                                  virtual;
end;


CellGrid = object(control)
        CenterX,
        CenterY         : integer;
        ZoomFactor      : real;
        CellIndex       : integer;
        CancelProcedure : CellGridCancelProc;
        CellArray: ARRAY[1..444] OF cell;
        constructor Init(InitX, InitY, InitWidth, InitHeight : integer;
                         InitActionProc : CellActionProcedure;
                         InitShowProc : CellShowProcedure;
                         InitCancelProc : CellGridCancelProc;
                         initabovecolor,initbelowcolor,initbordercolor,
                        inithilitecolor:integer);
        Procedure Show; virtual;
        Procedure Hilite(Xpos,Ypos:word); virtual;
```

File Name: DEFS.PAS

```
        Function Action(Xpos,Ypos:word):boolean; virtual;
        procedure Cancel; virtual;
end;

CoverageGridPtr = ^CoverageGrid;
CoverageGrid = object(cellgrid)
        Procedure Show; virtual;
end;

Coordinate = record
               x,y : integer;
             end;

WeightsGridPtr = ^WeightsGrid;
WeightsGrid = object(cellgrid)
   HasBeenChanged,pickingweights,SelectingByGroup: boolean;
   FirstCoord,LastCoord : Coordinate;
   eptr: emenuptr;
   weightsfile: string;
   Procedure Show; virtual
end;

DiffCellPopWindow = object(menu)
  textcolor,
  selectedcolor  : word;
  HilitedCellPtr : ^Cell;


  constructor Init(InitX,InitY,InitWidth,InitHeight : integer;
                   InitColor,InitBorderColor,InitTextColor,
                   InitHiliteColor,InitSelectedColor: word);
  procedure   Show;                                    virtual;
  procedure   Cancel;                                  virtual;
end;

CellPopWindow = object(menu)

  subcells   : Array[1..12, 1..24] of cell;

  Pac            : real;
  Weight         : integer;
  CellNum        : integer;
  eptr           : emenuptr;
  month,
  hour,
  xoffset,
  yoffset,
  cellwidth,
  cellheight     : integer;
```

```
    textcolor,
    selectedcolor  : word;
    HilitedCellPtr : ^Cell;

    constructor Init(InitX, InitY, InitWidth, InitHeight: Integer;
                     InitColor,InitBorderColor,InitTextColor,
                     InitHiliteColor,InitSelectedColor: word;
                     InitActionProc:CellActionProcedure;
                     InitShowProc:CellShowProcedure;
                     InitCellColor:word;Initeptr:emenuptr);
    procedure   Show;                                   virtual;
    procedure   HiLite(Xpos,Ypos:word);                 virtual;
    function    Action(Xpos,Ypos:word)    : boolean;    virtual;
    procedure   Cancel;                                 virtual;
  end;

  SubCellPopWindow = object(menu)

    Pat             : real;
    xoffset,yoffset : integer;
    Stations        : MutExclusiveN;
    MDU             : linkedbutton;
    BottomMenuShown,
    MDU_shown       : boolean;
    BottomMenu      : window;

    TextColor,
    SelectedColor : word;
    HilitedCellPtr: ^cell;

    constructor Init(InitX, InitY, InitWidth, InitHeight: Integer;
                     InitColor,InitBorderColor,InitTextColor,
                     InitSelectedColor: word;
                     InitActionProc:ActionProcedure);
    procedure   FillBottomOfWindow;
    procedure   Show;                                   virtual;
    function    Action(Xpos,Ypos:word): boolean;        virtual;
    function    KeyAction(Xpos,Ypos:word;key:integer): boolean;  virtual;
    procedure   Hilite(Xpos,Ypos:word);                 virtual;
    procedure   Cancel;                                 virtual;
  end;

  lmenuptr = ^lmenu;
  lmenu = object(menu)
     Dlist              : listptr;
     listposition,
     listlength         : integer;

     DEFAULTPATH,
```

File Name: DEFS.PAS

```
      EString             : string;
      SelectedFile        : string;
      DirectoryMaskButton : EditButton;
      DirectoryList       : MutExclusiveN;
      VBar                : VerticalSlideBar;
      ViewButton,
      LoadButton          : linkedbutton3D;
      mc,mbc              : word;
      constructor Init(Xpos,Ypos:word;InitMenuColor,InitMenuBorderColor:word;
                       Vlink,Llink:ActionProcedure;
                       InitPath,InitEString:string);
      procedure   Show;                                              virtual;
      procedure   HiLite(Xpos,Ypos:word);                            virtual;
      function    Action(Xpos,Ypos:word)                : boolean;   virtual;
      function    KeyAction(Xpos,Ypos:word;Key:integer) : boolean;   virtual;
      procedure   Cancel;                                            virtual;
      procedure   DisposeOldDirectoryList;
      procedure   DisplayDirectoryList;
      procedure   GetNewDirectoryList;
   end;

   lbarptr = ^lowerstatusbar;
   lowerstatusbar = object(borderedarea)
      eptr: emenuptr;
      MinPsa,MeanPsa,MaxPsa,RegMinPsa,RegMeanPsa,RegMaxPsa:single;
      Archivefile,
      Weightsfile,
      QRfile        : string;
      constructor Init(InitX,InitY,InitWidth,InitHeight,InitColor,InitBorterColor:word;
           Initeptr:emenuptr);
      procedure Show; virtual;
   end;

   sidestatusbar = object(borderedarea)
      eptr: emenuptr;
      constructor Init(InitX,InitY,InitWidth,InitHeight,InitColor,InitBorderColor:word;
           Initeptr:emenuptr);
      procedure Show; virtual;
   end;

   stubobject = object(menu)
      constructor Init;
      procedure Show; virtual;
   end;

   MemoArea = object(borderedarea)
     Font            : word;
     TextLineHeight,
     BorderSize,
```

File Name: DEFS.PAS

```
    NumChars,
    NumLines        : integer;
    HiliteColor,
    TextColor,
    HiliteTextColor : word;
    Display_String,
    MemoText        : MemoTextType;

    constructor Init(InitX,InitY,InitFont,InitColor,InitHiLiteColor,
                     InitTextColor,InitHiliteTextColor,
                     InitBorderColor:word);
    procedure   GetNewText(Text:CommentAreaType);
    procedure   AssignChangedText(var Text:CommentAreaType);
    procedure   HideTheCursor(LineNum:integer);
    procedure   ShowTheCursor(LineNum,CursorPosition:integer);
    procedure   Show;                                          virtual;
    procedure   Hilite(Xpos,Ypos:word);                        virtual;
    procedure   DisplayText(NewText:MemoTextType;
                            LineNum,CursorPosition:integer);
    function    Action(Xpos,Ypos : word) : boolean;            virtual;
  end;

  SaveMenuPtr = ^SaveMenu;
  SaveMenu = object(menu)
    TextLabel,
    Title          : string;
    TextColor      : word;
    xoffset,
    yoffset        : integer;
    MemoBox        : MemoArea;
    SaveButton     : linkedbutton3D;
    FileButton     : EditButton;
    constructor Init(InitX, InitY, InitWidth, InitHeight: Integer;
                     InitColor,InitHiliteColor,InitMemoColor,
                     InitMemoHiliteColor,InitBorderColor,
                     InitTextColor,InitHiliteTextColor,InitFont: word;
                     Slink:ActionProcedure;InitFileString:string);
    procedure   Show;                                          virtual;
    procedure   HiLite(Xpos,Ypos:word);                        virtual;
    function    Action(Xpos,Ypos:word)              : boolean; virtual;
    function    KeyAction(Xpos,Ypos:word; key:integer): boolean; virtual;
    procedure   Cancel;                                        virtual;
  end;

  HelpMenu = object(menu)
    TextLabel,
    Title          : string;
    HelpFileName   : shortstring;
    TextColor      : word;
```

File Name: DEFS.PAS

```
    boxoffset,
    xoffset,
    yoffset        : integer;
    NextButton,
    PrevButton     : linkedbutton3D;
    GoToButton     : EditButton;
    constructor Init(InitX, InitY, InitWidth, InitHeight: Integer;
                     InitColor,InitHiliteColor,InitBorderColor,
                     InitTextColor,InitHiliteTextColor,InitFont: word;
                     Nlink,Plink:ActionProcedure);
    procedure   Show;                                              virtual;
    procedure   HiLite(Xpos,Ypos:word);                            virtual;
    function    IsHelpFile(filename : string): boolean;
    function    Action(Xpos,Ypos:word)              : boolean;     virtual;
    function    KeyAction(Xpos,Ypos:word; key:integer): boolean;   virtual;
    procedure   ClearWindow;
    procedure   ShowText;
    procedure   Cancel;                                            virtual;
  end;


  { the message box object takes an input string of max length 250
    and displays it in a window }


  messageboxobject = object(window)

     procedure displaymessage(Xpos,Ypos,nlines,nchars,backcolor,textcolor: integer;
                              message: string);
     procedure hidemessage;
     procedure changeattr(XX,YY,WW,HH,BK,TX: integer);
     procedure redisplaymessage(nlines,nchars:integer; message: string);
  end;




  procedure SaveEditParams(eptr:emenuptr;saved:savedparamsrecordptr);
  procedure RestoreEditParams(eptr:emenuptr;saved:savedparamsrecordptr);
  Procedure ShowCurrentPickChoice;
  procedure messagewait;
  function FileName (context : shortstring) : shortstring;

Var

(********** the definition for the QR path variable is moved to QRutil.PAS
   QRPATH,
**********)
```

File Name: DEFS.PAS

```
    messagebox: messageboxobject;

    batchqueue,topofbatchqueue: batchqueueentryptr;

    Basescreen,basescreen2:^borderedarea;
    stubwindow:^stubobject;
    h,q,f,s,o,r,difference,difference2,cr,dr:^linkedbutton3D;

    BIGmap,
    BIGmap2,
    LEFTmap,
    RIGHTmap : ^Map;

    BIGCellGrid,
    LEFTCellGrid,
    RIGHTCellGrid,
    diffcellgrid: CellGridPtr;
    WeightCellGrid : WeightsGridPtr;
    CoverageCellGrid: CoverageGridPtr;

    Filemenu,LeftFilemenu,RightFilemenu,optionsmenu:^pickmenu;
    Editmenu,AuxEditmenu: ^emenu;

    Diffmenu: dmenuptr;

    ListMenu,
    RightListmenu,
    ReliabilityListMenu,
    AuxReliabilityListMenu,
    WeightsListMenu          : ^lmenu;

    Leftsplit:^borderedarea;
    Rightsplit:^borderedarea;
    lf,rf:^linkedbutton3D;

    leftlowerstatus,rightlowerstatus:^lowerstatusbar;
    leftsidestatus,rightsidestatus:^sidestatusbar;

    leftexpand,rightexpand:^linkedbutton3D;

    DiffCellPopUp  : ^DiffCellPopWindow;

    CellPopUp : ^CellPopWindow;

    SubCellPopUp : ^SubCellPopWindow;

    SaveMenuWindow,auxsavemenuwindow,WeightsSaveMenuWindow,
    queuesavemenuwindow,auxqueuesavemenuwindow  : ^SaveMenu;
```

File Name: DEFS.PAS

```
   HelpMenuWindow : ^HelpMenu;

   QUIT:boolean;
```

File Name: ERRLOG.PAS

```
UNIT Errlog;
{*****************************************************************************
*                                                                            *
*        PROGRAM NAME - Omega Performance Assessment                         *
*                                and                                         *
*                        Coverage Evaluation                                 *
*                              (PACE)                                        *
*                           Workstation                                      *
*                                                                            *
*           UNIT NAME - ERRLOG                                               *
*                                                                            *
*****************************************************************************
*                                                                            *
*        This program was prepared by                                        *
*                                                                            *
*              The Analytic Science Corporation (TASC)                       *
*                    55 Walkers Brook Drive                                  *
*                Reading, Massachusetts 01867                                *
*                                                                            *
*        PACE has been developed to run on a IBM PC/AT or compatible         *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main           *
*        memory and an EGA or compatible graphics adapter and color          *
*        monitor. This work was performed under contract number              *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for           *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.        *
*                                                                            *
*****************************************************************************
*                                                                            *
*        PURPOSE                                                             *
*             General purpose error logging routine.  Error messages         *
*        are written to the file "ERROR.LOG" and a warning sound             *
*        is generated.                                                       *
*                                                                            *
*****************************************************************************}

INTERFACE
   Uses Crt;

   PROCEDURE Log_Error(Message: String);

IMPLEMENTATION
   VAR
      f: text;

   PROCEDURE Log_Error(Message: String);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
```

File Name: ERRLOG.PAS

```
      *          Write a message to the ERROR.LOG file and alert the user   *
      *          that a message has been recorded. The ERROR.LOG file is    *
      *          re-written each time PACE is run.                          *
      ******************************************************************************
      }
   VAR
      i, freq : Integer;
   BEGIN
      Assign(f, 'Error.Log');  Append(f);
      Writeln(f, Message);
      Close(f);

      { Make some phaser sounds }
      Nosound;
      FOR i := 1 TO 3 DO
         BEGIN
            Freq := 5000;
            REPEAT
               Sound(Freq);
               Freq := Round(Freq*0.995)
            UNTIL Freq <= 125;
            NoSound
         END
   END;

BEGIN
   { Create a new and empty error log file }
   Assign(f, 'Error.log'); Rewrite(f);
   Close(f);
END.
```

File Name: HELPMENU.PAS

```
(*************************************************************************
*-                                                                       *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                       *
*           UNIT NAME - Part of the PACEOBJS Unit                       *
*                                                                       *
*************************************************************************
*                                                                       *
*        This program was prepared by                                   *
*                                                                       *
*              The Analytic Science Corporation (TASC)                  *
*                    55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                          *
*                                                                       *
*        PACE has been developed to run on a IBM PC/AT or compatible    *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main      *
*        memory and an EGA or compatible graphics adapter and color     *
*        monitor. This work was performed under contract number         *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for      *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.   *
*                                                                       *
*************************************************************************
*                                                                       *
*        PURPOSE                                                        *
*           Collection of object methods and utility routines for the   *
*           PACE context sensetive help subsystem.                      *
*                                                                       *
*                                                                       *
*************************************************************************)

procedure GenerateContextHelpList(var f : text);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Read the list of help sections and associated contexts that*
      *           from the help information file. Store the section names and*
      *           associated contexts in a list so that we can access the    *
      *           appropriate help section when a particular context is      *
      *           active.                                                    *
      ************************************************************************
      }

var
  Endptr,temp : CHLPtr;
```

```
begin
  ContextHelpList := nil;

  While not EOF(f) do
  begin
    new(temp);
    if (temp = nil) then
    begin
      Log_Error('Not enough memory to Generate Context Help List');
    end;
    temp^.Next := nil;

    readln(f,temp^.ContextStr);
    readln(f,temp^.filename);

    if ContextHelpList = nil then       (* at beginning *)
    begin
      ContextHelpList := temp;
      EndPtr := ContextHelpList;
    end
    else
    begin                                (* otherwise, add at end *)
      Endptr^.Next := temp;
      Endptr := Endptr^.next;
    end;
  end;
end;

procedure GenerateHelpFilesList(var f : text);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Generate a list of help files that correspond to the          *
      *           various help sections that are available in                   *
      *           PACE help subsystem. Check for the existance of each help     *
      *           file as it is inserted into the list.                         *
      ***************************************************************************
      }

var
  FoundFile : boolean;
  hf : text;
  Endptr,temp : HFLPtr;

begin
  HelpFilesList := nil;
```

```
  While not EOF(f) do
  begin
    new(temp);
    if (temp = nil) then
    begin
      Log_Error('Not enough memory to Generate Help Files List');
    end;
    temp^.Next := nil;
    temp^.Prev := temp;
    readln(f,temp^.filename);

    (* test to see if files exist *)
    Assign(hf,HelpPath + '\' + temp^.filename + '.HLP');
    {$I-}
    Reset(hf);
    {$I+}
    if (IOResult <> 0) then
    begin
      Log_Error('Problem opening file: ' + HelpPath + '\' + temp^.filename + '.HLP' );
      FoundFile := FALSE;
    end
    else
    begin
      FoundFile := TRUE;
      close(hf);
    end;

    if FoundFile then   (* if file is found, then add to our list *)
    begin
      if HelpFilesList = nil then       (* at beginning *)
      begin
        HelpFilesList := temp;
        EndPtr := HelpFilesList;
        temp^.Prev := nil;
      end
      else
      begin                                (* otherwise, add at end *)
        Endptr^.Next := temp;
        temp^.Prev := Endptr;
        Endptr := Endptr^.next;
      end;
    end;
  end;
end;



procedure InitializeHelpSystem;
      {
```

File Name: HELPMENU.PAS

```
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Set up the help subsystem by checking for the help system          *
      *           information file and creating the help section lists               *
      ********************************************************************************
      }

var
  f : text;

begin
  Assign(f,HelpPath + '\' + ContextFile);
  {$I-}
  Reset(f);
  {$I+}
  if (IOResult <> 0) then
  begin
    Log_Error('Problem opening file: ' + HelpPath + '\' + ContextFile);
  end
  ELSE
  begin
     GenerateContextHelpList(f);
     Close(f);
  end;

  Assign(f,HelpPath + '\' + HelpFile);
  {$I-}
  Reset(f);
  {$I+}
  if (IOResult <> 0) then
  begin
    Log_Error('Problem opening file: ' + HelpPath + '\' + HelpFile);
  end
  ELSE
  begin
     GenerateHelpFilesList(f);
     Close(f);
  end;
end;

function FileName (context : shortstring) : shortstring;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Associates a help section/help file name for a given               *
      *           context. Provides the context-sensetive part of the help           *
      *           subsystem.                                                         *
```

```
      ********************************************************************************
      }

var
  pointer : CHLPtr;

begin
  pointer := ContextHelpList;

  While (pointer <> nil) AND (NOT (context = pointer^.ContextStr)) do
    pointer := pointer^.next;

  if (context = pointer^.ContextStr) then
    FileName := pointer^.filename
  else
    FileName := '';
end;


Constructor HelpMenu.Init(InitX, InitY, InitWidth, InitHeight: Integer;
                          InitColor,InitHiliteColor,InitBorderColor,
                          InitTextColor,InitHiliteTextColor,InitFont: word;
                          Nlink,Plink:ActionProcedure);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Object initialization code to set up the context sensitive *
      *           help subsystem interface screens.                                  *
      ********************************************************************************
      }

const
  bs      =  20;    (* space between buttons *)
  bw      =  60;    (* button width  *)
  bh      =  16;    (* button height *)
  ew      =  160;   (* edit button width  *)
  eh      =  15;    (* edit button height *)

begin
  TextColor   := InitTextColor;
  Title       := 'PACE Help Menu';
  TextLabel   := 'Section Number:';
  boxoffset   := 20;
  xoffset     := 20;
  yoffset     := 10;
  menu      .Init(InitX,InitY,InitWidth,InitHeight,InitColor,
                  InitBorderColor,2,WindowShadowWidth);
  NextButton.Init(InitX+xoffset,InitY+InitHeight-yoffset-bh,
```

```
                bw,bh,white,black,black,white,'Next',NLink,'N');
  PrevButton.Init(InitX+bw+xoffset+bs,InitY+InitHeight-yoffset-bh,
                bw,bh,white,black,black,white,'Prev',PLink,'P');
  GoToButton.Init(InitX+InitWidth-xoffset-ew,
                InitY+InitHeight-yoffset-eh,
                ew,eh,white,black,black,white,'',0,4,19,TRUE);

  InitializeHelpSystem;

end;


procedure HelpMenu.Show;
      (
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Display a page of the context sensitive help and the       *
      *           additional help controls.                                  *
      ************************************************************************
      )


var
  f : text;
  xx,yy: integer;
  ContextName  : shortstring;
  i : integer;
  helptext : string[80];


begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  Window.Show;
  setcolor(TextColor);
  rectangle(X+boxoffset,Y+boxoffset,X+Width-boxoffset,Y+Height-NextButton.Height-yoffset-(bo
  rectangle(X+boxoffset+2,Y+boxoffset+2,X+Width-boxoffset-2,Y+Height-NextButton.Height-yoffs
  settextjustify(centertext,toptext);
  outtextxy(X+(Width DIV 2),Y+(yoffset DIV 2)+2,Title);

  settextjustify(righttext,centertext);
  outtextxy(GoToButton.X-5,GoToButton.Y+(GoToButton.Height DIV 2)+1,TextLabel);
  NextButton.Show;
  PrevButton.Show;
  ShowText;
  GoToButton.Show;
  Show_Cursor(xx,yy);
end;
```

File Name: HELPMENU.PAS

```
procedure HelpMenu.HiLite(Xpos,Ypos:word);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method to highlight the help interface controls.                   *
      *                                                                              *
      ********************************************************************************
      }

begin
  NextButton.Hilite(Xpos,Ypos);
  PrevButton.Hilite(Xpos,Ypos);
  GoToButton.Hilite(Xpos,Ypos);
end;


function HelpMenu.IsHelpFile(filename:string): boolean;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Checks for the existance of help for a user selected               *
      *           help subsystem section.                                            *
      ********************************************************************************
      }

var
  i : integer;
  f : text;

begin
  for i := 1 to length(filename) do
  if filename[i] = '.' then
    filename[i] := '_';
  Assign(f,HelpPath + '\' + filename + '.hlp');
  {$I-}
  Reset(f);
  {$I+}
  if (IOResult = 0) then
  begin
    HelpFileName := filename;
    IsHelpFile := TRUE;
    close(f);
  end
  else
    IsHelpFile := FALSE;
```

```
end;

function HelpMenu.Action(Xpos,Ypos:word): boolean;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method for performing the actions associated with the help *
      *           subsystem controls to page through the help information.   *
      ********************************************************************************
      }

var
  dummy    : boolean;

begin
   dummy := NextButton.Action(Xpos,Ypos);
   dummy := PrevButton.Action(Xpos,Ypos);
   if GoToButton.Action(Xpos,Ypos) then
   begin
     If IsHelpFile(GoToButton.Text_String) then
     begin
       ClearWindow;
       ShowText;
     end
     else
       beep;
   end;
end;

function HelpMenu.KeyAction(Xpos,Ypos:word; key:integer): boolean;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method for performing the hotkey actions associated with   *
      *           the help subsystem controls to page through the help       *
      *           information.                                                       *
      ********************************************************************************
      }

var
  dummy : boolean;

begin
  KeyAction := false;
  if (NextButton.KeyAction(Xpos,Ypos,key)) OR
     (PrevButton.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
```

```
end;

procedure HelpMenu.ClearWindow;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Cleans up the help window so that the next page of help       *
      *           can be displayed.                                             *
      ***************************************************************************
      }
var
  xx,yy : integer;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  setfillstyle(solidfill,Color);
  bar(X+boxoffset+3,Y+boxoffset+3,X+Width-boxoffset-3,Y+Height-NextButton.Height-yoffset-(bo
  Show_Cursor(xx,yy);
end;

procedure HelpMenu.ShowText;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Writes the actual help information into the help window.      *
      *                                                                         *
      ***************************************************************************
      }

const
  Xtextoffset = 18;
  Ytextoffset = 10;
  maxnumlines = 23;

var
  f : text;
  xx,yy: integer;
  i : integer;
  temp : string;
  helptext : string[80];


begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  Assign(f,HelpPath + '\' + HelpFileName + '.HLP');
```

File Name: HELPMENU.PAS

```
  {$I-}
  Reset(f);
  {$I+}
  if (IOResult <> 0) then
    Log_Error('Problem opening file: ' + HelpPath + '\' + HelpFileName)
  else
  begin
    temp := HelpFileName;
    for i := 1 to length(temp) do
    begin
      if temp[i] = '_' then
        temp[i] := '.';
    end;
    GoToButton.Change_String(temp);
    GoToButton.ReShow;
(*  GoToButton.DisplayText(temp); *)

    i := 0;
    settextjustify(lefttext,toptext);
    setcolor(textcolor);
    settextstyle(smallfont,horizdir,4);

    while not EOF(f) and (i < maxnumlines) do
    begin
      inc(i);
      readln(f,helptext);
      outtextxy(X+boxoffset+Xtextoffset,Y+boxoffset+Ytextoffset*i,helptext);
    end;
    Close(f);
  end;
  settextstyle(defaultfont,horizdir,1);
  Show_Cursor(xx,yy);
end;

procedure HelpMenu.Cancel;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for removing the help subsystem window from the     *
      *           display.                                                   *
      ************************************************************************
      }

begin
  Menu.Cancel;
end;
```

File Name: LOAD_DEF.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the CELLUTIL Unit                        *
*                                                                        *
**************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
**************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Utilities to load the PACE 10 degree latitude by 10 degree   *
*           longitude cell definitions. Variation of the code contained  *
*           in the loaddef.pas program file.                             *
*                                                                        *
**************************************************************************}

PROCEDURE Load_Cell_Definition(     cellsize       : Integer;
                               VAR Cell_Defn_List : Cell_defn_ptr;
                               VAR Numb_Cells     : Integer);
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Load the cell grid definitions into a memory based linked-  *
      *           list.                                                       *
      *************************************************************************
      }
{
   Load cell definition for indicated cellsize into memory as a
   list attached to the pointer Cell_Defn_List.  Callsize is the basic size
   (size at the equator) in degrees (e.g., 5, 10).
```

```
File Name: LOAD_DEF.PAS

   Numb_Cells is set to the number of cells read in.

   The format of the file name is:    GRID_nn.DEF
      Where nn is the basic cell size in degrees

   Ver 0   05/06/88   KATench
   Ver 1   05/23/90   EMAustvold      for use in PACE code
}

VAR
   Cdefnfname     : string[40];
   Last_Cell,
   Temp_cell      : Cell_defn_Ptr;          { cell definition list }
   Temp_Cell_Defn : Cell_defn;              { for reading in the definition }
   fCdefn         : file of Cell_Defn;
   i              : Integer;

BEGIN
   { figure out the name of the file to use }
   Cdefnfname:='GRID_'+DIGITS[Cellsize div 10]+DIGITS[Cellsize mod 10]+'.DEF';
   { Read in and store the cell definition file }

   Assign(fCdefn, Cdefnfname); Reset(fCdefn);
   Read(fCdefn, Temp_cell_defn);
   New( Cell_Defn_List );
   Cell_Defn_List^.lat1    := Temp_cell_defn.lat1;
   Cell_Defn_List^.lat2    := Temp_cell_defn.lat2;
   Cell_Defn_List^.lon1    := Temp_cell_defn.lon1;
   Cell_Defn_List^.lon2    := Temp_cell_defn.lon2;
   Cell_Defn_List^.range   := 0;
   Cell_Defn_List^.bearing := 0;
(*
   Cell_Defn_List^.range := Temp_cell_defn.xlate[Station].range;
   Cell_Defn_List^.bearing := Temp_cell_defn.xlate[Station].bearing;
*)
   Last_Cell := Cell_Defn_List;
   Numb_Cells := 1;
   WHILE NOT EOF(fCdefn) DO BEGIN
      Inc(Numb_Cells);
      New( Temp_Cell );
      Read(fCdefn, Temp_cell_defn);      { get a new definition and store away }
      Temp_Cell^.lat1    := Temp_cell_defn.lat1;
      Temp_Cell^.lat2    := Temp_cell_defn.lat2;
      Temp_Cell^.lon1    := Temp_cell_defn.lon1;
      Temp_Cell^.lon2    := Temp_cell_defn.lon2;
      Temp_Cell^.range   := 0;
      Temp_Cell^.bearing := 0;
(*
      Temp_Cell^.range := Temp_cell_defn.xlate[Station].range;
```

File Name: LOAD_DEF.PAS

```
      Temp_Cell^.bearing := Temp_cell_defn_xlate[Station].bearing;
*)
      Last_cell^.next := Temp_cell;     { link new one into the list }
      Last_cell := Temp_cell;
   END;
   Last_cell^.next := nil;              { mark the last link as the end }
   Close(fCdefn);

(*****************************************************************************)
(*
  (print the list to the screen to see if it's in memory OK )
      Temp_Cell := Cell_Defn_List;
      i := 0;
      REPEAT
         Inc(i);
         WITH Temp_Cell^ DO BEGIN
            Write(i:4,'  Lats ',(Lat1/10):7:1, (Lat2/10):7:1);
            Write('  Lons ',(lon1/10):7:1, (Lon2/10):7:1);
            Writeln('  !  Station R/B: ',(Range/100):6:2, (Bearing/10):7:1);
         END;
         Temp_Cell := Temp_Cell^.next;
      UNTIL Temp_Cell = nil;
*)
(*****************************************************************************)

END;
```

File Name: LOAD_RB.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the RBCONV2 Program                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains utility code to read and parse the        *
*           radial format signal data needed by the range/bearing to     *
*           cell format database conversion program.                     *
*                                                                        *
*************************************************************************}

PROCEDURE Load_Range_Bearing_Data( Station, Month, GMT,
                                   Freq: Integer;
                                  VAR RB_List:RB_Ptr);
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Construct the appropriate range/bearing data file name,     *
      *           read in the range/bearing data and parse the input data     *
      *           to obtain the desired signal quantities.                    *
      *************************************************************************
      }
{
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
File Name: LOAD_RB.PAS

     Ver 1    05/88   KATench-
     ver 2    11/88   DPFrank      to process reformatted Range/Bearing file
     ver 3    05/90   KATench      added RB_List disposal and hardwired sta C
                                   & added file name creation from input params
     ver 4    10/90   EMAustvold   Remove hardwired station C.
                                   This version should hardwire the month of
                                   February.
     ver 5    11/90   EMAustvold   Change to read new data input format
                                   Hardwired NonStandardRadials

new format :
line  1 - 6 is header information

 STA=A MON=Feb GMT=01 S_PATH BEAR=000 FREQ=10.2

 DIS  SPM1 SPM1 SPMS SPMS LPMS
      AMP  PHS  AMP  PHS  AMP
 *10  *10       *1C       *10
 MM   DB   CEC  DB   CEC  DB
   0  999    0  999    0 -999

lines 7 - number of radials contains data in the columns

cols (01 - 04)  - distance along radial in tenths of a MegaMeter
                  (ie 1 = 0.1, 2 = 0.2,... )
cols (05 - 09)  - Short Path Mode 1 Amplitude (dB)
cols (10 - 14)  - Short Path Mode 1 Phase (cecs)
cols (15 - 19)  - Short Path Mode Sum Amplitude (dB)
cols (20 - 24)  - Short Path Mode Sum Phase (cecs)
cols (25 - 29)  - Long Path Mode Sum Amplitude (cecs)

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
)

const
  NumberOfNonStandardRadials = 14;

type
  NonStandardRadialType =
                 array[1..8,1..NumberOfNonStandardRadials] of integer;
                 (* 8 stations x 14 possible NonStandardRadials *)
const
  NumberRecordsPerRadial = 201;       (* records numbered 0 - 200 *)
  StepsizeAlongRadial    = 0.10;      (* one tenth of a megameter *)

  NonStandardRadials : NonStandardRadialType =
               (* A *)((  45, 225, 105, 285, 115, 295, 000,
                          000, 000, 000, 000, 000, 000, 000  ),
               (* B *) (  17, 197, 156, 336, 165, 345, 175,
```

File Name: LOAD_RB.PAS

```
                        355, 178, 358, 000, 000, 000, 000 ),
                (* C *) (   5, 185,  25, 205,  27, 207, 152,
                        332, 163, 343, 167, 347, 000, 000 ),
                (* D *) (   8, 188,  34, 214,  44, 224, 165,
                        345, 000, 000, 000, 000, 000, 000 ),
                (* E *) (  13, 193,  23, 203, 148, 328, 155,
                        335, 167, 347, 174, 354, 000, 000 ),
                (* F *) (   5, 185,  16, 196,  22, 202, 152,
                        332, 175, 355, 178, 358, 000, 000 ),
                (* G *) (   5, 185,  18, 198, 165, 345, 173,
                        353, 000, 000, 000, 000, 000, 000 ),
                (* H *) (   2, 182,   5, 185,  26, 206, 165,
                        345, 172, 352, 175, 355, 000, 000 ) );

VAR
   Last_RB,
   Temp_RB      : RB_Ptr;  { station range-bearing data list }
   fRB          : text;
   fDIAG        : text;
   RBfname      : string[40];
   i,j,k        : integer;
   Inrec        : string [80];
   RBFileName   : string [40];
   RealTemp     : real;
   IntTemp1     : integer;
   Code         : integer;
   Radials      : SortList;
   NumberRadials: integer;
   freqstr,
   MonthName    : String[3];
   stationstr   : string[1];

BEGIN
  { Dispose of old list if need be }
  Last_RB := RB_List;
  WHILE Last_RB <> nil DO BEGIN
    Temp_RB := Last_RB^.next;
    Dispose(Last_RB);
    Last_RB := Temp_RB;
  END;

  {
    Generate file name of the form smmmtt.fff
    where:
       s   = sta        (A,B,C,D,E,F,G,H)
       mmm = month      (FEB,MAY,AUG,NOV)  ( 4 months we're concerned with.)
       tt  = GMT        (01..24)
       fff = frequency  (102 or 136)
   }
```

File Name: LOAD_RB.PAS

```
  case Month of
    1 : MonthName := 'JAN';
    2 : MonthName := 'FEB';
    3 : MonthName := 'MAR';
    4 : MonthName := 'APR';
    5 : MonthName := 'MAY';
    6 : MonthName := 'JUN';
    7 : MonthName := 'JUL';
    8 : MonthName := 'AUG';
    9 : MonthName := 'SEP';
   10 : MonthName := 'OCT';
   11 : MonthName := 'NOV';
   12 : MonthName := 'DEC';
  else
    MonthName := 'XXX';
  end; (* case stmt *)

   str(freq:3,freqstr );
   stationstr := chr(ord(digits[station])+16);

   RBFileName := rbpath +
                 stationstr +
                 MonthName +
                 digits[GMT DIV 10] +
                 digits[GMT MOD 10] +
                 '.' +
                 freqstr;

 {$I-}
 Assign(fRB, RBFileName);
 Reset(fRB);
 {$I+}
 if (IOResult = 0) then       (* no error *)
 begin
   writeln;
   Writeln('Reading Range/Bearing file ',RBFileName,' . . .');

   {Assign Bearings for "Standard" Radials}
   For i := 1 to NUMSTANDARD_RADIALS Do    (* 36 standard radials *)
     Radials[i] := 10*(i-1);               (* 0..350, every 10 degrees *)

   {*********************************************************}
   {Input Bearings for "Additional" Radials                   }
   {*********************************************************}

   NumberRadials := NUMSTANDARD_RADIALS;
   i := 1;
   While (i <= NumberOfNonStandardRadials ) and
         (NonStandardRadials[Station,i] <> 0) Do
```

File Name: LOAD_RB.PAS

```
    BEGIN
      Radials[NUMSTANDARD_RADIALS+i] := NonStandardRadials[Station,i];
      Inc(i);
    END;

    NumberRadials := NUMSTANDARD_RADIALS + (i-1);

                                {Sort Radials if Necessary}

    If (NumberRadials > NUMSTANDARD_RADIALS) Then
      Qksort(Radials, 1, NumberRadials);

    {*******************************}
    {Process Range/Bearing Records:}
    {*******************************}

    Write('@');

    for j := 1 to NumberRadials do
    begin
      Write('.');
      New( Temp_RB );
      Temp_RB^.NumAtoms := NumberRecordsPerRadial;
      If j = 1 Then
      BEGIN
        RB_List := Temp_RB;
        Last_RB := RB_List;
      END
      Else
      BEGIN
        Last_RB^.next := Temp_RB;
        Last_RB := Temp_RB;
      END;

      Temp_RB^.RB.Bearing := Radials[j];

      For i := 0 to (NumberRecordsPerRadial-1)Do
      BEGIN
        if i = 0 then              (* if processing a new radial then *)
        begin                      (* read past the 6 lines of header *)
          for k := 1 to 6 do
            readln(fRB, Inrec);
        end;

        Readln(fRB, Inrec);

        Val(Copy(Inrec,1,4), IntTemp1, Code);
        if (Code = 0) then
          Temp_RB^.RB.a[i].Range := IntTemp1
```

```
      else
      begin
        writeln(i,j);
        writeln(flog,'***************');
        writeln(flog,'Error reading Range Bearing file: ',RBFileName);
        writeln(flog,'Attempting to read Range values');
        writeln(flog,'Radial Number = ',j:4);
        writeln(flog,'Record Number along Radial = ',i:4);
        writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
        writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
        writeln(flog,'Range has been assigned the value of zero.');
        writeln(flog,'***************');
        Temp_RB^.RB.a[i].Range := 0;
      end;

      Val(Copy(Inrec,6,4), IntTemp1, Code);   (Short-path Mode 1 Amp)
      if (Code = 0) then
        Temp_RB^.RB.a[i].SPM1Amp := IntTemp1
      else
      begin
        writeln(i,j);
        writeln(flog,'***************');
        writeln(flog,'Error reading Range Bearing file: ',RBFileName);
        writeln(flog,'Attempting to read Short Path Mode 1 Amplitude');
        writeln(flog,'Radial Number = ',j:4);
        writeln(flog,'Record Number along Radial = ',i:4);
        writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
        writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
        writeln(flog,'Short Path Mode 1 Amplitude has been assigned the value of zero.');
        writeln(flog,'***************');
        Temp_RB^.RB.a[i].SPM1Amp := 0;
      end;

      Val(Copy(Inrec,11,4), IntTemp1, Code);  (Short-path Mode 1 Phase)
      if (Code = 0) then
        Temp_RB^.RB.a[i].SPM1Phase := IntTemp1
      else
      begin
        writeln(i,j);
        writeln(flog,'***************');
        writeln(flog,'Error reading Range Bearing file: ',RBFileName);
        writeln(flog,'Attempting to read Short Path Mode 1 Phase');
        writeln(flog,'Radial Number = ',j:4);
        writeln(flog,'Record Number along Radial = ',i:4);
        writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
        writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
        writeln(flog,'Short Path Mode 1 Phase has been assigned the value of zero.');
        writeln(flog,'***************');
        Temp_RB^.RB.a[i].SPM1Phase := 0;
```

```
        end;

        Val(Copy(Inrec,16,4), IntTemp1, Code);  {Short-path Modesum Amp}
        if (Code = 0) then
          Temp_RB^.RB.a[i].SPMSumAmp := IntTemp1
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Short Path Mode Sum Amplitude');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Short Path Mode Sum Amplitude has been assigned the value of zero.')
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].SPMSumAmp := 0;
        end;

        Val(Copy(Inrec,21,4), IntTemp1, Code);  {Short-path Modesum Phase}
        if (Code = 0) then
          Temp_RB^.RB.a[i].SPMSumPhase := IntTemp1
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Short Path Mode Sum Phase');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Short Path Mode Sum Phase has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].SPMSumPhase := 0;
        end;

        Val(Copy(Inrec,26,4), IntTemp1, Code);  {Long-path Modesum Amp}
        if (Code = 0) then
          Temp_RB^.RB.a[i].LPMSumAmp := IntTemp1
        else
        begin
          writeln(i,j);
          writeln(flog,'***************');
          writeln(flog,'Error reading Range Bearing file: ',RBFileName);
          writeln(flog,'Attempting to read Long Path Mode Sum Amplitude');
          writeln(flog,'Radial Number = ',j:4);
          writeln(flog,'Record Number along Radial = ',i:4);
```

```
          writeln(flog,'Radial Bearing = ',Temp_RB^.RB.Bearing:4);
          writeln(flog,'Value read in from file = ^',IntTemp1:4,'^');
          writeln(flog,'Long Path Mode Sum Amplitude has been assigned the value of zero.');
          writeln(flog,'***************');
          Temp_RB^.RB.a[i].LPMSumAmp := 0;
        end;


      END;
    END;
    Last_RB^.next := nil;
    Close(fRB);


(**********************)
{
    Assign(fDIAG, 'DIAG.OUT');
    Rewrite(fDIAG);
    Temp_RB := RB_List;
    writeln;
    Writeln(fDIAG, Temp_RB^.NumAtoms);
    While (Temp_RB<>nil) Do
    BEGIN
      Writeln('@', Temp_RB^.RB.Bearing:5);
      Writeln(fDIAG,Temp_RB^.RB.Bearing:5);
      For i := 0 to (Temp_RB^.NumAtoms-1) Do
        Writeln(fDIAG,i:5,Temp_RB^.RB.a[i].Range:5,
                          Temp_RB^.RB.a[i].SPM1Amp:5,
                          Temp_RB^.RB.a[i].SPM1Phase:5,
                          Temp_RB^.RB.a[i].SPMSumAmp:5,
                          Temp_RB^.RB.a[i].SPMSumPhase:5,
                          Temp_RB^.RB.a[i].LPMSumAmp:5);
        Temp_RB := Temp_RB^.next;

    END;
    Close(fDIAG);
}
(**********************)


  end             { Range/Bearing Processing }
  else            { error reading file }
  begin
    Writeln(chr(7),'/// Range/Bearing file "',RBFileName,'" not found ///');
    Halt(1);
  end;
end;
```

File Name: MEMOAREA.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                                (PACE)                                  *
*                             Workstation                                *
*                                                                        *
*           UNIT NAME - Part of the PACEOBJS Unit                        *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the methods and initialization code       *
*           for the user memo input window that is used for user-input   *
*           documentation in the PACE load and save commands.            *
*                                                                        *
*************************************************************************}

constructor MemoArea.Init(InitX,InitY,InitFont,InitColor,InitHiLiteColor,
                          InitTextColor,InitHiliteTextColor,
                          InitBorderColor:word);
     {
     *************************************************************************
     *                                                                       *
     *        PURPOSE                                                        *
     *           Initialization code for the memo text input/edit area.      *
     *                                                                       *
     *************************************************************************
     }

const
  InitBorderSize =  4;   (* pixel border around MemoArea *)
  InitNumChars   = 50;   (* 50 characters per line *)
```

```
  InitNumLines  =  4;   (* 4 lines of text *)

var
  InitWidth,
  InitHeight        : word;
  OldStyle          : TextSettingsType;
  i                 : integer;

begin
  BorderSize := InitBorderSize;
  Font       := InitFont;
  NumChars   := InitNumChars;
  NumLines   := InitNumLines;
  Color      := InitColor;
  HiliteColor := InitHiliteColor;
  TextColor   := InitTextColor;
  HiliteTextColor := InitHiliteTextColor;

  GetTextSettings(OldStyle);
  settextstyle(Font,horizdir,1);

  TextLineHeight := textheight('1') + 4;       (* pixels above and below *)

  InitWidth  := NumChars * textwidth('1') + BorderSize*2;
  InitHeight := NumLines * TextLineHeight + BorderSize*2;

  BorderedArea.Init(InitX,InitY,InitWidth,InitHeight,Initcolor,
                    InitBorderColor,2);

  with OldStyle do begin
    settextjustify(Horiz,Vert);
    settextstyle(Font,Direction,CharSize);
  end;

end;

(* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> *)

procedure MemoArea.GetNewText(Text:CommentAreaType);
      {
      ************************************************************************
      *                                                                      *
      *       PURPOSE                                                        *
      *          Method to place altered text into the memo object.          *
      *                                                                      *
      ************************************************************************
      }

var
```

```
  i : integer;

begin
  for i := 1 to NumLines do
    MemoText[i] := copy(Text,1+((i-1)*NumChars),i*NumChars);
end;

(* >-------------------------------------------------------------------< *)

procedure MemoArea.AssignChangedText(var Text:CommentAreaType);
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           Method to save the text that is currently stored in the        *
      *           memo object so that we can retrieve it after the edit.         *
      ****************************************************************************
      }

var
  i,j : integer;

begin
  Text := '';
  for i := 1 to NumLines do begin
    for j := length(MemoText[i])+1 to NumChars do
      MemoText[i] := MemoText[i] + ' ';
    Text:= Text + MemoText[i];
  end;
end;

(* >-------------------------------------------------------------------< *)

procedure MemoArea.Show;
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           Method to display the memo area and it's current text          *
      *           in a window on the screen.                                     *
      ****************************************************************************
      }

var
  xx,yy,
  i          : integer;
  OldStyle   : TextSettingsType;
```

File Name: MEMOAREA.PAS

```
begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  GetTextSettings(OldStyle);
  settextstyle(Font,horizdir,1);
  setcolor(TextColor);
  settextjustify(lefttext,centertext);
  setfillstyle(solidfill,color);

  BorderedArea.show;

  for i := 1 to NumLines do begin
    bar(X+BorderSize, Y+BorderSize+(i-1)*TextLineHeight,
        X+Width-BorderSize, Y+i*TextLineHeight);
    outtextxy(X+BorderSize+1,
              Y+BorderSize+((i-1)*TextLineHeight)+(TextLineHeight DIV 2),
              MemoText[i]);
  end;
  with OldStyle do begin
    settextjustify(Horiz,Vert);
    settextstyle(Font,Direction,CharSize);
  end;
  Show_Cursor(xx,yy);
end;

(* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> *)

procedure MemoArea.Hilite(Xpos,Ypos:word);
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *          Method to highlight the memo area when the cursor enters   *
      *          it's area.                                                 *
      ***********************************************************************
      }

var
  i          : integer;
  OldStyle   : TextSettingsType;

begin
   IF (Xpos >= X)        AND
      (Xpos < X+Width)   AND
      (Ypos >= Y)        AND
      (Ypos < Y+Height)  THEN
   BEGIN
     if not(Hilited) then begin
       hide_cursor;
```

```
        Hilited := TRUE;
        GetTextSettings(OldStyle);
        settextstyle(Font,horizdir,1);
        setfillstyle(solidfill,hilitecolor);
        setcolor(hiliteTextcolor);
        settextjustify(lefttext,centertext);
        for i := 1 to NumLines do begin
          bar(X+BorderSize, Y+BorderSize+(i-1)*TextLineHeight,
              X+Width-BorderSize, Y+BorderSize+i*TextLineHeight);
          outtextxy(X+BorderSize+1,
                    Y+BorderSize+((i-1)*TextLineHeight)+(TextLineHeight DIV 2),
                    MemoText[i]);
        end;
        show_cursor(Xpos,Ypos);
        with OldStyle do begin
          settextjustify(Horiz,Vert);
          settextstyle(Font,Direction,CharSize);
        end;
      end;
    END
    ELSE
      IF Hilited THEN
      begin
        hilited := FALSE;
        Show;
      end;

end;

(* >◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇◇< *)

procedure MemoArea.HideTheCursor(LineNum:integer);
      {
      ************************************************************************
      *                                                                      *
      *       PURPOSE                                                        *
      *          Method to remove the text cursor from the indicated line    *
      *          of the memo area.                                           *
      *                                                                      *
      ************************************************************************
      }

begin
  IF Hilited THEN
     setcolor(HiliteColor)
  else
     setcolor(Color);
  line (X+BorderSize,Y+BorderSize+LineNum*TextLineHeight-1,
        X+Width-BorderSize,Y+BorderSize+LineNum*TextLineHeight-1);
```

```
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

procedure MemoArea.ShowTheCursor(LineNum,CursorPosition:integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to draw the text cursor at the appropriate line/       *
      *           character in the memo area.                                   *
      *                                                                         *
      ***************************************************************************
      }

var
  x1 : integer;

begin
  x1 := X + BorderSize + (CursorPosition * textwidth('1'));
  line (x1-textwidth('1'),
        Y+BorderSize+LineNum*TextLineHeight-1,
        x1,
        Y+BorderSize+LineNum*TextLineHeight-1);
end;

(* >><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><<><< *)

procedure MemoArea.DisplayText(NewText:MemoTextType;
                               LineNum,CursorPosition:integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to re-display the memo area text.                      *
      *                                                                         *
      ***************************************************************************
      }

var
  OldStyle   : TextSettingsType;

begin
  GetTextSettings(OldStyle);

  IF Hilited THEN begin
    setcolor(HiliteColor);
    setfillstyle(solidfill,HiliteColor);
  end
```

```
  else begin
    setcolor(Color);
    setfillstyle(solidfill,Color);
  end;

  settextstyle(Font,horizdir,1);
  settextjustify(lefttext,centertext);

(* hide the edit cursor and erase old text with the background color *)

  HideTheCursor(LineNum);

  if Display_String[LineNum] <> NewText[LineNum] then begin
    bar(X+BorderSize,Y+BorderSize+((LineNum-1)*TextLineHeight),
        X+Width-BorderSize,Y+BorderSize+((LineNum)*TextLineHeight));
    Display_String[LineNum] := NewText[LineNum];
  end;

  if Hilited then
    setcolor(HiliteTextColor)
  else
    setcolor(TextColor);

  outtextxy(X+BorderSize+1,
            Y+BorderSize+((LineNum-1)*TextLineHeight)+(TextLineHeight DIV 2),
            Display_String[LineNum]);

  ShowTheCursor(LineNum,CursorPosition);

  with OldStyle do begin
    settextjustify(Horiz,Vert);
    settextstyle(Font,Direction,CharSize);
  end;
end;

(* >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>< *)

function MemoArea.Action(Xpos,Ypos : word): boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to handle the editing of text in the memo area.     *
      *                                                                      *
      ************************************************************************
      }

const
  DEL      = 210;
```

File Name: MEMOAREA.PAS

```
  HOME_    = 198;
  END_     = 206;
  CTRL_END = 244;

var
  tmp              : string[1];
  TempText         : MemoTextType;
  CursorPosition,
  i,
  LineNum,
  key,
  button           : Integer;

begin
  Action := FALSE;
  if Control.Action(Xpos,Ypos) then begin
    Hide_Cursor;
    Action := true;
    for i := 1 to NumLines do begin
      Display_String[i] := MemoText[i];
      TempText[i]       := MemoText[i];
    end;
    CursorPosition := 1;
    LineNum        := 1;

    repeat
      DisplayText(TempText,LineNum,CursorPosition);

      key := 0;
      button := 0;

      while (key = 0) do begin
        get_key(key);
        button := get_buttons;
        if (button = left_button_pressed) then
          key := enter;
        if (button = right_button_pressed) then
          key := esc;
      end;

      case key of

        BACKSPACE  : begin
                       if (CursorPosition > 1) then begin
                         delete(TempText[LineNum], CursorPosition-1, 1);
                         dec(CursorPosition);
                       end
                       else begin
                         if (LineNum > 1) then begin
```

```
                        HideTheCursor(LineNum);
                        dec(LineNum);
                        CursorPosition := NumChars;
                        delete(TempText[LineNum], CursorPosition, 1);
                      end
                      else
                        beep;
                    end;
                  end;

DEL             : begin
                    if (TempText[LineNum][CursorPosition] <> '') then
                      delete(TempText[LineNum], CursorPosition, 1)
                    else
                      beep;
                  end;

HOME            : begin
                    CursorPosition := 1;
                  end;

END_            : begin
                    CursorPosition := length(TempText[LineNum]);
                    if CursorPosition = 0 then
                      CursorPosition := 1;
                  end;

CTRL_END        : begin
                    delete(TempText[LineNum],CursorPosition,NumChars);
                  end;

UP_ARROW        : begin
                    if LineNum > 1 then begin
                      HideTheCursor(LineNum);
                      dec(LineNum);
                    end
                    else
                      beep;
                  end;

DOWN_ARROW      : begin
                    if LineNum < NumLines then begin
                      HideTheCursor(LineNum);
                      inc(LineNum);
                    end
                    else
                      beep;
                  end;
```

File Name: MEMOAREA.PAS

```
        LEFT_ARROW : begin
                       if (CursorPosition > 1) then
                         dec(CursorPosition)
                       else
                         if (LineNum > 1) then begin
                           HideTheCursor(LineNum);
                           dec(LineNum);
                           CursorPosition := NumChars;
                         end
                         else beep;
                     end;

      RIGHT_ARROW : begin
                       if (CursorPosition < NumChars) then
                         inc(CursorPosition)
                       else
                         if (LineNum < NumLines) then begin
                           HideTheCursor(LineNum);
                           inc(LineNum);
                           CursorPosition := 1;
                         end
                         else beep;
                     end;

    else begin
      if (chr(key) >= ' ') and (chr(key) <= 'z') then begin
        tmp := chr(key);
        if CursorPosition > length(TempText[LineNum]) then
          for i := length(TempText[LineNum]) to CursorPosition do
            insert(' ',TempText[LineNum],CursorPosition);
        insert(tmp, TempText[LineNum],CursorPosition);
        if (CursorPosition < NumChars) then
          inc(CursorPosition)
        else begin
          DisplayText(TempText,LineNum,CursorPosition);
          HideTheCursor(LineNum);
          inc(LineNum);
          CursorPosition := 1;
        end;
      end;
    end;    (* of case statement *)
  end;
  until (( key = ENTER ) or ( key = ESC ));

  HideTheCursor(LineNum);

  if (key = ENTER) then begin
    for i := 1 to NumLines do
      MemoText[i] := TempText[i];
```

```
    end;

(* rewrite the memo area with the original text *)

    if (key = ESC) then
      for i := 1 to NumLines do begin
        DisplayText(MemoText,i,CursorPosition);
        HideTheCursor(i);
      end;


    Show_Cursor(Xpos,Ypos);
  end;
end;

(* ><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>< *)
```

File Name: PACE.PAS

```
{*****************************************************************
*                                                                *
*       PROGRAM NAME - Omega Performance Assessment              *
*                              and                               *
*                      Coverage Evaluation                       *
*                           (PACE)                               *
*                        Workstation                             *
*                                                                *
*         UNIT NAME - PACE PROGRAM                               *
*                                                                *
*****************************************************************
*                                                                *
*       This program was prepared by                             *
*                                                                *
*            The Analytic Science Corporation (TASC)             *
*                  55 Walkers Brook Drive                        *
*              Reading, Massachusetts 01867                      *
*                                                                *
*       PACE has been developed to run on a IBM PC/AT or compatible *
*       under MS-DOS 3.3 or higher with a minimum of 640K of main  *
*       memory and an EGA or compatible graphics adapter and color *
*       monitor. This work was performed under contract number     *
*       DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for  *
*       the Omega Navigation System Center (ONSCEN), Alexandria, VA. *
*                                                                *
*****************************************************************
*                                                                *
*       PURPOSE                                                  *
*          This file contains the main routine of the PACE workstation*
*          code. It handles the initialization of all PACE objects *
*          contexts and tab lists and contains the main action     *
*          dispatch loop.                                        *
*                                                                *
*****************************************************************}
{$N+,E+}
program Performance_Assessment_Coverage_Evaluator;

Uses PaceInit,Crt,CursrObj,Graph,ErrLog,CellUtil,QRUtil,WorldMap,
        Controls,ConMan,Paceobjs,Procs,TascLogo;

VAR
   grmode,grdriver:integer;
   X,Y,InitIndex:integer;
   button:integer;
   key:integer;
   first:word;
   ma:longint;
   avail:shortstring;
```

```
File Name: PACE.PAS


Begin

   Grdriver := Detect;
   Initgraph(grdriver,grmode,'');
   Setgraphmode(EGAHI);                (* 640 x 350 *)

   DisplayLogo;

(*
   ma := memavail;
   str(ma,avail);
   Log_Error('memory avail = '+avail);
   ma := maxavail;
   str(ma,avail);
   Log_Error('max avail = '+avail);
*)

   InitWorldMap;
   InitCellGrid;

   rightexpanded := FALSE; { start off with the left edit menu }
   leftexpanded := TRUE;




   SetActivePage(1);

   { base screen and controls }
   new(basescreen,Init(0,0,639,349,     blue, white,1));
   new(BIGmap,Init(37,20,564,278,       blue, white));
   new(BIGCellGrid,Init(37,20,564,278,BigCellGridAction,ShowCellData,
                        DoNothingProc, blue,      red,   lightgray, yellow));

   new(basescreen2,Init(0,0,639,349,    blue, white,1));
   new(BIGmap2,Init(37,20,564,278,      blue, lightgray));
   new(WeightCellGrid,Init(37,20,564,278,WeightsCellGridAction,
       DrawCells,WeightsCellGridCancel, blue,     red,    lightgray, yellow));
   new(CoverageCellGrid,Init(37,20,564,278,CoverageCellGridAction,
       ShowCoverageCellData,DoNothingProc, lightblue,     red,    lightgray, yellow));
   new(diffCellGrid,Init(37,20,564,278,DiffCellGridAction,
       ShowDiffCellData,DoNothingProc, lightblue,     red,    lightgray, yellow));

   new(h,Init(25,4,60,16,white,black,black,white,'Help',helpbutton,'H'));
   new(q,Init(105,4,60,16,white,black,black,white,'Quit',quitbutton,'Q'));
   new(f,Init(185,4,60,16,white,black,black,white,'File',filebutton,'F'));
   new(s,Init(265,4,60,16,white,black,black,white,'Split',splitbutton,'S'));
```

```
  new(o,Init(185,4,70,16,white,black,black,white,'Options',optionsbutton,'O'));

  { options file menu for the weights screen }
  optionsmenunamelist[1] := 'Group';
  optionsmenunamelist[2] := 'Region';
  optionsmenunamelist[3] := 'Load';
  optionsmenunamelist[4] := 'Save';
  optionsmenuhotkeylist[1] := 'G';
  optionsmenuhotkeylist[2] := 'R';
  optionsmenuhotkeylist[3] := 'L';
  optionsmenuhotkeylist[4] := 'S';
  optionsmenuprocedurelist[1] := WeightsGroupButton;
  optionsmenuprocedurelist[2] := Weight_or_Region_Button;
  optionsmenuprocedurelist[3] := WeightsLoadbutton;
  optionsmenuprocedurelist[4] := WeightsSaveButton;

  new(optionsmenu,Init(o^.X, o^.Y+o^.Height+5,4,optionsmenunamelist,
                       optionsmenuprocedurelist,optionsmenuhotkeylist));
  new(Weightslistmenu,Init(227,83,cyan,black,Weightslistmenuview,
                        Weightslistmenuaction,WEIGHTSPATH+'\','*.*'));

  new(r,Init(275,4,60,16,white,black,black,white,'Return',   returnbutton,'R'));
  new(cr,Init(185,4,60,16,white,black,black,white,'Return',   CoverageReturnButton,'R'));
  new(dr,Init(185,4,60,16,white,black,black,white,'Return',   DifferenceReturnButton,'R'));
  new(WeightsSaveMenuWindow,Init(110,145,418,100,cyan,black,white,black,
                          black,black,white,defaultfont,WeightsSaveAction,
                          ''));
  WeightsSaveMenuWindow^.Title := 'Weights/Region File Annotation';

  { split screens and controls }
  new(leftsplit,Init(0,21,319,328,lightblue,white,1));
  new(LEFTmap, Init(35,24,281,314,lightblue,white));
  new(LEFTCellGrid, Init(35,24,281,314,LeftCellGridAction,ShowCellData,DoNothingProc,blue,r
  new(rightsplit,Init(320,21,318,328,lightblue,white,1));
  new(RIGHTmap, Init(323,24,281,314,lightblue,white));
  new(RIGHTCellGrid, Init(323,24,281,314,RightCellGridAction,ShowCellData,DoNothingProc,blu
  new(leftexpand,Init(leftsplit^.X+105,leftsplit^.Y+4,60,16,white,black,
                      black,white,'Expand',leftexpandbutton,'E'));
  new(rightexpand,Init(rightsplit^.X+105,rightsplit^.Y+4,60,16,white,black,
                       black,white,'eXpand',rightexpandbutton,'X'));
  new(lf,Init(leftsplit^.X+25,leftsplit^.Y+4,60,16,white,black,black,white,
              'File',leftfilebutton,'F'));
  new(rf,Init(rightsplit^.X+25,rightsplit^.Y+4,60,16,white,black,black,
              white,'fIle',rightfilebutton,'I'));
  new(difference,Init(265,4,105,16,white,black,black,white,'Difference',differencebutton,'D
  new(difference2,Init(265,4,105,16,white,black,black,white,'Difference',differencebutton,'

  { file and auxiliary (right split) file menus }
  filemenunamelist[1] := 'Load';
```

File Name: PACE.PAS

```
   filemenunamelist[2] := 'Save';
   filemenunamelist[3] := 'Edit';
   filemenunamelist[4] := 'Batch';
   filemenuhotkeylist[1] := 'L';
   filemenuhotkeylist[2] := 'S';
   filemenuhotkeylist[3] := 'E';
   filemenuhotkeylist[4] := 'B';
   filemenuprocedurelist[1] := loadbutton;
   filemenuprocedurelist[2] := savebutton;
   filemenuprocedurelist[3] := editbutton;
   filemenuprocedurelist[4] := processbatchqueue;

   new(filemenu,     Init(f^.X,f^.Y+f^.Height+5,4,filemenunamelist,
                          filemenuprocedurelist,filemenuhotkeylist));
   new(leftfilemenu, Init(lf^.X+15,lf^.Y+lf^.Height+5,4,filemenunamelist,
                          filemenuprocedurelist,filemenuhotkeylist));

   filemenuprocedurelist[1] := auxloadbutton;
   filemenuprocedurelist[2] := auxsavebutton;
   filemenuprocedurelist[3] := auxeditbutton;
   filemenuprocedurelist[4] := auxprocessbatchqueue;

   new(rightfilemenu,Init(rf^.X,rf^.Y+rf^.Height+5,4,filemenunamelist,
                          filemenuprocedurelist,filemenuhotkeylist));


   { primary and auxiliary (left & right split) list menus }

   new(              listmenu,Init(227,83,cyan,black,listmenuview,
                                   listmenuaction,ARCHIVEPATH+'\','*.*'));
   new(        rightlistmenu,Init(227,83,cyan,black,rightlistmenuview,
                                  rightlistmenuaction,ARCHIVEPATH+'\','*.*'));

   { edit and auxiliary (right split) edit menus }

   new(   editmenu,Init(54,100,   ReliabilityAction,WeightAction,QueueAction,
                        editmenuprocess));
   new(auxeditmenu,Init(54,100,AuxReliabilityAction,AuxWeightAction,AuxQueueAction,
                        auxeditmenuprocess));

   new(   diffmenu,Init(180,25, differenceacceptbutton));

   new(   ReliabilityListMenu,Init(editmenu^.Reliability.X-5,
                                   editmenu^.Reliability.Y-175,
                                   cyan,black,reliabilitymenuview,
                                   RelListMenuAction,QRPATH+'\','*.*'));
   new(AuxReliabilityListMenu,Init(auxeditmenu^.Reliability.X+5,
                                   auxeditmenu^.Reliability.Y-175,
                                   cyan,black,auxreliabilitymenuview,
```

```
                                    AuxRelListMenuAction,QRPATH+'\','*.*'));

  new(leftlowerstatus,Init(3,296,309,49,black,white,editmenu));
  new(rightlowerstatus,Init(327,296,309,49,black,white,auxeditmenu));

  new(leftsidestatus,Init(3,52,30,244,black,white,editmenu));
  new(rightsidestatus,Init(606,52,30,244,black,white,auxeditmenu));

  new(CellPopUp,Init(150,21,172,271,lightgray,black,black,yellow,white,
                     SubCellAction,ShowCellData,red,editmenu));
  new(SubCellPopUp,Init(150,23,150,48,lightgray,black,black,yellow,CoverageAction));

  new(DiffCellPopUp,Init(150,21,170,60,lightgray,black,
                         black,yellow,white));



  new(SaveMenuWindow,Init(110,145,418,100,cyan,black,white,black,
                          black,black,white,defaultfont,SaveAction,
                          ''));
  new(AuxSaveMenuWindow,Init(110,145,418,100,cyan,black,white,black,
                          black,black,white,defaultfont,AuxSaveAction,
                          ''));
  new(QueueSaveMenuWindow,Init(110,145,418,100,cyan,black,white,black,
                          black,black,white,defaultfont,QueueSaveAction,
                          ''));
  new(AuxQueueSaveMenuWindow,Init(110,145,418,100,cyan,black,white,black,
                          black,black,white,defaultfont,AuxQueueSaveAction,
                          ''));
  new(HelpMenuWindow,Init(50,25,540,300,cyan,black,white,black,white,
                          defaultfont,NextAction,PreviousAction));

  new(stubwindow,init);

  SetActivePage(0);

  createcontext('basescreen');
  addtocontext('basescreen',basescreen);
  addtocontext('basescreen',h);
  addtocontext('basescreen',q);
  addtocontext('basescreen',f);
  addtocontext('basescreen',s);
  addtocontext('basescreen',leftlowerstatus);
  addtocontext('basescreen',leftsidestatus);
  addtocontext('basescreen',BIGCellGrid);
  addtocontext('basescreen',BIGmap);
  addtotablist('basescreen',h);
  addtotablist('basescreen',q);
  addtotablist('basescreen',f);
```

File Name: PACE.PAS

```
addtotablist('basescreen',s);

createcontext('filemenu');
addtocontext('filemenu',filemenu);
addtocontext('filemenu',h);
addtocontext('filemenu',q);
addtotablist('filemenu',@filemenu^.buttonarray[1]);
addtotablist('filemenu',@filemenu^.buttonarray[2]);
addtotablist('filemenu',@filemenu^.buttonarray[3]);
addtotablist('filemenu',@filemenu^.buttonarray[4]);
addtotablist('filemenu',h);
addtotablist('filemenu',q);

createcontext('leftfilemenu');
addtocontext('leftfilemenu',leftfilemenu);
addtocontext('leftfilemenu',h);
addtocontext('leftfilemenu',q);
addtotablist('leftfilemenu',@leftfilemenu^.buttonarray[1]);
addtotablist('leftfilemenu',@leftfilemenu^.buttonarray[2]);
addtotablist('leftfilemenu',@leftfilemenu^.buttonarray[3]);
addtotablist('leftfilemenu',@leftfilemenu^.buttonarray[4]);
addtotablist('leftfilemenu',h);
addtotablist('leftfilemenu',q);

createcontext('rightfilemenu');
addtocontext('rightfilemenu',rightfilemenu);
addtocontext('rightfilemenu',h);
addtocontext('rightfilemenu',q);
addtotablist('rightfilemenu',@rightfilemenu^.buttonarray[1]);
addtotablist('rightfilemenu',@rightfilemenu^.buttonarray[2]);
addtotablist('rightfilemenu',@rightfilemenu^.buttonarray[3]);
addtotablist('rightfilemenu',@rightfilemenu^.buttonarray[4]);
addtotablist('rightfilemenu',h);
addtotablist('rightfilemenu',q);

createcontext('splitscreen');
addtocontext('splitscreen',Leftsplit);
addtocontext('splitscreen',RightSplit);
addtocontext('splitscreen',h);
addtocontext('splitscreen',q);
addtocontext('splitscreen',difference);
addtocontext('splitscreen',leftlowerstatus);
addtocontext('splitscreen',rightlowerstatus);
addtocontext('splitscreen',leftsidestatus);
addtocontext('splitscreen',rightsidestatus);
addtocontext('splitscreen',leftexpand);
addtocontext('splitscreen',rightexpand);
addtocontext('splitscreen',lf);
addtocontext('splitscreen',rf);
```

File Name: PACE.PAS

```
   addtocontext('splitscreen',LEFTCellGrid);
   addtocontext('splitscreen',RIGHTCellGrid);
   addtocontext('splitscreen',LEFTmap);
   addtocontext('splitscreen',RIGHTmap);
   addtotablist('splitscreen',difference);
   addtotablist('splitscreen',lf);
   addtotablist('splitscreen',leftexpand);
   addtotablist('splitscreen',rf);
   addtotablist('splitscreen',rightexpand);
   addtotablist('splitscreen',h);
   addtotablist('splitscreen',q);

   createcontext('editmenu');
   addtocontext('editmenu',editmenu);
   addtocontext('editmenu',h);
   addtocontext('editmenu',q);
   addtotablist('editmenu',@editmenu^.Psa.value);
   addtotablist('editmenu',@editmenu^.SNR.value);
   addtotablist('editmenu',@editmenu^.ShortLongRatio.value);
   addtotablist('editmenu',@editmenu^.XAngle.value);
   addtotablist('editmenu',@editmenu^.PhaseDev.value);
   addtotablist('editmenu',@editmenu^.GDOP.value);
   addtotablist('editmenu',@editmenu^.DM.ButtonArray[1]);
   addtotablist('editmenu',@editmenu^.DM.ButtonArray[2]);
{
   addtotablist('editmenu',@editmenu^.PsaCalculateMode.buttonarray[1]);
   addtotablist('editmenu',@editmenu^.PsaCalculateMode.buttonarray[2]);
}
   addtotablist('editmenu',@editmenu^.PsaReportMode.buttonarray[1]);
   addtotablist('editmenu',@editmenu^.PsaReportMode.buttonarray[2]);
   addtotablist('editmenu',@editmenu^.PsaReportMode.buttonarray[3]);
   addtotablist('editmenu',@editmenu^.StationReliabilityModel.buttonarray[1]);
   addtotablist('editmenu',@editmenu^.StationReliabilityModel.buttonarray[2]);
   addtotablist('editmenu',@editmenu^.StationReliabilityModel.buttonarray[3]);
   FOR initindex := 1 TO 8 DO
   begin
      addtotablist('editmenu',@editmenu^.StationPower[initindex].Sld.Value);
      addtotablist('editmenu',@editmenu^.StationPower[initindex].Off);
      addtotablist('editmenu',@editmenu^.StationPower[initindex].On);
   end;
   addtotablist('editmenu',@editmenu^.Freq.buttonarray[1]);
   addtotablist('editmenu',@editmenu^.Freq.buttonarray[2]);
   addtotablist('editmenu',@editmenu^.Freq.buttonarray[3]);
   addtotablist('editmenu',@editmenu^.Freq.buttonarray[4]);
   FOR initindex := 1 TO 12 DO
      addtotablist('editmenu',@editmenu^.MonthSelectors[initindex]);
   FOR initindex := 1 TO 24 DO
      addtotablist('editmenu',@editmenu^.HourSelectors[initindex]);
   addtotablist('editmenu',@editmenu^.Reliability);
```

```
   addtotablist('editmenu',@editmenu^.Weights);
   addtotablist('editmenu',@editmenu^.Batch);
   addtotablist('editmenu',@editmenu^.Process);
   addtotablist('editmenu',h);
   addtotablist('editmenu',q);

   createcontext('auxeditmenu');
   addtocontext('auxeditmenu',auxeditmenu);
   addtocontext('auxeditmenu',h);
   addtocontext('auxeditmenu',q);
   addtotablist('auxeditmenu',@auxeditmenu^.Psa.value);
   addtotablist('auxeditmenu',@auxeditmenu^.SNR.value);
   addtotablist('auxeditmenu',@auxeditmenu^.ShortLongRatio.value);
   addtotablist('auxeditmenu',@auxeditmenu^.XAngle.value);
   addtotablist('auxeditmenu',@auxeditmenu^.PhaseDev.value);
   addtotablist('auxeditmenu',@auxeditmenu^.GDOP.value);
   addtotablist('auxeditmenu',@auxeditmenu^.DM.ButtonArray[1]);
   addtotablist('auxeditmenu',@auxeditmenu^.DM.ButtonArray[2]);
{
   addtotablist('auxeditmenu',@auxeditmenu^.PsaCalculateMode.buttonarray[1]);
   addtotablist('auxeditmenu',@auxeditmenu^.PsaCalculateMode.buttonarray[2]);
}
   addtotablist('auxeditmenu',@auxeditmenu^.PsaReportMode.buttonarray[1]);
   addtotablist('auxeditmenu',@auxeditmenu^.PsaReportMode.buttonarray[2]);
   addtotablist('auxeditmenu',@auxeditmenu^.PsaReportMode.buttonarray[3]);
   addtotablist('auxeditmenu',@auxeditmenu^.StationReliabilityModel.buttonarray[1]);
   addtotablist('auxeditmenu',@auxeditmenu^.StationReliabilityModel.buttonarray[2]);
   addtotablist('auxeditmenu',@auxeditmenu^.StationReliabilityModel.buttonarray[3]);
   FOR initindex := 1 TO 8 DO
   begin
      addtotablist('auxeditmenu',@auxeditmenu^.StationPower[initindex].Sld.Value);
      addtotablist('auxeditmenu',@auxeditmenu^.StationPower[initindex].Off);
      addtotablist('auxeditmenu',@auxeditmenu^.StationPower[initindex].On);
   end;
   addtotablist('auxeditmenu',@auxeditmenu^.Freq.buttonarray[1]);
   addtotablist('auxeditmenu',@auxeditmenu^.Freq.buttonarray[2]);
   addtotablist('auxeditmenu',@auxeditmenu^.Freq.buttonarray[3]);
   addtotablist('auxeditmenu',@auxeditmenu^.Freq.buttonarray[4]);
   FOR initindex := 1 TO 12 DO
      addtotablist('auxeditmenu',@auxeditmenu^.MonthSelectors[initindex]);
   FOR initindex := 1 TO 24 DO
      addtotablist('auxeditmenu',@auxeditmenu^.HourSelectors[initindex]);
   addtotablist('auxeditmenu',@auxeditmenu^.Reliability);
   addtotablist('auxeditmenu',@auxeditmenu^.Weights);
   addtotablist('auxeditmenu',@auxeditmenu^.Batch);
   addtotablist('auxeditmenu',@auxeditmenu^.Process);
   addtotablist('auxeditmenu',h);
   addtotablist('auxeditmenu',q);
```

```
createcontext('diffmenu');
addtocontext('diffmenu',diffmenu);
addtocontext('diffmenu',h);
addtocontext('diffmenu',q);
addtotablist('diffmenu',@diffmenu^.lothresh.value);
addtotablist('diffmenu',@diffmenu^.hithresh.value);
addtotablist('diffmenu',@diffmenu^.DiffMode.buttonarray[1]);
addtotablist('diffmenu',@diffmenu^.DiffMode.buttonarray[2]);
addtotablist('diffmenu',@diffmenu^.DiffMode.buttonarray[3]);
addtotablist('diffmenu',@diffmenu^.Accept);
addtotablist('diffmenu',h);
addtotablist('diffmenu',q);

createcontext('listmenu');
addtocontext('listmenu',listmenu);
addtocontext('listmenu',h);
addtocontext('listmenu',q);
addtotablist('listmenu',@listmenu^.DirectoryMaskButton);
FOR initindex := 1 TO maxlistsize DO
   addtotablist('listmenu',@listmenu^.directorylist.buttonarray[initindex]);
addtotablist('listmenu',@listmenu^.VBar.Up);
addtotablist('listmenu',@listmenu^.VBar.Down);
addtotablist('listmenu',@listmenu^.ViewButton);
addtotablist('listmenu',@listmenu^.LoadButton);
addtotablist('listmenu',h);
addtotablist('listmenu',q);

createcontext('rightlistmenu');
addtocontext('rightlistmenu',rightlistmenu);
addtocontext('rightlistmenu',h);
addtocontext('rightlistmenu',q);
addtotablist('rightlistmenu',@rightlistmenu^.DirectoryMaskButton);
FOR initindex := 1 TO maxlistsize DO
   addtotablist('rightlistmenu',@rightlistmenu^.directorylist.buttonarray[initindex]);
addtotablist('rightlistmenu',@rightlistmenu^.VBar.Up);
addtotablist('rightlistmenu',@rightlistmenu^.VBar.Down);
addtotablist('rightlistmenu',@rightlistmenu^.ViewButton);
addtotablist('rightlistmenu',@rightlistmenu^.LoadButton);
addtotablist('rightlistmenu',h);
addtotablist('rightlistmenu',q);

createcontext('reliabilitylistmenu');
addtocontext('reliabilitylistmenu',reliabilitylistmenu);
addtocontext('reliabilitylistmenu',h);
addtocontext('reliabilitylistmenu',q);
addtotablist('reliabilitylistmenu',@reliabilitylistmenu^.DirectoryMaskButton);
FOR initindex := 1 TO maxlistsize DO
   addtotablist('reliabilitylistmenu',@reliabilitylistmenu^.directorylist.buttonarray[ini
addtotablist('reliabilitylistmenu',@reliabilitylistmenu^.VBar.Up);
```

```
   addtotablist('reliabilitylistmenu',@reliabilitylistmenu^.VBar.Down);
   addtotablist('reliabilitylistmenu',@reliabilitylistmenu^.ViewButton);
   addtotablist('reliabilitylistmenu',@reliabilitylistmenu^.LoadButton);
   addtotablist('reliabilitylistmenu',h);
   addtotablist('reliabilitylistmenu',q);

   createcontext('auxreliabilitylistmenu');
   addtocontext('auxreliabilitylistmenu',auxreliabilitylistmenu);
   addtocontext('auxreliabilitylistmenu',h);
   addtocontext('auxreliabilitylistmenu',q);
   addtotablist('auxreliabilitylistmenu',@auxreliabilitylistmenu^.DirectoryMaskButton);
   FOR initindex := 1 TO maxlistsize DO
      addtotablist('auxreliabilitylistmenu',@auxreliabilitylistmenu^.directorylist.buttonarr
   addtotablist('auxreliabilitylistmenu',@auxreliabilitylistmenu^.VBar.Up);
   addtotablist('auxreliabilitylistmenu',@auxreliabilitylistmenu^.VBar.Down);
   addtotablist('auxreliabilitylistmenu',@auxreliabilitylistmenu^.ViewButton);
   addtotablist('auxreliabilitylistmenu',@auxreliabilitylistmenu^.LoadButton);
   addtotablist('auxreliabilitylistmenu',h);
   addtotablist('auxreliabilitylistmenu',q);

   createcontext('weightslistmenu');
   addtocontext('weightslistmenu',weightslistmenu);
   addtocontext('weightslistmenu',h);
   addtocontext('weightslistmenu',q);
   addtotablist('weightslistmenu',@weightslistmenu^.DirectoryMaskButton);
   FOR initindex := 1 TO maxlistsize DO
      addtotablist('weightslistmenu',@weightslistmenu^.directorylist.buttonarray[initindex])
   addtotablist('weightslistmenu',@weightslistmenu^.VBar.Up);
   addtotablist('weightslistmenu',@weightslistmenu^.VBar.Down);
   addtotablist('weightslistmenu',@weightslistmenu^.ViewButton);
   addtotablist('weightslistmenu',@weightslistmenu^.LoadButton);
   addtotablist('weightslistmenu',h);
   addtotablist('weightslistmenu',q);

   createcontext('weightscreen');
   addtocontext('weightscreen',basescreen2);
   addtocontext('weightscreen',h);
   addtocontext('weightscreen',q);
   addtocontext('weightscreen',o);
   addtocontext('weightscreen',r);
   addtocontext('weightscreen',WeightCellGrid);
   addtocontext('weightscreen',BIGmap2);
   addtotablist('weightscreen',h);
   addtotablist('weightscreen',q);
   addtotablist('weightscreen',o);
   addtotablist('weightscreen',r);

   createcontext('diffscreen');
   addtocontext('diffscreen',basescreen2);
```

```
addtocontext('diffscreen',h);
addtocontext('diffscreen',q);
addtocontext('diffscreen',dr);
addtocontext('diffscreen',difference2);
addtocontext('diffscreen',leftlowerstatus);
addtocontext('diffscreen',rightlowerstatus);
addtocontext('diffscreen',leftsidestatus);
addtocontext('diffscreen',rightsidestatus);
addtocontext('diffscreen',DiffCellGrid);
addtocontext('diffscreen',BIGmap2);
addtotablist('diffscreen',h);
addtotablist('diffscreen',q);
addtotablist('diffscreen',dr);
addtotablist('diffscreen',difference2);

createcontext('coveragescreen');
addtocontext('coveragescreen',basescreen2);
addtocontext('coveragescreen',h);
addtocontext('coveragescreen',q);
addtocontext('coveragescreen',cr);
addtocontext('coveragescreen',CoverageCellGrid);
addtocontext('coveragescreen',BIGmap2);
addtotablist('coveragescreen',h);
addtotablist('coveragescreen',q);
addtotablist('coveragescreen',cr);

createcontext('optionsmenu');
addtocontext('optionsmenu',optionsmenu);
addtocontext('optionsmenu',h);
addtocontext('optionsmenu',q);
addtotablist('optionsmenu',@optionsmenu^.buttonarray[1]);
addtotablist('optionsmenu',@optionsmenu^.buttonarray[2]);
addtotablist('optionsmenu',@optionsmenu^.buttonarray[3]);
addtotablist('optionsmenu',@optionsmenu^.buttonarray[4]);
addtotablist('optionsmenu',h);
addtotablist('optionsmenu',q);

createcontext('weightsavemenuwindow');
addtocontext('weightsavemenuwindow',WeightsSaveMenuWindow);
addtocontext('weightsavemenuwindow',h);
addtocontext('weightsavemenuwindow',q);
addtotablist('weightsavemenuwindow',@weightssavemenuwindow^.MemoBox);
addtotablist('weightsavemenuwindow',@weightssavemenuwindow^.FileButton);
addtotablist('weightsavemenuwindow',@weightssavemenuwindow^.SaveButton);
addtotablist('weightsavemenuwindow',h);
addtotablist('weightsavemenuwindow',q);

createcontext('CellPopUp');
addtocontext('CellPopUp',CellPopUp);
```

```
addtocontext('CellPopUp',h);
addtocontext('CellPopUp',q);
addtotablist('CellPopUp',h);
addtotablist('CellPopUp',q);

createcontext('DiffCellPopUp');
addtocontext('DiffCellPopUp',DiffCellPopUp);
addtocontext('DiffCellPopUp',h);
addtocontext('DiffCellPopUp',q);
addtotablist('DiffCellPopUp',h);
addtotablist('DiffCellPopUp',q);

createcontext('SubCellPopUp');
addtocontext('SubCellPopUp',SubCellPopUp);
addtocontext('SubCellPopUp',h);
addtocontext('SubCellPopUp',q);
FOR initindex := 1 TO 8 DO
   addtotablist('SubCellPopUp',@subCellPopUp^.Stations.ButtonArray[initindex]);
addtotablist('SubCellPopUp',h);
addtotablist('SubCellPopUp',q);

createcontext('savemenuwindow');
addtocontext('savemenuwindow',SaveMenuWindow);
addtocontext('savemenuwindow',h);
addtocontext('savemenuwindow',q);
addtotablist('savemenuwindow',@savemenuwindow^.MemoBox);
addtotablist('savemenuwindow',@savemenuwindow^.FileButton);
addtotablist('savemenuwindow',@savemenuwindow^.SaveButton);
addtotablist('savemenuwindow',h);
addtotablist('savemenuwindow',q);

createcontext('auxsavemenuwindow');
addtocontext('auxsavemenuwindow',AuxSaveMenuWindow);
addtocontext('auxsavemenuwindow',h);
addtocontext('auxsavemenuwindow',q);
addtotablist('auxsavemenuwindow',@auxsavemenuwindow^.MemoBox);
addtotablist('auxsavemenuwindow',@auxsavemenuwindow^.FileButton);
addtotablist('auxsavemenuwindow',@auxsavemenuwindow^.SaveButton);
addtotablist('auxsavemenuwindow',h);
addtotablist('auxsavemenuwindow',q);

createcontext('queuemenuwindow');
addtocontext('queuemenuwindow',queuesaveMenuWindow);
addtocontext('queuemenuwindow',h);
addtocontext('queuemenuwindow',q);
addtotablist('queuemenuwindow',@queuesavemenuwindow^.MemoBox);
addtotablist('queuemenuwindow',@queuesavemenuwindow^.FileButton);
addtotablist('queuemenuwindow',@queuesavemenuwindow^.saveButton);
addtotablist('queuemenuwindow',h);
```

```
   addtotablist('queuemenuwindow',q);

   createcontext('auxqueuemenuwindow');
   addtocontext('auxqueuemenuwindow',auxqueuesaveMenuWindow);
   addtocontext('auxqueuemenuwindow',h);
   addtocontext('auxqueuemenuwindow',q);
   addtotablist('auxqueuemenuwindow',@auxqueuesavemenuwindow^.MemoBox);
   addtotablist('auxqueuemenuwindow',@auxqueuesavemenuwindow^.FileButton);
   addtotablist('auxqueuemenuwindow',@auxqueuesavemenuwindow^.saveButton);
   addtotablist('auxqueuemenuwindow',h);
   addtotablist('auxqueuemenuwindow',q);

   createcontext('helpmenuwindow');
   addtocontext('helpmenuwindow',HelpMenuWindow);
   addtocontext('helpmenuwindow',q);
   addtotablist('helpmenuwindow',@helpmenuwindow^.NextButton);
   addtotablist('helpmenuwindow',@helpmenuwindow^.PrevButton);
   addtotablist('helpmenuwindow',@helpmenuwindow^.GoToButton);
   addtotablist('helpmenuwindow',q);

   createcontext('stub');
   addtocontext('stub',stubwindow);

(*
   ma := memavail;
   str(ma,avail);
   Log_Error('memory avail = '+avail);
   ma := maxavail;
   str(ma,avail);
   Log_Error('max avail = '+avail);
*)

   pushcontext('basescreen');

   ShowContext;



   show_cursor(300,200);
   QUIT := FALSE;
   repeat
   begin
      Get_Key(Key);
      get_cursorXY(X,Y);
      IF (Key IN _Cursor_set) THEN
         Move_Cursor(Key, X, Y);
      HiLiteContext(X,Y);
      button := get_buttons;
      IF (button = left_button_pressed) OR (key = ENTER) THEN
```

```
         ActionContext(X,Y);
      IF (button = right_button_pressed) OR (key = ESC) THEN
         CancelContext;
      if (key <> 0) then
      begin                          { key has been pressed... }
         CheckMemory(key);
         KeyActionContext(X, Y, key);
      end;
   end
   until QUIT;
   restorecrtmode;
end.
```

File Name: PACEINIT.PAS

```
{*************************************************************************
*                                                                        *
*       PROGRAM NAME - Omega Performance Assessment                      *
*                                and                                     *
*                       Coverage Evaluation                              *
*                            (PACE)                                      *
*                         Workstation                                    *
*                                                                        *
*          UNIT NAME - PACEINIT                                          *
*                                                                        *
*************************************************************************
*                                                                        *
*       This program was prepared by                                     *
*                                                                        *
*             The Analytic Science Corporation (TASC)                    *
*                   55 Walkers Brook Drive                               *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*       PACE has been developed to run on a IBM PC/AT or compatible      *
*       under MS-DOS 3.3 or higher with a minimum of 640K of main        *
*       memory and an EGA or compatible graphics adapter and color       *
*       monitor. This work was performed under contract number           *
*       DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for        *
*       the Omega Navigation System Center (ONSCEN), Alexandria, VA.     *
*                                                                        *
*************************************************************************
*                                                                        *
*       PURPOSE                                                          *
*          This file contains the procedures to initialize the PACE      *
*          operational environment including the display device          *
*          initialization and the internal PACE database directory       *
*          paths.                                                        *
*************************************************************************}
Unit PaceInit;

Interface

  Uses Crt, Dos, Grapn, CursrObj, QRUtil, TascLogo, ErrLog;

CONST
   GDOPDATABASE = 'GDOP.GDP';

VAR
   HELPPATH,
   PACEPATH,
   ARCHIVEPATH,
   DEFAULTQRFILE,
   WEIGHTSPATH,
   DEFAULTWEIGHTSFILE,
```

File Name: PACEINIT.PAS

```
   DATABASEPATH,
   DATABASE102,
   DATABASE136,
   CONTEXTFILE,
   HELPFILE            : STRING;


Implementation

  const
    InitFileName = 'INITPACE.TXT';

(*****************************************************************************)

  procedure ReadInitFile;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Initialization routine to read the PACE setup file and        *
      *           assign the appropriate directory paths to the PACE            *
      *           internal variables.                                           *
      ***************************************************************************
      }

  var
    F : text;
    tmp  : string;

  begin
    HELPPATH            := '';
    PACEPATH            := '';
    ARCHIVEPATH         := '';
    QRPATH              := '';
    WEIGHTSPATH         := '';
    DATABASEPATH        := '';
    DEFAULTQRFILE       := '';
    DEFAULTWEIGHTSFILE  := '';
    DATABASE102         := '';
    DATABASE136         := '';
    CONTEXTFILE         := '';
    HELPFILE            := '';

    assign(F,InitFileName);
    reset(F);

    readln(F,tmp);
      PACEPATH := PACEPATH + tmp;           (* read the drive letter *)
    readln(F,tmp);
```

```
      PACEPATH := PACEPATH + tmp;              (* get the path *)

    readln(F,tmp);
      ARCHIVEPATH := ARCHIVEPATH + tmp;        (* read the drive letter *)
    readln(F,tmp);
      ARCHIVEPATH := ARCHIVEPATH + tmp;        (* get the path *)

    readln(F,tmp);
      QRPATH := QRPATH + tmp;                  (* read the drive letter *)
    readln(F,tmp);
      QRPATH := QRPATH + tmp;                  (* get the path *)

    readln(F,tmp);
      WEIGHTSPATH := WEIGHTSPATH + tmp;    (* read the drive letter *)
    readln(F,tmp);
      WEIGHTSPATH := WEIGHTSPATH + tmp;    (* get the path *)

    readln(F,tmp);
      DATABASEPATH := DATABASEPATH + tmp;    (* read the drive letter *)
    readln(F,tmp);
      DATABASEPATH := DATABASEPATH + tmp;    (* get the path *)

    readln(F,tmp);
      HELPPATH := HELPPATH + tmp;    (* read the drive letter *)
    readln(F,tmp);
      HELPPATH := HELPPATH + tmp;    (* get the path *)

    readln(F,DEFAULTQRFILE);
    readln(F,DEFAULTWEIGHTSFILE);
    readln(F,DATABASE102);
    readln(F,DATABASE136);
    readln(F,CONTEXTFILE);
    readln(F,HELPFILE);
    close(F);
  end;

(*****************************************************************************)

  procedure PaceInitialization;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Initialization routine to set up the graphics device driver*
      *           and check for the existance of the various PACE directories*
      ***************************************************************************
      }

  var
```

File Name: PACEINIT.PAS

```
    DirInfo          : SearchRec;  (* as defined in the DOS UNIT *)
    CurrentDrive     : string[3];
    DriverString,
    CurrentPath,
    errorstring      : string;
    PaceInitProblem  : boolean;
    TotalDiskSpace,
    FreeDiskSpace,
    ver              : longint;
    key,
    button,
    GraphDriver,
    GraphMode        : integer;

  begin
    PaceInitProblem := FALSE;
(*
    clrscr;
    writeln;
    writeln('/************************/');
    writeln('/  PACE INITIALIZATION  /');
    writeln('/************************/');
    writeln;
    ver := DosVersion;
    writeln('DOS Version: ',Lo(ver),'.',Hi(Ver)); writeln;
    GetDir(0,CurrentPath);
    CurrentDrive := CurrentPath;
                                        { 0 indicates the default drive }
    TotalDiskSpace := DiskSize(0);
    FreeDiskSpace := DiskFree(0);
    writeln(CurrentDrive+' is ',TotalDiskSpace,' bytes in size'); writeln;
    writeln(CurrentDrive+' has ', FreeDiskSpace,' bytes of available disk space'); writeln;

    DetectGraph(GraphDriver,GraphMode);
    case GraphDriver of
      CGA      : DriverString := 'CGA';
      MCGA     : DriverString := 'MCGA';
      EGA      : DriverString := 'EGA';
      EGA64    : DriverString := 'EGA64';
      EGAMono  : DriverString := 'EGAMono';
      IBM8514  : DriverString := 'IBM8514';
      HercMono : DriverString := 'HercMono';
      ATT400   : DriverString := 'ATT400';
      VGA      : DriverString := 'VGA';
      PC3270   : DriverString := 'PC3270';
    else
      begin
        beep;
        DriverString := '<<< Unknown >>>';
```

```
      end;
    end;
    writeln(DriverString + ' graphics hardware detected'); writeln;
*)

    findfirst(InitFileName, AnyFile, DirInfo);
    if (DosError <> 0) then
    begin
      errorstring := 'File not found...<<'+InitFileName+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;

    if not(PaceInitProblem) then begin
      ReadInitFile;

      findfirst(ARCHIVEPATH, Directory, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'Path not found...<<'+ARCHIVEPATH+'>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst(QRPATH, Directory, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'Path not found...<<'+QRPATH+'>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst(WEIGHTSPATH, Directory, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'Path not found...<<'+WEIGHTSPATH+'>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst(DATABASEPATH, Directory, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'Path not found...<<'+DATABASEPATH+'>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst(HELPPATH, Directory, DirInfo);
```

File Name: PACEINIT.PAS

```
    if (DosError <> 0) then
    begin
      errorstring := 'Path not found...<<'+HELPPATH+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;

    findfirst(QRPATH+'\'+DEFAULTQRFILE ,Anyfile, DirInfo);
    if (DosError <> 0) then
    begin
      errorstring := 'File not found...<<'+QRPATH+'\'+DEFAULTQRFILE+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;


    findfirst(WEIGHTSPATH+'\'+DEFAULTWEIGHTSFILE, Anyfile, DirInfo);
    if (DosError <> 0) then
    begin
      errorstring := 'File not found...<<'+WEIGHTSPATH+'\'+DEFAULTWEIGHTSFILE+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;


    findfirst(DATABASEPATH+'\'+DATABASE102, Anyfile, DirInfo);
    if (DosError <> 0) then
    begin
      errorstring := 'File not found...<<'+DATABASEPATH+'\'+DATABASE102+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;

    findfirst(DATABASEPATH+'\'+DATABASE136, Anyfile, DirInfo);
    if (DosError <> 0) then
    begin
      errorstring := 'File not found...<<'+DATABASEPATH+'\'+DATABASE136+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;

    findfirst(HELPPATH+'\'+CONTEXTFILE, Anyfile, DirInfo);
    if (DosError <> 0) then
    begin
      errorstring := 'File not found...<<'+HELPPATH+'\'+CONTEXTFILE+'>>';
      writeln(errorstring);
      PaceInitProblem := TRUE;
    end;
```

File Name: PACEINIT.PAS

```
      findfirst(HELPPATH+'\'+HELPFILE, Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<'+HELPPATH+'\'+HELPFILE+'>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst('EGAVGA.BGI', Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<EGAVGA.BGI>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst('GRID_10.DEF', Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<GRID_10.DEF>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst('LITT.CHR', Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<LITT.CHR>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst(LOGOFILE, Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<'+LOGOFILE+'>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
    end;

      findfirst('WORLD.MAP', Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<WORLD.MAP>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst('GDOP.GDP', Anyfile, DirInfo);
```

```
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<GDOP.GDP>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

      findfirst('TRIP.CHR', Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<TRIP.CHR>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;


      findfirst('TASCLOGO.VEC', Anyfile, DirInfo);
      if (DosError <> 0) then
      begin
        errorstring := 'File not found...<<TASCLOGO.VEC>>';
        writeln(errorstring);
        PaceInitProblem := TRUE;
      end;

    end;

    if PaceInitProblem then
    begin
      writeln('/*********************************/');
      writeln('/  PROBLEM IN PACE INITIALIZATION  /');
      writeln('/*********************************/');
      writeln;
      beep;
      halt(0);
(*
    end
    else
    begin
      writeln('/*******************************/');
      writeln('/  END OF PACE INITIALIZATION  /');
      writeln('/*******************************/');
      writeln;
*)
    end;
(*
    writeln('            <Press any key to Continue>');

    repeat
       button := get_buttons;
```

File Name: PACEINIT.PAS

```
      Get_key(key);
    until (key <> 0) or (button = left_button_pressed) or
          (button = right_button_pressed);
*)
  end;

(*****************************************************************************)

begin
  PaceInitialization;
end.
```

File Name: PACEOBJS.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME - PACEOBJS                                         *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                     55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This unit contains the declatation and methods for all of    *
*           the objects used within PACE. The declarations for the       *
*           objects whose methods are contained herein are found in      *
*           the defs.pas program file.                                   *
*************************************************************************}
{$N+,E+}
UNIT PaceObjs;

Interface

Uses    CRT,Dos,Graph,ErrLog,Controls,ConMan,
          cursrobj,Cellutil,WorldMap,Paceinit,datautil,QRutil;

  procedure checkmemory(key: integer);

{$I defs.pas}

TYPE
     CHLPtr = ^CHL;                    (* Context Help List *)
     CHL = Record
             Next        : CHLPtr;
```

File Name: PACEOBJS.PAS

```
            ContextStr;
            filename    : shortstring;
          end;

    HFLPtr = ^HFL;                    (* Help File List *)
    HFL = Record
            Next,Prev   : HFLPtr;
            filename    : shortstring;
          end;

VAR
   ContextHelpList : CHLPtr;
   HelpFilesList   : HFLPtr;

Implementation


{$I checkmem.pas}


CONST
   stations: array[0..7,0..1] of integer =
            ((6642,1313),
             (630,-1066),
             (2140,-15783),
             (4637,-9834),
             (-2097,5529),
             (-4305,-6519),
             (-3848,14694),
             (3461,12945));



VAR

   savedparams: savedparamsrecordptr;
   rightexpanded: boolean;
   currenteditcontext:shortstring;
   LastLatLonX, LastLatLonY: Integer;

   procedure jingle;
      {
      ********************************************************************
      *                                                                  *
      *        PURPOSE                                                   *
      *          Utility routine to produce a jingle sound for debugging *
      *          purposes.                                               *
      ********************************************************************
      }
```

```
   begin
      sound(1000);
      delay(50);
      nosound;
      sound(1500);
      delay(50);
      nosound;
   end;

   procedure messagewait;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility routine that waits for a user keypress. Usefull       *
      *           when waiting for the user to acknowledge a displayed message
      ***************************************************************************
      }
   VAR
    dummy: char;
   begin
    repeat
    until keypressed OR (get_buttons <> 0);
    IF keypressed THEN dummy := readkey;
   end;


(***************************************************************************)


   Constructor Cell.Init(InitX,InitY,InitWidth,InitHeight,InitLat,InitLon,
                         InitCellNumber: integer;
                         InitAboveColor,InitBelowColor,InitBorderColor,
                         InitBorderHiliteColor:word;
                         InitActionProc : CellActionProcedure;
                         InitShowProc : CellShowProcedure);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to initialize the basic 10 by 10 degree cell object.*
      *                                                                         *
      ***************************************************************************
      }
   begin
      Control.Init(InitX,InitY,InitWidth,InitHeight,InitAboveColor);
      AboveColor := InitAboveColor;
      BelowColor := InitBelowColor;
      AboveThreshold := TRUE;
```

```
    Region := False;
    Hilited := False;
    ActionProc := InitActionProc;
    ShowProc   := InitShowProc;
    CellNumber := InitCellNumber;
    Weight_or_Coverage := 1;
    Pac := 0.0;
    BorderColor := InitBorderColor;
    BorderHiliteColor := InitBorderHiliteColor;
    lat := initlat;
    lon := initlon;
end;

Procedure Cell.Hilite(Xpos,Ypos:word);
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *          Method to highlight a cell in the cell grid object used for*
    *          all PACE cell displays.                                             *
    ********************************************************************************
    }

begin
    Hilited := TRUE;
    hide_cursor;

{ Imagesize returns 0 if > 64K }
    IF imagesize(X-2,Y-2,X+Width+2,Y+Height+2) <> 0 THEN
    begin
       getmem(BitImg,imagesize(X-2,Y-2,X+Width+2,Y+Height+2));
       getimage(X-2,Y-2,X+Width+2,Y+Height+2,BitImg^);
    end
    else
    begin
       getmem(BitImg,imagesize(X-2,Y-2,X+Width+2,(Y+Height+2)DIV 2));
       getimage(X-2,Y-2,X+Width+2,(Y+Height+2)DIV 2,BitImg^);
       getmem(AuxBitImg,imagesize(X-2,(Y+Height+2)DIV 2,X+Width+2,Y+Height+2));
       getimage(X-2,(Y+Height+2)DIV 2,X+Width+2,Y+Height+2,AuxBitImg^);
    end;

    setcolor(BorderHiliteColor);
    rectangle(X,Y,X+Width,Y+Height);
    rectangle(X-1,Y-1,X+Width+1,Y+Height+1);
    rectangle(X-2,Y-2,X+Width+2,Y+Height+2);
    IF region THEN
      rectangle(X+1,Y+1,X+Width-1,Y+Height-1);
    show_cursor(Xpos,Ypos);
end;
```

File Name: PACEOBJS.PAS

```
Procedure Cell.UnHilite;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Method to unhighlight a cell in the cell grid object.         *
   ***************************************************************************
   }

var
  xx,yy : integer;

begin
    HILITED := FALSE;
    Get_CursorXY(xx,yy);
    Hide_Cursor;
      { Imagesize returns 0 if > 64K }
    IF imagesize(X-2,Y-2,X+Width+2,Y+Height+2) <> 0 THEN
    begin
       putimage(X-2,Y-2,BitImg^,CopyPut);
       freemem(BitImg,imagesize(X-2,Y-2,X+Width+2,Y+Height+2));
    end
    else
    begin
       putimage(X-2,Y-2,BitImg^,CopyPut);
       putimage(X-2,(Y+Height+2)DIV 2,AuxBitImg^,CopyPut);
       freemem(BitImg,imagesize(X-2,Y-2,X+Width+2,(Y+Height+2)DIV 2));
       freemem(AuxBitImg,imagesize(X-2,(Y+Height+2)DIV 2,X+Width+2,Y+Height+2));
    end;
    Show_Cursor(xx,yy);
end;


Procedure Cell.Show;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Method that causes a cell to display itself by calling        *
   *           the objects display process.                                  *
   ***************************************************************************
   }
begin
   ShowProc(@self);
end;

Procedure Cell.ShowRegion;
   {
```

File Name: PACEOBJS.PAS

```
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method that draws a colored border around any cells that      *
      *           are part of a user defined region.                            *
      ***************************************************************************
      }
   var xx,yy : integer;
   begin
     Get_CursorXY(xx,yy);
     Hide_Cursor;
     setcolor(lightcyan);
{     rectangle(x,y,x+width,y+Height);}
     rectangle(X+1,Y+1,X+Width-1,Y+Height-1);
     Show_Cursor(xx,yy);
   end;

   Procedure Cell.HideRegion;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method that removes the colored border around any cells       *
      *           that are part of a user defined region.                       *
      ***************************************************************************
      }
   var xx,yy : integer;
   begin
     Get_CursorXY(xx,yy);
     Hide_Cursor;
{     setcolor(bordercolor);
     rectangle(x,y,x+width,y+Height);}
     setcolor(abovecolor);
     rectangle(X+1,Y+1,X+Width-1,Y+Height-1);
     Show_Cursor(xx,yy);
   end;

   Function Cell.IsAboveThreshold : boolean;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility function to have a cell check itself against the      *
      *           user selected threshold.                                      *
      ***************************************************************************
      }
   begin
      IsAboveThreshold := AboveThreshold;
   end;
```

File Name: PACEOBJS.PAS

```
Function Cell.Action(Xpos,Ypos:word):boolean;
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to have the cell perform it's defined action when   *
   *           the user clicks within it's boundary.                      *
   ************************************************************************
   }
begin
    IF Control.Action(Xpos,Ypos) THEN begin
      action := TRUE;
(*      Hilite(Xpos,Ypos); *)
      ActionProc(@self, cellnumber);
    end
    ELSE
      action := FALSE;
end;

Procedure Cell.Cancel;
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to remove a cell from the cell grid display.        *
   *                                                                      *
   ************************************************************************
   }
begin
  UnHilite;
end;

Constructor CellGrid.Init(InitX, InitY, InitWidth, InitHeight : integer;
                          InitActionProc : CellActionProcedure;
                          InitShowProc  : CellShowProcedure;
                          InitCancelProc : CellGridCancelProc;
                          initabovecolor,initbelowcolor,initbordercolor,
                         inithilitecolor:integer);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to initialize the cell grid that is used for all    *
   *           PACE cell grid displays. Contains 444 cell objects.        *
   ************************************************************************
   }

VAR
```

File Name: PACEOBJS.PAS

```
    i, x1, y1, x2, y2, xwidth, yheight, temploc : Integer;
    temp1, temp2                  : xypt;
    tmplst: cell_defn_ptr;
    initlon,initlat: integer;

  begin
    CancelProcedure := InitCancelProc;
    Control.Init(InitX, InitY, InitWidth, InitHeight, black);
    CenterX         := InitX + (InitWidth DIV 2);
    CenterY         := InitY + (InitHeight DIV 2);
    ZoomFactor      := 640 / InitWidth;
    temp1 := Map_Center;
    Map_Center.x := CenterX;
    Map_Center.y := CenterY;
    temp2 := Scale_Factor;
    Scale_Factor.x := Round(Scale_Factor.x * ZoomFactor);
    Scale_Factor.y := Round(Scale_Factor.y * ZoomFactor);
    i := 1;
    tmplst := Cell_lst;
    WHILE tmplst <> NIL DO begin
      Convert_to_Screen_Position( tmplst^.lon1, tmplst^.lat1, x1, y1);
      Convert_to_Screen_Position( tmplst^.lon2,tmplst^.lat2, x2, y2);
      xwidth := abs(x2 - x1);
      yheight := abs(y2 - y1);
      IF x1 > x2 THEN
      begin
        temploc := x1;
        x1 := x2;
        x2 := temploc;
      end;
      IF y1 > y2 THEN
      begin
        temploc := y1;
        y1 := y2;
        y2 := temploc;
      end;
      convert_to_real_position(x1+(xwidth DIV 2),y1 + (yheight DIV 2),
                               initlon,initlat);
      CellArray[i].Init(x1,y1,xwidth,yheight,initlat,initlon,
                        i,initabovecolor,initbelowcolor,initbordercolor,
                        inithilitecolor,
                      (*  blue,red,lightgray,yellow, *)
                        InitActionProc,InitShowProc);
      inc(i);
      tmplst := tmplst^.next;
    END;
    Map_Center :=  temp1;
    Scale_Factor := temp2;
  END;
```

File Name: PACEOBJS.PAS

```
   Procedure CellGrid.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to display the basic cell grid and in turn to cause *
      *           all cells within the grid to display themselfs.            *
      ************************************************************************
      }
   VAR
      xx,yy     : integer;
      i: integer;
   begin
      Get_CursorXY(xx,yy);
      Hide_Cursor;
      FOR i := 1 TO NCELLS DO
         CellArray[i].Show;
      FOR i := 1 TO NCELLS DO
         if CellArray[i].Region then
            CellArray[i].ShowRegion;
      Show_Cursor(xx,yy);
   end;


{ ********* include the method for the coverage grid show ********* }
{$I covgrid.pas}



   Procedure CellGrid.Hilite(Xpos,Ypos: word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to highlight cells in the cell grid display as the  *
      *           cursor passes over them. Not used in operation because     *
      *           it is too inefficient.                                     *
      ************************************************************************
      }
   var i: integer;
   begin
   (*   this takes a WICKED LONG TIME to update the screen... *)
   (*   so I just left it out for now...  EMA 05-27-90        *)
(*
     for i := 1 to NCells do
       CellArray[i].Hilite(Xpos,Ypos);
*)
   end;
```

File Name: PACEOBJS.PAS

```
Function CellGrid.Action(Xpos,Ypos: word): boolean;
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to perform the gell grid associated action by       *
   *           invoking the action methods for the cell objects that make *
   *           up the cell grid object.                                   *
   ************************************************************************
   }
var
  dummy.: boolean;
  i :  integer;
begin
   IF Control.Action(Xpos,Ypos) THEN
      FOR i := 1 TO NCells DO
        dummy := CellArray[i].Action(Xpos,Ypos);
end;

procedure CellGrid.Cancel;
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to remove the cell grid from the display.           *
   ************************************************************************
   }
var i,xx,yy :integer;
begin
  Get_CursorXY(xx,yy);
  for i := 1 to NCells do
    if ((xx > CellArray[i].X) and
        (xx < (CellArray[i].X + CellArray[i].width))) and
       ((yy > CellArray[i].Y) and
        (yy < (CellArray[i].Y + CellArray[i].height))) then
      CancelProcedure(@CellArray[i]);
end;

Procedure ShowCurrentPickChoice;
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Utility routine to show the current operational mode for   *
   *           the weight editor. Changes the text info at the bottom of  *
   *           the display.                                               *
   ************************************************************************
   }
```

File Name: PACEOBJS.PAS

```
VAR
  st1,st2:string;
begin
   IF WeightCellGrid^.pickingweights THEN
   begin
      st1 := 'WEIGHTS';
      st2 := 'REGIONS';
   end
   ELSE
   begin
      st1 := 'REGIONS';
      st2 := 'WEIGHTS';
   end;
   settextjustify(lefttext,toptext);
   setcolor(Basescreen2^.color);
   outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
             'Currently choosing '+st2);
   setcolor(white);
   outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
             'Currently choosing '+st1);
   outtextxy(WeightCellGrid^.X+20,
             WeightCellGrid^.Y+WeightCellGrid^.Height+10,
             'File: '+WeightCellGrid^.Weightsfile);
end;

(*****************************************************************************)

procedure WeightsGrid.Show;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Method to display the weight editor cell grid and also        *
   *           show the current operational mode of the weight editor.       *
   ***************************************************************************
   }
begin
   CellGrid.Show;
   ShowCurrentPickChoice;
end;

constructor Map.Init(InitX, InitY, InitWidth, InitHeight : integer;
                     InitBackgroundColor, InitMapOutlineColor: word);
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Method to initialize the map object.                          *
   *                                                                         *
```

File Name: PACEOBJS.PAS

```
    ********************************************************************************
    }
  begin
    Control.Init(InitX, InitY, InitWidth, InitHeight, InitBackgroundColor);
    CenterX         := InitX + (InitWidth DIV 2);
    CenterY         := InitY + (InitHeight DIV 2);
    ZoomFactor      := 640 / InitWidth;
    MapOutlineColor := InitMapOutlineColor;
    MapBackgroundColor := InitBackgroundColor;
  end;

  procedure Map.Show;
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *           Method that draws the map on the screen according to what          *
    *           is currently stored in it's map vertex list. Also shows            *
    *           the positions of the 8 Omega stations on the screen.               *
    ********************************************************************************
    }

  { this routine will display all of the map boundaries, rivers }
  { city markers, etc , correcting for zoom level by using a different }
  { map for each zoom level to minimize run time computation }
  VAR
    xx, yy, i, xval, yval, oldlon : Integer;
    temp1, temp2                  : xypt;
    tempviewport                  : viewporttype;
  BEGIN
    hide_cursor;
    Get_CursorXY(xx,yy);
    temp1 := Map_Center;
    Map_Center.x := CenterX;
    Map_Center.y := CenterY;
    temp2 := Scale_Factor;
    Scale_Factor.x := Round(Scale_Factor.x * ZoomFactor);
    Scale_Factor.y := Round(Scale_Factor.y * ZoomFactor);
    SetColor(MapOutlineColor);
    setlinestyle(solidln,0,normwidth);
    i := 1;
    oldlon := Map_Points[i].Lon;
    WHILE Map_Points[i].Dtype <> 'Z' DO WITH Map_Points[i] DO BEGIN
      Convert_to_Screen_Position( lon, lat, xval, yval);
      CASE Dtype OF
        'A': Moveto(xval, yval);
        'B': IF abs(oldlon-lon) < 9000 THEN   { watch for wraparound }
               Lineto(xval, yval)
             ELSE
```

```
                    Moveto(xval, yval);
            END;
            Inc(i);
            oldlon := lon;
        END;

        setcolor(lightmagenta);
        FOR i := 0 TO 7 DO
        begin
            Convert_to_Screen_Position( stations[i,1],stations[i,0], xval, yval);
            putpixel(xval,yval,lightmagenta);
            circle(xval,yval,1);
            circle(xval,yval,2);
        end;

        Map_Center :=  temp1;
        Scale_Factor := temp2;

        Show_Cursor(xx,yy);
    END;

    Procedure Map.HiLite(Xpos,Ypos:word);
        {
        ***************************************************************************
        *                                                                         *
        *        PURPOSE                                                          *
        *           Method that displays and updates the lat/lon 'odometer'       *
        *           in the upper right hand corner of the PACE screen.            *
        ***************************************************************************
        }
    VAR
        Lat, Lon: Integer;
        LLStr: String[20];
        OutStr : String[20];
        temp1, temp2                    : xypt;

        PROCEDURE Convert_to_string( ll: Integer);
        {
        ***************************************************************************
        *                                                                         *
        *        PURPOSE                                                          *
        *           Utility routine to convert a lat/lon value to a string        *
        *           with the degree symbol following it.                          *
        ***************************************************************************
        }
        VAR
            i: LongInt;
            TempStr: String[20];
        BEGIN
```

```
        i := LL DIV 100;
        Str(i:3, LLstr);
        i := LL - (i*100);
        i := (i * 60) DIV 100;
        Str(i:2, Tempstr);
        LLstr := LLstr + '½'+Tempstr+'''';
      END;

   BEGIN
      IF (Ypos >= Y) AND (Ypos <= Y+Height) AND
         (Xpos >= X) AND (Xpos <= X+Width) AND
         ((Xpos <> LastlatlonX) OR (Ypos <> LastlatlonY)) THEN
      BEGIN
         LastlatlonX := Xpos;
         LastlatlonY := Ypos;
         temp1 := Map_Center;
         Map_Center.x := CenterX;
         Map_Center.y := CenterY;
         temp2 := Scale_Factor;
         Scale_Factor.x := Round(Scale_Factor.x * ZoomFactor);
         Scale_Factor.y := Round(Scale_Factor.y * ZoomFactor);
         Convert_to_Real_Position(Xpos, Ypos, Lon, Lat);
         IF (Lon <= 18000) AND (Lon >= -18000) AND
            (Lat <= 9000) AND (Lat >= -9000) THEN
         begin
            Hide_cursor;
            setfillstyle(solidfill,blue(*MapBackgroundcolor*));
            bar(450,7,600,18);
            SetColor(mapoutlinecolor);
            IF (Lat < 0) THEN
            begin
               Lat := Lat * -1;
               Convert_to_String(Lat);
               OutStr := LLstr + 'S ';
            end
            ELSE
            begin
               Convert_to_String(Lat);
               OutStr := LLstr + 'N ';
            end;
            IF (Lon < 0) THEN
            begin
               Lon := Lon * -1;
               Convert_to_String(Lon);
               OutStr := OutStr + LLstr + 'W';
            end
            ELSE
            begin
               Convert_to_String(Lon);
```

```
            OutStr := OutStr + LLstr + 'E';
          end;
          SetTextJustify(LeftText, TopText);
          OutTextXY(450,8, OutStr);
          Show_Cursor(Xpos, Ypos);
        end;
        Map_Center :=  temp1;
        Scale_Factor := temp2;
    END;
END;


{ window and menu special purpose methods }
procedure Menu.Cancel;
    {
    ************************************************************************
    *                                                                      *
    *        PURPOSE                                                       *
    *           Method to remove a generic menu object from the display.   *
    *                                                                      *
    ************************************************************************
    }
begin
    HideContext;
    PopContext;
end;

Procedure messageboxobject.changeattr(XX,YY,WW,HH,BK,TX: integer);
    {
    ************************************************************************
    *                                                                      *
    *        PURPOSE                                                       *
    *           Method to initialize a message/alert object.               *
    *                                                                      *
    ************************************************************************
    }
begin
    X := XX;
    Y := YY;
    WIDTH := WW;
    HEIGHT := HH;
    COLOR := BK;
    BORDERCOLOR := TX;
end;

procedure messageboxobject.redisplaymessage(nlines,nchars:integer; message: string);
    {
    ************************************************************************
    *                                                                      *
```

```
      *        PURPOSE                                                    *
      *          Method to re-display the text in a message box that is   *
      *          already on the screen.                                   *
      ***************************************************************************
      }
   VAR
      t: array[1..MaxNumMessageBoxLines] OF string;
      i,limit,XX,YY: integer;
   begin
      IF nlines > MaxNumMessageBoxLines THEN
         limit := MaxNumMessageBoxLines
      ELSE
         limit := nlines;
      FOR i := 1 TO limit DO
         t[i] := copy(message,(i-1)*nchars+1,nchars);
      borderedarea.show;
      SetColor(bordercolor);
      Get_CursorXY(xx,yy);
      Hide_Cursor;
      settextjustify(lefttext,toptext);
      FOR i := 1 TO limit DO
         outtextXY(X+textwidth('12'),Y+(textheight('1')+2)*(i),t[i]);
      Show_Cursor(XX,YY);
   end;

   Procedure messageboxobject.displaymessage(Xpos,Ypos,nlines,nchars,backcolor,textcolor: in
      (
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Method to re-display the text in a message box.                *
      *                                                                         *
      ***************************************************************************
      )
   VAR
      boxwidth,boxheight: integer;
   begin
      boxwidth := textwidth('1') * (nchars + 4);
      boxheight := (textheight('1')+2) * (nlines + 2);
      changeattr(Xpos,Ypos,boxwidth,boxheight,backcolor,textcolor);
      window.show;
      redisplaymessage(nlines,nchars,message);
   end;

   procedure messageboxobject.hidemessage;
      (
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
```

```
      *          Method to remove a message box from the display.          *
      *                                                                    *
      **********************************************************************
      }
   begin
      window.hide;
   end;

   constructor stubobject.init;
      {
      **********************************************************************
      *                                                                    *
      *        PURPOSE                                                     *
      *          Method to initialize a dummy object. Used for debug.      *
      *                                                                    *
      **********************************************************************
      }
   begin
      menu.init(320-textwidth('  Not implemented yet...  ') DIV 2,
                150,
                textwidth('  Not implemented yet...  '),20,
                red,
                white,
                1,
                WindowShadowWidth);
   end;

   procedure stubobject.show;
      {
      **********************************************************************
      *                                                                    *
      *        PURPOSE                                                     *
      *          Method to display    a dummy object. Used for debug.      *
      *                                                                    *
      **********************************************************************
      }
   var
      xx,yy:integer;
   begin
      menu.show;
      settextjustify(centertext,centertext);
      get_cursorxy(xx,yy);
      hide_cursor;
      outtextxy(X+width DIV 2,Y+height DIV 2,'Not implemented yet...');
      show_cursor(xx,yy);
      jingle;
   end;

   constructor linkedbutton3D.Init(InitX,InitY,InitWidth,InitHeight,
```

```
                InitColor,InitHilitecolor,
                InitNameColor,InitHiliteNameColor:integer;
                InitName:shortstring;InitActionProc:ActionProcedure;InitHotKey:char);
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Method to initialize a 3D button that also has an action       *
    *          that can be dynamically associated with it.                    *
    ***************************************************************************
    }
  begin
    actionproc := InitActionProc;
    textbutton3d.Init(InitX,InitY,InitWidth,InitHeight,
                InitColor,InitHilitecolor,
                InitNameColor,InitHiliteNameColor,
                InitName,InitHotKey);
  end;

  function linkedbutton3D.action(XX,YY:word):boolean;
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Method to activate a 3D button that has an dynamically         *
    *          associated action.                                             *
    ***************************************************************************
    }
  begin
    Action := FALSE;
    if textbutton3d.action(XX,YY) then
    begin
        ActionProc;
        Action := TRUE;
    end;
  end;

  constructor linkedbutton.Init(InitX,InitY,InitWidth,InitHeight,
                InitColor,InitHilitecolor,
                InitNameColor,InitHiliteNameColor:integer;
                InitName:shortstring;InitActionProc:ActionProcedure;InitHotKey:char);
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Method to initialize a button that also has an action          *
    *          that can be dynamically associated with it.                    *
    ***************************************************************************
    }
```

```
begin
   actionproc := InitActionProc;
   textbutton.Init(InitX,InitY,InitWidth,InitHeight,
               InitColor,InitHilitecolor,
               InitNameColor,InitHiliteNameColor,
               InitName,InitHotKey);
end;

function linkedbutton.action(XX,YY:word):boolean;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Method to activate a    button that has an dynamically        *
   *           associated action.                                            *
   ***************************************************************************
   }
begin
   Action := FALSE;
   if textbutton.action(XX,YY) then
   begin
       ActionProc;
       Action := TRUE;
   end;
end;

{ utility functions to save/restore info into the different contexts }
procedure SaveEditParams(eptr:emenuptr;saved:savedparamsrecordptr);
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Utility to save the edit screen parameters so that we         *
   *           can get back to them later.                                   *
   ***************************************************************************
   }
VAR
   i: integer;
begin
   saved^.Psa := eptr^.Psa.value.value;
   saved^.SNR := eptr^.SNR.value.value;
   saved^.ShortLongRatio := eptr^.ShortLongRatio.value.value;
   saved^.XAngle := eptr^.XAngle.value.value;
   saved^.PhaseDev := eptr^.PhaseDev.value.value;
   saved^.DMoff := eptr^.DM.buttonarray[1].selected;
   saved^.DMon := eptr^.DM.buttonarray[2].selected;
   saved^.DMpicked := eptr^.DM.picked;
   saved^.SRMbest := eptr^.StationReliabilityModel.buttonarray[1].selected;
   saved^.SRMnom := eptr^.StationReliabilityModel.buttonarray[2].selected;
```

```
      saved^.SRMworst := eptr^.StationReliabilityModel.buttonarray[3].selected;
      saved^.SRMpicked := eptr^.StationReliabilityModel.picked;
      saved^.Fr102 := eptr^.Freq.buttonarray[1].selected;
      saved^.Fr136 := eptr^.Freq.buttonarray[2].selected;
      saved^.FrAND := eptr^.Freq.buttonarray[3].selected;
      saved^.FrOR := eptr^.Freq.buttonarray[4].selected;
      saved^.Frpicked := eptr^.Freq.picked;
      saved^.GDOP := eptr^.GDOP.value.value;
{
      saved^.PsaRAN := eptr^.PsaCalculateMode.buttonarray[1].selected;
      saved^.PsaDET := eptr^.PsaCalculateMode.buttonarray[2].selected;
      saved^.PsaCalpicked := eptr^.PsaCalculateMode.picked;
}
      saved^.PsaMIN := eptr^.PsaReportMode.buttonarray[1].selected;
      saved^.PsaMEAN := eptr^.PsaReportMode.buttonarray[2].selected;
      saved^.PsaMAX := eptr^.PsaReportMode.buttonarray[3].selected;
      saved^.PsaReppicked := eptr^.PsaReportMode.picked;

      FOR i := 1 TO 8 DO
         saved^.StationPower[i] := eptr^.StationPower[i].sld.value.value;

      FOR i := 1 TO 8 DO
      begin
         saved^.StationPowerOn[i] := eptr^.StationPower[i].On.Selected;
         saved^.StationPowerOff[i] := eptr^.StationPower[i].Off.Selected;
      end;

      FOR i := 1 TO 12 DO
         saved^.MonthSelectors[i] := eptr^.MonthSelectors[i].selected;
      FOR i := 1 TO 24 DO
         saved^.HourSelectors[i] := eptr^.HourSelectors[i].selected;
      saved^.QRFile := eptr^.QRFile;
      saved^.WeightsFile := eptr^.WeightsFile;
   end;

   procedure RestoreEditParams(eptr:emenuptr;saved:savedparamsrecordptr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Utility to read back the edit screen parameters that we    *
      *           saved with saveeditparams.                                 *
      ************************************************************************
      }
   VAR
      i: integer;
   begin
      eptr^.Psa                    .value.value :=      saved^.Psa;
      eptr^.SNR                    .value.value :=      saved^.SNR;
```

```
      eptr^.ShortLongRatio          .value.value :=      saved^.ShortLongRatio;
      eptr^.XAngle                  .value.value :=      saved^.XAngle;
      eptr^.PhaseDev                .value.value :=      saved^.PhaseDev;
      eptr^.DM                      .buttonarray[1].selected := saved^.DMoff;
      eptr^.DM                      .buttonarray[2].selected := saved^.DMon;
      eptr^.DM                      .picked := saved^.DMpicked;
      eptr^.StationReliabilityModel.buttonarray[1].selected := saved^.SRMbest;
      eptr^.StationReliabilityModel.buttonarray[2].selected := saved^.SRMnom;
      eptr^.StationReliabilityModel.buttonarray[3].selected := saved^.SRMworst;
      eptr^.StationReliabilityModel.picked := saved^.SRMpicked;
      eptr^.Freq                    .buttonarray[1].selected := saved^.Fr102;
      eptr^.Freq                    .buttonarray[2].selected := saved^.Fr136;
      eptr^.Freq                    .buttonarray[3].selected := saved^.FrAND;
      eptr^.Freq                    .buttonarray[4].selected := saved^.FrOR;
      eptr^.Freq.picked := saved^.Frpicked;

      eptr^.GDOP                     .value.value :=       saved^.GDOP;
{
      eptr^.PsaCalculateMode.buttonarray[1].selected :=          saved^.PsaRAN;
      eptr^.PsaCalculateMode.buttonarray[2].selected :=          saved^.PsaDET;
      eptr^.PsaCalculateMode.picked := saved^.PsaCalpicked;
}
      eptr^.PsaReportMode.buttonarray[1].selected := saved^.PsaMIN;
      eptr^.PsaReportMode.buttonarray[2].selected := saved^.PsaMEAN;
      eptr^.PsaReportMode.buttonarray[3].selected := saved^.PsaMAX;
      eptr^.PsaReportMode.picked := saved^.PsaReppicked;

      FOR i := 1 TO 8 DO
         eptr^.StationPower[i].sld.value.value := saved^.StationPower[i];

      FOR i := 1 TO 8 DO
      begin
         eptr^.StationPower[i].On.Selected := saved^.StationPowerOn[i];
         eptr^.StationPower[i].Off.Selected := saved^.StationPowerOff[i];
      end;

      FOR i := 1 TO 12 DO
         eptr^.MonthSelectors[i].selected := saved^.MonthSelectors[i];
      FOR i := 1 TO 24 DO
         eptr^.HourSelectors[i].selected := saved^.HourSelectors[i];
      eptr^.QRFile := saved^.QRFile;
      eptr^.WeightsFile := saved^.WeightsFile;
   end;



   { control button special purpose methods }

   constructor MutExclusiveN.Init(InitX,InitY,InitWidth,InitHeight,
```

```
                InitColor,InitHilitecolor,InitSelectedColor,
                InitNameColor,InitHiliteNameColor,InitSelectedNameColor:Integer;
                InitControlLabel:shortstring;InitNumberButtons,InitButtonSelected:integer;
                ButtonNameList:namearray;InitOrientation:boolean);
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Initialization routine for an object that allows the        *
    *          selectio of one of n buttons.                               *
    ************************************************************************
    }
  var
    dummy:boolean;
    i: integer;
  begin
    xoffset := 5;
    yoffset := 1;
    Orientation := InitOrientation;
    NumberButtons := InitNumberButtons;
    FOR i := 1 TO NumberButtons DO begin
      if (Orientation = Horizontal) then
        ButtonArray[i].Init(InitX+(i-1)*(InitWidth+xoffset),
                            InitY,InitWidth,InitHeight,InitColor,
                            InitHilitecolor,InitSelectedColor,
                            InitNameColor,InitHiliteNameColor,
                            InitSelectedNameColor,ButtonNameList[i])
      else
        ButtonArray[i].Init(InitX,InitY+(i-1)*(InitHeight+yoffset),
                            InitWidth,InitHeight,InitColor,
                            InitHilitecolor,InitSelectedColor,
                            InitNameColor,InitHiliteNameColor,
                            InitSelectedNameColor,ButtonNameList[i]);
    end;
    IF InitButtonSelected > 0 THEN
      ButtonArray[InitButtonSelected].TurnOn;
    ControlLabel := InitControlLabel;
    Picked := InitButtonSelected;
  end;

procedure MutExclusiveN.Show;
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Method to display the mutually exclusive set of selector    *
    *          buttons.                                                    *
    ************************************************************************
    }
```

```
VAR
   i:integer;
begin
   FOR i := 1 TO NumberButtons DO
      ButtonArray[i].Show;
   if orientation = horizontal then begin
     SetTextJustify(RightText,CenterText);
     SetColor(ButtonArray[1].ControlColor);
     OutTextXY(ButtonArray[1].X-5,
               ButtonArray[1].Y+1+ButtonArray[1].Height DIV 2,
               ControlLabel);
   end
   else begin
     SetTextJustify(CenterText,CenterText);
     SetColor(ButtonArray[1].ControlColor);
     OutTextXY(ButtonArray[1].X+ButtonArray[1].Width DIV 2,
               ButtonArray[1].Y+1+ButtonArray[1].Height DIV 2,
               ControlLabel);
   end;
end;

procedure MutExclusiveN.HiLite(Xpos,Ypos:word);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to highlight the mutually exclusive set of selector *
   *           buttons.                                                   *
   ************************************************************************
   }
VAR
   i:integer;
begin
   FOR i := 1 TO NumberButtons DO
      ButtonArray[i].HiLite(Xpos,Ypos);
end;

procedure MutExclusiveN.ChangeXY(Xpos,Ypos:word);
   {
   ************************************************************************
   *                                                                      *
   *        PURPOSE                                                       *
   *           Method to change the X Y position of a mutually exclusive  *
   *           set of selector buttons.                                   *
   ************************************************************************
   }
var
  i : integer;
begin
```

```
      FOR i := 1 TO NumberButtons DO
         if Orientation = Horizontal then
           ButtonArray[i].ChangeXY(Xpos+
                                   (i-1)*(ButtonArray[i].Width+xoffset),Ypos)
         else
           ButtonArray[i].ChangeXY(Xpos,Ypos+
                                   (i-1)*(ButtonArray[i].Height+yoffset));
   end;

   function MutExclusiveN.Action(Xpos,Ypos:word):boolean;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method to invoke the action associated with a mutually        *
      *           exclusive set of selector buttons.                            *
      ***************************************************************************
      }
   var
      i,j:integer;
   begin
      Action := FALSE;
      FOR i := 1 TO NumberButtons DO
      begin
         IF ButtonArray[i].Action(Xpos,Ypos) THEN
         begin
            Action := TRUE;
            FOR j := 1 TO NumberButtons DO
               IF ButtonArray[j].selected THEN
                  ButtonArray[j].TurnOff;
            ButtonArray[i].TurnOn;
            picked := i;
         end;
      end;
   end;

   constructor OnOffSlideControl.Init(InitX,InitY,InitColor,InitHiliteColor,InitSelectedColo
               InitControlColor,InitHiliteControlColor,InitSelectedControlColor:word;
               InitLoLim,InitHiLim,InitInc:longint;
               InitAccel:integer;InitName,InitUnits:shortstring);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Initialization for a slider bar control that can be turned    *
      *           on and off.                                                   *
      ***************************************************************************
      }
   begin
```

```
    Off.Init(InitX+H_barwidth+90,InitY,30,H_barheight,InitColor,InitHiLiteColor,InitSelect
             InitControlColor,InitHiliteControlColor,InitSelectedControlColor,
             'Off');
    On.Init(Off.X+1+30,InitY,30,H_barheight,InitColor,InitHiLiteColor,InitSelectedColor,
             InitControlColor,InitHiliteControlColor,InitSelectedControlColor,
             'On');
    SLD.Init(InitX,InitY,InitColor,InitHiLiteColor,
             InitControlColor,InitHiLiteControlColor,InitLoLim,InitHiLim,
             InitInc,InitAccel,InitName,InitUnits,5,0);
    On.TurnOn;
end;

procedure OnOffSlideControl.HiLite(Xpos,Ypos:word);
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Highlight method for slider bar control that can be turned *
    *          on and off.                                                 *
    ************************************************************************
    }
begin
    Off.HiLite(Xpos,Ypos);
    On.HiLite(Xpos,Ypos);
    IF On.Selected THEN
       SLD.HiLite(Xpos,Ypos);
end;

procedure OnOffSlideControl.Show;
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Show      method for slider bar control that can be turned *
    *          on and off.                                                 *
    ************************************************************************
    }
begin
    Off.Show;
    On.Show;
    SLD.Show;
    IF Off.selected THEN
       SLD.Value.Changename('Off');
end;

function OnOffSlideControl.Action(Xpos,Ypos:word):boolean;
    {
    ************************************************************************
    *                                                                      *
```

File Name: PACEOBJS.PAS

```
      *        PURPOSE:                                                    *
      *           Action    method for slider bar control that can be turned *
      *           on and off.                                              *
      ***************************************************************************
      }
   var
      st:shortstring;
      sldaction,onoffaction:boolean;
   begin
      onoffAction := FALSE;
      sldaction := FALSE;
      IF Off.Action(Xpos,Ypos) THEN
      begin
         On.TurnOff;
         Off.TurnOn;
         SLD.Value.Changename('Off');
         onoffAction := TRUE;
      end;
      IF On.Action(Xpos,Ypos) THEN
      begin
         On.TurnOn;
         Off.TurnOff;
         Str(SLD.Value.Value:SLD.Value.int:SLD.Value.frac,st);
         SLD.Value.Changename(St);
         onoffAction := TRUE;
      end;
      IF On.Selected THEN
         sldAction := SLD.Action(Xpos,Ypos);
      Action := onoffaction OR sldaction;
   end;

   function SelectButton.Action(Xpos,Ypos:word):boolean;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Initialization for a set of buttons where the user can        *
      *           choose as many as desired.                                    *
      ***************************************************************************
      }
   begin
      Action := FALSE;
      IF StatusButton.Action(Xpos,Ypos) THEN
      begin
         IF selected THEN
            statusbutton.Turnoff
         Else
            Statusbutton.Turnon;
         Action := TRUE;
```

```
    end;
  end;

  constructor pickmenu.Init(Xpos,Ypos:word;InitNumberButtons:integer;
      InitNameList:PickMenuNameArray;InitActions:PickMenuActionArray;
      InitHotkeys:PickMenuHotkeyArray);
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Initialization for a list object where the user can select *
    *          a particular item of interest and also scroll the list.    *
    ************************************************************************
    }
  Const
    menucolor = cyan;
    menubordercolor = black;
    buttoncolor = white;
    hilitecolor = lightblue;
    textcolor = black;
    hilitetextcolor = white;
  VAR
    i,menuheight: integer;
  begin
    NumberButtons := InitNumberButtons;
    menuheight := 5 + NumberButtons*20;
    menu.Init(Xpos,Ypos,80,menuheight,menucolor,menubordercolor,2,WindowShadowWidth);
    FOR i := 1 TO NumberButtons DO
      ButtonArray[i].Init(X+10,Y+4+(i-1)*20,60,16,buttoncolor,hilitecolor,textcolor,hilit
        InitNameList[i],InitActions[i],InitHotKeys[i]);
  end;

  function pickmenu.Action(Xpos,Ypos:word):boolean;
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Action method  for a list object where the user can select *
    *          a particular item of interest and also scroll the list.    *
    ************************************************************************
    }
  var
    XX,YY,i: integer;
  begin
    Action := FALSE;
    FOR i := 1 TO NumberButtons DO
      IF ButtonArray[i].action(Xpos,Ypos) THEN
        Action := TRUE;
  end;
```

```
function pickmenu.KeyAction(Xpos,Ypos:word; key:integer):boolean;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           'Hotkey' method for  list object where the user can select *
   *           a particular item of interest and also scroll the list.    *
   ***************************************************************************
   }
var
    i:integer;
begin
   KeyAction := false;
   FOR i := 1 TO NumberButtons DO
      IF ButtonArray[i].keyaction(Xpos,Ypos,key) THEN
         KeyAction := TRUE;
end;

procedure pickmenu.Show;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Show     method for  list object where the user can select *
   *           a particular item of interest and also scroll the list.    *
   ***************************************************************************
   }
VAR
   i:integer;
begin
   menu.show;
   FOR i := 1 TO NumberButtons DO
      ButtonArray[i].Show
end;

procedure pickmenu.HiLite(Xpos,Ypos:word);
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Highlight method for list object where the user can select *
   *           a particular item of interest and also scroll the list.    *
   ***************************************************************************
   }
VAR
   i:integer;
begin
   FOR i := 1 TO NumberButtons DO
```

```
        ButtonArray[i].HiLite(Xpos,Ypos);
    end;

procedure lmenu.DisposeOldDirectoryList;
      (
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method for getting rid of the currently displayed list of          *
      *           file names so that we can get and display a new list.              *
      ********************************************************************************
      )

var
  tmp, ptr : listptr;

begin
  tmp := Dlist;
  if tmp <> nil then
  repeat
    ptr := tmp^.next;
    dispose(tmp);
    tmp := ptr;
  until (ptr = nil);
end;

procedure lmenu.DisplayDirectoryList;
      (
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method for displaying the current list of file names.              *
      *                                                                              *
      ********************************************************************************
      )

var
  ptr,tmp    : listptr;
  i          : integer;
  Xpos,Ypos  : integer;

begin
  ptr := Dlist;
  i := 1;
  if (listposition <> i) then
    repeat
      if ptr^.Next <> nil then begin
        ptr := ptr^.Next;
        inc(i);
```

```
      end;
    until (i = listposition);

  i:=1;
  while (i <= maxlistsize) do begin
    if ptr <> nil then begin
      DirectoryList.ButtonArray[i].Name := ptr^.FName;
      ptr := ptr^.Next;
    end
    else
      DirectoryList.ButtonArray[i].Name := '';
    inc(i);
  end;
  Get_CursorXY(Xpos,Ypos);
  Hide_Cursor;
  FOR i := 1 TO maxlistsize DO
     IF directorylist.buttonarray[i].selected AND
        ((directorylist.buttonarray[i].name = '') OR
         (directorylist.buttonarray[i].name = '<top>') OR
         (directorylist.buttonarray[i].name = '<bottom>')) THEN
         begin
           directorylist.buttonarray[i].TurnOff;
           directorylist.picked := -1;
         end;
  DirectoryList.Show;
  Show_Cursor(Xpos,Ypos);
end;

procedure lmenu.GetNewDirectoryList;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method for getting a new          list of file names.     *
      *                                                                      *
      ************************************************************************
      }

var
  tmp,ptr : Listptr;
  DirInfo : SearchRec;        (* SearchRec is defined in the Dos Unit *)

begin
  new(Dlist);
  Dlist^.FName := '<top>';
  Dlist^.Next  := nil;
  ptr := Dlist;
  listlength := 1;
  listposition := 1;
```

File Name: PACEOBJS.PAS

```
  findfirst(DEFAULTPATH+DirectoryMaskButton.Text_String,SysFile,DirInfo,;
  while (DosError = 0) do begin
    new(tmp);
    tmp^.FName := DirInfo.Name;
    tmp^.Next  := nil;
    ptr^.Next  := tmp;
    ptr        := tmp;
    inc(listlength);
    findnext(DirInfo);
  end;
  new(tmp);
  tmp^.FName := '<bottom>';
  tmp^.Next  := nil;
  ptr^.Next  := tmp;
  ptr := Dlist;
  inc(listlength);
  VBar.HiLim := listlength(*-maxlistsize+1*);
  VBar.Value := listposition;
end;

Constructor lmenu.Init(Xpos, Ypos   : word;
                       InitMenuColor,InitMenuBorderColor: word;
                       Vlink, Llink : ActionProcedure;
                       InitPath,InitEString:string);
     {
     ************************************************************************
     *                                                                      *
     *        PURPOSE                                                       *
     *           Initialization for an object that lets the user pick a file*
     *           name from a scrollable list.                               *
     ************************************************************************
     }
Const
   bc = white;          { button color }
   hc = lightblue;      { hilite color }
   sc = yellow;         { selected color }
   tc = black;          { text color }
   htc = white;         { hilited text color }
   stc = black;         { selected text color }
   ems = 18;            { edit menu spacing constant }

var
   labelarray:namearray;
   i : integer;

begin
   mc := InitMenuColor;
   mbc := InitMenuBorderColor;
   EString := InitEString;
```

```
   DEFAULTPATH    := InitPath;

   listposition := 1;
   listlength   := 0;
   Dlist := nil;

   menu               .Init(Xpos, Ypos,180,180,mc,mbc,2,WindowShadowWidth);
   DirectoryMaskButton.Init(X+10, Y+6,160,15,bc,hc,tc,htc,EString,0,4,19,FALSE);
   Vbar               .Init(X+138,Y+27, bc,hc,tc,htc,1.0,1.0,1.0,linearscale);
   DirectoryList      .Init(X+28, Y+27, 100,12,bc,hc,sc,tc,htc,stc,
                            '',maxlistsize,-1,
                            labelarray,vertical);
   ViewButton         .Init(X+20, Y+159,60,16,bc,hc,tc,htc,'View',Vlink,'V');
   LoadButton         .Init(X+100,Y+159,60,16,bc,hc,tc,htc,'Load',Llink,'L');
   selectedfile := '';
end;

procedure lmenu.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *          Method for displaying a list of scrollable items.           *
      *                                                                      *
      ************************************************************************
      }
begin
   menu.show;
   DirectoryMaskButton.Show;
   ViewButton.Show;
   LoadButton.Show;
   GetNewDirectoryList;
   DisplayDirectoryList;
   VBar.Show;
end;

procedure lmenu.HiLite(Xpos,Ypos:word);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *          Method for highlighting a list of scrollable items.         *
      *                                                                      *
      ************************************************************************
      }
begin
   menu.HiLite(Xpos,Ypos);
   DirectoryMaskButton.HiLite(Xpos,Ypos);
   VBar.HiLite(Xpos,Ypos);
```

```
   DirectoryList.HiLite(Xpos,Ypos);
   ViewButton.HiLite(Xpos,Ypos);
   LoadButton.HiLite(Xpos,Ypos);
end;

function lmenu.Action(Xpos,Ypos:word):boolean;
      (
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Action method for a scrollable list of items.                      *
      *                                                                              *
      ********************************************************************************
      )
var
   dummy      : boolean;
   i          : integer;
   path: pathstr;
   dir: dirstr;
   name: namestr;
   ext: extstr;

begin
   dummy := DirectoryList.Action(Xpos,Ypos);

   if VBar.Action(Xpos,Ypos) then begin
     listposition := round(VBar.Value);
     DisplayDirectoryList;
   end;

   if DirectoryMaskButton.Action(Xpos,Ypos) then begin
     DisposeOldDirectoryList;
     GetNewDirectoryList;
     DisplayDirectoryList;
     VBar.Show;
   end;

   FOR i := 1 TO maxlistsize DO
     IF directorylist.buttonarray[i].selected THEN
     begin
        IF ((directorylist.buttonarray[i].name = '') OR
         (directorylist.buttonarray[i].name = '<top>') OR
         (directorylist.buttonarray[i].name = '<bottom>')) THEN
        begin
          directorylist.buttonarray[i].TurnOff;
          directorylist.picked := -1;
        end
        ELSE
        begin
```

```
            path := directorymaskbutton.text^.string;
            Fsplit(path,dir,name,ext);
            selectedfile := dir + directorylist.buttonarray[i].name;
          end;
        end;

    dummy := ViewButton.Action(Xpos,Ypos);

    dummy := LoadButton.Action(Xpos,Ypos);
end;

function lmenu.KeyAction(Xpos,Ypos:word; key:integer):boolean;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Hotkey action method for a scrollable list of items.          *
      *                                                                         *
      ***************************************************************************
      }
var dummy : boolean;
begin
  KeyAction := false;
  if (ViewButton.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
  if (LoadButton.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
end;

procedure lmenu.Cancel;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Method for removing the scrollable list from the display.     *
      *                                                                         *
      ***************************************************************************
      }
var i:integer;
begin
  DisposeOldDirectoryList;
  menu.Cancel;
end;

{***************** Include the methods for the side and lower status bars ****}
{$I statbars.pas}


   Constructor dmenu.Init(Xpos,Ypos:word;Alink:ActionProcedure);
```

File Name: PAC3OBJS.PAS

```
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Method for initializing the difference menu selections.       *
    *                                                                         *
    ***************************************************************************
    }
  Const
    mc = cyan;            { menu color }
    mbc = black;          { menu border color }
    bc = white;           { button color }
    hc = lightblue;       { hilite color }
    sc = yellow;          { selected color }
    tc = black;           { text color }
    htc = white;          { hilited text color }
    stc = black;          { selected text color }
    ems = 18;             { edit menu spacing constant }
  var
    labelarray:namearray;
    first:integer;
  begin
    menu.Init(Xpos,Ypos,305,90,mc,mbc,2,WindowShadowWidth);
    first := Y+10;
    lothresh.Init(X+ 75,first,bc,hc,tc,htc,      -1,1,0.001,linearscale,'LoLim','',0,3);
    hithresh.Init(X+ 75,first+ems*1,bc,hc,tc,htc,-1,1,0.001,linearscale,'HiLim','',0,3);
    labelarray[1] := chr(127) + 'PSA';
    labelarray[2] := '%'+chr(127)+'PSA';
    labelarray[3] := '%'+chr(127)+'(1-PSA)';
    DiffMode.Init(X+ 50,first+ems*2,75,10,bc,hc,sc,tc,htc,stc,'Mode',3,1,labelarray,horizo
    accept.Init(X+100    ,first+ems*3,100,16,bc,hc,tc,htc,'Accept',Alink,'A');
  end;

  procedure dmenu.Show;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Method for displaying   the difference menu selections.       *
    *                                                                         *
    ***************************************************************************
    }
  begin
    menu.show;
    diffmode.show;
    lothresh.show;
    hithresh.show;
    accept.show;
  end;
```

File Name: PACEOBJS.PAS

```
procedure dmenu.HiLite(Xpos,Ypos:word);
   {
   ********************************************************************************
   *                                                                              *
   *        PURPOSE                                                               *
   *           Method for highlighting the difference menu selections.            *
   *                                                                              *
   ********************************************************************************
   }
begin
   menu.hilite(Xpos,Ypos);
   diffmode.hilite(Xpos,Ypos);
   lothresh.hilite(Xpos,Ypos);
   hithresh.hilite(Xpos,Ypos);
   accept.hilite(Xpos,Ypos);
end;

function dmenu.Action(Xpos,Ypos:word):boolean;
   {
   ********************************************************************************
   *                                                                              *
   *        PURPOSE                                                               *
   *           Action method for the difference menu.                             *
   *                                                                              *
   ********************************************************************************
   }
var
   dummy: boolean;
begin
   action := false;
   dummy := menu.action(Xpos,Ypos);
   dummy := diffmode.action(Xpos,Ypos);
   IF (lothresh.action(Xpos,Ypos)) THEN
      IF lothresh.value.value > hithresh.value.value THEN
      begin
         hithresh.value.value := lothresh.value.value;
         hithresh.show;
      end;
   IF (hithresh.action(Xpos,Ypos)) THEN
      IF hithresh.value.value < lothresh.value.value THEN
      begin
         lothresh.value.value := hithresh.value.value;
         lothresh.show;
      end;
   dummy := accept.action(Xpos,Ypos);
end;

function dmenu.KeyAction(Xpos,Ypos:word;Key:integer):boolean;
```

File Name: PACEOBJS.PAS

```
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Hotkey action method for the difference menu.                 *
    *                                                                         *
    ***************************************************************************
    }
  var
    dummy: boolean;
  begin
    Keyaction := FALSE;
    dummy := menu.keyaction(Xpos,Ypos,key);
    dummy := diffmode.keyaction(Xpos,Ypos,key);
    dummy := lothresh.keyaction(Xpos,Ypos,key);
    dummy := hithresh.keyaction(Xpos,Ypos,key);
    keyaction := accept.keyaction(Xpos,Ypos,key);
  end;

  Constructor emenu.Init(Xpos,Ypos:word;Rlink,Wlink,Blink,Plink:ActionProcedure);
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Method to initialize the controls on the edit menu.           *
    *                                                                         *
    ***************************************************************************
    }
  Const
    mc = cyan;            { menu color }
    mbc = black;          { menu border color }
    bc = white;           { button color }
    hc = lightblue;       { hilite color }
    sc = yellow;          { selected color }
    tc = black;           { text color }
    htc = white;          { hilited text color }
    stc = black;          { selected text color }
    ems = 18;             { edit menu spacing constant }
  var
    first:word;
    labelarray:namearray;
    i:integer;
  begin
    menu.Init(Xpos,Ypos,525,230,mc,mbc,2,WindowShadowWidth);
    first := Y+10;
    Psa                   .Init(X+ 40,first,bc,hc,tc,htc,0.5,0.999,0.001,linearscale,'Psa
    SNR                   .Init(X+ 40,first+ems*1,bc,hc,tc,htc,-99,99,1,linearscale,'SNR'
    ShortLongRatio        .Init(X+ 40,first+ems*2,bc,hc,tc,htc,-99,99,1,linearscale,'S/L'
    XAngle                .Init(X+ 40,first+ems*3,bc,hc,tc,htc,0,90,1,linearscale,'ANG','
```

File Name: PACEOBJS.PAS

```
      PhaseDev              .Init(X+ 40,first+ems*4,bc,hc,tc,htc,0,50,1,linearscale,'DEV','
      GDOP                  .Init(X+ 40,first+ems*5,bc,hc,tc,htc,0,25,1,linearscale,'GDP','
      labelarray[1] := 'OFF';
      labelarray[2] := 'ON';
      DM                    .Init(X+ 40,first+ems*6,35,10,bc,hc,sc,tc,htc,stc,'DM',2,2,labe
(
      labelarray[1] := 'RAN';
      labelarray[2] := 'DET';
      PsaCalculateMode      .Init(X+ 40,first+ems*6,35,10,bc,hc,sc,tc,htc,stc,'CAL',2,2,lab
)
      labelarray[1] := 'MIN';
      labelarray[2] := 'MEAN';
      labelarray[3] := 'MAX';

      PsaReportMode         .Init(X+ 40,first+ems*7,35,10,bc,hc,sc,tc,htc,stc,'REP',3,1,lab
      labelarray[1] := 'BEST';
      labelarray[2] := 'NOM';
      labelarray[3] := 'WRST';
      StationReliabilityModel.Init(X+ 40,first+ems*8,35,10,bc,hc,sc,tc,htc,stc,'SRM',3,2,lab

      MonthSelectors[1].Init(    X+40,first+ems*9,30,10,bc,hc,sc,tc,htc,stc,MonthNames[1]);
      FOR i := 2 TO 12 DO
         MonthSelectors[i].Init(MonthSelectors[i-1].X+31,first+ems*9,30,10,
         bc,hc,sc,tc,htc,stc,MonthNames[i]);

      HourSelectors[1].Init(    X+10,first+ems*10,20,10,bc,hc,sc,tc,htc,stc,HourNames[1]);
      FOR i := 2 TO 24 DO
         HourSelectors[i].Init(HourSelectors[i-1].X+21,first+ems*10,20,10,
         bc,hc,sc,tc,htc,stc,HourNames[i]);

      first := y+10;
      FOR i := 1 TO 8 DO
      StationPower[i].Init(X+260,first+ems*(i-1),bc,hc,sc,tc,htc,stc,-20,20,1,linearscale,st

      labelarray[1] := '10.2';
      labelarray[2] := '13.6';
      labelarray[3] := 'AND';
      labelarray[4] := 'OR';
      Freq.Init(X+260,first+ems*8,35,10,bc,hc,sc,tc,htc,stc,'FRQ',4,1,labelarray,horizontal)

      Reliability.Init(X+10    ,first+ems*11,100,16,bc,hc,tc,htc,'Reliability',Rlink,'R');
      Weights    .Init(X+10+120,first+ems*11,100,16,bc,hc,tc,htc,'Weights',Wlink,'W');
      Batch      .Init(X+10+240,first+ems*11,100,16,bc,hc,tc,htc,'qUeue',Blink,'U');
      Process    .Init(X+10+360,first+ems*11,100,16,bc,hc,tc,htc,'Process',Plink,'P');
      QRfile := DEFAULTQRFILE;
      Weightsfile := DEFAULTWEIGHTSFILE;
   end;

   procedure emenu.Show;
```

File Name: PACEOBJS.PAS

```
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to display    the controls on the edit menu.       *
      *                                                                      *
      ************************************************************************
      }
   VAR
      i: integer;
   begin
      recompute := FALSE;
      rethreshold := FALSE;

      menu.show;

      SaveEditParams(@self,savedparams);

      Psa                     .Show;
      SNR                     .Show;
      ShortLongRatio          .Show;
      XAngle                  .Show;
      PhaseDev                .Show;
      DM                      .Show;
      GDOP                    .Show;
{
      PsaCalculateMode        .Show;
}
      StationReliabilityModel.Show;
      Freq                    .Show;
      PsaReportMode           .Show;

      FOR i := 1 TO 8 DO
         stationpower[i].show;

      FOR i := 1 TO 12 DO
         MonthSelectors[i].Show;

      FOR i := 1 TO 24 DO
         HourSelectors[i].Show;
      Reliability.Show;
      Weights    .Show;
      Batch      .Show;
      Process    .Show;
   end;

   procedure emenu.HiLite(Xpos,Ypos:word);
      {
      ************************************************************************
```

File Name: PACEOBJS.PAS

```
      *                                                                      *
      *        PURPOSE                                                       *
      *          Method to highlight  the controls on the edit menu.         *
      *                                                                      *
      ************************************************************************
      }
   VAR
      i:integer;
   begin
      menu.HiLite(Xpos,Ypos);

      Psa                     .HiLite(Xpos,Ypos);
      SNR                     .HiLite(Xpos,Ypos);
      ShortLongRatio          .HiLite(Xpos,Ypos);
      XAngle                  .HiLite(Xpos,Ypos);
      PhaseDev                .HiLite(Xpos,Ypos);
      DM                      .HiLite(Xpos,Ypos);
      GDOP                    .HiLite(Xpos,Ypos);
{
      PsaCalculateMode        .HiLite(Xpos,Ypos);
}
      StationReliabilityModel.HiLite(Xpos,Ypos);
      Freq                    .HiLite(Xpos,Ypos);
      PsaReportMode           .HiLite(Xpos,Ypos);

      FOR i := 1 TO 12 DO
         MonthSelectors[i].Hilite(Xpos,Ypos);

      FOR i := 1 TO 24 DO
         HourSelectors[i].Hilite(Xpos,Ypos);

      FOR i := 1 TO 8 DO
         stationpower[i].HiLite(Xpos,Ypos);

      Reliability.HiLite(Xpos,Ypos);
      Weights    .HiLite(Xpos,Ypos);
      Batch      .HiLite(Xpos,Ypos);
      Process    .HiLite(Xpos,Ypos);
   end;

   function emenu.Action(Xpos,Ypos:word):boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *          Method to activate   the controls on the edit menu.         *
      *                                                                      *
      ************************************************************************
      }
```

File Name: PACEOBJS.PAS

```
   var
      dummy: boolean;
      st: shortstring;
      i:integer;
   begin
      { if any of the following change then the cell data must be rethresholded }
      rethreshold := Psa                    .Action(Xpos,Ypos) OR rethreshold;
      rethreshold := PsaReportMode          .Action(Xpos,Ypos) OR rethreshold;
      FOR i := 1 TO 12 DO
         rethreshold := MonthSelectors[i].Action(Xpos,Ypos) OR rethreshold;
      FOR i := 1 TO 24 DO
         rethreshold := HourSelectors[i].Action(Xpos,Ypos) OR rethreshold;

      { if any of the following change then the cell data must be recomputed }
      recompute := SNR                      .Action(Xpos,Ypos) OR recompute;
      recompute := ShortLongRatio           .Action(Xpos,Ypos) OR recompute;
      recompute := XAngle                   .Action(Xpos,Ypos) OR recompute;
      recompute := PhaseDev                 .Action(Xpos,Ypos) OR recompute;
      recompute := DM                       .Action(Xpos,Ypos) OR recompute;
      recompute := GDOP                     .Action(Xpos,Ypos) OR recompute;
{
      recompute := PsaCalculateMode         .Action(Xpos,Ypos) OR recompute;
}
      recompute := StationReliabilityModel.Action(Xpos,Ypos) OR recompute;
      recompute := Freq                     .Action(Xpos,Ypos) OR recompute;
      FOR i := 1 TO 8 DO
         recompute := stationpower[i].Action(Xpos,Ypos) OR recompute;

(*  REPLACE THE FOLLOWING TWO LINES WITH SOMETHING LIKE THIS *)

      recompute   := Reliability.Action(Xpos,Ypos) OR recompute;
      rethreshold := Weights    .Action(Xpos,Ypos) OR rethreshold;

(*    WHEN THE RELIABILITY BUTTON AND WEIGHTS BUTTONS GET HOOKED UP *)
(*
      dummy := Reliability.Action(Xpos,Ypos);
      dummy := Weights    .Action(Xpos,Ypos);
*)
      dummy := Batch      .Action(Xpos,Ypos);
      dummy := Process    .Action(Xpos,Ypos);
end;

function emenu.KeyAction(Xpos,Ypos:word; key:integer):boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to activate   the controls on the edit menu based   *
      *           on a hotkey input.                                         *
```

File Name: PACEOBJS.PAS

```
      ********************************************************************************
      }
var dummy : boolean;
begin
  KeyAction := false;
  if (Reliability.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
  if (Weights.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
  if (Batch.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
  if (Process.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
end;

procedure emenu.Cancel;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method to remove the edit screen from the display.                 *
      *                                                                              *
      ********************************************************************************
      }
var
  key,button : integer;

begin
  if (rethreshold or recompute) then
  begin
    messagebox.displaymessage((GetMaxX DIV 2) -
         (textwidth('123456789012345') DIV 2),
         (GetMaxY DIV 2), 1,
         length('123456789012345')
         ,lightmagenta,black,
       (*'123456789012345'*)
         '  Quit Edit ?  ');
    beep;
    repeat
      Get_Key(key);
      button := Get_Buttons;
    until (key = ENTER) or
          (key = ESC)   or
          (upcase(chr(key)) = 'Y') or
          (upcase(chr(key)) = 'N') or
          (button <> 0);

    messagebox.hidemessage;
```

```
    if (key = ENTER) or
       (button = left_button_pressed) or
       (upcase(chr(key)) = 'Y') then
    begin
      menu.Cancel;
      RestoreEditParams(@self,savedparams);
    end;
  end
  else
  begin
    menu.Cancel;
    RestoreEditParams(@self,savedparams);
  end;
end;


(************** Include auxiliary object method code here *****************)

{$I cellwin.pas}
{$I memoarea.pas}
{$I savemenu.pas}
{$I helpmenu.pas}

var
  DirInfo : SearchRec;       (* SearchRec is defined in the Dos unit *)

begin
   { allocate area to hold edit parameters for the edit menus etc }
   new(savedparams);
   { initialize a utility message box }
   messagebox.init(0,0,0,0,0,0,1,WindowShadowWidth);
   batchqueue := NIL;
   topofbatchqueue := NIL;
end.
```

File Name: PROCS.PAS

```
(*************************************************************************
*                                                                       *
*        PROGRAM NAME - Omega Performance Assessment                    *
*                                and                                    *
*                          Coverage Evaluation                          *
*                                (PACE)                                 *
*                             Workstation                               *
*                                                                       *
*           UNIT NAME - PROCS Unit                                      *
*                                                                       *
*************************************************************************
*                                                                       *
*        This program was prepared by                                   *
*                                                                       *
*              The Analytic Science Corporation (TASC)                  *
*                    55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                          *
*                                                                       *
*        PACE has been developed to run on a IBM PC/AT or compatible    *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main      *
*        memory and an EGA or compatible graphics adapter and color     *
*        monitor. This work was performed under contract number         *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for      *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.   *
*                                                                       *
*************************************************************************
*                                                                       *
*        PURPOSE                                                        *
*           This unit contains a collection of procedures that are      *
*           dynamically assigned to object methods for the objects      *
*           declared in the PACEOBJS unit. Basically, all of the        *
*           PACE specific action procedures are contained herein.       *
*           The files PROCS2.PAS and PROCS3.PAS also contains portions  *
*           of this unit.                                               *
*************************************************************************)
(SN+,E+)
unit procs;

Interface

Uses        Crt,Graph,CursrObj,CellUtil,QRutil,Controls,
                ConMan,Pacecbjs,PaceInit,DataUtil,DOS,ErrLog;

TYPE


   WeightsDataType = record
                       Comment : string;
                       WeightData : array[1..444] of byte;
```

```
                    end;

   WeightsFile = File OF WeightsDataType;

var
   leftexpanded,rightexpanded:boolean;
                          { indicated which window is currently expanded }

   QueueSaveComment,WeightsSaveComment,LeftSaveComment,RightSaveComment: CommentAreaType;

   QRDatabase:QRDatabaseptr;

   filemenunamelist,optionsmenunamelist:PickMenuNameArray;
   filemenuproceduralist,optionsmenuproceduralist:PickMenuActionArray;
   filemenuhotkeylist,optionsmenuhotkeylist:PickMenuHotKeyArray;


   Procedure ComputePsa(eptr:emenuptr; lptr:lbarptr);
   procedure CopyCells(fromptr,toptr:cellgridptr);
   procedure quitbutton;
   procedure helpbutton;
   procedure filebutton;
   procedure optionsbutton;
   procedure leftfilebutton;
   procedure rightfilebutton;
   procedure editbutton;
   procedure loadbutton;
   procedure savebutton;
   procedure returnbutton;
   procedure Coveragereturnbutton;
   procedure DifferenceReturnButton;
   procedure auxsavebutton;
   procedure saveaction;
   procedure auxsaveaction;
   procedure auxloadbutton;
   procedure auxeditbutton;
   procedure splitbutton;
   procedure rightexpandbutton;
   procedure leftexpandbutton;
   procedure editmenuprocess;
   procedure auxeditmenuprocess;
   procedure ShowCellData(selfptr : CellPtr);
   procedure ShowCoverageCellData(selfptr: CellPtr);
   procedure stub;
   procedure SaveArchiveParameters(VAR AF:ArchiveFile; eptr:emenuptr;
                                   VAR comment:commentareatype);
   procedure RestoreArchiveParameters(VAR AF: Archivefile;eptr:emenuptr;
                                      VAR comment:commentareatype);
   procedure ReThresholdCells(CGptr: CellGridPtr; eptr:emenuptr;
```

File Name: PROCS.PAS

```
                                    lptr:lbarptr);
    procedure RightCellGridAction(selfptr : CellPtr; cellnumber:integer);
    procedure LeftCellGridAction(selfptr : CellPtr; cellnumber:integer);
    procedure BigCellGridAction(selfptr : CellPtr; cellnumber:integer);
    procedure SubCellAction(selfptr : CellPtr; cellnumber:integer);
    procedure listmenuaction;
    procedure rightlistmenuaction;
    procedure listmenuview;
    procedure rightlistmenuview;
    procedure reliabilitymenuview;
    procedure auxreliabilitymenuview;
    procedure ReliabilityAction;
    procedure AuxReliabilityAction;
    procedure QueueAction;
    procedure AuxQueueAction;
    procedure queuesaveAction;
    procedure auxqueuesaveAction;
    procedure processbatchqueue;
    procedure auxprocessbatchqueue;
    procedure CoverageAction;
    procedure CoverageCellGridAction(selfptr : CellPtr; CellNumber: integer);
    procedure RelListMenuAction;
    procedure AuxRelListMenuAction;
    procedure DoNothingProc(selfptr : CellPtr);
    procedure Weightslistmenuaction;
    procedure Weightsloadbutton;
    procedure ShowWeightsCellData(selfptr : CellPtr);
    procedure WeightsSaveButton;
    procedure Weightssaveaction;
    procedure Weightslistmenuview;
    procedure WeightAction;
    procedure AuxWeightAction;
    procedure GetWeightSet(eptr:emenuptr;CGptr : CellGridPtr; fname : string);
    procedure WeightsCellGridAction(selfptr : CellPtr; cellnumber:integer);
    procedure WeightsCellGridCancel(selfptr : CellPtr);
    procedure Weight_or_Region_Button;
    procedure DrawCells(selfptr : CellPtr);
    procedure WeightsGroupButton;
    procedure differencebutton;
    procedure differenceacceptbutton;
    procedure DiffCellGridAction(selfptr : CellPtr; cellnumber:integer);
    procedure ShowDiffCellData(selfptr : CellPtr);
    procedure NextAction;
    procedure PreviousAction;

Implementation

CONST NUMBERCELLS = 444;
```

File Name: PROCS.PAS

```
VAR
      SNRThreshold,ShortLongThreshold,PhaseThreshold,
         XAngleThreshold,GDOPThreshold: integer;
      DMThreshold: boolean;


{$F+}


(*****************************************************************************)

   procedure CopyCells(fromptr,toptr:cellgridptr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           This procedure will copy the contents of a cellgrid to        *
      *           another when we split or expand screens                       *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
   var
      i: integer;
   begin
      for i := 1 TO NUMBERCELLS do
      begin
         toptr^.cellarray[i].abovethreshold := fromptr^.cellarray[i].abovethreshold ;
         toptr^.cellarray[i].region := fromptr^.cellarray[i].region ;
         toptr^.cellarray[i].weight_or_coverage :=
                                    fromptr^.cellarray[i].weight_or_coverage;
         toptr^.cellarray[i].Pac := fromptr^.cellarray[i].Pac ;
      end;
   end;

(*****************************************************************************)

   procedure quitbutton;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           This procedure handles the action of exiting the PACE         *
      *           system. It checks to ensure that the user will not lose       *
      *           results accidentally by forgetting to save them.              *
      *                                                                         *
      ***************************************************************************
      }
```

File Name: PROCS.PAS

```
var
  f : file;
  key,button : integer;

begin
  if (LeftLowerStatus^.archivefile = LEFTDEFAULTARCHIVE) or
     (RightLowerStatus^.archivefile = RIGHTDEFAULTARCHIVE) or
     (topofbatchqueue <> NIL) then begin
    messagebox.displaymessage((GetMaxX DIV 2) -
        (textwidth('12345678901234567890123456789012345678') DIV 2),
        (GetMaxY DIV 2), 7,
        length('12345678901234567890123456789012345678')
        ,lightred,black,
      (*'12345678901234567890123456789012345678'*)
        '  ********************************  '+
        ' <<< WARNING - DATA MAY BE LOST >>> '+
        '  ********************************  '+
        '                                    '+
        '       Unsaved Scenarios or         '+
        '         Batch Files Exist          '+
        '           Exit PACE ?              ');
    beep;
  end
  else begin
    messagebox.displaymessage((GetMaxX DIV 2) -
        (textwidth('Exit PACE ?') DIV 2),
        GetMaxY DIV 2,1,
        length('Exit PACE ?')
        ,lightred,white,
        'Exit PACE ?');
    beep;
  end;

  repeat
    Get_Key(key);
    button := Get_Buttons;
  until (key = ENTER) or
        (key = ESC)   or
        (upcase(chr(key)) = 'Y') or
        (upcase(chr(key)) = 'N') or
        (button <> 0);

  messagebox.hidemessage;

  if (key = ENTER) or
     (button = left_button_pressed) or
     (upcase(chr(key)) = 'Y') then
  begin
    QUIT := TRUE;
```

```
    else
      QUIT := FALSE;
  end;
(*****************************************************************************)

  procedure helpbutton;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *          This procedure invokes the help subsystem by pushing the       *
    *          help system context onto the context stack and then showing    *
    *          the context.                                                   *
    *                                                                         *
    ***************************************************************************
    }
  begin
    HelpMenuWindow^.HelpFileName := FileName(CurrentContext);
    pushcontext('helpmenuwindow');
    showcontext;
  end;

(*****************************************************************************)

  procedure filebutton;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *          This procedure activates the file menu for the main            *
    *          cell display screen.                                           *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }
  begin
    pushcontext('filemenu');
    showcontext;
  end;

(*****************************************************************************)

  procedure optionsbutton;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *          This procedure activates the options menu on the weights       *
    *          editor screen.                                                 *
```

```
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      pushcontext('optionsmenu');
      showcontext;
   end;

(************************************************************************)

   procedure leftfilebutton;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure activates the file menu for the left split  *
      *           cell display screen.                                       *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      pushcontext('leftfilemenu');
      showcontext;
   end;

(************************************************************************)

   procedure rightfilebutton;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure activates the file menu for the right split *
      *           cell display screen.                                       *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      pushcontext('rightfilemenu');
      showcontext;
   end;

(************************************************************************)

   procedure editbutton;
      {
```

File Name: PROCS.PAS

```
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure activates the edit menu for the main        *
      *           cell display screen.                                       *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      IF rightexpanded THEN
         pushcontext('auxeditmenu')
      ELSE
         pushcontext('editmenu');
      showcontext;
   end;

(************************************************************************)

   procedure loadbutton;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure activates the load menu for the main and    *
      *           left cell display screen.                                  *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      IF rightexpanded THEN
         pushcontext('rightlistmenu')
      ELSE
         pushcontext('listmenu');
      showcontext;
   end;

(************************************************************************)

   procedure savebutton;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure activates the save menu for the main        *
      *           cell display screen.                                       *
      *                                                                      *
      *                                                                      *
```

```
      ********************************************************************************
      }
   begin
      IF Rightexpanded THEN
      begin
         if RightLowerStatus^.archivefile <> RIGHTDEFAULTARCHIVE then
           SaveMenuWindow^.FileButton.Change_String(RightLowerStatus^.archivefile);
         SaveMenuWindow^.MemoBox.GetNewText(RightSaveComment);
         pushcontext('savemenuwindow');
      end
      ELSE
      begin
         if LeftLowerStatus^.archivefile <> LEFTDEFAULTARCHIVE then
           SaveMenuWindow^.FileButton.Change_String(LeftLowerStatus^.archivefile);
         SaveMenuWindow^.MemoBox.GetNewText(LeftSaveComment);
         pushcontext('savemenuwindow');
      end;

      showcontext;
   end;

(*******************************************************************************)

   procedure auxsavebutton;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           This procedure activates the save menu for the right split         *
      *           cell display screen.                                               *
      *                                                                              *
      *                                                                              *
      ********************************************************************************
      }
   begin
      if RightLowerStatus^.archivefile <> RIGHTDEFAULTARCHIVE then
        AuxSaveMenuWindow^.FileButton.Change_String(RightLowerStatus^.archivefile);
      AuxSaveMenuWindow^.MemoBox.GetNewText(RightSaveComment);
      pushcontext('auxsavemenuwindow');
      showcontext;
   end;

(*******************************************************************************)

   procedure returnbutton;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
```

File Name: PROCS.PAS

```
      *          This procedure causes a return from the weights editor to  *
      *          the edit menu. It checks to see if the user has changed    *
      *          anything and warns the user if data may be lost.           *
      *                                                                     *
      ***********************************************************************
      }
   VAR
      key,button : integer;
   begin
      IF WeightCellGrid^.HasBeenChanged THEN
      begin
         messagebox.displaymessage((GetMaxX DIV 2) -
              (textwidth('12345678901234567890123456789012345678') DIV 2),
              (GetMaxY DIV 2), 7,
              length('12345678901234567890123456789012345678')
              ,lightred,black,
            (*'12345678901234567890123456789012345678'*)
              '                                        '+
              '  ********************************    '+
              ' <<< WARNING - DATA MAY BE LOST >>>   '+
              '  ********************************    '+
              '                                        '+
              '  Unsaved Weight/Region Set Exists    '+
              '    Exit Weight/Region Editor ?       ');
         beep;

         repeat
           Get_Key(key);
           button := Get_Buttons;
         until (key = ENTER) or
               (key = ESC)   or
               (upcase(chr(key)) = 'Y') or
               (upcase(chr(key)) = 'N') or
               (button <> 0);

         messagebox.hidemessage;

         if NOT((key = ENTER) or
            (button = left_button_pressed) or
            (upcase(chr(key)) = 'Y')) then
            exit;
      end;
      hide_cursor;
      popcontext;
      optionsmenu^.buttonarray[2].HotKey := 'R';
      optionsmenu^.buttonarray[2].ChangeName('Region');
      WeightCellGrid^.eptr^.weightsfile := WeightCellGrid^.weightsfile;
      setvisualpage(0);
      setactivepage(0);
```

```
      set_cursor_page(0);
      Set_Cursor_Shape(arrow);
      show_cursor((GetMaxX DIV 2),(GetMaxY DIV 2));
   end;

(*************************************************************************)

   procedure CoverageReturnButton;
      {
      ************************************************************************
      *                                                                      *
      *       PURPOSE                                                        *
      *          This procedure causes a return from the coverage display    *
      *          screen to the detailed cell query level.                    *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      hide_cursor;
      popcontext;
(*      switchcontext('editmenu'); *)
      CoverageCellGrid^.cellarray[CellPopUp^.HilitedCellPtr^.cellnumber].unhilite;

      setvisualpage(0);
      setactivepage(0);
      set_cursor_page(0);
      show_cursor((GetMaxX DIV 2),(GetMaxY DIV 2));
   end;

(*************************************************************************)
   procedure DifferenceReturnButton;
      {
      ************************************************************************
      *                                                                      *
      *       PURPOSE                                                        *
      *          This procedure causes a return from the difference display  *
      *          screen to the split screen display.                         *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      hide_cursor;
      popcontext;
      setvisualpage(0);
      setactivepage(0);
      set_cursor_page(0);
      show_cursor((GetMaxX DIV 2),(GetMaxY DIV 2));
```

```
   end;

(*****************************************************************************)

  procedure saveaction;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           This procedure performs the actions necessary to save a       *
      *           scenario and it's results to an archive file for the main     *
      *           and left split cell displays.                                 *
      *                                                                         *
      ***************************************************************************
      }
  var
    filename  : string;
    Source,AF : ArchiveFile;
    i         : integer;
    temp      : archivedcelltype;
    dir       : DirStr;   (* as defined in the DOS unit *)
    name      : NameStr;
    ext       : ExtStr;

  begin
    IF leftlowerstatus^.archivefile <> '' THEN
    begin
      filename := SaveMenuWindow^.FileButton.Text_String;
      fsplit(filename,dir,name,ext);
      filename := name + ext;
      if filename = '' then
      begin
        messagebox.displaymessage((GetMaxX DIV 2) -
             (textwidth('Please Select An Appropriate Filename before saving...') DIV 2),
             GetMaxY DIV 2,1,
             length('Please Select An Appropriate Filename before saving...')
             ,lightred,white,
             'Please Select An Appropriate Filename before saving...');
        beep;
        messagewait;
        messagebox.hidemessage;
      end
      else
      begin
        SaveMenuWindow^.MemoBox.AssignChangedText(LeftSaveComment);
        messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Saving to file: '+ARCHIVEPAT
                                  (GetMaxY DIV 2) - (textheight('1') DIV 2),
                                  1,
                                  length('Saving to file: '+ARCHIVEPATH+'\'+filename),
```

```
                                lightred;
                                white,
                                'Saving to file: '+ARCHIVEPATH+'\'+filename);
      Assign(AF,ARCHIVEPATH+'\'+filename);
      IF (fexpand(filename) = fexpand(leftlowerstatus^.archivefile)) THEN
      begin { if saving to the same name as currently loaded, just rewrite the
              parameter info and don't save anything else ('cause it's already
              there }
         Reset(AF);
         SaveArchiveParameters(AF,editmenu,Leftsavecomment);
         Close(AF);
      end
      ELSE
      begin { storing to a different file so it's OK to rewrite and copy }
         Rewrite(AF);
         Assign(source, ARCHIVEPATH+'\'+leftlowerstatus^.archivefile);
         Reset(source);
         SaveArchiveParameters(AF,editmenu,LeftSaveComment);
         FOR i := 1 TO NCells DO
         begin
            loadcellfromarchive(source,i,temp);
            storecelltoarchive(AF,i,temp);
         end;
         Close(AF);
         Close(source);
         leftlowerstatus^.archivefile := filename;
         leftlowerstatus^.show;
      end;
      messagebox.hidemessage;
      hidecontext; popcontext; (* wipe out the save memo window *)
      hidecontext; popcontext; (* wipe out the file pick list drop down menu *)
    end;
  end
  else
  begin
    messagebox.displaymessage((GetMaxX DIV 2) -
        (textwidth('Save Data Not Available, Process a Scenario First') DIV 2),
        GetMaxY DIV 2,1,
        length('Save Data Not Available, Process a Scenario First')
        ,lightred,white,
        'Save Data Not Available, Process a Scenario First');
    beep;
    messagewait;
    messagebox.hidemessage;
  end;
end;

(*****************************************************************************)
```

File Name: PROCS.PAS

```
procedure SaveWeightsParameters(filename : string;
                                Comment : CommentAreaType );
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *           This procedure performs the actions necessary to save a       *
     *           weight set to a weight file.                                  *
     *                                                                         *
     *                                                                         *
     ***************************************************************************
     }
var
  wd : WeightsDataType;
  WF : WeightsFile;
  i  : integer;

begin
  Assign(WF,filename);
  rewrite(WF);
  wd.Comment := Comment;
  for i := 1 to 444 do begin
    wd.WeightData[i] := WeightCellGrid^.CellArray[i].Weight_or_Coverage;
    IF WeightCellGrid^.CellArray[i].region THEN
    wd.WeightData[i] := wd.WeightData[i] OR $80;
  end;
  write(WF,WD);
  close(WF);
end;

(***************************************************************************)

  procedure auxsaveaction;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           This procedure performs the actions necessary to save a       *
      *           scenario and it's results to an archive file for the          *
      *           right    split cell displays.                                 *
      *                                                                         *
      ***************************************************************************
      }
  var
    dummy: char;
    errorstring,
    filename : string;
    Source,AF: ArchiveFile;
    i: integer;
```

File Name: PROCS.PAS

```
    temp:archivedcelltype;

  begin
    if rightlowerstatus^.archivefile <> '' THEN
    begin
      filename := AuxSaveMenuWindow^.FileButton.Text_String;
      if filename = '' then
      begin
        messagebox.displaymessage((GetMaxX DIV 2) -
             (textwidth('Please Select An Appropriate Filename before saving...') DIV 2),
             GetMaxY DIV 2,1,
             length('Please Select An Appropriate Filename before saving...')
             ,lightred,white,
             'Please Select An Appropriate Filename before saving...');
        beep;
        messagewait;
        messagebox.hidemessage;
      end
      else
      begin
        AuxSaveMenuWindow^.MemoBox.AssignChangedText(RightSaveComment);
        messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Saving to file: '+ARCHIVEPAT
                                  (GetMaxY DIV 2) - (textheight('1') DIV 2),
                                  1,
                                  length('Saving to file: '+ARCHIVEPATH+'\'+filename),
                                  lightred,
                                  white,
                                  'Saving to file: '+ARCHIVEPATH+'\'+filename);
        Assign(AF,ARCHIVEPATH+'\'+filename);
        IF (fexpand(filename) = fexpand(rightlowerstatus^.archivefile)) THEN
        begin { if saving to the same name as currently loaded, just rewrite the
                parameter info and don't save anything else ('cause it's already
                there }
           Reset(AF);
           SaveArchiveParameters(AF,auxeditmenu,RightSaveComment);
           Close(AF);
        end
        ELSE
        begin { storing to a different file so it's OK to rewrite and copy }
           Rewrite(AF);
           Assign(source,ARCHIVEPATH+'\'+Rightlowerstatus^.archivefile);
           Reset(source);
           SaveArchiveParameters(AF,auxeditmenu,RightSaveComment);
           FOR i := 1 TO NCells DO
           begin
              loadcellfromarchive(source,i,temp);
              storecelltoarchive(AF,i,temp);
           end;
           Close(AF);
```

```
          Close(source);
          rightlowerstatus^.archivefile := filename;
          rightlowerstatus^.show;
        end;
        messagebox.hidemessage;
        hidecontext; popcontext; (* wipe out the save memo window *)
        hidecontext; popcontext; (* wipe out the file pick list drop down menu *)
      end
    end
    else
    begin
      messagebox.displaymessage((GetMaxX DIV 2) -
          (textwidth('Save Data Not Available, Process a Scenario First') DIV 2),
          GetMaxY DIV 2,1,
          length('Save Data Not Available, Process a Scenario First'),
          lightred,white,
          'Save Data Not Available, Process a Scenario First');
      beep;
      messagewait;
      messagebox.hidemessage;
    end;
  end;

(*****************************************************************************)

  procedure auxloadbutton;
    {
    ******************************************************************************
    *                                                                            *
    *        PURPOSE                                                             *
    *          This procedure activates the load menu for the right split *
    *          cell display screen.                                              *
    *                                                                            *
    *                                                                            *
    ******************************************************************************
    }
  begin
    pushcontext('rightlistmenu');
    showcontext;
  end;

(*****************************************************************************)

  procedure auxeditbutton;
    {
    ******************************************************************************
    *                                                                            *
    *        PURPOSE                                                             *
    *          This procedure activates the edit menu for the right split *
```

```
      **          cell display screen.                                    *
      *                                                                   *
      *                                                                   *
      *********************************************************************
      }
   begin
      pushcontext('auxeditmenu');
      showcontext;
   end;

(*************************************************************************)

   procedure splitbutton;
      {
      *********************************************************************
      *                                                                   *
      *        PURPOSE                                                    *
      *           This procedure activates the split screen cell display  *
      *           mode.                                                   *
      *                                                                   *
      *                                                                   *
      *********************************************************************
      }
   begin
      IF rightexpanded THEN
      begin
         deletefromcontext('basescreen',rightlowerstatus);
         addtocontext('basescreen',leftlowerstatus);
         deletefromcontext('basescreen',rightsidestatus);
         addtocontext('basescreen',leftsidestatus);
         CopyCells(BigCellGrid,RightCellGrid);
      end
      ELSE
      begin
         CopyCells(BigCellGrid,LeftCellGrid);
      end;
      pushcontext('splitscreen');
      basescreen^.show;
      ShowContext;
      rightexpanded := FALSE;
      leftexpanded := FALSE;
   end;

(*************************************************************************)

   procedure rightexpandbutton;
      {
      *********************************************************************
      *                                                                   *
```

```
      *        PURPOSE                                                    *
      *           This procedure causes the right split screen display to be *
      *           displayed in full screen mode.                          *
      *                                                                   *
      *                                                                   *
      *********************************************************************
      }
   begin
      deletefromcontext('basescreen',leftlowerstatus);
      addtocontext('basescreen',rightlowerstatus);
      deletefromcontext('basescreen',leftsidestatus);
      addtocontext('basescreen',rightsidestatus);
      CopyCells(RightCellGrid,BigCellGrid);
      rightexpanded := TRUE;
      leftexpanded := FALSE;
      HideContext;
      popcontext;
      ShowContext;
   end;

(*********************************************************************)

   procedure leftexpandbutton;
      {
      *********************************************************************
      *                                                                   *
      *        PURPOSE                                                    *
      *           This procedure causes the left  split screen display to be *
      *           displayed in full screen mode.                          *
      *                                                                   *
      *                                                                   *
      *********************************************************************
      }
   begin
      CopyCells(LeftCellGrid,BigCellGrid);
      rightexpanded := FALSE;
      leftexpanded := TRUE;
      HideContext;
      popcontext;
      ShowContext;
   end;

(*********************************************************************)


   procedure stub;
      {
      *********************************************************************
      *                                                                   *
```

```
      *        PURPOSE                                                      *
      *           This is a dummy procedure that is used to debug object    *
      *           definitions and operation.                                *
      *                                                                     *
      *                                                                     *
      ***********************************************************************
      }
   begin
      pushcontext('stub');
      showcontext;
   end;

(***************************************************************************)

   procedure SaveArchiveParameters(VAR AF:Archivefile; eptr:emenuptr;
                                   VAR comment:commentareatype);
      {
      ***********************************************************************
      *                                                                     *
      *        PURPOSE                                                      *
      *           This procedure stores the parameters associated with a    *
      *           particular scenario (i.e., a particular edit menu set)    *
      *           to be stored in a PACE archive file.                      *
      *                                                                     *
      ***********************************************************************
      }
   VAR
      params : record CASE Parameteraccess: boolean OF
                     TRUE: (ParameterStuff: SavedParamsRecord;
                            CommentArea:CommentAreaType);
                     FALSE:(CellInfo:ArchivedCellType);
               end;
   begin
      { Write the parameters into the first (0th) cell location. The
        parameters are written using a variant record format.
        The routine also re-creates the indicated archive file,
        wiping out all previous data in the file. }
      params.parameteraccess := TRUE;
      SaveEditParams(eptr,@params.parameterstuff);
      params.commentarea := comment;
      Seek(AF,0);
      params.parameteraccess := FALSE;
      write(AF,params.CellInfo);
   end;

(***************************************************************************)

   procedure RestoreArchiveParameters(VAR AF: Archivefile;eptr:emenuptr;
                                      VAR comment:commentareatype);
```

File Name: PROCS.PAS

```
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *           This procedure stores the parameters associated with a             *
    *           particular scenario (i.e., a particular edit menu set)             *
    *           to be loaded from a PACE archive file.                             *
    *                                                                              *
    ********************************************************************************
    }
  VAR
    params : record CASE Parameteraccess: boolean OF
                  TRUE: (ParameterStuff: SavedParamsRecord;
                         CommentArea:CommentAreaType);
                  FALSE:(CellInfo:ArchivedCellType);
             end;
  begin
    { Read the parameters from the first (0th) cell location. The
      storage is used as a placeholder for the comment field and
      the parameter area. }
    params.parameteraccess := FALSE;
    seek(AF,0);
    read(AF,params.CellInfo);
    params.parameteraccess := TRUE;
    RestoreEditParams(eptr,@params.ParameterStuff);
    comment := params.commentarea;
  end;

(*******************************************************************************)

  procedure ViewArchiveComment(VAR AF: Archivefile;VAR comment:commentareatype);
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *           This procedure reads the textual comment area associated           *
    *           with an archive file.                                              *
    *                                                                              *
    *                                                                              *
    ********************************************************************************
    }
  VAR
    params : record CASE Parameteraccess: boolean OF
                  TRUE: (ParameterStuff: SavedParamsRecord;
                         CommentArea:CommentAreaType);
                  FALSE:(CellInfo:ArchivedCellType);
             end;
  begin
    { Read the parameters from the first (0th) cell location. The
```

```
        storage is used as a placeholder for the comment field and
        the parameter area. }
      params.parameteraccess := FALSE;
      seek(AF,0);
      read(AF,params.CellInfo);
      params.parameteraccess := TRUE;
      comment := params.commentarea;
   end;

(*************************************************************************)

   procedure ViewWeightsComment(VAR WF: Weightsfile;VAR comment:commentareatype);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure reads the textual comment area associated   *
      *           with a  weights file.                                      *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }

   var
     wd : WeightsDataType;

   begin
      read(WF,wd);
      comment := wd.comment;
   end;

(*************************************************************************)
   procedure ViewQRComment(VAR AF: file;VAR comment:commentareatype);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure reads the textual comment area associated   *
      *           with a QR set.                                             *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      seek(AF,0);
      blockread(AF,comment,1);
   end;

(*************************************************************************)
```

File Name: PROCS.PAS

```
   procedure AddToQueue(eptr:emenuptr;smenu:savemenuptr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          This procedure adds the scenario parameters associated         *
      *          with an edit menu to the batch queue.                          *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
  var
    filename  : string;
    Source,AF : ArchiveFile;
    i         : integer;
    temp      : archivedcelltype;
    dir       : DirStr;   (* as defined in the DOS unit *)
    name      : NameStr;
    ext       : ExtStr;
    q         : batchqueueentryptr;
begin
    { get the name of the archive file and a comment using the save menu }
    filename := SMenu^.FileButton.Text_String;
    fsplit(filename,dir,name,ext);
    filename := name + ext;
    if filename = '' then
    begin
      messagebox.displaymessage((GetMaxX DIV 2) -
           (textwidth('Please Select An Appropriate Filename before queueing...') DIV 2),
           GetMaxY DIV 2,1,
           length('Please Select An Appropriate Filename before queueing...')
           ,lightred,white,
           'Please Select An Appropriate Filename before queueing...');
      beep;
      messagewait;
      messagebox.hidemessage;
    end
    else
      begin
        new(q);
        SMenu^.MemoBox.AssignChangedText(q^.CommentArea);
        messagebox.displaymessage((GetMaxX DIV 2) -
                                  (textwidth('Archive will be saved to file: '+ARCHIVEPATH+'
                                  (GetMaxY DIV 2) - (textheight('1') DIV 2),
                                  1,
                                  length('Archive will be saved to file: '+ARCHIVEPATH+'\'-f
                                  lightred,
                                  white,
                                  'Archive will be saved to file: '+ARCHIVEPATH+'\'+filename
```

```
        Delay(2000);
        { save the edit parameters }
        SaveEditParams(eptr,@q^.parameterstuff);
        q^.filename := filename;
        { put the info into the queue linked list for later processing }
        q^.next := NIL;
        IF topofbatchqueue = NIL THEN
        begin
           topofbatchqueue := q;
           batchqueue := q;
        end
        ELSE
           batchqueue^.next := q;
        batchqueue := q;
        messagebox.hidemessage;
        HideContext;
        CancelContext;
     end;
  end;
(*****************************************************************************)
 procedure QueueAction;
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *           This procedure obtains the scenario archive information to *
     *           be associated with a batch scenario.                          *
     *                                                                         *
     *                                                                         *
     ***************************************************************************
     }
 var
  savecomment: commentareatype;

 begin
     { pop up the save window to get the file name and comment }
     QueueSaveMenuWindow^.FileButton.Change_String('');
     QueueSaveMenuWindow^.MemoBox.GetNewText(QueueSaveComment);
     pushcontext('queuemenuwindow');
     showcontext;
 end;
(*****************************************************************************)
 procedure auxQueueAction;
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *           This procedure obtains the scenario archive information to *
     *           be associated with a batch scenario.                          *
```

```
      *                                                                    *
      *                                                                    *
      **********************************************************************
      }
  var
   savecomment: commentareatype;

  begin
      { pop up the save window to get the file name and comment }
      AuxQueueSaveMenuWindow^.FileButton.Change_String('');
      AuxQueueSaveMenuWindow^.MemoBox.GetNewText(QueueSaveComment);
      pushcontext('auxqueuemenuwindow');
      showcontext;
  end;
(**********************************************************************)
   procedure queuesaveAction;
      {
      **********************************************************************
      *                                                                    *
      *        PURPOSE                                                     *
      *          This procedure invokes the save to batch queue routine for *
      *          the left split and main cell displays.                    *
      *                                                                    *
      *                                                                    *
      **********************************************************************
      }
   begin
      AddToQueue(editmenu,QueueSaveMenuWindow);
   end;
(**********************************************************************)
procedure auxqueuesaveAction;
      {
      **********************************************************************
      *                                                                    *
      *        PURPOSE                                                     *
      *          This procedure invokes the save to batch queue routine for *
      *          the left split    cell display.                           *
      *                                                                    *
      *                                                                    *
      **********************************************************************
      }
   begin
      AddToQueue(auxeditmenu,AuxQueueSaveMenuWindow);
   end;
(**********************************************************************)
   procedure processbatchqueue;
      {
      **********************************************************************
      *                                                                    *
```

File Name: PROCS.PAS

```
      *        PURPOSE:                                                    *
      *           This procedure causes all scenarios in the batch queue to  *
      *           be processed and stored in archive files when finished when*
      *           invoked from the left split or main cell display.          *
      *                                                                      *
      ************************************************************************
      }
   VAR
      temp,temp2: batchqueueentryptr;
      i,xx,yy: integer;
      AF: ArchiveFile;
      t: archivedcelltype;
   begin
      temp := topofbatchqueue;
      get_cursorxy(xx,yy);
      hidecontext;
      popcontext;
      HiliteContext(xx,yy);
      WHILE temp <> NIL DO
      begin
         IF rightexpanded THEN
         begin
            RestoreEditParams(auxeditmenu,@temp^.parameterstuff);
            rightSaveComment := temp^.commentarea;
            rightLowerStatus^.ArchiveFile := temp^.filename;
            rightlowerstatus^.show;
            ComputePsa(auxeditmenu,rightlowerstatus);
         end
         ELSE
         begin
            RestoreEditParams(editmenu,@temp^.parameterstuff);
            LeftSaveComment := temp^.commentarea;
            LeftLowerStatus^.ArchiveFile := temp^.filename;
            leftlowerstatus^.show;
            ComputePsa(editmenu,leftlowerstatus);
         end;
         IF rightexpanded THEN
         begin
            rethresholdcells(BIGCellGrid,auxeditmenu,rightlowerstatus);
            BigCellGrid^.Show;
            BigMap^.SHOW;
            rightlowerstatus^.show;
         end
         ELSE
         begin
            IF leftexpanded THEN
               rethresholdcells(BIGCellGrid,editmenu,leftlowerstatus)
            ELSE
                rethresholdcells(LeftCellGrid,editmenu,leftlowerstatus);
```

```
            IF LeftExpanded THEN
            begin
               BigCellGrid^.Show;
               BigMap^.SHOW;
            end
            ELSE
            begin
               LeftCellGrid^.Show;
               LeftMap^.SHOW;
            end;
            leftlowerstatus^.show;
         end;
         { now save the scenario }

         IF rightexpanded THEN
         begin
            Assign(AF,ARCHIVEPATH + '\' + rightlowerstatus^.archivefile);
            Reset(AF);
            SaveArchiveParameters(AF,auxeditmenu,temp^.commentarea);
            Close(AF);
         end
         ELSE
         begin
            Assign(AF,ARCHIVEPATH + '\' + leftlowerstatus^.archivefile);
            Reset(AF);
            SaveArchiveParameters(AF,editmenu,temp^.commentarea);
            Close(AF);
         end;

         temp := Temp^.next;
      end;
      { clean up the linked list }
      temp := topofbatchqueue;
      WHILE temp <> NIL DO
      begin
         temp2 := temp^.next;
         dispose(temp);
         temp := temp2;
      end;
      topofbatchqueue := nil;
   end;


(*****************************************************************************)
   procedure auxprocessbatchqueue;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
```

File Name: PROCS.PAS

```
      *          This procedure causes all scenarios in the batch queue to  *
      *          be processed and stored in archive files when finished when*
      *          invoked from the right split cell display.                  *
      *                                                                      *
      ************************************************************************
      }
   VAR
      temp,temp2: batchqueueentryptr;
      i,xx,yy: integer;
      AF: ArchiveFile;
      t: archivedcelltype;
   begin
      temp := topofbatchqueue;
      get_cursorxy(xx,yy);
      hidecontext;
      popcontext;
      HiliteContext(xx,yy);
      WHILE temp <> NIL DO
      begin
         RestoreEditParams(auxeditmenu,@temp^.parameterstuff);
         rightSaveComment := temp^.commentarea;
         rightLowerStatus^.ArchiveFile := temp^.filename;
         rightlowerstatus^.show;
         ComputePsa(auxeditmenu,rightlowerstatus);
         rethresholdcells(rightCellGrid,auxeditmenu,rightlowerstatus);
         rightCellGrid^.Show;
         rightMap^.SHOW;
         rightlowerstatus^.show;
         { now save the scenario }

         Assign(AF,ARCHIVEPATH + '\' + rightlowerstatus^.archivefile);
         Reset(AF);
         SaveArchiveParameters(AF,auxeditmenu,temp^.commentarea);
         Close(AF);

         temp := Temp^.next;
      end;
      { clean up the linked list }
      temp := topofbatchqueue;
      temp2 := topofbatchqueue;
      WHILE temp <> NIL DO
      begin
         temp2 := temp^.next;
         dispose(temp);
         temp := temp2;
      end;
      topofbatchqueue := nil;
   end;
```

File Name: PROCS.PAS

```
(*****************************************************************************)

   procedure editmenuprocess;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           This procedure causes the scenario that has been set up in *
      *           the main or left split screen cell display to be processed.*
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
   VAR
      XX,YY:integer;
   begin
      HideContext; PopContext; { don't use cancel context here because the
                                 emenu cancel restores the edit params from
                                 the previous set }
      CancelContext;
      Get_CursorXY(XX,YY);
      HiliteContext(XX,YY);
      leftsidestatus^.show;
      IF (editmenu^.recompute) OR (LeftLowerStatus^.archivefile = '') THEN
      begin
         LeftLowerStatus^.ArchiveFile := LEFTDEFAULTARCHIVE;
         leftlowerstatus^.show;
         ComputePsa(editmenu,leftlowerstatus);
      end
      else
        leftlowerstatus^.show;

      { IF we are re-thresholding an already loaded archive, then
        use that archive name. Else, use the default archive. If there
        is NO archive because we have not loaded any and we have
        not yet performed a Psa calulation since starting pace,
         then create one using the temporary archive file. }
      IF leftexpanded THEN
         rethresholdcells(BIGCellGrid,editmenu,leftlowerstatus)
      ELSE
          rethresholdcells(LeftCellGrid,editmenu,leftlowerstatus);
      IF LeftExpanded THEN
      begin
         BigCellGrid^.Show;
         BigMap^.SHOW;
      end
      ELSE
      begin
         LeftCellGrid^.Show;
```

```
        LeftMap^.SHOW;
      end;
      leftlowerstatus^.show;
   end;

(*************************************************************************)

   procedure auxeditmenuprocess;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure causes the scenario that has been set up in *
      *           the right split screen cell display to be processed.       *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   VAR
      XX,YY:integer;
   begin
      HideContext; PopContext; { don't use cancelcontext here because the
                                 emenu cancel restores the edit params from
                                 the previous set }
      CancelContext;
      Get_CursorXY(XX,YY);
      HiliteContext(XX,YY);
      rightsidestatus^.show;
      { load up the process from the right edit menu }
      IF (auxeditmenu^.recompute) OR
            (RightLowerStatus^.archivefile = '') THEN
      begin
         RightLowerStatus^.ArchiveFile := RIGHTDEFAULTARCHIVE;
         rightlowerstatus^.show;
         ComputePsa(auxeditmenu,rightlowerstatus);
      end
      else
         rightlowerstatus^.show;
      { IF we are re-thresholding an already loaded archive, then
        use that archive name. Else, use the default archive. If there
        is NO archive because we have not loaded any and we have
        not yet performed a Psa calulation since starting pace,
        then create one using the temporary archive file. }
      IF rightexpanded THEN
         rethresholdcells(BIGCellGrid,auxeditmenu,rightlowerstatus)
      ELSE
         rethresholdcells(RightCellGrid,auxeditmenu,rightlowerstatus);
      IF RightExpanded THEN
      begin
```

```
        BigCellGrid^.Show;
        BigMap^.SHOW;
      end
      ELSE
      begin
        RightCellGrid^.Show;
        RightMap^.SHOW;
      end;
      rightlowerstatus^.show;
   end;

(*****************************************************************************)

procedure ShowCellData(selfptr: CellPtr);
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *          Utility routine to display the appropriate colors for a           *
      *          cell. Gets bound to all cells in the cell grid object.            *
      *                                                                            *
      *                                                                            *
      ******************************************************************************
      }

var
  xx,yy : integer;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
   if selfptr^.IsHilited then
       selfptr^.SetHilite(False)
   else
   begin
      if selfptr^.IsAboveThreshold then
         selfptr^.color := selfptr^.abovecolor
      else
         selfptr^.color := selfptr^.belowcolor;
      setfillstyle(solidfill,selfptr^.color);
      bar(selfptr^.X,
          selfptr^.Y,
          selfptr^.X +
          selfptr^.Width,
          selfptr^.Y+
          selfptr^.Height);
   end;
  setcolor(selfptr^.BorderColor);
  setlinestyle(solidln,0,normwidth);
```

```
  rectangle(selfptr^.x,
            selfptr^.y,
            selfptr^.x +
            selfptr^.width,
            selfptr^.y +
            selfptr^.Height);
  Show_Cursor(xx,yy);
end;

(*****************************************************************************)

procedure DrawCells(selfptr: CellPtr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Utility routine to display the appropriate colors for a        *
      *          cell for the weights editor.                                   *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }

begin
  setcolor(selfptr^.BorderColor);
  setlinestyle(solidln,0,normwidth);
  rectangle(selfptr^.x, selfptr^.y,
            selfptr^.x + selfptr^.width, selfptr^.y + selfptr^.Height);
  ShowWeightsCellData(selfptr);
end;

(*****************************************************************************)

procedure ShowWeightsCellData(selfptr: CellPtr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *          Utility routine to display the appropriate weight for a        *
      *          cell for the weights editor.                                   *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }

var
  xx,yy : integer;
  tmp   : string;
```

File Name: PROCS.PAS

```
begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  STR(selfptr^.Weight_or_Coverage:2, tmp);
  settextstyle(smallfont,horizdir,4);
  setcolor(Yellow);
  settextjustify(centertext,centertext);
  outtextxy(selfptr^.X + (selfptr^.Width DIV 2),
            selfptr^.Y + (selfptr^.Height DIV 2), tmp);
  settextstyle(defaultfont,horizdir,1);
  IF NOT WeightCellGrid^.pickingweights THEN
     IF selfptr^.region THEN
        selfptr^.showregion
     ELSE
        selfptr^.hideregion;
  Show_Cursor(xx,yy);
end;

(************************.*************************************************)

procedure ShowCoverageCellData(selfptr: CellPtr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility routine to display the appropriate color  for a       *
      *           cell for the coverage display.                                *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
VAR
   xx,yy: integer;
begin
   { get the database data for each cell and color it appropriately.
     The coloring scheme is as follows:

     SNR - red
     S/L ratio - light red
     Modal - M in the cell center
     XAngle - X across the cell

     The various object variables associated with a cell type variable are
     used to hold flags that indicate the condition of the above threshold
     tests. These are related as follows:

     selfptr^.abovecolor - SNR or S/L color (red or light red or blue of OK)
     selfptr^.belowcolor - 0 if not modal, modal
     selfptr^.weight or coverage - 0 if Xangle OK, else not OK
```

```
   }

   Get_CursorXY(xx,yy);
   Hide_Cursor;
   setcolor(selfptr^.BorderColor);
   setlinestyle(solidln,0,normwidth);
   rectangle(selfptr^.x,
             selfptr^.y,
             selfptr^.x +
             selfptr^.width,
             selfptr^.y +
             selfptr^.Height);

   setfillstyle(solidfill,selfptr^.abovecolor);
   bar(selfptr^.X+1,
         selfptr^.Y+1,
         selfptr^.X +
         selfptr^.Width-1,
         selfptr^.Y+
         selfptr^.Height-1);
   IF selfptr^.weight_or_coverage <> 0 THEN
   begin
      setcolor(black);
      line(selfptr^.x+1,selfptr^.y+1,
           selfptr^.x+selfptr^.width-1,selfptr^.y+selfptr^.height-1);
      line(selfptr^.x+selfptr^.width-1,selfptr^.y+1,
           selfptr^.x+1,selfptr^.y+selfptr^.height-1);
   end;
   IF selfptr^.belowcolor <> 0 THEN
   begin
      setcolor(white);
      settextjustify(centertext,toptext);
      outtextxy(selfptr^.x+selfptr^.width DIV 2,
                selfptr^.y+selfptr^.height DIV 2,
                'M');
   end;
   Show_Cursor(xx,yy);
end;

(*****************************************************************************)

   procedure CellAction(cellnumber: integer;
                        eptr:emenuptr; archivename: string);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Procedure to build and display the temporal (hour/month)      *
      *           display of the summary cell query display.                    *
```

File Name: PROCS.PAS

```
      *                                                                    *
      *                                                                    *
      **********************************************************************
      }

   var
     errorstring : string;
     mon,hr,month,hour:integer;
     cellstuff: archivedcelltype;
     AF: Archivefile;
   begin

      { First copy the cell info into the global info areas.
        Next, reach into the archive file and pull out the info
        for the particular cell in question and fill in the
        cells of the CellPopWindow object. }

      { Open up the archive file indicated and get the Pa values
        for each hour,and the coverage for each hour also. Check the
        archive name and, if it is null, do not pop up the info window.}

      IF archivename <> '' THEN
      begin
         Assign(AF,ARCHIVEPATH+'\'+archivename);
         Reset(AF);
         LoadCellFromArchive(AF,cellnumber,cellstuff);
         Close(AF);

         FOR month := 1 TO 12 DO
            FOR hour := 1 TO 24 DO
            begin
               CellPopup^.subcells[month,hour].Pac := CellStuff[month,hour].Pat;
               CellPopup^.subcells[month,hour].Weight_or_coverage :=
                  CellStuff[month,hour].coverage;
               IF CellStuff[month,hour].Pat < eptr^.Psa.Value.Value THEN
                  CellPopup^.subcells[month,hour].abovethreshold := FALSE
               ELSE
                  CellPopup^.subcells[month,hour].abovethreshold := TRUE;
            end;

         for mon := 1 to 12 do
           for hr := 1 to 24 do
            with CellPopUp^ do begin
               SubCells[mon,hr].ChangeXY(X + xoffset +
                                         ((mon-1) * cellwidth),
                                         Y + yoffset +
                                         ((hr-1) * cellheight) );
            end;
```

```
        CellPopUp^.CellNum := cellnumber;
        CellPopUp^.eptr := eptr;
        PushContext('CellPopUp');
        ShowContext;
      end
      else begin
        messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Archive File Not Loaded...')
                                  (GetMaxY DIV 2) - (textheight('1')),
                                  1,
                                  length('Archive File Not Loaded...'),
                                  lightred,
                                  white,
                                  ('Archive File Not Loaded...'));
        beep;
        messagewait;
        messagebox.hidemessage;
        CellPopUp^.HilitedCellPtr^.Cancel;
      end;
    end;

(*****************************************************************************)

    procedure RightCellGridAction(selfptr : CellPtr; cellnumber:integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Procedure to set up for a summary cell display using cell     *
      *           action procedure for the right split cell grid display.       *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }

    const
      WindowOffset = 10;  (* distance between the cell and the window edge *)

    var
      mon, hr, xx, yy : integer;

    begin
      Get_CursorXY(xx,yy);
      selfptr^.Hilite(xx,yy);
      with RightCellGrid^ do begin
        if (CellArray[cellnumber].X < (GetMaxX - RightSideStatus^.width -
            CellArray[cellnumber].Width -
            CellPopUp^.Width - WindowOffset - WindowShadowWidth) ) then
          CellPopUp^.X := CellArray[cellnumber].X +
                          CellArray[cellnumber].Width +
```

```
                                WindowOffset
        else
          CellPopUp^.X := CellArray[cellnumber].X -
                                CellPopUp^.Width -
                                WindowOffset;
       end;
     CellPopUp^.HilitedCellPtr := @RightCellGrid^.CellArray[cellnumber];
     CellAction(cellnumber,auxeditmenu,rightlowerstatus^.archivefile);
  end;

(*****************************************************************************)
  procedure LeftCellGridAction(selfptr : CellPtr; cellnumber:integer);
     {
     ******************************************************************************
     *                                                                            *
     *        PURPOSE                                                             *
     *           Procedure to set up for a summary cell display using cell        *
     *           action procedure for the left  split cell grid display.          *
     *                                                                            *
     *                                                                            *
     ******************************************************************************
     }

  const
    WindowOffset = 10;  (* distance between the cell and the window edge *)

  var
    mon, hr,xx,yy : integer;

  begin
      Get_CursorXY(xx,yy);
      selfptr^.Hilite(xx,yy);
      with LeftCellGrid^ do begin
        if (CellArray[cellnumber].X < (GetMaxX - RightSideStatus^.width -
            CellArray[cellnumber].Width -
            CellPopUp^.Width - WindowOffset - WindowShadowWidth) ) then
          CellPopUp^.X := CellArray[cellnumber].X +
                          CellArray[cellnumber].Width +
                          WindowOffset
        else
          CellPopUp^.X := CellArray[cellnumber].X -
                          CellPopUp^.Width -
                          WindowOffset;
      end;
     CellPopUp^.HilitedCellPtr := @LeftCellGrid^.CellArray[cellnumber];
     CellAction(cellnumber,editmenu,leftlowerstatus^.archivefile);
  end;

(*****************************************************************************)
```

File Name: PROCS.PAS

```
procedure BigCellGridAction(selfptr : CellPtr; cellnumber:integer);
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           Procedure to set up for a summary cell display using cell     *
   *           action procedure for the main        cell grid display.      *
   *                                                                         *
   *                                                                         *
   ***************************************************************************
   }

(* CellAction defines the action to occur when either the <ENTER> key *)
(* or the <LEFT MOUSE BUTTON> is hit in full-screen mode or when the *)
(* cursor XY position is over the left map in split-screen mode.      *)
(*      This procedure will determine the X coordinate for the        *)
(* CellPopUp window (the Y coordinate stays fixed because the window  *)
(* extends from the map top to the map bottom) and will display the   *)
(* pop up window.                                                     *)

const
  WindowOffset = 10;  (* distance between the cell and the window edge *)

var
  mon, hr,xx,yy : integer;

begin
    Get_CursorXY(xx,yy);
    selfptr^.Hilite(xx,yy);
    with BigCellGrid^ do begin
        if (CellArray[cellnumber].X < (GetMaxX - RightSideStatus^.width -
            CellArray[cellnumber].Width - CellPopUp^.Width -
            WindowOffset-WindowShadowWidth) ) then
          CellPopUp^.X := CellArray[cellnumber].X +
                          CellArray[cellnumber].Width +
                          WindowOffset
        else
          CellPopUp^.X := CellArray[cellnumber].X -
                          CellPopUp^.Width -
                          WindowOffset;
    end;
   CellPopUp^.HilitedCellPtr := @BigCellGrid^.CellArray[cellnumber];
   IF leftexpanded THEN
      CellAction(cellnumber,editmenu,leftlowerstatus^.archivefile)
   ELSE { right expanded }
      CellAction(cellnumber,auxeditmenu,rightlowerstatus^.archivefile)
end;
```

File Name: PROCS.PAS

```
(*****************************************************************************)

   procedure SubCellAction(selfptr : CellPtr; cellnumber:integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Procedure to display the detailed cell query when a query     *
      *           on the summary cell query display is performed.               *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
   const
     WindowOffset = 10;
     BottomWindowOffset = 110; (*for vertical positioning *)
     BottomWindowPosition = 50;(*add to top window Y position to determine *)
                               (*the Y position for the bottom window *)

   var xx,yy : integer;

   begin

       get_cursorxy(xx,yy);
       selfptr^.Hilite(xx,yy);

       with CellPopUp^ do begin
         if (SubCells[Month,Hour].X < (GetMaxX - RightSideStatus^.width -
             SubCells[Month,Hour].Width - SubCellPopUp^.Width -
             WindowOffset-WindowShadowWidth) ) then
         begin
                   (* put pop-up window to the RIGHT of the selected cell *)
           SubCellPopUp^.X := SubCells[Month,Hour].X +
                              SubCells[Month,Hour].Width +
                              WindowOffset;
           SubCellPopUp^.BottomMenu.X := SubCellPopUp^.X;
         end
         else
         begin
                   (* put pop-up window to the LEFT of the selected cell *)
           SubCellPopUp^.X := SubCells[Month,Hour].X -
                              SubCellPopUp^.Width -
                              WindowOffset;
           SubCellPopUp^.BottomMenu.X := SubCellPopUp^.X;
         end;

         if ((SubCells[Month,Hour].Y+SubCells[Month,Hour].Height+
              BottomWindowOffset) < (GetMaxY - LeftLowerStatus^.Height -
             SubCellPopUp^.Height - WindowOffset-WindowShadowWidth) ) then
```

```
        begin
                (* put pop-up window BELOW the selected cell *)
          SubCellPopUp^.Y := SubCells[Month,Hour].Y +
                             SubCells[Month,Hour].Height +
                             WindowOffset;
          SubCellPopUp^.BottomMenu.Y := SubCellPopUp^.Y + BottomWindowPosition;
        end
        else
        begin
                (* put pop-up window ABOVE the selected cell *)
          SubCellPopUp^.Y := SubCells[Month,Hour].Y - SubCellPopUp^.Height -
                             BottomWindowOffset - WindowOffset;
          SubCellPopUp^.BottomMenu.Y := SubCellPopUp^.Y + BottomWindowPosition;
        end;

        if SubCellPopUp^.Y < (q^.Y + q^.Height + WindowOffset) then
        begin
          SubCellPopUp^ Y := (q^.Y + q^.Height + WindowOffset);
          SubCellPopUp^.BottomMenu.Y := SubCellPopUp^.Y + BottomWindowPosition;
        end;
      end;

    PushContext('SubCellPopUp');
    ShowContext;

  end;

(*****************************************************************************)

  Procedure archivemenuview(lmenu:lmenuptr);
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Procedure to display the comment associated with an         *
    *          archived scenario file.                                     *
    *                                                                      *
    *                                                                      *
    ************************************************************************
    }
  VAR
    errorstring : string;
    tempstring: commentareatype;
    AF: Archivefile;
  begin
    IF lmenu^.directorylist.picked > 0 THEN
    begin { read the stuff in }
      Assign(AF,ARCHIVEPATH+'\'+lmenu^.selectedfile);
      Reset(AF);
```

```
        ViewArchiveComment(AF,tempstring);
        Close(AF);
        messagebox.displaymessage((GetMaxX DIV 2) - (25*textwidth(' ')),
                                  GetMaxY DIV 2,
                                  5,50,cyan,black,tempstring);
        messagewait;
        messagebox.hidemessage;
      end
      ELSE
      begin
       messagebox.displaymessage((GetMaxX DIV 2) -
            (textwidth('File Not Selected') DIV 2),
            GetMaxY DIV 2,1,
            length('File Not Selected'),lightred,white,
            'File Not Selected');
       beep;
       messagewait;
       messagebox.hidemessage;
     end;
   end;

   Procedure QRmenuview(lmenu:lmenuptr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Procedure to display the comment associated with a            *
      *           QR set file.                                                  *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
   VAR
      tempstring: commentareatype;
      AF: file;
   begin
      IF lmenu^.directorylist.picked > 0 THEN
      begin { read the stuff in }
        Assign(AF,QRPATH+'\'+lmenu^.selectedfile);
        Reset(AF,Sizeof(commentareatype));
        ViewQRComment(AF,tempstring);
        Close(AF);
        messagebox.displaymessage((GetMaxX DIV 2) - (25*textwidth(' ')),
                                  GetMaxY DIV 2,
                                  5,50,cyan,black,tempstring);
        messagewait;
        messagebox.hidemessage;
      end
      ELSE
```

```
      begin
       messagebox.displaymessage((GetMaxX DIV 2) -
             (textwidth('File Not Selected') DIV 2),
             GetMaxY DIV 2,1,
             length('File Not Selected'),lightred,white,
             'File Not Selected');
       beep;
       messagewait;
       messagebox.hidemessage;
     end;
   end;

   procedure listmenuview;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Procedure to display the comment associated with an                *
      *           archive file for the left split and main cell displays.            *
      *                                                                              *
      *                                                                              *
      ********************************************************************************
      }
   begin
      archivemenuview(listmenu);
   end;

   procedure rightlistmenuview;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Procedure to display the comment associated with an                *
      *           archive file for the right split cell display.                     *
      *                                                                              *
      *                                                                              *
      ********************************************************************************
      }
   begin
      archivemenuview(rightlistmenu);
   end;

   procedure reliabilitymenuview;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Procedure to display the comment associated with an                *
      *           QR set  file for the left and main cell display.                   *
```

```
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      QRmenuview(reliabilitylistmenu);
   end;

   procedure auxreliabilitymenuview;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Procedure to display the comment associated with an        *
      *           QR set  file for the right cell display.                   *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
   begin
      QRmenuview(auxreliabilitylistmenu);
   end;

(*************************************************************************)

   procedure listmenuaction;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Procedure to get the name of the archive file to be loaded *
      *           and to perform the actions necessary to load the info into *
      *           the left and main cell grid displays.                      *
      *                                                                      *
      ************************************************************************
      }
   VAR
      errorstring : string;
      AF: ArchiveFile;
   begin
      { Get the name of the archive to read from the listmenu and
        read the parameters and the archive data into the exitmenu
        or auxeditmenu areas (depending upon which is currently active
        in the split screen or on the main screen) and fill in the
        appropriate cell grid data for the hours selected in the loaded
        parameters }
      IF listmenu^.directorylist.picked > 0 THEN
      begin
         leftlowerstatus^.archivefile := listmenu^.selectedfile;
```

```
        Assign(AF,ARCHIVEPATH+'\'+leftlowerstatus^.archivefile);
        Reset(AF);
        RestoreArchiveParameters(AF,editmenu,LeftSaveComment);
        Close(AF);
        CancelContext; { pop down the list menu }
        CancelContext; { pop down the file menu }
        LeftSideStatus^.Show;
        LeftLowerStatus^.Show;
        IF leftexpanded THEN
           ReThresholdCells(BigCellGrid, editmenu, leftlowerstatus)
        ELSE
           ReThresholdCells(LeftCellGrid, editmenu, leftlowerstatus);
        IF LeftExpanded THEN
        begin
           BigCellGrid^.Show;
           BigMap^.SHOW;
        end
        ELSE
        begin
           LeftCellGrid^.Show;
           LeftMap^.SHOW;
        end;
        LeftLowerStatus^.eptr := editmenu;
        LeftLowerStatus^.Show;
     end
     ELSE
     begin
      messagebox.displaymessage((GetMaxX DIV 2) -
            (textwidth('File Not Selected') DIV 2),
           GetMaxY DIV 2,1,
           length('File Not Selected'),lightred,white,
           'File Not Selected');
      beep;
      messagewait;
      messagebox.hidemessage;
    end;
  end;

(*****************************************************************************)

  procedure rightlistmenuaction;
     {
     ***************************************************************************
     *                                                                         *
     *        PURPOSE                                                          *
     *           Procedure to get the name of the archive file to be loaded *
     *           and to perform the actions necessary to load the info into *
     *           the right split   cell grid display.                          *
     *                                                                         *
```

File Name: PROCS.PAS

```
      ********************************************************************************
      }
   VAR
      errorstring : string;
      AF: ArchiveFile;
   begin
      { Get the name of the archive to read from the listmenu and
        read the parameters and the archive data into the exitmenu
        or auxeditmenu areas (depending upon which is currently active
        in the split screen or on the main screen) and fill in the
        appropriate cell grid data for the hours selected in the loaded
        parameters }
      IF rightlistmenu^.directorylist.picked > 0 THEN
      begin
         rightlowerstatus^.archivefile := rightlistmenu^.selectedfile;
         Assign(AF,ARCHIVEPATH+'\'+rightlowerstatus^.archivefile);
         Reset(AF);
         RestoreArchiveParameters(AF,auxeditmenu,RightSaveComment);
         Close(AF);
         CancelContext; { pop down the list menu }
         CancelContext; { pop down the file menu }
         RightSideStatus^.Show;
         RightLowerStatus^.Show;
         IF Rightexpanded THEN
            ReThresholdCells(BigCellGrid, auxeditmenu, Rightlowerstatus)
         ELSE
            ReThresholdCells(RightCellGrid, auxeditmenu, Rightlowerstatus);
         IF RightExpanded THEN
         begin
            BigCellGrid^.Show;
            BigMap^.SHOW;
         end
         ELSE
         begin
            RightCellGrid^.Show;
            RightMap^.SHOW;
         end;
         RightLowerStatus^.eptr := auxeditmenu;
         RightLowerStatus^.Show;
      end
      ELSE
      begin
       messagebox.displaymessage((GetMaxX DIV 2) -
             (textwidth('File Not Selected') DIV 2),
             GetMaxY DIV 2,1,
             length('File Not Selected'),lightred,white,
             'File Not Selected');
       beep;
       messagewait;
```

```
      messageocx.hidemessage;
    end;
  end;

(*****************************************************************************)

  procedure ReliabilityAction;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Procedure to display the reliability (QR) menu from the       *
    *           left split and main edit screens.                             *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }
  begin
    pushcontext('reliabilitylistmenu');
    showcontext;
  end;

(*****************************************************************************)

  procedure AuxReliabilityAction;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Procedure to display the reliability (QR) menu from the       *
    *           right split edit screen.                                      *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }
  begin
    pushcontext('auxreliabilitylistmenu');
    showcontext;
  end;

(*****************************************************************************)

  procedure RelListMenuAction;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Procedure to get the name of the QR set file  to be loaded    *
    *           and to perform the actions necessary to load the info into    *
```

```
      *          the left and main edit screens.                          *
      *                                                                   *
      *********************************************************************
      }
   begin
      IF reliabilitylistmenu^.directorylist.picked > 0 THEN
      begin
         editmenu^.QRFile := reliabilitylistmenu^.SelectedFile;
         CancelContext;
      end
      ELSE
      begin
       messagebox.displaymessage((GetMaxX DIV 2) -
             (textwidth('File Not Selected') DIV 2),
             GetMaxY DIV 2,1,
             length('File Not Selected'),lightred,white,
             'File Not Selected');
       beep;
       messagewait;
       messagebox.hidemessage;
     end;
   end;

(*************************************************************************)

   procedure AuxRelListMenuAction;
      {
      *********************************************************************
      *                                                                   *
      *        PURPOSE                                                    *
      *           Procedure to get the name of the QR set file  to be loaded *
      *           and to perform the actions necessary to load the info into *
      *           the right split   edit screens.                         *
      *                                                                   *
      *********************************************************************
      }
   begin
      IF auxreliabilitylistmenu^.directorylist.picked > 0 THEN
      begin
         auxeditmenu^.QRFile := auxreliabilitylistmenu^.SelectedFile;
         CancelContext;
      end
      ELSE
      begin
       messagebox.displaymessage((GetMaxX DIV 2) -
             (textwidth('File Not Selected') DIV 2),
             GetMaxY DIV 2,1,
             length('File Not Selected'),lightred,white,
             'File Not Selected');
```

```
      beep;
      messagewait;
      messagebox.hidemessage;
    end;
  end;

(*****************************************************************************)
  procedure CoverageAction;
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Procedure to display the coverage screen for the station/      *
    *          time currently selected on the detailed cell query window.     *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }
  begin
    hide_cursor;
    setactivepage(1);
    set_cursor_page(1);
    setvisualpage(1);
    pushcontext('coveragescreen');
    showcontext;
    show_cursor((GetMaxX DIV 2),(GetMaxY DIV 2));
  end;

(*****************************************************************************)
  procedure CoverageCellGridAction(selfptr : CellPtr; CellNumber: integer);
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Dummy procedure for cell queries on the coverage screen.       *
    *          Placeholder for later implementation.                          *
    *                                                                         *
    *                                                                         *
    ***************************************************************************
    }
  begin
  end;
(*****************************************************************************)

procedure DoNothingProc(selfptr : CellPtr);
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
```

File Name: PROCS.PAS

```
      *          Dummy procedure for right button clicks on the various cell*
      *          grid displays. Keeps the right button from cancelling      *
      *          the celldisplay contexts.                                  *
      *                                                                     *
      ***********************************************************************
      }

(* do nothing on the right mouse button hit for the cell grid *)

begin
end;

(*********************************************************************)

{$I procs2.pas}

{$F-}

begin
   new(QRDatabase);
   WeightsSaveComment := '';
   LeftSaveComment := '';
   RightSaveComment := '';
   QueueSaveComment := '';
end.
```

File Name: PROCS2.PAS

```
(*************************************************************************
*                                                                       *
*        PROGRAM NAME - Omega Performance Assessment                    *
*                                and                                    *
*                        Coverage Evaluation                            *
*                              (PACE)                                   *
*                           Workstation                                 *
*                                                                       *
*           UNIT NAME - Part of the PROCS Unit                          *
*                                                                       *
*************************************************************************
*                                                                       *
*        This program was prepared by                                   *
*                                                                       *
*              The Analytic Science Corporation (TASC)                  *
*                    55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                          *
*                                                                       *
*        PACE has been developed to run on a IBM PC/AT or compatible    *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main      *
*        memory and an EGA or compatible graphics adapter and color     *
*        monitor. This work was performed under contract number         *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for      *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.   *
*                                                                       *
*************************************************************************
*                                                                       *
*        PURPOSE                                                        *
*           Continuation of the routines defined in the PROCS Unit.     *
*                                                                       *
*                                                                       *
*                                                                       *
*************************************************************************)

(******** include the routines in procs3.pas file ************)
{$I procs3.pas}


Procedure Weightslistmenuaction;
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Invokes the list menu of available weights files from the   *
      *           weights editor.                                             *
      *************************************************************************
      }
var
  eptr : emenuptr;
```

File Name: PROCS2.PAS

```
begin
   IF Weightslistmenu^.directorylist.picked > 0 THEN
   begin
      WeightCellGrid^.HasBeenChanged := FALSE;
      WeightCellGrid^.weightsfile := Weightslistmenu^.selectedfile;
      getweightset(eptr,Weightcellgrid,WeightsPath + '\' + Weightslistmenu^.selectedfile);
      CancelContext; { pop down the list menu }
      CancelContext; { pop down the file menu }
      ShowContext;
   end
   else
   begin
    messagebox.displaymessage((GetMaxX DIV 2) -
          (textwidth('File Not Selected') DIV 2),
          GetMaxY DIV 2,1,
          length('File Not Selected'),lightred,white,
          'File Not Selected');
    beep;
    messagewait;
    messagebox.hidemessage;
  end;
end;

(*************************************************************************)

procedure Weightsloadbutton;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Displays the weight load menu for  the weight editor.         *
      *                                                                         *
      ***************************************************************************
      }

begin
  pushcontext('weightslistmenu');
  showcontext;
end;

(*************************************************************************)

procedure GetWeightSet(eptr:emenuptr; CGptr : CellGridPtr; fname : string);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Loads the weight information from the selected weight file.*
```

File_Name: PROCS2.PAS

```
      *                                                                      *
      ************************************************************************
      }

var
  WD : WeightsDataType;
  WF : WeightsFile;
  i  : integer;
  s  : string;

begin
  { ASSUMES that the fname already contains the path to the weights dir }
  {$I-}
  assign(WF,fname);
  reset(WF);
  {$I+}
  if ioresult <> 0 then
  begin
    s := fname + ' NOT FOUND. PRESS ANY KEY.';

    messagebox.displaymessage((GetMaxX DIV 2) - (textwidth(s) DIV 2),
                              GetMaxY DIV 2,
                              1,
                              length('12345678901234567890123456789012345678901234567890'
                              lightmagenta,
                              black,
                              s);
    messagewait;
    messagebox.hidemessage;

    fname := WeightsPath + '\'+ DEFAULTWEIGHTSFILE;
    {$I-}
    assign(WF,fname);
    reset(WF);
    {$I+}
    if (ioresult = 0) then
    begin
       read(WF,WD);
       for i := 1 to 444 do begin
         CGptr^.CellArray[i].weight_or_coverage := WD.weightdata[i] AND $7F;
         CGptr^.CellArray[i].region := ((WD.weightdata[i] AND $80) <> 0);
       end;
       WeightsSaveComment := WD.Comment;
       close(WF);
       eptr^.weightsfile := DEFAULTWEIGHTSFILE;
    end
    else
      Log_Error(fname + ' file not found...');
  end
```

```
  else
  begin
    read(WF,WD);
    for i := 1 to 444 do begin
      CGptr^.CellArray[i].weight_or_coverage := WD.weightdata[i] AND $7F;
      CGptr^.CellArray[i].region := ((WD.weightdata[i] AND $80) <> 0);
    end;
    WeightsSaveComment := WD.Comment;
    close(WF);
  end;
end;
(*****************************************************************************)

   procedure WeightsSaveButton;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Displays the save menu for the weight editor.                 *
      *                                                                         *
      ***************************************************************************
      }
   begin
     WeightsSaveMenuWindow^.FileButton.Change_String(WeightCellGrid^.eptr^.Weightsfile);
     WeightsSaveMenuWindow^.MemoBox.GetNewText(WeightsSaveComment);
     pushcontext('weightsavemenuwindow');
     showcontext;
   end;

(*****************************************************************************)

   Procedure Weightsmenuview(lmenu:lmenuptr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Displays the comment associated with a weight file.           *
      *                                                                         *
      ***************************************************************************
      }
   VAR
      errorstring : string;
      tempstring: commentareatype;
      WF: Weightsfile;

   begin
      IF lmenu^.directorylist.picked > 0 THEN
      begin { read the stuff in }
         Assign(WF,WEIGHTSPATH+'\'+lmenu^.selectedfile);
```

File Name: PROCS2.PAS

```
        Reset(WF);
        ViewWeightsComment(WF,tempstring);
        Close(WF);
        messagebox.displaymessage((GetMaxX DIV 2) - (25*textwidth(' ')),
                                  GetMaxY DIV 2,
                                  5,50,cyan,black,tempstring);
        messagewait;
        messagebox.hidemessage;
      end
      ELSE
      begin
       messagebox.displaymessage((GetMaxX DIV 2) -
            (textwidth('File Not Selected') DIV 2),
            GetMaxY DIV 2,1,
            length('File Not Selected'),lightred,white,
            'File Not Selected');
       beep;
       messagewait;
       messagebox.hidemessage;
     end;
   end;

(*****************************************************************************)

  procedure Weightssaveaction;
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           Causes the weight set to be saved to the selected file.        *
      *                                                                          *
      ****************************************************************************
      }
  var
    filename  : string;
    Source,WF : WeightsFile;
    i         : integer;
    dir       : DirStr;   (* as defined in the DOS unit *)
    name      : NameStr;
    ext       : ExtStr;

  begin
      filename := WeightsSaveMenuWindow^.FileButton.Text_String;
      fsplit(filename,dir,name,ext);
      filename := name + ext;

      if filename = '' then
      begin
        messagebox.displaymessage((GetMaxX DIV 2) -
```

```
            (textwidth('Please Select An Appropriate Filename before saving...') DIV 2),
            GetMaxY DIV 2,1,
            length('Please Select An Appropriate Filename before saving...')
            ,lightred,white,
            'Please Select An Appropriate Filename before saving...');
        beep;
        messagewait;
        messagebox.hidemessage;
      end
      else
      begin
        WeightCellGrid^.HasBeenChanged := FALSE;
        WeightCellGrid^.weightsfile := filename;
        WeightsSaveMenuWindow^.MemoBox.AssignChangedText(WeightsSaveComment);
        messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Saving to file: '+WEIGHTSPAT
                                  (GetMaxY DIV 2) - (textheight('1') DIV 2),
                                  1,
                                  length('Saving to file: '+WEIGHTSPATH+'\'+filename),
                                  lightred,
                                  white,
                                  'Saving to file: '+WEIGHTSPATH+'\'+filename);

        SaveWeightsParameters(WEIGHTSPATH+'\'+filename,WeightsSaveComment);
        messagebox.hidemessage;
        hidecontext; popcontext; { wipe out the save memo window }
        hidecontext; popcontext; { wipe out the file pick list drop down menu }
      end;
  end;

(*****************************************************************************)

function isin(x1,y1,x2,y2,xp,yp:integer): boolean;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility function that checks to see if a point is within      *
      *           a defined rectangular region.                                 *
      ***************************************************************************
      }
VAR
  tx,ty: integer;

begin
   { first order the coordinates }
   IF (x1 > x2) THEN
   begin
      tx := x2;
      x2 := x1;
```

```
      x1 := tx;
   end;
   IF (y1 > y2) THEN
   begin
      ty := y2;
      y2 := y1;
      y1 := ty;
   end;
   { now see if the point is in the coordinate box }
   IF (x1 <= xp) AND (xp <= x2) AND (y1 <= yp) AND (yp <= y2) THEN
      isin := TRUE
   ELSE
      isin := FALSE;
end;

(*****************************************************************************\
Function GetGroupRegionValue(var Region : boolean):boolean;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Utility function to assign a region flag to all cells in         *
      *           a defined rectangular region.                                    *
      ******************************************************************************
      }
VAR
  st1 : string;
  CorrectWeightFound : boolean;
  key,oldkey,button,
  xx,yy : integer;

begin
   settextjustify(lefttext,toptext);
   Get_CursorXY(xx,yy);
   Hide_Cursor;
   setcolor(white);
   outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
             'Press <ENTER> when finished');
   outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
              'Turn region on? <Y/N> ');
   oldkey := -1;
   repeat
     Get_Key(Key);
     IF ((Key = ord('Y')) OR (Key = ord('N')) OR
         (Key = ord('y')) OR (Key = ord('n'))) THEN
     begin
        IF oldkey <> -1 THEN
        begin
           setcolor(basescreen2^.color);
```

```
            outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
                      'Turn region on? <Y/N> '+chr(oldkey));
          end;
          setcolor(white);
          outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
                    'Turn region on? <Y/N> '+chr(key));
          oldkey := key;
        end;
        button := get_buttons;
      Until (button = left_button_pressed) OR (button = right_button_pressed) OR
            (key = ENTER) OR (key = ESC);
      IF ((button = left_button_pressed) OR (key = ENTER)) AND
         ((oldKey = ord('Y')) OR (oldkey = ord('y')) OR
         (oldKey = ord('N')) OR (oldkey = ord('n'))) THEN
      begin
         IF ((oldKey = ord('N')) OR (oldkey = ord('n'))) THEN
            Region := FALSE
         ELSE
            Region := TRUE;
         GetGroupRegionValue := TRUE;
      end
      ELSE
      begin
         GetGroupRegionValue := FALSE;
      end;
      Show_Cursor(xx,yy);
end;

(*************************************************************************)

Function GetGroupWeightValue(var Weight : integer):boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Utility function to assign a weight value to all cells in  *
      *           a defined rectangular region.                              *
      ************************************************************************
      }
VAR
  st1 : string;
  CorrectWeightFound : boolean;
  key,oldkey,button,
  xx,yy : integer;

begin
   settextjustify(lefttext,toptext);
   Get_CursorXY(xx,yy);
   Hide_Cursor;
```

File Name: PROCS2.PAS

```
    setcolor(white);
    outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
              'Press <ENTER> when finished');
    outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
              'Enter weight number <0..9> ');
    oldkey := -1;
    repeat
      Get_Key(Key);
      IF ((Key >= ord('0')) and (Key <= ord('9'))) THEN
      begin
         IF oldkey <> -1 THEN
         begin
            setcolor(basescreen2^.color);
            outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
                      'Enter weight number <0..9> '+chr(oldkey));
         end;
         setcolor(white);
         outtextxy(WeightCellGrid^.X+350,WeightCellGrid^.Y+WeightCellGrid^.Height+20,
                   'Enter weight number <0..9> '+chr(key));
         oldkey := key;
      end;
      button := get_buttons;
    Until (button = left_button_pressed) OR (button = right_button_pressed) OR
          (key = ENTER) OR (key = ESC);
    IF ((button = left_button_pressed) OR (key = ENTER)) AND
       (oldKey >= ord('0')) and (oldKey <= ord('9')) THEN
    begin
       weight := oldkey - ord('0');
       GetGroupWeightValue := TRUE;
    end
    ELSE
    begin
       GetGroupWeightValue := FALSE;
    end;
    Show_Cursor(xx,yy);
end;

(*****************************************************************************)

procedure ProcessGroup;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Utility function to assign weight or region values to cells*
      *           in a defined rectangular region.                              *
      ***************************************************************************
      }
var
```

```
File Name: PROCS2.PAS

  i,xx,yy,key,button : integer;
  regionvalue: boolean;
  weightvalue: integer;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  setcolor(Basescreen2^.color);
  outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
          'Please select 1st Corner of GROUP...');
  setcolor(white);
  outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
          'Please select 2nd Corner of GROUP...');
  Set_Cursor_Shape(finger);
  Show_Cursor(xx,yy);
  WeightCellGrid^.LastCoord.x := xx;
  WeightCellGrid^.LastCoord.y := yy;
  WeightCellGrid^.FirstCoord.x := xx;
  WeightCellGrid^.FirstCoord.y := yy;
  SetWriteMode(XORPUT);
  SetColor(white);
  Repeat
     Get_CursorXY(xx,yy);
     Get_Key(Key);
     IF (Key IN _Cursor_set) THEN
        Move_Cursor(Key, xx, yy);
     button := get_buttons;
     IF (xx <> WeightCellGrid^.LastCoord.x) OR (yy <> WeightCellGrid^.LastCoord.y) THEN
     begin
        Hide_Cursor;                                    rectangle(WeightCellGrid^.FirstCoord.x,
                WeightCellGrid^.LastCoord.x,WeightCellGrid^.LastCoord.y);
        rectangle(WeightCellGrid^.FirstCoord.x,WeightCellGrid^.FirstCoord.y,
                xx,yy);
        WeightCellGrid^.LastCoord.x := xx;
        WeightCellGrid^.LastCoord.y := yy;
        Show_Cursor(xx,yy);
     end;
  Until (button = left_button_pressed) OR (button = right_button_pressed) OR
        (key = ENTER) OR (key = ESC);

  SetWriteMode(NORMALPUT);

  setcolor(Basescreen2^.color);
  outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
          'Please select 2nd Corner of GROUP...');

  { now process the case }
  IF (button = left_button_pressed) OR (key = enter) THEN
  begin
```

File Name: PROCS2.PAS

```
      IF WeightCellGrid^.pickingweights THEN
      begin
         IF GetGroupWeightValue(WeightValue) THEN
         FOR i := 1 to NCells DO
            IF isin(WeightCellGrid^.firstcoord.x,WeightCellGrid^.firstcoord.y,
                    WeightCellGrid^.lastcoord.x,WeightCellGrid^.lastcoord.y,
                    WeightCellGrid^.CellArray[i].X + WeightCellGrid^.CellArray[i].Width DIV 2
                    WeightCellGrid^.CellArray[i].Y + WeightCellGrid^.CellArray[i].Height DIV
                    THEN
               WeightCellGrid^.CellArray[i].weight_or_coverage := WeightValue;
      end
      ELSE { picking regions }
      begin
         IF GetGroupRegionValue(RegionValue) THEN
         FOR i := 1 to NCells DO
            IF isin(WeightCellGrid^.firstcoord.x,WeightCellGrid^.firstcoord.y,
                    WeightCellGrid^.lastcoord.x,WeightCellGrid^.lastcoord.y,
                    WeightCellGrid^.CellArray[i].X + WeightCellGrid^.CellArray[i].Width DIV 2
                    WeightCellGrid^.CellArray[i].Y + WeightCellGrid^.CellArray[i].Height DIV
                    THEN
               WeightCellGrid^.CellArray[i].region := RegionValue;
      end;
      ShowContext;
      WeightCellGrid^.SelectingByGroup := FALSE;
      Get_CursorXY(xx,yy);
      Hide_Cursor;
      Set_Cursor_Shape(arrow);
      Show_Cursor(xx,yy);
   end
   else
   begin
      SetWriteMode(XORPUT);
      SetColor(white);
      Get_CursorXY(xx,yy);
      Hide_Cursor;
      rectangle(WeightCellGrid^.FirstCoord.x,WeightCellGrid^.FirstCoord.y,
                WeightCellGrid^.LastCoord.x,WeightCellGrid^.LastCoord.y);
      SetWriteMode(NORMALPUT);
      setcolor(white);
      outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
                'Please select 1st Corner of GROUP...');
      Show_Cursor(xx,yy);
   end;
end;

(*****************************************************************************)

procedure WeightsCellGridAction(selfptr : CellPtr; cellnumber:integer);
      {
```

File Name: PROCS2.PAS

```
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Procedure to perform the weight increment or region assign-*
      *           ment to single cells on the weight editor grid.            *
      ************************************************************************
      }
var
  xx,yy : integer;
  tmp   : string;
begin
  WeightCellGrid^.HasBeenChanged := TRUE;
  IF (WeightCellGrid^.SelectingByGroup) THEN
  begin
    ProcessGroup;
  end
  else
  begin
    Get_CursorXY(xx,yy);
    Hide_Cursor;
    IF (WeightCellGrid^.PickingWeights) THEN
    begin
      STR(selfptr^.Weight_or_Coverage:2, tmp);
      setcolor(selfptr^.AboveColor);
      settextstyle(smallfont,horizdir,4);
      settextjustify(centertext,centertext);
      outtextxy(selfptr^.X + (selfptr^.Width DIV 2),
                selfptr^.Y + (selfptr^.Height DIV 2), tmp);
      settextstyle(defaultfont,horizdir,1);
      if selfptr^.Weight_or_Coverage < 9 then
         selfptr^.Weight_or_Coverage := selfptr^.Weight_or_Coverage + 1;
    end
    ELSE { picking regions }
    begin
      selfptr^.region := TRUE;
    end;
    ShowWeightsCellData(selfptr);
    Show_Cursor(xx,yy);
  end;
end;

(************************************************************************)

procedure WeightsCellGridCancel(selfptr : CellPtr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Procedure to return to the edit screen from the weight     *
```

```
      *          editor.                                                    *
      ******************************************************************************
      }
var
  xx,yy : integer;
  tmp   : string;
begin
  WeightCellGrid^.HasBeenChanged := TRUE;
  IF (WeightCellGrid^.SelectingByGroup) THEN
  begin
     WeightCellGrid^.SelectingByGroup := FALSE;
     setcolor(Basescreen2^.color);
     outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
               'Please select 1st Corner of GROUP...');
     Set_Cursor_Shape(arrow);
  end
  else
  begin
    Get_CursorXY(xx,yy);
    Hide_Cursor;
    IF (WeightCellGrid^.PickingWeights) THEN
    begin
      STR(selfptr^.Weight_or_Coverage:2, tmp);
      setcolor(selfptr^.AboveColor);
      settextstyle(smallfont,horizdir,4);
      settextjustify(centertext,centertext);
      outtextxy(selfptr^.X + (selfptr^.Width DIV 2),
                selfptr^.Y + (selfptr^.Height DIV 2), tmp);
      settextstyle(defaultfont,horizdir,1);
      if selfptr^.Weight_or_Coverage > 0 then
        selfptr^.Weight_or_Coverage := selfptr^.Weight_or_Coverage - 1;
    end
    ELSE { picking regions }
    begin
      selfptr^.region := FALSE;
    end;
    ShowWeightsCellData(selfptr);
    Show_Cursor(xx,yy);
  end;
end;

(******************************************************************************)

   procedure Weight_or_Region_Button;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Procedure to toggle between the selection of weights or          *
```

```
      *          regions in the weight editor.                              *
      ******************************************************************************
      }
   VAR
      xx,yy:integer;
   begin
      get_cursorxy(xx,yy);
      hide_cursor;
      WeightCellGrid^.pickingweights := NOT WeightCellGrid^.pickingweights;
      IF WeightCellGrid^.pickingweights THEN
      begin
         optionsmenu^.buttonarray[2].HotKey := 'R';
         optionsmenu^.buttonarray[2].ChangeName('Region');
      end
      ELSE
      begin
         optionsmenu^.buttonarray[2].HotKey := 'W';
         optionsmenu^.buttonarray[2].ChangeName('Weight');
      end;
      hidecontext;
      popcontext;
      ShowCurrentPickChoice;
      show_cursor(xx,yy);
   end;
(*****************************************************************************)

   procedure WeightAction;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           Procedure to invoke the weight editor.                           *
      *                                                                            *
      ******************************************************************************
      }

   begin
     hide_cursor;
     WeightCellGrid^.pickingweights := TRUE;
     WeightCellGrid^.HasBeenChanged := FALSE;
     WeightCellGrid^.SelectingByGroup := FALSE;
     WeightCellGrid^.eptr := editmenu;
     WeightCellGrid^.weightsfile := editmenu^.weightsfile;
     GetWeightSet(editmenu,WeightCellGrid,WeightsPath+'\'+editmenu^.weightsfile);
     setactivepage(1);
     set_cursor_page(1);
     setvisualpage(1);
     pushcontext('weightscreen');
     showcontext;
```

```
    show_cursor((GetMaxX DIV 2),(GetMaxY DIV 2));
  end;

(*****************************************************************************)

  procedure AuxWeightAction;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Procedure to invoke the weight editor.                        *
    *                                                                         *
    ***************************************************************************
    }

  begin
    hide_cursor;
    WeightCellGrid^.pickingweights := TRUE;
    WeightCellGrid^.HasBeenChanged := FALSE;
    WeightCellGrid^.SelectingByGroup := FALSE;
    WeightCellGrid^.eptr := auxeditmenu;
    WeightCellGrid^.weightsfile := auxeditmenu^.weightsfile;
    GetWeightSet(auxeditmenu,WeightCellGrid,WeightsPath+'\'+auxeditmenu^.weightsfile);
    setactivepage(1);
    set_cursor_page(1);
    setvisualpage(1);
    pushcontext('weightscreen');
    showcontext;
    show_cursor((GetMaxX DIV 2),(GetMaxY DIV 2));
  end;

(*****************************************************************************)

  procedure Weightslistmenuview;
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           Procedure to cause the weight comment to be displayed.        *
    *                                                                         *
    ***************************************************************************
    }
  begin
    weightsmenuview(weightslistmenu);
  end;

(*****************************************************************************)

  procedure WeightsGroupButton;
```

File Name: PROCS2.PAS

```
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *          Utility function to rubber band a group of cells into               *
    *          a defined rectangular region.                                       *
    ********************************************************************************
    }
  var xx,yy : integer;
  begin
    cancelcontext;
    Get_CursorXY(xx,yy);
    Hide_Cursor;
    WeightCellGrid^.SelectingByGroup := NOT WeightCellGrid^.SelectingByGroup;
    IF WeightCellGrid^.SelectingByGroup THEN
    begin
      Set_Cursor_Shape(finger);
      setcolor(white);
      outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
                'Please select 1st Corner of GROUP...');
    end
    else
    begin
      setcolor(Basescreen2^.color);
      outtextxy(WeightCellGrid^.X+20,WeightCellGrid^.Y+WeightCellGrid^.Height+20+10,
                'Please select 1st Corner of GROUP...');
      Set_Cursor_Shape(arrow);
    end;
    Show_Cursor(xx,yy);
  end;
```

File Name: PROCS3.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                              (PACE)                                    *
*                           Workstation                                  *
*                                                                        *
*          UNIT NAME - Part of the PROCS Unit                            *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           Continuation of the object utility routines of the PROCS     *
*           Unit. Contains mostly routines for computing Psa.            *
*                                                                        *
*                                                                        *
*************************************************************************)

{ Continuation of the Procs.pas procedure file.
  Contains the main routines for calculating PSA and for
  re-thresholding the cell grid.

  9/24/90 GRDesrochers
}

(*************************************************************************)

   procedure ReThresholdCells(CGptr: CellGridPtr; eptr:emenuptr;
                              lptr:lbarptr);
      {
      ************************************************************************
      *                                                                      *
```

File Name: PROCS3.PAS

```
      *        PURPOSE                                                    *
      *           This procedure goes through all of the cells and computes  *
      *           the global and regional Psa statistics from the Pat and    *
      *           Pac values. Also perform the cell weighting and normalizat-*
      *           ion.                                                       *
      ***************************************************************************
      }
   VAR
      Cellstuff: ArchivedCellType;
      month,hour,cellnumber:integer;
      AvgVal,MinVal,MaxVal,Psa,RegPsa,Pac: single;
      NumberMonthsHours,WeightSum,RegWeightSum:integer;
      abovethreshold: boolean;
      AF: Archivefile;
      errorstring,
      s: string;
   begin
      Assign(AF,ARCHIVEPATH+'\'+lptr^.archivefile);
      Reset(AF);
      GetWeightSet(eptr,CGPtr,WeightsPath+'\'+eptr^.weightsfile);
      WeightSum := 0;
      RegWeightSum := 0;
      Psa := 0;
      RegPsa := 0;
      lptr^.MinPsa := 99;
      lptr^.MaxPsa := 0;
      lptr^.RegMinPsa := 99;
      lptr^.RegMaxPsa := 0;
      s := 'Loading ...Cell   ';
      messagebox.displaymessage((GetMaxX DIV 2) - (textwidth(s) DIV 2),GetMaxY DIV 2,1,lengt
      FOR cellnumber := 1 TO NUMBERCELLS DO
      begin
         numbermonthshours := 0;
         AvgVal := 0;
         MinVal := 99;
         MaxVal := 0;
         str(cellnumber,s);
         messagebox.redisplaymessage(1,50,'Loading...Cell '+s);
         LoadCellFromArchive(AF,cellnumber,CellStuff);
         FOR month := 1 TO 12 DO
            FOR hour := 1 TO 24 DO
            begin
               IF eptr^.MonthSelectors[month].selected AND
                  eptr^.HourSelectors[hour].selected THEN
                  begin
                     IF CellStuff[Month,hour].Pat < MinVal THEN
                        MinVal := CellStuff[month,hour].Pat;
                     IF CellStuff[Month,hour].Pat > MaxVal THEN
                        MaxVal := CellStuff[month,hour].Pat;
```

File Name: PROCS3.PAS

```
                    AvgVal := AvgVal + CellStuff[month,hour].Pat;
                    inc(numbermonthshours);
                  end;
            end;
            IF numbermonthshours <> 0 THEN
               AvgVal := AvgVal/numbermonthshours;
         IF MinVal < lptr^.MinPsa THEN
           lptr^.MinPsa := MinVal;
         IF MaxVal > lptr^.MaxPsa THEN
            lptr^.MaxPsa := MaxVal;
         Psa := Psa + AvgVal*CGptr^.cellarray[cellnumber].weight_or_coverage;
         WeightSum := weightSum + CGPtr^.cellarray[cellnumber].Weight_or_coverage;
         IF CGptr^.CellArray[cellnumber].region THEN
         begin
            IF MinVal < lptr^.RegMinPsa THEN
               lptr^.RegMinPsa := MinVal;
            IF MaxVal > lptr^.RegMaxPsa THEN
               lptr^.RegMaxPsa := MaxVal;
            RegPsa := RegPsa + AvgVal*CGptr^.cellarray[cellnumber].weight_or_coverage;
            RegWeightSum := RegWeightSum +
               CGPtr^.cellarray[cellnumber].Weight_or_coverage;
            IF MinVal < lptr^.RegMinPsa THEN
               lptr^.RegMinPsa := MinVal;
            IF MaxVal > lptr^.RegMaxPsa THEN
               lptr^.RegMaxPsa := MaxVal;
         end;
         CASE eptr^.PsaReportMode.Picked OF
            PsaMin:
            begin
               IF MinVal < eptr^.Psa.Value.Value THEN
                  abovethreshold := FALSE
               ELSE
                  abovethreshold := TRUE;
               Pac := MinVal;
            end;
            PsaMean:
            begin
               IF AvgVal < eptr^.Psa.Value.Value THEN
                  abovethreshold := FALSE
               ELSE
                  abovethreshold := TRUE;
               Pac := AvgVal;
            end;
            PsaMax:
            begin
               IF MaxVal < eptr^.Psa.Value.Value THEN
                  abovethreshold := FALSE
               ELSE
                  abovethreshold := TRUE;
```

```
            Pac := MaxVal;
          end;
      end; { CASE }
      CGptr^.cellarray[cellnumber].abovethreshold := abovethreshold;
      CGptr^.cellarray[cellnumber].Pac := Pac;
   end; { for cellnumber := 1 to NUMBERCELLS }
   messagebox.hidemessage;
   IF weightsum <> 0 THEN
      Psa := Psa/WeightSum;
   IF regweightsum <> 0 THEN
      RegPsa := RegPsa/RegWeightSum;
   IF lptr^.MinPsa = 99 THEN { didn't find a min }
      lptr^.MinPsa := 0;
   IF lptr^.RegMinPsa = 99 THEN { didn't find a min }
      lptr^.RegMinPsa := 0;
   lptr^.MeanPsa := Psa;
   lptr^.RegMeanPsa := RegPsa;
   Close(AF);
end;

(*****************************************************************************)

Function CheckCoverage(covinfo: covcell):boolean;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           This function  performs the coverage calculation for a        *
   *           given set of coverage parameters and criteria.                *
   ***************************************************************************
   }
VAR
   SSNR,LSNR: integer;
begin
   CheckCoverage := FALSE; { Assume FALSE to start }
   SSNR := Get_SNR(Covinfo.SNR_S);
   LSNR := Get_SNR(Covinfo.SNR_L);
   IF (SSNR >= SNRThreshold) AND
      ((covinfo.Phase AND $7F <= PhaseThreshold) OR
       (covinfo.Phase AND $7F >= 100 - PhaseThreshold)) AND
      (covinfo.X_ang >= XAngleThreshold) AND
      (SSNR - LSNR >= ShortLongThreshold) THEN
   begin { all criteria except DM checked out OK }
      CheckCoverage := TRUE;
      IF (DMThreshold) THEN { all criteria OK, is DM check ON? }
      begin
         IF (covinfo.Phase AND $80) <> 0 THEN { all criteria passed, DM check is on }
            CheckCoverage := FALSE;          { DM = mode 0 IF bit 8 = 0 }
      end;
```

```
    end;
  end;

  Procedure ComputePsa(eptr:emenuptr; lptr:lbarptr);
    {
    ***************************************************************************
    *                                                                         *
    *        PURPOSE                                                          *
    *           This procedure performs the computation of the Psa value      *
    *           by looping through all cells/times, checking coverage, and    *
    *           retrieving the Pat values from the QR set.                    *
    ***************************************************************************
    }
  TYPE
    GDOPForACell = ARRAY[0..255] of byte;
  VAR
    CellStuff: ArchivedCellType;
    stationcoverage102,stationcoverage136: stationcoverageinfoptr;
    cellnum,month,hour,station: integer;
    errorstring,
    s: string;
    AF: Archivefile;
    DB102,DB136: DBFile; { file variables for the 10.2 and 13.6 KHz databases }
    GDOPvalues: GDOPForACell;
    GDOPfile: file of GDOPForACell;
  begin
    { For each cell and combination of parameters, get the cell
      coverage info from the database, calculate the coverage, and
      then read the Pa value from the QR matrix. Store the results in
      a temporary work file (one for each split screen). }
    { First open up the two database files (one each for 10.2 and 13.6 KHz.
      The files are read in the readfromdatabase routine }


    Assign(DB102,DATABASEPATH+'\'+DATABASE102);
    Reset(DB102);
    Assign(DB136,DATABASEPATH+'\'+DATABASE136);
    Reset(DB136);
    Assign(GDOPfile,PACEPATH+'\'+GDOPDATABASE);
    Reset(GDOPFile);

    Assign(AF,ARCHIVEPATH+'\'+lptr^.archivefile);
    Rewrite(AF);
    s := 'Processing ...Cell   ';
     messagebox.displaymessage((GetMaxX DIV 2) - (textwidth(s) DIV 2),GetMaxY DIV 2,1,leng

    { save the parameters that we are computing with }
    SaveArchiveParameters(AF,editmenu,LeftSaveComment);
```

```
{ convert coverage thresholds into 1/8 dB units and round into integers }
ShortLongThreshold := round(eptr^.ShortLongRatio.value.value)*8;
Phasethreshold := round(eptr^.PhaseDev.value.value);
XAngleThreshold := round(eptr^.XAngle.Value.Value);
DMThreshold := eptr^.DM.picked = DMON;
{ GDOP values are scaled by 10 }
GDOPThreshold := round(eptr^.GDOP.value.value) * 10;

new(stationcoverage102);
new(stationcoverage136);
ReadQrDatabase(eptr^.QRFile,QRDatabase);

FOR cellnum := 1 TO NUMBERCELLS DO
begin { compute the cell Pac for each cell and store result in archive }
   str(cellnum,s);
   messagebox.redisplaymessage(1,50,'Processing...Cell '+s);

   { initialize the cell coverage for each time to 0 }
   FOR month := 1 TO 12 DO
      FOR hour := 1 TO 24 DO
         cellstuff[month,hour].coverage := 0;

   { read 8 stations worth of coverage for the year for the cell we are on }
   CASE eptr^.Freq.picked OF
      FRQ102:
         ReadFromDatabase(DB102,cellnum,stationcoverage102);
      FRQ136:
         ReadFromDatabase(DB136,cellnum,stationcoverage136);
      ELSE
      begin
         ReadFromDatabase(DB102,cellnum,stationcoverage102);
         ReadFromDatabase(DB136,cellnum,stationcoverage136);
      end;
   end; { CASE }

   { read in the GDOP values for all coverage vectors for this cell }
   { seek to the cellnum-1 position because the file positions start
     at 0 whereas the cell numbers start at 1 }
   seek(GDOPfile,cellnum-1);
   read(GDOPfile,GDOPvalues);

   FOR station := 1 TO 8 DO { Compute the coverage for all hours/months }
   begin
      FOR month := 1 TO 12 DO
      begin
         { first check to see if the station is on and, if it is, make sure
           worst case has not been selected and, if it is, the station is
           not in anual maintenance for this month }
```

File Name: PROCS3.PAS

```
            IF eptr^.StationPower[station].on.selected AND NOT
               ((eptr^.StationReliabilityModel.picked = SRMWrst) AND
                (QRdatabase^.stationreliabilities[month,station,maintenance] <> 0)) THEN
            begin
               SNRThreshold := (round(eptr^.SNR.value.value) -
                               round(eptr^.stationpower[station].Sld.Value.Value))*8;
               FOR hour := 1 TO 24 DO
               begin
                  CASE eptr^.Freq.picked OF
                     FRQ102:
                        IF CheckCoverage(stationcoverage102^[station,month,hour]) THEN
                           cellstuff[month,hour].coverage :=
                              cellstuff[month,hour].coverage OR ($80 SHR (station-1));
                     FRQ136:
                        IF CheckCoverage(stationcoverage136^[station,month,hour]) THEN
                           cellstuff[month,hour].coverage :=
                              cellstuff[month,hour].coverage OR ($80 SHR (station-1));
                     FRQAND:
                        IF CheckCoverage(stationcoverage102^[station,month,hour]) AND
                           CheckCoverage(stationcoverage136^[station,month,hour]) THEN
                           cellstuff[month,hour].coverage :=
                              cellstuff[month,hour].coverage OR ($80 SHR (station-1));
                     FRQOR:
                        IF CheckCoverage(stationcoverage102^[station,month,hour]) OR
                           CheckCoverage(stationcoverage136^[station,month,hour]) THEN
                           cellstuff[month,hour].coverage :=
                              cellstuff[month,hour].coverage OR ($80 SHR (station-1));
                  end; { CASE }
               end; { FOR hour := 1 TO 24 }
            end; { IF station on and etc }
         end; { FOR month := 1 TO 12 }
      end; { FOR station := 1 TO 8 }
      { now that we have coverage for each time, retrieve Pat for each
        from the QR file }
      FOR month := 1 TO 12 DO
      begin
         FOR hour := 1 TO 24 DO
         begin
            { check GDOP, if above threshold then the cell is "not" covered }
            IF GDOPvalues[cellstuff[month,hour].coverage] > GDOPthreshold THEN
               Cellstuff[month,hour].Pat := 0
            ELSE
               CASE eptr^.StationReliabilityModel.Picked OF
                  SRMbest: Cellstuff[month,hour].Pat :=
                     QRDatabase^.best[month,cellstuff[month,hour].coverage];
                  SRMnom: Cellstuff[month,hour].Pat :=
                    QRDatabase^.nom[month,cellstuff[month,hour].coverage];
                  SRMwrst: Cellstuff[month,hour].Pat :=
                     QRDatabase^.nom[month,cellstuff[month,hour].coverage];
```

File Name: PROCS3.PAS

```
                end; { CASE }
            end; { FOR hour := 1 TO 24 }
        end; { FOR month := 1 TO 12 }
        StoreCellToArchive(AF,cellnum,CellStuff);
    end; { FOR cellnum := 1 TO NUMBERCELLS }
    messagebox.hidemessage;
    dispose(stationcoverage102);
    dispose(stationcoverage136);
    Close(AF);

    Close(DB102);
    Close(DB136);
    Close(GDOPFile);

end;

procedure differencebutton;
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *           This procedure activates the difference menu display.              *
    ********************************************************************************
    }
var
  st: string;
begin
  IF currentcontext = 'splitscreen' THEN
  begin
    hide_cursor;
    setactivepage(1);
    set_cursor_page(1);
    setvisualpage(1);
    pushcontext('diffscreen');
    showcontext;
    difference2^.hilite(difference2^.X+1,Difference2^.y+1);
    Case diffmenu^.diffmode.picked of
      1:  st := ' '+chr(127)+'PSA';
      2:  st := '%'+chr(127)+'PSA';
      3:  st := '%'+chr(127)+'(1-PSA)';
    end; {case}
    setcolor(white);
    settextjustify(lefttext,toptext);
    outtextxy(difference2^.X + difference2^.Width + 10,
              difference2^.Y + 5, st);
    show_cursor(getmaxx div 2,getmaxy div 2);
  end;
  pushcontext('diffmenu');
  showcontext;
```

```
File Name: PROCS3.PAS

end;

procedure differenceacceptbutton;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure initiates the computation of the difference *
      *           display cell values.                                       *
      ************************************************************************
      }
var
 st : string;

begin
   hidecontext;
   popcontext;
   showcontext;
   Case diffmenu^.diffmode.picked of
     1:  st := ' '+chr(127)+'PSA';
     2:  st := '%'+chr(127)+'PSA';
     3:  st := '%'+chr(127)+'(1-PSA)';
   end; {case}
   setcolor(white);
   settextjustify(lefttext,toptext);
   outtextxy(difference2^.X + difference2^.Width + 10,
             difference2^.Y + 5, st);
end;

procedure DiffCellGridAction(selfptr : CellPtr; cellnumber:integer);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           This procedure invokes the display of difference display   *
      *           cell query window.                                         *
      ************************************************************************
      }

const
  WindowOffset = 10;  (* distance between the cell and the window edge *)

var
  mon, hr,xx,yy : integer;

begin
    Get_CursorXY(xx,yy);
    selfptr^.Hilite(xx,yy);
    with DiffCellGrid^ do begin
```

```
      if (CellArray[cellnumber].X < (GetMaxX - RightSideStatus^.width -
           CellArray[cellnumber].Width - DiffCellPopUp^.Width -
           WindowOffset-WindowShadowWidth) ) then
         DiffCellPopUp^.X := CellArray[cellnumber].X +
                             CellArray[cellnumber].Width +
                             WindowOffset
        else
          DiffCellPopUp^.X := CellArray[cellnumber].X -
                             DiffCellPopUp^.Width -
                             WindowOffset;

      if ((CellArray[cellnumber].Y+CellArray[cellnumber].Height) < (GetMaxY -
           LeftLowerStatus^.Height - DiffCellPopUp^.Height -
           WindowOffset-WindowShadowWidth) ) then
      begin
               (* put pop-up window BELOW the selected cell *)
        DiffCellPopUp^.Y := CellArray[cellnumber].Y +
                            CellArray[cellnumber].Height +
                            WindowOffset;
      end
      else
      begin
               (* put pop-up window ABOVE the selected cell *)
        DiffCellPopUp^.Y := CellArray[cellnumber].Y - DiffCellPopUp^.Height -
                          - WindowOffset;
      end;

      if DiffCellPopUp^.Y < (q^.Y + q^.Height + WindowOffset) then
      begin
        DiffCellPopUp^.Y := (q^.Y + q^.Height + WindowOffset);
      end;
    end;



   DiffCellPopUp^.HilitedCellPtr := @DiffCellGrid^.CellArray[cellnumber];

   PushContext('DiffCellPopUp');
   ShowContext;

end;

procedure ShowDiffCellData(selfptr : CellPtr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *          This procedure displays the actual difference data in the   *
```

File Name: PROCS3.PAS

```
      *-.          cell query window:                                          *
      ************************************************************************
      }
var
  col,xx,yy : integer;
  denom,val,lval,rval: single;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
   if selfptr^.IsHilited then
       selfptr^.SetHilite(False)
   else
   begin
      lval :=  leftcellgrid^.cellarray[selfptr^.cellnumber].pac;
      rval := rightcellgrid^.cellarray[selfptr^.cellnumber].pac;
      IF abs(lval - rval) < 0.00001 THEN
         val := 0
      ELSE
      CASE diffmenu^.diffmode.picked OF
         1: val := lval - rval;
         2:
            begin
               IF lval > rval THEN
                  denom := lval
               ELSE
                  denom := rval;
               IF denom <> 0 THEN
                  val := (lval-rval)/denom
               ELSE
                  val := 1;
            end;
         3:
            begin
               IF (1-lval) > (1-rval) THEN
                  denom := 1 - lval
               ELSE
                  denom := 1 - rval;
               IF (denom) <> 0 THEN
                  val := ((1-lval)-(1-rval))/denom
               ELSE
                  val := 1;
            end;
      end; { case }
      col := basescreen2^.color;
      IF val < diffmenu^.lothresh.value.value THEN
         col := red;
      IF val > diffmenu^.hithresh.value.value THEN
         col := green;
```

```
      selfptr^.color := col;
      setfillstyle(solidfill,selfptr^.color);
      bar(selfptr^.X,
          selfptr^.Y,
          selfptr^.X +
          selfptr^.Width,
          selfptr^.Y+
          selfptr^.Height);
   end;
  setcolor(selfptr^.BorderColor);
  setlinestyle(solidln,0,normwidth);
  rectangle(selfptr^.x,
            selfptr^.y,
            selfptr^.x +
            selfptr^.width,
            selfptr^.y +
            selfptr^.Height);
  Show_Cursor(xx,yy);
end;


procedure NextAction;
      {
      ****************************************************************************
      *                                                                          *
      *        PURPOSE                                                           *
      *           This procedure retrieves the next page of help for display *
      *           in the help window.                                            *
      ****************************************************************************
      }

VAR
  pointer : HFLPtr;
  FoundNext : boolean;

begin
  pointer := HelpFilesList;
  FoundNext := FALSE;
  while (pointer <> nil) and NOT (FoundNext) do
  begin
    if (pointer^.filename = HelpMenuWindow^.HelpFileName) then
      if (pointer^.next <> nil) then
      begin
        HelpMenuWindow^.HelpFileName := pointer^.next^.filename;
        FoundNext := TRUE;
      end;
    pointer := pointer^.next;
  end;
  if NOT(FoundNext) then
```

```
    beep;
  else
  begin
    HelpMenuWindow^.ClearWindow;
    HelpMenuWindow^.ShowText;
  end;
end;

procedure PreviousAction;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           This procedure retrieves the previous help page for display*
      *           in the help window.                                                *
      ********************************************************************************
      }

VAR
  pointer : HFLPtr;
  FoundNext : boolean;

begin
  pointer := HelpFilesList;
  FoundNext := FALSE;
  while (pointer <> nil) and NOT (FoundNext) do
  begin
    if (pointer^.filename = HelpMenuWindow^.HelpFileName) then
      if (pointer^.prev <> nil) then
      begin
        HelpMenuWindow^.HelpFileName := pointer^.prev^.filename;
        FoundNext := TRUE;
      end;
    pointer := pointer^.next;
  end;
  if NOT(FoundNext) then
    beep
  else
  begin
    HelpMenuWindow^.ClearWindow;
    HelpMenuWindow^.ShowText;
  end;
end;
```

File Name: SAVEMENU.PAS

```
(*************************************************************************
*                                                                       *
*        PROGRAM NAME - Omega Performance Assessment                    *
*                                and                                    *
*                          Coverage Evaluation                          *
*                                (PACE)                                 *
*                             Workstation                               *
*                                                                       *
*           UNIT NAME - Part of the PACEOBJS Unit                       *
*                                                                       *
*************************************************************************
*                                                                       *
*        This program was prepared by                                   *
*                                                                       *
*              The Analytic Science Corporation (TASC)                  *
*                    55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                          *
*                                                                       *
*        PACE has been developed to run on a IBM PC/AT or compatible    *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main      *
*        memory and an EGA or compatible graphics adapter and color     *
*        monitor. This work was performed under contract number         *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for      *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.   *
*                                                                       *
*************************************************************************
*                                                                       *
*        PURPOSE                                                        *
*           This file contains object methods for the generic save      *
*           window.                                                     *
*                                                                       *
*                                                                       *
*************************************************************************)

constructor SaveMenu.Init(InitX,InitY,InitWidth,InitHeight : integer;
                          InitColor,InitHiliteColor,InitMemoColor,
                          InitMemoHiliteColor,InitBorderColor,
                          InitTextColor,InitHiliteTextColor,InitFont: word;
                          Slink:ActionProcedure;InitFileString:String);
      {
      *************************************************************************
      *                                                                       *
      *        PURPOSE                                                        *
      *           Initialization for the save menu window.                    *
      *                                                                       *
      *************************************************************************
      }
const
  sw      =  60;     (* save button width  *)
```

```
  sh      = 16;     (* save button height *)
  ew      = 160;    (* edit button width  *)
  eh      = 15;     (* edit button height *)

begin
  TextColor   := InitTextColor;
  Title       := 'Archive File Annotation';
  TextLabel   := 'FILENAME:';
  xoffset     := 5;
  yoffset     := 6 + textheight(TextLabel);
  menu      .Init(InitX,InitY,InitWidth,InitHeight,InitColor,
                  InitBorderColor,2,WindowShadowWidth);
  MemoBox   .Init(InitX+xoffset,InitY+yoffset,InitFont,InitMemoColor,
                  InitMemoHiLiteColor,InitTextColor,InitHiliteTextColor,
                  black);
  SaveButton.Init(InitX+InitWidth-sw-4*xoffset,
                  InitY+yoffset+MemoBox.Height+5,
                  sw,sh,white,black,black,white,'Save',SLink,'S');
  FileButton.Init(InitX+xoffset+textwidth(TextLabel)+12,
                  InitY+Yoffset+MemoBox.Height+5,
                  ew,eh,white,black,black,white,InitFileString,0,4,19,FALSE);
end;


procedure SaveMenu.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Method to display the save menu window.                    *
      *                                                                      *
      ************************************************************************
      }

var
  xx,yy: integer;

begin
  Get_CursorXY(xx,yy);
  Hide_Cursor;
  Window.Show;
  MemoBox.Show;
  setcolor(TextColor);
  settextjustify(centertext,centertext);
  outtextxy(X+(Width DIV 2),Y+(yoffset DIV 2)+2,Title);

  settextjustify(righttext,centertext);
  outtextxy(FileButton.X-5,FileButton.Y+(FileButton.Height DIV 2)+1,TextLabel);
  SaveButton.Show;
```

```
  FileButton.Show;
  Show_Cursor(xx,yy);
end;

procedure SaveMenu.Hilite(Xpos,Ypos: word);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Method to highlight the save menu window.                          *
      *                                                                              *
      ********************************************************************************
      }
begin
  MemoBox.Hilite(Xpos,Ypos);
  SaveButton.Hilite(Xpos,Ypos);
  FileButton.Hilite(Xpos,Ypos);
end;

function SaveMenu.Action(Xpos, Ypos: word):boolean;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Action method for the save menu that handles the memobox,          *
      *           save, and file name buttons.                                       *
      ********************************************************************************
      }

var
  dummy    : boolean;

begin
   dummy := MemoBox.Action(Xpos,Ypos);
   dummy := SaveButton.Action(Xpos,Ypos);
   dummy := FileButton.Action(Xpos,Ypos);
end;

function SaveMenu.KeyAction(Xpos,Ypos:word; key:integer):boolean;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Keyaction method for the save menu that handles the memobox*
      *           save, and file name buttons.                                       *
      ********************************************************************************
      }

var
```

File Name: SAVEMENU.PAS

```
  dummy : boolean;

begin
  KeyAction := false;
  if (SaveButton.KeyAction(Xpos,Ypos,key)) then
    KeyAction := true;
end;

procedure SaveMenu.Cancel;
      {
      ********************************************************************************
      *                                                                              *
      *         PURPOSE                                                              *
      *            Method to remove the save menu from the screen.                   *
      ********************************************************************************
      }

begin
  menu.Cancel;
end;
```

File Name: STATBARS.PAS

```
(*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                              (PACE)                                    *
*                           Workstation                                  *
*                                                                        *
*           UNIT NAME - Part of the PACEOBJS Unit.                       *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the methods associated with the lower     *
*           and side status bar objects.                                 *
*                                                                        *
*                                                                        *
*************************************************************************)

Function contolocase(st:string):string;
      (
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Utility routine to convert a string to all lower case.     *
      *                                                                      *
      ************************************************************************
      )
VAR
  i: byte;
  temp:string;
begin
   temp := st;
   FOR i := 1 to length(st) DO
```

```
    IF (ord(temp[i]) >= ord('A')) THEN  { if it's a character, convert it }
      temp[i] := chr(ord(temp[i])+32);
  contolocase := temp;
end;

function contostr(v:real;a,b:integer): string;
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Utility routine to convert a real value to a string.           *
    *                                                                         *
    ***************************************************************************
    }
VAR
  st:string;
begin
  str(v:a:b,st);
  contostr := st;
end;

constructor lowerstatusbar.Init(InitX,InitY,InitWidth,InitHeight,InitColor,InitBorterColor:w
          Initeptr:emenuptr);
    {
    ***************************************************************************
    *                                                                         *
    *       PURPOSE                                                           *
    *          Initialization method for the lower status bar.                *
    *                                                                         *
    ***************************************************************************
    }
begin
  borderedarea.init(InitX,InitY,InitWidth,InitHeight,InitColor,InitBorterColor,1);
  eptr := initeptr;
  MinPsa := 0;
  MeanPsa := 0;
  MaxPsa := 0;
  RegMinPsa := 0;
  RegMeanPsa := 0;
  RegMaxPsa := 0;
  Archivefile := '';
  Weightsfile := '';
  QRfile      := '';
end;

procedure lowerstatusbar.show;
    {
    ***************************************************************************
    *                                                                         *
```

File Name: STATBARS.PAS

```
      *        PURPOSE                                                    *
      *           Display        method for the lower status bar.        *
      *                                                                   *
      *********************************************************************
      }
CONST
  xi = 4;
  yi = 3(2);
  xi2 =-1;
VAR
   i,xx,yy:integer;
   xp,yp:integer;
   tmpstr,st: shortstring;
   ts:textsettingstype;
   d:dirstr;
   n:namestr;
   e:extstr;

begin
   get_cursorXY(xx,yy);
   hide_cursor;
   borderedarea.show;
   settextstyle(smallfont,horizdir,4);
   settextjustify(lefttext,toptext);
   { row 1 }
   xp := X + xi;
   yp := y + yi;

   CASE eptr^.PsaReportMode.picked OF
      PsaMIN  : st := ' MIN';
      PsaMEAN : st := 'MEAN';
      PsaMAX  : st := ' MAX';
   end; { case }
   outtextxy(xp,yp,st);
   xp := xp+xi+textwidth('12345');

   FOR i := 1 TO 8 DO
   begin
      IF eptr^.stationpower[i].ON.selected THEN
      begin
         outtextxy(xp,yp,stationnames[i] + ' ');
         st := contostr(eptr^.stationpower[i].sld.value.value,
                        eptr^.stationpower[i].sld.value.int,
                        eptr^.stationpower[i].sld.value.frac);
         outtextxy(xp+textwidth('  OFF')-textwidth(st),yp,st)
      end
      else
         outtextxy(xp,yp,stationnames[i] + ' OFF');
      xp := xp + textwidth('A OFF') + xi;
```

```
    end;


    { row 2 }
    xp := X+xi;
    yp := Y+2*yi+textheight('1');
    outtextXY(xp,yp,' P  ');
    outtextXY(xp,yp+2,'  SA');
    xp := xp+xi+textwidth('12345');
    outtextxy(xp,yp,'SNR');
    xp := xp+xi+textwidth('SNR');
    outtextxy(xp,yp,'S/L');
    xp := xp+xi+textwidth('S/L');
    outtextxy(xp,yp,'ANG');
    xp := xp+xi+textwidth('ANG');
    outtextxy(xp,yp,' DM');
    xp := xp+xi+textwidth(' DM');
    outtextxy(xp,yp,'DEV');
    xp := xp+xi+textwidth('DEV');
    outtextxy(xp,yp,'FRQ');
    xp := xp+xi+textwidth('FRQ');
    outtextxy(xp,yp,'SRM ');
    xp := xp+xi+textwidth('SRM');
    { adjusted width of PSA display to allow for
      addition of GDOP criteria 9/13/90 GRD }
    outtextxy(xp,yp,'GDP ');
    xp := xp+xi+textwidth('GDP');
    outtextxy(xp,yp,'P  ');
    outtextxy(xp,yp+2,' SA');
    xp := xp+xi+textwidth('PSA');
    outtextxy(xp,yp,'MIN');
    xp := xp+xi+textwidth('MIN');
    outtextxy(xp,yp,' MEAN');
    xp := xp+xi+textwidth(' MEAN');
    outtextxy(xp,yp,'MAX');

    { row 3 }
    xp := X+xi;
    yp := Y+3*yi+2*textheight('1');
    outtextxy(xp,Y+3*yi+2*textheight('1'),contostr(eptr^.PSA.value.value,
            eptr^.PSA.value.int,eptr^.PSA.value.frac));
    xp := xp+xi+textwidth('12345');
    outtextxy(xp+textwidth('SNR')-textwidth(contostr(eptr^.SNR.value.value,eptr^.SNR.value.in
            ,yp,contostr(eptr^.SNR.value.value,eptr^.SNR.value.int,eptr^.SNR.value.frac));
    xp := xp+xi+textwidth('SNR');
    outtextxy(xp+textwidth('S/L')-
              textwidth(contostr(eptr^.ShortLongRatio.value.value,
              eptr^.ShortLongRatio.value.int,eptr^.ShortLongRatio.value.frac))
              ,yp,
```

```
            contostr(eptr^.ShortLongRatio.value.value,eptr^.ShortLongRatio.value.int,eptr^.
   xp := xp+xi+textwidth('S/L');
   outtextxy(xp+textwidth('ANG')-textwidth(contostr(eptr^.XAngle.value.value,
            eptr^.XAngle.value.int,eptr^.XAngle.value.frac)),yp,
            contostr(eptr^.XAngle.value.value,eptr^.XAngle.value.int,eptr^.XAngle.value.fra
   CASE eptr^.DM.picked OF
      DMOFF : st := 'OFF';
      DMON  : st := ' ON';
   end; { case }
   xp := xp+xi+textwidth('ANG');
   outtextxy(xp+textwidth(' DM')-textwidth(st),yp,st);
   xp := xp+xi+textwidth(' DM');
   outtextxy(xp+textwidth('DEV')-
      textwidth(contostr(eptr^.PhaseDev.value.value,eptr^.PhaseDev.value.int,eptr^.PhaseDev.
      yp,contostr(eptr^.PhaseDev.value.value,eptr^.PhaseDev.value.int,eptr^.PhaseDev.value.f
   CASE eptr^.Freq.picked OF
      FRQ102 : st := '102';
      FRQ136 : st := '136';
      FRQAND : st := 'AND';
      FRQOR  : st := ' OR';
   end; { case }
   xp := xp+xi+textwidth('DEV');
   outtextxy(xp,yp,st);
   CASE eptr^.StationReliabilityModel.picked OF
      SRMBEST : st := 'BST';
      SRMNOM  : st := 'NOM';
      SRMWRST : st := 'WST';
   end; { case }
   xp := xp+xi+textwidth('DEV');
   outtextxy(xp,yp,st);
   xp := xp+xi+textwidth('FRQ');
   { adjusted width of PSA display to allow for
     addition of GDOP criteria 9/13/90 GRD }
   outtextxy(xp+textwidth('GDP')-textwidth(contostr(eptr^.GDOP.value.value,
            eptr^.GDOP.value.int,eptr^.GDOP.value.frac)),
            yp,contostr(eptr^.GDOP.value.value,
            eptr^.GDOP.value.int,eptr^.GDOP.value.frac));
   xp := xp+xi+textwidth('GDP');
   outtextxy(xp,yp,'GLB');
   xp := xp+xi+textwidth('GLB');
   tmpstr := contostr(MinPsa*1000,3,0);
   IF MinPsa > 0.999 THEN
      tmpstr := '1.0';
   IF MinPsa = 0 THEN
      tmpstr := '0.0';
   outtextxy(xp,yp,tmpstr);
   xp := xp+xi+textwidth('123');
   tmpstr := contostr(MeanPsa*100000,5,0);
   IF MeanPsa > 0.99999 THEN
```

```
      tmpstr := '  1.0';
   IF MeanPsa = 0 THEN
      tmpstr := '  0.0';
   outtextxy(xp,yp,tmpstr);
   xp := xp+xi+textwidth('12345');
   tmpstr := contostr(MaxPsa*1000,3,0);
   IF MaxPsa > 0.999 THEN
      tmpstr := '1.0';
   IF MaxPsa = 0 THEN
      tmpstr := '0.0';
   outtextxy(xp,yp,tmpstr);

   { row 4 }
   xp := X + xi;
   yp := Y+4*yi+3*textheight('1');

   fsplit(eptr^.qrfile,d,n,e);
   e := contolocase(copy(e,2,length(e)));
   outtextxy(xp,yp,n+e);

   xp := xp+xi2+textwidth('DDDDDDDD.mmm');
   fsplit(eptr^.weightsfile,d,n,e);
   e := contolocase(copy(e,2,length(e)));
   outtextxy(xp,yp,n+e);

   xp := xp+xi2+textwidth('DDDDDDDD.mmm');
   fsplit(archivefile,d,n,e);
   e := contolocase(copy(e,2,length(e)));
   outtextxy(xp,yp,n+e);

   { adjusted width of PSA display to allow for
     addition of GDOP criteria 9/13/90 GRD }
   xp := X+10*xi+textwidth('12345SNRS/LAngDM DEVSRMFrqGDP');
   outtextxy(xp,yp,'REG');
   xp := xp+xi+textwidth('REG');
   tmpstr := contostr(RegMinPsa*1000,3,0);
   IF RegMinPsa > 0.999 THEN
      tmpstr := '1.0';
   IF RegMinPsa = 0 THEN
      tmpstr := '0.0';
   outtextxy(xp,yp,tmpstr);
   xp := xp+xi+textwidth('123');
   tmpstr := contostr(RegMeanPsa*100000,5,0);
   IF RegMeanPsa > 0.99999 THEN
      tmpstr := '  1.0';
   IF RegMeanPsa = 0 THEN
      tmpstr := '  0.0';
   outtextxy(xp,yp,tmpstr);
   xp := xp+xi+textwidth('12345');
```

File Name: STATBARS.PAS

```
   tmpstr := contostr(RegMaxPsa*1000,3,0);
   IF RegMaxPsa > 0.999 THEN
      tmpstr := '1.0';
   IF RegMaxPsa = 0 THEN
      tmpstr := '0.0';
   outtextxy(xp,yp,tmpstr);
   line(X+1,Y+3*yi+3*textheight('1')+2,
                        X+1+9*xi+textwidth('12345SNRS/LAngDM DEVSRMFrqGDP'),Y+3*yi+3*texthe
   line(X+1+9*xi+textwidth('12345SNRS/LAngDM DEVSRMFrqGDP'),Y+2*yi+textheight('1'),
        X+1+9*xi+textwidth('12345SNRS/LAngDM DEVSRMFrqGDP'),Y+Height-1);

   xp := X+1+xi+textwidth('MEAN ');
   yp := Y+2*yi+textheight('1')-1;
   line(xp,yp,X+width-1,yp);
   FOR i := 1 to 8 DO
   begin
      line(xp,Y+1,xp,yp);
      xp := xp + xi + textwidth('A OFF');
   end;

   show_cursor(xx,yy);
   settextstyle(defaultfont,horizdir,1);
end;

constructor sidestatusbar.Init(InitX,InitY,InitWidth,InitHeight,InitColor,InitBorderColor:wo
            Initeptr:emenuptr);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Initialization method for the side  status bar.            *
      *                                                                      *
      ************************************************************************
      }
begin
   borderedarea.init(InitX,InitY,InitWidth,InitHeight,InitColor,InitBorderColor,1);
   eptr := initeptr;
end;

procedure sidestatusbar.show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Display        method for the side  status bar.           *
      *                                                                      *
      ************************************************************************
      }
var
```

File Name: STATBARS.PAS

```
   xx,yy,i: integer;
   th,tw,yi,xi: word;
begin
   get_cursorXY(xx,yy);
   hide_cursor;
   borderedarea.show;
   settextstyle(smallfont,horizdir,4);
   settextjustify(lefttext,toptext);

   th := textheight('1')+2;
   tw := textwidth('1')+1;
   yi := 2;
   xi := 4;
   setcolor(bordercolor);
   FOR i := 1 TO 12 DO
      outtextxy(X+xi,Y+yi+(i-1)*2*th,MA[i]);

   setfillstyle(solidfill,bordercolor);
   FOR i := 1 TO 12 DO
      IF eptr^.MonthSelectors[i].selected THEN
         BAR(X+xi-2,Y+1+yi+(i-1)*2*th,X+xi+tw,Y+yi+(i-1)*2*th+th);

   setcolor(color);
   FOR i := 1 TO 12 DO
      IF eptr^.MonthSelectors[i].selected THEN
         outtextxy(X+xi,Y+yi+(i-1)*2*th,MA[i]);


   setcolor(bordercolor);
   line(X+xi+tw+3,Y+2,X+xi+tw+3,Y+height-2);

   { now write the hour choices }
   tw := textwidth('11')+1;
   xi := 2*xi+textwidth('1') + 4;
   setcolor(bordercolor);
   FOR i := 1 TO 24 DO
      outtextxy(X+xi,Y+yi+(i-1)*th,HourNames[i]);

   setfillstyle(solidfill,bordercolor);
   FOR i := 1 TO 24 DO
      IF eptr^.HourSelectors[i].selected THEN
         bar(X+xi-2,Y+1+yi+(i-1)*th,X+xi+tw,Y+yi+(i-1)*th+th);

   setcolor(color);
   FOR i := 1 TO 24 DO
      IF eptr^.hourselectors[i].selected THEN
         outtextxy(X+xi,Y+yi+(i-1)*th,HourNames[i]);

   show_cursor(xx,yy);
```

```
   settextstyle(defaultfont,horizdir,1);
end;
```

File Name: TASCLOGO.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                              (PACE)                                    *
*                           Workstation                                  *
*                                                                        *
*           UNIT NAME - TASCLOGO Unit                                    *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by:                                   *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This unit handles the generation and display of the          *
*           PACE initialization screen and also of the TASC logo         *
*           that is on the initialization screen.                        *
*                                                                        *
*************************************************************************}
UNIT tasclogo;

Interface

Uses CRT,Graph,CursrObj,ErrLog;

CONST
   logofile = 'TASCLOGO.VEC';

   PROCEDURE DisplayTASCLogo(Xpos,Ypos:integer;scale:real);

   PROCEDURE DisplayLogo;

Implementation
```

```
File Name: TASCLOGO.PAS

TYPE
   logolistptr = ^logolist;
   logolist = RECORD
              fwptr,bkptr: logolistptr;
              X, Y: Integer;
              Flag: Byte;
              count : integer;
            END;

VAR
   logo_List: logolistptr;

procedure DisplayLogo;

      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Utility routine to display the PACE specific initialization*
      *           screen.                                                            *
      ********************************************************************************
      }
var
  key,button,
  yoffset,xoffset,
  x,y                : integer;
  ch :char;
begin

   setfillstyle(solidfill,white);
   settextstyle(Triplexfont, HorizDir,5);
   settextjustify(lefttext,toptext);
   bar(0,0,getmaxx,getmaxy);
   SetLineStyle(solidln,0,ThickWidth);
   setcolor(blue);
   X := (GETMAXX DIV 2) - (textwidth('P A C E') DIV 2) ;
   Y := 50;
   yoffset := textheight('1');
   rectangle(X-8,Y+4,X+textwidth('P A C E')+4,Y+textheight('C')+8);
   Outtextxy(X,Y,'P A C E');
   settextstyle(Triplexfont, HorizDir,3);

   x := x - 65;

   setcolor(red);
   Outtextxy(X,Y+yoffset+textheight('1')*1 ,'P');
   setcolor(blue);
   Outtextxy(X+textwidth('P'),Y+yoffset+textheight('1')*1 ,'erformance');
   setcolor(red);
```

File Name: TASCLOGO.PAS

```
   Outtextxy(X+textwidth('Performance  '),Y+yoffset+textheight('1')*1 ,'A');
   setcolor(blue);
   Outtextxy(X+textwidth('Performance  A'),Y+yoffset+textheight('1')*1 ,'ssessment');

   Outtextxy(GetMaxX DIV 2 - TEXTWIDTH('&') DIV 2, Y+yoffset+textheight('1')*2,'&');

   x := x + 30;
   setcolor(red);
   Outtextxy(X,Y+yoffset+textheight('1')*3 ,'C');
   setcolor(blue);
   Outtextxy(X+textwidth('C'),Y+yoffset+textheight('1')*3 ,'overage');
   setcolor(red);
   Outtextxy(X+textwidth('Coverage  '),Y+yoffset+textheight('1')*3 ,'E');
   setcolor(blue);
   Outtextxy(X++textwidth('Coverage  E'),Y+yoffset+textheight('1')*3 ,'valuation');

   Y := Y+yoffset+textheight('1')*5 + 15;

   settextstyle(Triplexfont, HorizDir,1);

   outtextxy(GetMaxX DIV 2 - textwidth('developed for') DIV 2,Y,'developed for');
   outtextxy(GetMaxX DIV 2 - textwidth('United States Coast Guard') DIV 2,Y+textheight('1')*
   outtextxy(GetMaxX DIV 2 - textwidth('OMEGA Navigation System Center') DIV 2,
             Y+textheight('1')*3,'OMEGA Navigation System Center');




   rectangle(0+4,0+4,getmaxx-4,getmaxy-4);
   DisplayTascLogo(getmaxx-150,getmaxy-40,0.13);
   settextstyle(defaultfont,horizdir,1);

   repeat
      button := get_buttons;
      Get_key(key);
   until (key <> 0) or (button = left_button_pressed) or
         (button = right_button_pressed);

end;

PROCEDURE Read_logolist;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Procedure to get the list of vertices for the TASC logo.   *
      *                                                                      *
      ************************************************************************
      }
```

File Name: TASCLOGO.PAS

```
                              {    Read a list file and attach to the listptr }
VAR
   Temp: logolistptr;
   Last: logolistptr;
   flist: Text;
   counter : integer;

BEGIN
   {$I-}
   Assign(flist, logofile);
   Reset(flist);
   {$I+}
   IF IOResult <> 0 THEN
      Log_Error('logo file not available')
   ELSE BEGIN
      logo_List := NIL;
      Last := logo_list;
      counter := 0;
      WHILE NOT EOF(flist) DO BEGIN
         inc(counter);
         New(Temp);
         IF (Temp = nil) THEN BEGIN
            Log_Error('Insufficient memory for reading in TASC Logo');
            Exit;
         END;
         Readln(flist, Temp^.flag, Temp^.X,Temp^.Y);
         Temp^.count := counter;
         Temp^.fwptr := nil;
         IF Last = nil THEN                  { Our first node ?}
            logo_List := Temp
         ELSE                                { nope.  Link it in }
            Last^.fwptr := Temp;
         temp^.bkptr := Last;
         Last := Temp;
      END;
      Close(flist);
   END;
END;  {Read_List}

PROCEDURE Deallocate_logo_List;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Utility routine to release the memory allocated to the             *
      *           TASC logo.                                                         *
      ********************************************************************************
      }
VAR
```

```
File Name: TASCLOGO.PAS

   Last: logolistptr;

BEGIN
   Last := logo_List;
   WHILE Last^.fwptr^.fwptr <> NIL DO     {get to the next to last list item}
      Last := Last^.fwptr;
   While Last <> logo_list DO
   BEGIN
      dispose(last^.fwptr);
      last := last^.bkptr;
   END;
   dispose(last^.fwptr);
   dispose(last);
END;


PROCEDURE DisplayTASCLogo(Xpos,Ypos:integer;scale:real);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Procedure to draw the logo on the screen.                     *
      *                                                                         *
      ***************************************************************************
      }

VAR
   X, Y: Integer;
   X_Center, Y_Center: Integer;
   listptr: logolistptr;
   n:integer;
   pts: ARRAY[0..1000] of pointtype;
   xasp,yasp:word;
   col:word;

BEGIN
   read_logolist;
   SetViewPort(0,0,GetMaxX,GetMaxY,ClipOn);
   listptr := logo_list;
   setcolor(blue);                    { the OFFICIAL TASC color }
   SetLineStyle(SolidLn,0,normwidth);
   setfillstyle(solidfill,blue);

   REPEAT
      getaspectratio(xasp,yasp);
      n := 0;
      REPEAT
         pts[n].x := Xpos+Round(listptr^.X*scale);;
         pts[n].y := Ypos+round(listptr^.y*xasp/yasp*scale);
```

```
        inc(n);
        listptr := listptr^.fwptr;
      UNTIL (listptr = nil) OR (listptr^.flag <> 2);
      fillpoly(n,pts);
   UNTIL listptr = nil;

   deallocate_logo_list;
END;

BEGIN
   ( no initialization section )
END.
```

File Name: WORLDMAP.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                             (PACE)                                     *
*                          Workstation                                   *
*                                                                        *
*          UNIT NAME - WORLDMAP Unit                                     *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (CNSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*          This unit contains utility routines for initializing          *
*          and displaying the world map and conversion utilities         *
*          for lat/lon to screen conversion.                             *
*                                                                        *
*************************************************************************}

UNIT WorldMap;

INTERFACE

USES
   Graph, Errlog;

TYPE
   xypt = RECORD
       x,y: integer;
   END;

VAR
   Map_Center,
   Screen_Center,
```

```
File Name: WORLDMAP.PAS

   Scale_Factor   : xypt;

PROCEDURE Convert_To_Screen_Position(xin,yin:integer; VAR xout,yout:integer);
             { converts from real world coordinates to screen coordinates }
             { accounting for current zoom level and current screen center }

PROCEDURE Convert_To_Real_Position(xin,yin:Integer; VAR xout,yout:integer);
             { converts from screen coordinates to real world coordinates }
             { accounting for current zoom level and current screen center }

PROCEDURE InitWorldMap;
             { reads in map points }

CONST
   NUMMAPPTS = 7500;

TYPE
   mappt = RECORD
       dtype: char;
       lat, lon: Integer;
    END;

VAR
   fmap: file (*of mappt*);
   Map_Points: array[1..NUMMAPPTS] of mappt;

IMPLEMENTATION

PROCEDURE Convert_To_Screen_Position(xin,yin:integer; VAR xout,yout:integer);
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           converts from real world coordinates to screen coordinates *
      *           inputs are lat/lon in deg*100 units                        *
      *           accounting for current zoom level                          *
      *                                                                      *
      *                                                                      *
      ************************************************************************
      }
  BEGIN

(************************************************************************
  not needed as of 04-09-90 for the PACE project...

      CASE Sel_Map OF
         Mercator_Projection: BEGIN
            IF xin < -18000 THEN xin := -18000;
            IF xin >  18000 THEN xin := 18000;
```

```
            xout := X_Mercator[xin DIV 50];
            yout := Y_Mercator[yin DIV 25];
         END;
         CED_Projection: BEGIN
*****************************************************************************)

    xout := ((xin - Screen_Center.x) DIV Scale_Factor.x) + Map_Center.x;
    yout := Map_Center.y - ((yin - Screen_Center.y) DIV Scale_Factor.y);

   END;



PROCEDURE Convert_To_Real_Position(xin,yin:Integer; VAR xout,yout:integer);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           converts from screen coordinates to real world coordinates *
      *           accounting for current zoom level and current screen center*
      *           output is in lat/lon in units of deg*100                      *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }
   BEGIN

(*************************************************************************
  not needed as of 04-09-90 for the PACE project...

      CASE Sel_Map OF
         Mercator_Projection: BEGIN
            xout := -360;
            WHILE (X_Mercator[xout] < xin) AND (xout < 360) DO Inc(xout);
            xout := xout * 50;
            yout := -360;
            WHILE (Y_Mercator[yout] > yin) AND (yout < 360) DO Inc(yout);
            yout := yout * 25;
         END;
         CED_Projection: BEGIN
   *************************************************************************)

   xout := ((xin - Map_Center.x) * Scale_Factor.x) + Screen_Center.x;
   yout := Screen_Center.y - ((yin - Map_Center.y) * Scale_Factor.y);

   END;

procedure InitWorldMap;
      {
```

File Name: WORLDMAP.PAS

```
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Initialize the unit by setting default zoom levels, etc            *
      *                                                                              *
      ********************************************************************************
      }


var
   i: Integer;
   NumRead : word;

begin
                                          { Load the map point data into memory }
   for i := 1 to NUMMAPPTS do
      Map_Points[i].Dtype := 'Z';                { initialize to invalid type }

   {$I-}
   assign(fmap, 'World.map');
   reset(fmap,SizeOf(Map_Points));
   {$I+}

   if IOResult = 0 then begin
     BlockRead(fmap,Map_Points,SizeOf(Map_Points),NumRead);
     if NumRead = 0 then begin
       i := 0;
       while i < NUMMAPPTS do begin
         Inc(i);
         with Map_Points[i] do begin
           Lat := lat * 10;
           Lon := lon * 10;
           if Lat >   9000 then Lat :=   9000;
           if Lat <  -9000 then Lat :=  -9000;
           if Lon >  18000 then Lon :=  18000;
           if Lon < -18000 then Lon := -18000;
         end;
       end;
       close(fmap);
     end
     else begin
       Log_Error('Problem reading Map Data');
    end
  end
  else
    Log_Error('File World.Map NOT FOUND');
end;

BEGIN
```

File Name: WORLDMAP.PAS

```
   Screen_Center.x := 0;        (* map coordinates are relative to the *)
   Screen_Center.y := 0;        (*    center at (0,0)                  *)

   Scale_factor.x  := 56;       (* initial scale factor setting that   *)
   Scale_factor.y  := 58;       (*    utilizes the whole screen        *)

   Map_Center.x    := 320;      (* This represents the center of the   *)
   Map_Center.y    := 175;      (*    screen in EGA mode 640 x 350     *)
END.
```

APPENDIX G
QRBUILD LISTINGS

This appendix contains the Pascal code listing for the QR builder program. A discussion of this program is provided in Section 5.4.

File Name: QRBUILD.SRC

The QRBUILD program uses the following source program files as well as other PACE units:

QRBUILD.PAS
QRCOMPUT.PAS
QRUTIL.PAS

File Name: QRBUILD.PAS

```
{*************************************************************************
*                                                                        *
*       PROGRAM NAME - Omega Performance Assessment                      *
*                                and                                     *
*                          Coverage Evaluation                           *
*                              (PACE)                                    *
*                           Workstation                                  *
*                                                                        *
*         UNIT NAME - QRBUILD Program                                    *
*                                                                        *
*************************************************************************
*                                                                        *
*       This program was prepared by                                     *
*                                                                        *
*             The Analytic Science Corporation (TASC)                    *
*                   55 Walkers Brook Drive                               *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*       PACE has been developed to run on a IBM PC/AT or compatible      *
*       under MS-DOS 3.3 or higher with a minimum of 640K of main        *
*       memory and an EGA or compatible graphics adapter and color       *
*       monitor. This work was performed under contract number           *
*       DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for        *
*       the Omega Navigation System Center (ONSCEN), Alexandria, VA.     *
*                                                                        *
*************************************************************************
*                                                                        *
*       PURPOSE                                                          *
*         This program serves as an editor for the station reliability   *
*         values, a builder for precomputing QR matrix sets, and an      *
*         archiver/loader for QR set archive management.                 *
*                                                                        *
*************************************************************************}
{$N+,E+}
program QR_builder_and_station_reliability_editor_for_PACE;
Uses PaceInit,DOS,Crt,ConMan,Controls,Paceobjs,
     cursrobj,Graph,ErrLog,QRcomput,QRUtil,TascLogo;

Const
   mc = cyan;           { menu color }
   mbc = black;         { menu border color }
   bc = white;          { button color }
   hc = blue;           { hilite color }
   sc = yellow;         { selected color }
   tc = black;          { text color }
   htc = white;         { hilited text color }
   stc = black;         { selected text color }
   ems = 18;            { edit menu spacing constant }
```

File Name: QRBUILD.PAS

```
   DEFAULTQRFILE = 'DEFAULT:QR';
VAR
   { Really wanted to declare monthofparams an array of monthofreliabilities
     but... it would be a forward. I fake it out later by filling it in
     using monthofparamsinit, an array of monthofreliabilities
     (defined later). }
   monthofparams:array[1..12] of ^Control;

   QRDatabase: QRDatabaseType;
   SaveComment:CommentAreaType;

TYPE
   MonthOfReliabilities = object(control)
      unscheduled,scheduled,maintenance: Array [1..8] of NumberButton;
      Constructor  Init;
      Function Action(Xpos,Ypos:word):boolean;virtual;
      Procedure Show;virtual;
      Procedure HiLite(Xpos,Ypos:word);virtual;
   end;

   MonthSelectionBar = object(MutExclusiveN)
      function Action(Xpos,Ypos:word):boolean;virtual;
   end;

   Function MonthSelectionBar.Action(Xpos,Ypos:word):boolean;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Action method for the user month selection bar.               *
      *                                                                         *
      ***************************************************************************
      }
   VAR
      i:integer;
      currentlyselected:integer;
   begin
      FOR i := 1 to NumberButtons DO
         IF ButtonArray[i].selected THEN
            currentlyselected := i;
      IF MutExclusiveN.Action(Xpos,Ypos) THEN
      begin
         deletefromcontext('basescreen',monthofparams[currentlyselected]);
         FOR i := 1 TO NumberButtons DO
            IF ButtonArray[i].Selected THEN
            begin
               {show the parameters for the month selected }
               addtocontext('basescreen',monthofparams[i]);
               monthofparams[i]^.show;
```

```
            end;
      end;
   end;

   Constructor MonthOfReliabilities.Init;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Initialization for the object that contains reliability    *
      *           values for a month for all stations.                       *
      ************************************************************************
      }
   VAR
      i:integer;
   begin
      control.init(30,100,100,100,white);
      FOR i := 1 to 8 DO
      begin
         unscheduled[i].Init(X+ 20,Y+(i-1)*ems,110,10,bc,hc,tc,htc,0.0,1,10);
         scheduled[i].Init  (X+220,Y+(i-1)*ems,110,10,bc,hc,tc,htc,0.0,1,10);
         maintenance[i].Init(X+420,Y+(i-1)*ems,110,10,bc,hc,tc,htc,0.0,1,10);
      end;
   end;

   Function MonthOfReliabilities.Action(Xpos,Ypos:word):boolean;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Action method  for the object that contains reliability    *
      *           values for a month for all stations.                       *
      ************************************************************************
      }
   VAR
      i:integer;
      dummy:boolean;
      s:string;
   begin
      FOR i := 1 to 8 DO
      begin
         IF unscheduled[i].Action(Xpos,Ypos) THEN
         begin
            Action := TRUE;
            IF unscheduled[i].value > 1.0 THEN
               unscheduled[i].value := 1.0;
            str(unscheduled[i].value:unscheduled[i].int:unscheduled[i].frac,s);
            Hide_Cursor;
            unscheduled[i].ChangeName(s);
```

```
            Show_Cursor(Xpos,Ypos);
         end;
         IF scheduled[i]  .Action(Xpos,Ypos) THEN
         begin
            Action := TRUE;
            IF scheduled[i].value > 1.0 THEN
               scheduled[i].value := 1.0;
            str(scheduled[i].value:scheduled[i].int:scheduled[i].frac,s);
            Hide_Cursor;
            scheduled[i].ChangeName(s);
            Show_Cursor(Xpos,Ypos);
         end;
         IF maintenance[i].Action(Xpos,Ypos) THEN
         begin
            Action := TRUE;
            IF maintenance[i].value > 1.0 THEN
               maintenance[i].value := 1.0;
            str(maintenance[i].value:maintenance[i].int:maintenance[i].frac,s);
            Hide_Cursor;
            maintenance[i].ChangeName(s);
            Show_Cursor(Xpos,Ypos);
         end;
      end;
   end;

   Procedure MonthOfReliabilities.Show;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Display method for the object that contains reliability    *
      *           values for a month for all stations.                       *
      ************************************************************************
      }
   VAR
      i:integer;
   begin
      settextjustify(lefttext,toptext);
      setcolor(black);
      outtextxy(unscheduled[1].X,unscheduled[1].Y-ems*2,'UNSCHEDULED');
      outtextxy(unscheduled[1].X,unscheduled[1].Y-ems,'PROBABILITY');
      outtextxy(scheduled[1].X,scheduled[1].Y-ems*2,'SCHEDULED');
      outtextxy(scheduled[1].X,scheduled[1].Y-ems,'PROBABILITY');
      outtextxy(maintenance[1].X,maintenance[1].Y-ems*2,'MAINTENANCE');
      outtextxy(maintenance[1].X,maintenance[1].Y-ems,'PROBABILITY');
      FOR i := 1 to 8 DO
      begin
         unscheduled[i].Show;
         scheduled[i]  .Show;
```

```
        maintenance[i].Show;
        settextjustify(lefttext,toptext);
        outtextxy(unscheduled[i].X - 15,unscheduled[i].Y+2,stationnames[i]);
        outtextxy(scheduled[i].X - 15,scheduled[i].Y+2,stationnames[i]);
        outtextxy(maintenance[i].X - 15,maintenance[i].Y+2,stationnames[i]);
      end;
   end;

   Procedure MonthOfReliabilities.Hilite(Xpos,Ypos:word);
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Highlight method for the object that contains reliability          *
      *           values for a month for all stations.                               *
      ********************************************************************************
      }
   VAR
      i:integer;
   begin
      FOR i := 1 to 8 DO
      begin
         unscheduled[i].HiLite(Xpos,Ypos);
         scheduled[i]  .HiLite(Xpos,Ypos);
         maintenance[i].HiLite(Xpos,Ypos);
      end;
   end;

VAR
   { The following declaration is used only to 'help' load
     up the monthofparams array with the pointers to the
     correct object types. This is done this way because
     monthofparams is an array of control object type pointers, not
     monthofreliability type pointers, so it cannot be used in the
     new statement for initialization. }
   monthofparamsinit:array[1..12] of ^MonthOfReliabilities;


{$F+}
   procedure filebutton;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           Action procedure to enable the file menu context.                  *
      *                                                                              *
      ********************************************************************************
      }
   begin
```

```
    pushcontext('filemenu');
    showcontext;
  end;

  procedure quitbutton;
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *          Action procedure for the quit button.                               *
    *                                                                              *
    ********************************************************************************
    }
  begin
    QUIT := TRUE;
  end;

  procedure savebutton;
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *          Action procedure to enable the save menu context.                   *
    *                                                                              *
    ********************************************************************************
    }
  begin
    SaveMenuWindow^.MemoBox.GetNewText(SaveComment);
    pushcontext('savemenuwindow');
    showcontext;
  end;

  procedure compute_and_write_QR;
    {
    ********************************************************************************
    *                                                                              *
    *        PURPOSE                                                               *
    *          Action procedure that controls the computation of the QR            *
    *          set and writes it out to a QR save file.                            *
    ********************************************************************************
    }
  VAR
    QR_file:file of QRDatabaseType;
    coverage,ncm,i,j,xx,yy: integer;
    s,m:string;
    dirinfo: searchrec;
    filename : string;
    dir: dirstr;
    name: namestr;
```

File Name: QRBUILD.PAS

```
    ext: extstr;

  begin
    { At this point, we should have alread chosen the QR file to write
      to, and should already have entered a comment to store
      at the beginning of the file. }
    { the following code is a first cut until the save pop up window
      is finished. CHECK THIS CODE WHEN THE SAVE POPUP WINDOW IS
      FINISHED !!!!!!!!!!!!!!!!!!!! }

      { write the reliability values to the QRDatabase first }
      FOR i := 1 TO 12 DO
        FOR j := 1 TO 8 DO
        begin
          QRDatabase.stationreliabilities[i,j,1] :=
            MonthOfParamsInit[i]^.unscheduled[j].value;
          QRDatabase.stationreliabilities[i,j,2] :=
            MonthOfParamsInit[i]^.scheduled[j].value;
          QRDatabase.stationreliabilities[i,j,3] :=
            MonthOfParamsInit[i]^.maintenance[j].value;
        end;

    { The Pa values will be stored in the following order:
       the 'best case' Pa numbers for stations A..H for January
       the 'best case' Pa numbers for stations A..H for February
                          .
                          .
       the 'best case' Pa numbers for stations A..H for December
       the 'nominal case' Pa numbers for stations A..H for January
       the 'nominal case' Pa numbers for stations A..H for February
                          .
                          .
       the 'nominal case' Pa numbers for stations A..H for December }

    { First compute the 'best case' which does not use maintenance
      probabilities in the probability of scheduled off air. The nominal
      case, done second, uses the sum of the maintenance probability and
      scheduled off air for the total probability of scheduled off air.
    }

    { pop up a little status window }
    messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Building Nominal case for Jan'
                              (GetMaxY DIV 2) - (textheight('1')),
                              1,
                              length('Building Nominal case for Jan'),
                              lightred,
                              white,
                              '');
    { Run through each month and compute the QR values for that month's
```

File Name: QRBUILD.PAS

```
      particular reliability numbers )
    FOR Nom := 0 TO 1 DO ( 0: do not include maint, 1: include maint )
    begin
      IF nom = 0 THEN
        s := 'BEST'
      else
        s := 'NOMINAL';
      FOR i := 1 TO 12 DO
      begin
        m := MonthNames[i];
        messagebox.redisplaymessage(1,
                          Length('Building Nominal case for    '),
                          'Building '+s+' case for '+m);

        ( Copy the scheduled and unscheduled maintenance probabilities
          for each station into the data area in the QR builder. )
        FOR j := 1 to 8 DO
        begin
          ( the scheduled and unscheduled arrays in QRCOMPUT are indexed
            [0..7], so subtract one from the index )
          ( This is kind of kludgie but ... MonthOfParams and
            MonthOfParamsInit contain the same set of pointers. I really
            want to use the MonthOfParams array here BUT, it is not
            visible since it is defined below. So... I'll cheat a
            little and use MonthOfParamsInit instead. )
          sched[j-1] := MonthOfParamsInit[i]^.scheduled[j].value +
                        MonthOfParamsInit[i]^.maintenance[j].value * nom;
          unsched[j-1] := MonthOfParamsInit[i]^.unscheduled[j].value;
        end; ( FOR j := 1 TO 8 )
        ( now do the QR computation for this month )
        Compute_QR;
        ( and store the results in the QRDatabase )
        IF nom = 0 ( BEST CASE ) THEN
          FOR coverage := 0 To 255 DO
            QRDatabase.Best[i,coverage] := QR[coverage]
        ELSE
          FOR coverage := 0 To 255 DO
            QRDatabase.Nom[i,coverage] := QR[coverage];
      end; ( FOR i := 1 TO 12 )
    end; ( FOR nom := 0 TO 1 )
    messagebox.hidemessage;
    ( now write  the entire QR database record to the QR file )
    filename := SaveMenuWindow^.FileButton.Text_String;
    fsplit(filename,dir,name,ext);
    filename := name + ext;
    if filename = '' then
    begin
      messagebox.displaymessage((GetMaxX DIV 2) -
           (textwidth('Please Select An Appropriate Filename before saving...') DIV 2),
```

```
            GetMaxY DIV 2,1,
            length('Please Select An Appropriate Filename before saving...')
            ,lightred,white,
            'Please Select An Appropriate Filename before saving...');
        beep;
        messagewait;
        messagebox.hidemessage;
      end
      else
      begin
        SaveMenuWindow^.MemoBox.AssignChangedText(SaveComment);
        messagebox.displaymessage((GetMaxX DIV 2) - (textwidth('Saving to file: '+QRPATH+'\'
                                  (GetMaxY DIV 2) - (textheight('1') DIV 2),
                                  1,
                                  length('Saving to file: '+QRPATH+'\'+filename),
                                  lightred,
                                  white,
                                  'Saving to file: '+QRPATH+'\'+filename);
        Assign(QR_File,QRPATH+'\'+filename);
        Rewrite(QR_file);
        QRDatabase.Comment := SaveComment;
        write(QR_file,QRDatabase);
        Close(QR_File);
        messagebox.hidemessage;
        hidecontext; popcontext; (* wipe out the save memo window *)
        hidecontext; popcontext; (* wipe out the file pick list drop down menu *)
      end
    end;

    procedure loadbutton;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Action procedure to activate the load menu context for     *
      *           the loading of existing QR sets.                           *
      ************************************************************************
      }
    begin
      pushcontext('listmenu');
      showcontext;
    end;

    Procedure listmenuview;
      {
      ************************************************************************
      *                                                                      *
      *        PURPOSE                                                       *
      *           Action procedure for displaying the comment that is stored *
```

```
    *          with a QR set.                                              *
    ************************************************************************
    }
  VAR
    tempstring: commentareatype;
    F: file;
    dummy: char;
  begin
    IF listmenu^.directorylist.picked > 0 THEN
    begin { read the stuff in }
      Assign(F,QRPath+'\'+listmenu^.selectedfile);
      Reset(F,Sizeof(commentareatype));
      Blockread(F,tempstring,1);
      Close(F);
      messagebox.displaymessage(listmenu^.X-100,listmenu^.y+30,5,50,cyan,black,tempstring
      repeat
      until keypressed OR (get_buttons <> 0);
      IF keypressed THEN dummy := readkey;
      messagebox.hidemessage;
    end
    ELSE
    begin
      { indicate and error condition that no file is selected }
      beep;
    end;
  end;

  procedure listmenuaction;
    {
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *          Action procedure to select a QR set for loading.            *
    *                                                                      *
    ************************************************************************
    }
  VAR
    i,j: integer;
  begin
    { Get the name of the archive to read from the listmenu and
      read the parameters and the archive data into the exitmenu
      or auxeditmenu areas (depending upon which is currently active
      in the split screen or on the main screen) and fill in the
      appropriate cell grid data for the hours selected in the loaded
      parameters }
    IF listmenu^.directorylist.picked > 0 THEN
    begin { read the stuff in }
      ReadQrDatabase(listmenu^.selectedfile,@QRDatabase);
      { now assign the data into the number buttons }
```

```
        FOR i := 1 TO 12 DO
          FOR j := 1 TO 8 DO
          begin
             MonthOfParamsInit[i]^.unscheduled[j].value :=
                QRDatabase.stationreliabilities[i,j,1];
             MonthOfParamsInit[i]^.scheduled[j].value :=
                QRDatabase.stationreliabilities[i,j,2];
             MonthOfParamsInit[i]^.maintenance[j].value :=
                QRDatabase.stationreliabilities[i,j,3];
          end;
        { pop down the list menu }
        CancelContext;
        { pop down the file menu }
        CancelContext;
        { and reshow the base screen }
        ShowContext;
      end
      ELSE
      begin
        { indicate and error condition that no file is selected }
        beep;
      end;
   end;

(*************************************************************************)
   procedure stub;
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           Dummy method action procedure used for debug.                 *
      *                                                                         *
      ***************************************************************************
      }
   begin
      pushcontext('stub');
      showcontext;
   end;
{$F-}

VAR
   grmode,grdriver:integer;
   X,Y:integer;
   button:integer;
   index,station,key:integer;
   labelarray:namearray;
   monthselector: ^MonthSelectionBar;
   filemenunamelist:PickMenuNameArray;
   filemenuprocedurelist:PickMenuActionArray;
```

```
   filemenuhotkeylist:PickMenuHotKeyArray;

Begin
   Grdriver := Detect;
   Initgraph(grdriver,grmode,'');
   Setgraphmode(EGAHI);

(*
   setfillstyle(widedotfill,darkgray);
   bar(0,0,getmaxx,getmaxy);
   setcolor(blue);
   rectangle(0,0,getmaxx,getmaxy);
   DisplayTascLogo(40,50,0.6);
*)

   SetActivePage(1);

   SaveComment := '';

   { base screen and controls }
   new(basescreen,Init(0,0,639,349,lightblue,white,1));
   new(h,Init(25,4,60,16,white,black,black,white,'Help',stub,'H'));
   new(q,Init(105,4,60,16,white,black,black,white,'Quit',quitbutton,'Q'));
   new(f,Init(185,4,60,16,white,black,black,white,'File',filebutton,'F'));

   { file menu }
   filemenunamelist[1] := 'Load';
   filemenunamelist[2] := 'Save';
   filemenuhotkeylist[1] := 'L';
   filemenuhotkeylist[2] := 'S';
   filemenuprocedurelist[1] := loadbutton;
   filemenuprocedurelist[2] := savebutton;
   new(filemenu,Init(f^.X,f^.Y+f^.Height+5,2,filemenunamelist,filemenuprocedurelist,filemenu

   new(SaveMenuWindow,Init(110,145,418,100,cyan,black,white,black,
                           black,black,white,defaultfont,Compute_And_Write_QR,
                           'DEFAULT.QR'));


   FOR index := 1 TO 12 DO
   begin
      { initialize with object of type monthofreliabilities and
        copy into object of type control. }
      new(monthofparamsinit[index],Init);
      monthofparams[index] := monthofparamsinit[index];
   end;
   { Initialize the reliabilities with the defaults of table 2.1-1 contained
     in TASC TIM-5834-1-1, March 1990. All values not initialized here
     are initialized to zero in the object instantiations. }
```

File Name: QRBUILD.PAS

```
   FOR index := 1 TO 12 DO { initialize all unscheduled probs to 0.001 }
      FOR station := 1 TO 8 DO
         MonthOfParamsInit[index]^.unscheduled[station].value := 0.001;
   FOR index := 1 TO 12 DO { initialize scheduled probs to station dep. valuse }
   begin
      MonthOfParamsInit[index]^.scheduled[1].value := 0.00269;
      MonthOfParamsInit[index]^.scheduled[2].value := 0.00037;
      MonthOfParamsInit[index]^.scheduled[3].value := 0.03604;
      MonthOfParamsInit[index]^.scheduled[4].value := 0.00024;
      MonthOfParamsInit[index]^.scheduled[5].value := 0.00163;
      MonthOfParamsInit[index]^.scheduled[6].value := 0.00030;
      MonthOfParamsInit[index]^.scheduled[7].value := 0.00061;
      MonthOfParamsInit[index]^.scheduled[8].value := 0.00005;
   end;
   { now do the station/month dependent maintenance off air probs }
   MonthOfParamsInit[8]^.maintenance[1].value := 0.11057; { station A in Aug }
   MonthOfParamsInit[2]^.maintenance[2].value := 0.34569; { station B in Feb }
   MonthOfParamsInit[6]^.maintenance[3].value := 0.28628; { station C in Jun }
   MonthOfParamsInit[7]^.maintenance[4].value := 0.07895; { station D in Jul }
   MonthOfParamsInit[9]^.maintenance[5].value := 0.61490; { station E in Sep }
   MonthOfParamsInit[3]^.maintenance[6].value := 0.20511; { station F in Mar }
   MonthOfParamsInit[11]^.maintenance[7].value := 0.01726;{ station G in Nov }
   MonthOfParamsInit[10]^.maintenance[8].value := 0.32515;{ station H in Oct }


   FOR index := 1 to 12 DO
      labelarray[index] := MonthNames[index];
   new(monthselector,init(70,325,35,10,bc,hc,sc,tc,htc,stc,'MONTH',12,1,labelarray,horizonta

   new(   listmenu,Init(filemenu^.buttonarray[1].X,
                        filemenu^.buttonarray[1].y+85, cyan,black,listmenuview,
                        listmenuaction,QRPATH+'\','*.QR'));

   new(stubwindow,init);

   QRdatabase.comment := 'Default QR parameter set. GRDesrochers, 5/31/90.';

   SetActivePage(0);

   createcontext('basescreen');
   addtocontext('basescreen',basescreen);
{   addtocontext('basescreen',h);}
   addtocontext('basescreen',q);
   addtocontext('basescreen',f);
   addtocontext('basescreen',monthselector);
   addtocontext('basescreen',monthofparams[1]); { start off with JAN }
{   addtotablist('basescreen',h);}
   addtotablist('basescreen',q);
   addtotablist('basescreen',f);
```

```
   FOR index := 1 to 8 DO begin
      addtotablist('basescreen',@MonthOfParamsInit[1]^.unscheduled[index]);
      addtotablist('basescreen',@MonthOfParamsInit[1]^.scheduled[index]);
      addtotablist('basescreen',@MonthOfParamsInit[1]^.maintenance[index]);
   end;
   FOR index := 1 To 12 DO
      addtotablist('basescreen',@MonthSelector^.ButtonArray[index]);

   createcontext('filemenu');
   addtocontext('filemenu',filemenu);
{   addtocontext('filemenu',h);}
   addtocontext('filemenu',q);
   addtotablist('filemenu',@filemenu^.buttonarray[1]);
   addtotablist('filemenu',@filemenu^.buttonarray[2]);
{   addtotablist('filemenu',h);}
   addtotablist('filemenu',q);

   createcontext('listmenu');
   addtocontext('listmenu',listmenu);
{   addtocontext('listmenu',h);}
   addtocontext('listmenu',q);
   addtotablist('listmenu',@listmenu^.DirectoryMaskButton);
   FOR index := 1 TO maxlistsize DO
      addtotablist('listmenu',@listmenu^.directorylist.buttonarray[index]);
   addtotablist('listmenu',@listmenu^.VBar.Up);
   addtotablist('listmenu',@listmenu^.VBar.Down);
   addtotablist('listmenu',@listmenu^.ViewButton);
   addtotablist('listmenu',@listmenu^.LoadButton);
{   addtotablist('listmenu',h);}
   addtotablist('listmenu',q);

   createcontext('savemenuwindow');
   addtocontext('savemenuwindow',SaveMenuWindow);
{   addtocontext('savemenuwindow',h);}
   addtocontext('savemenuwindow',q);
   addtotablist('savemenuwindow',@savemenuwindow^.MemoBox);
   addtotablist('savemenuwindow',@savemenuwindow^.FileButton);
   addtotablist('savemenuwindow',@savemenuwindow^.SaveButton);
{   addtotablist('savemenuwindow',h);}
   addtotablist('savemenuwindow',q);

   createcontext('stub');
   addtocontext('stub',stubwindow);

   pushcontext('basescreen');

   ShowContext;

   show_cursor(300,200);
```

File Name: QRBUILD.PAS

```
   QUIT := FALSE;
   repeat
   begin
      Get_Key(Key);
      get_cursorXY(X,Y);
      IF (Key IN _Cursor_set) THEN
         Move_Cursor(Key, X, Y);
      HiLiteContext(X,Y);
      button := get_buttons;
      IF (button = left_button_pressed) OR (key = ENTER) THEN
         ActionContext(X,Y);
      IF (button = right_button_pressed) OR (key = ESC) THEN
         CancelContext;
      if (key <> 0) then { key has been pressed... }
         KeyActionContext(X, Y, key);
   end
   until QUIT;
   restorecrtmode;
end.
```

File Name: QRCOMPUT.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                          Coverage Evaluation                           *
*                               (PACE)                                   *
*                            Workstation                                 *
*                                                                        *
*           UNIT NAME -  QRCOMPUT Unit                                   *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by:                                   *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                     55 Walkers Brook Drive                             *
*                  Reading, Massachusetts 01867                          *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This unit contains the code for actually building a QR       *
*           set for use within PACE. The heart of the Psa algorithm      *
*           computations are performed here.                             *
*                                                                        *
*************************************************************************}
{$N+}
UNIT QRCOMPUT;

Interface
VAR
   unsched: array[0..7] of single; { unscheduled off air probabilities by station }
   sched: array[0..7] of single; { scheduled off air probabilities by station }

   QR: ARRAY[0..255] OF single;


   Procedure Compute_QR;

Implementation

VAR
```

File Name: QRCOMPUT.PAS

```
   R: ARRAY[0..255] OF single;

   { TuBarProducts holds results of prod(j=1 to n)P(TuBarij) and is indexed
     by ijklmnop (from Rijklmnop). This is part of the equation for
     Rijklmnop on p. A-10 }
   TuBarProducts : ARRAY[0..255] OF single;

   { PofTsW holds results of P(TbarSik Ti1Ti2..Tin), and is indexed by k and
     the complement of ijklmnop (from Rijklmnop). See equation A.1-11 }
   PofTsW : ARRAY[0..7,0..255] of single;

   { PofW holds the results of P(Ti1,Ti2,...Tin-1) and in indexed by
     ilklmnop, the combination of stations that are off air (from Rijklmnop).
     This holds the results of equation A.1-12 }
   PofW : ARRAY[0..255] of single;


PROCEDURE Compute_NRFs;
      {
      ***************************************************************************
      *                                                                         *
      *       PURPOSE                                                           *
      *   This procedure computes the network reliability factors (NRFs)        *
      *   for a given set of reliability numbers. Since the reliability         *
      *   numbers for a given station are indexed by month, the calculations    *
      *   are performed for all months and for all station combinations. The    *
      *   results are used in the computation of PA = P(X3). The algorithm      *
      *   given below is based upon Appendix 1 of TASC TR-5351-8-1, Omega       *
      *   System Performance Assessment, March 1989 (also listed as report      *
      *   no. CG-ONSCEN-01-89). In particular, equations A.1-11, A.1-12, and    *
      *   the general expression for the NRF contained on p. A-10 of the        *
      *   report are used.                                                      *
      *                                                                         *
      *   GRDesrochers 5/9/90  initial coding                                   *
      *                                                                         *
      *                                                                         *
      ***************************************************************************
      }


VAR
   on_air_vector: byte;
   intermediate_sum,partial_product: single;
   i,j,k:integer;
   first_on_air,on_air,calculation_index,recursive_index: byte;
   partial_product_formed: boolean;

begin
```

File Name: QRCOMPUT.PAS

```
  { The following loop computes the TuProducts quantities.
    For each particular index, the products of the unscheduled off air
    probabilities is computed for the stations whose index bit is set. For
    example, for index 10101010 (decimal 170), TuBarProducts[170] =
    U1*U3*U5*U7, where Ui is the unscheduled off air probability for
    station i }
  FOR i := 0 To 255 DO
  begin
     TuBarProducts[i] := 1;
     FOR j := 0 TO 7 DO
        IF (i SHL j) AND $80 = $80 THEN { if this term is in this product }
           TuBarProducts[i] := TuBarProducts[i] * unsched[j];
  end;

  { The following loop computes all combinations of equation A.1-11.
    Some computed terms are never used, but it is easier to compute
    them anyway and just ignore them later }
  FOR i := 0 TO 255 DO
  begin
     on_air := 0;
     WHILE on_air <= 7 DO { continue until all on airs for this vector are calculated }
     begin
        WHILE (i AND ($80 SHR on_air) = 0) AND (on_air <= 7) DO
           inc(on_air);
        IF on_air <= 7 THEN { if there is another on air for this on air vector }
        begin
           calculation_index := i XOR ($80 SHR on_air);
           PofTsW[on_air,calculation_index] := 1;
           FOR j := 0 TO 7 DO
              IF (calculation_index SHL j) AND $80 = $80 THEN { if this term is in this pro
                 PofTsW[on_air,calculation_index] :=
                    PofTsW[on_air,calculation_index] * (1 - unsched[j]);
           PofTsW[on_air,calculation_index] := PofTsW[on_air,calculation_index] * sched[on_
           inc(on_air); { bump to the next station }
        end;
     end; { WHILE on_air <= 7 }
  end;

  { The following section computes the results of equation A.1-12
    for all combinations of Tij stations and stores them in the
    PofW array. The computation has been re-ordered from the
    equation A.1-12 form to the following:

        P(Ti1 Ti2 ...Tin) = P(Ti2 Ti3 ...Tin)(1 - P(Ti1))
                          - P(Ti1)(prod j = 2 to n)(1 - P(Tij))

    This allows the use of the results of the previous loop that
    computes PofTsW because the final term in the above expression
    is equavalent to equation A.1-11 as shown below.
```

```
        P(Ti1)(prod j = 2 to n)(1 - P(Tij)) = PofTsW[1,(i2 i3 ...in)]

      The loop also uses the results of the previous iterations to build
      the next term. This handles the recursion relationship in
      equation A.1-12.
    }

    FOR i := 0 TO 255 DO
       PofW[i] := 1; { initialize all entries }
    FOR i := 1 TO 255 DO
    begin
       { First find the first indicated station by finding the first
         bit that is on. Save the results. }
       first_on_air := 0;
       WHILE (i AND ($80 SHR first_on_air) = 0) AND (first_on_air < 7) DO
          inc(first_on_air);
       recursive_index := i XOR ($80 SHR first_on_air);
       PofW[i] := PofW[recursive_index] * (1 - unsched[first_on_air])
                  - PofTsW[first_on_air,recursive_index];
    end;

    { Now compute the NRFs for all combination of station off airs }
    FOR i := 0 TO 255 DO
    begin
       on_air_vector := i XOR $FF;
       intermediate_sum := 0;
       FOR j := 0 TO 7 DO
          IF i AND ($80 SHR j) <> 0 THEN
          begin
             partial_product := 1;
             partial_product_formed := FALSE;
             FOR k := 0 TO 7 DO
               IF i AND ($80 SHR k) <> 0 THEN
                begin
                   partial_product_formed := TRUE;
                   IF j = k THEN
                      partial_product := partial_product * PofTsW[k,on_air_vector]
                   else
                      partial_product := partial_product * unsched[k];
             end; { FOR k := 0 TO 7 }
             IF partial_product_formed THEN
                intermediate_sum := intermediate_sum + partial_product;
       end; { FOR j := 0 TO 7 }
       R[i] := PofW[on_air_vector] * TuBarProducts[i] + intermediate_sum;
    end;
end;
```

File Name: QRCOMPUT.PAS

```
Procedure Compute_QR;
    (
    ************************************************************************
    *                                                                      *
    *       PURPOSE                                                        *
    *   This routine implements the calculation of the Pa values for       *
    *   individual cells. It uses the following formula                    *
    *                                                                      *
    *     Pa = P(X3) = Q0R0 + QiRi + QijRij + ... + QijklmRijklm           *
    *                                                                      *
    *   where the Qi factors are binary valued local coverage elements     *
    *   that indicate, for a particular on air condition of all eight stations,*
    *   whether a cell is 'covered' by the given combination of station    *
    *   subscripts, and the Ri factors are the network reliability factors.*
    *                                                                      *
    *   The Qijklm factors are read as the probability that                *
    *   three or more station signals are accessible given that stations   *
    *   i,j,k,l,m are off air for a given set of stations that cover that  *
    *   cell. For example, if a cell is covered by stations A, B, C, and D,*
    *   QABC = 0 because only D is left covering the cell, whereas QAEFGH = 1*
    *   because the cell is still covered by BCD.                          *
    *                                                                      *
    *   The Rijklm factors are read as the network reliability factor      *
    *   when stations ijklm are off air. These are computed separately     *
    *   for all combinations of ijklm.                                     *
    *                                                                      *
    *   The approach for computing PA for each cell is as follows:         *
    *                                                                      *
    *      1. Given a reliability set for each station/month that          *
    *         indicates the probability of scheduled and unscheduled       *
    *         off air probabilities, calculate and save all possible       *
    *         Rijklm values (there are 219 of them total). These are       *
    *         saved in a 256 element array that is indexed by the          *
    *         combinations of ijklmnop. Some of the entries are null       *
    *         because for fewer than 3 stations on air, the network        *
    *         reliability factor is 0 (there are 256 - 219 = 37 entries    *
    *         equal to 0).                                                 *
    *                                                                      *
    *      2. For each station coverage condition (e.g., say a cell is     *
    *         covered by stations ABC), and for each month,                *
    *         form the sum indicated above, using the Rijklm values from (1*
    *         and evaluating the Qijklm quantities for each iteration      *
    *         of each summation on the fly.                                *
    *                                                                      *
    *      3. Store each PA quantity computed above in a two dimensional   *
    *         data structure indexed by station coverage and month.        *
    *                                                                      *
    *                                                                      *
    *                                                                      *
```

File Name: QRCOMPUT.PAS

```
      ********************************************************************
      }

VAR
   coverage_vector,off_air_vector: byte;

   function signal_accessible: boolean;
   VAR
      i,j,combination: byte;
   begin
      combination := coverage_vector AND (off_air_vector XOR $FF);
      i := 0;
      FOR j := 0 TO 7 DO
      begin
         IF (combination SHR j) AND $01 > 0 THEN
            inc(i);
      end;
      IF i >= 3 THEN
         signal_accessible := TRUE
      ELSE
         signal_accessible := FALSE;
   end;


begin
   { First compute the network reliability factors for this month of
     station reliability numbers. }
   Compute_NRFs;

   { Next form the combinations of the NRFs using the binary values
     for the Qs. Here, each Q is indicated by a vector that contains
     a bit for each station. For example, the Q vector 01110001
     indicates that stations B, C, D and H are off air, denoted
     Q(01110001). Thus, for a particular Q vector and a given coverage
     vector (a vector indicating which stations are covering the cell),
     the value of the Q is determined by comparing the station coverage
     vector and the off air vector. If three or more of the same bits
     in each vector are the same, the value for that Q is 1 and the
     corresponding R term is included in the sum/product. The R term
     is selected using the Q off air vector for each term. }

   FOR coverage_vector := 0 TO 255 DO
   begin
      QR[coverage_vector] := 0;
      FOR off_air_vector := 0 TO 255 DO
      begin
         IF signal_accessible THEN
            QR[coverage_vector] := QR[coverage_vector] + R[off_air_vector];
      end;
```

```
   end;
end;

begin
{ no initialization section }
end.
```

File Name: QRUTIL.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                        Coverage Evaluation                             *
*                              (PACE)                                    *
*                           Workstation                                  *
*                                                                        *
*           UNIT NAME - QRUTIL Unit                                      *
*                                                                        *
*************************************************************************
*                                                                        *
*        This program was prepared by                                    *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                     55 Walkers Brook Drive                             *
*                 Reading, Massachusetts 01867                           *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This unit contains utility routines that are used for        *
*           accessing the QR files and the data in them.                 *
*                                                                        *
*                                                                        *
*************************************************************************}
UNIT QRUTIL;

Interface

CONST
   Scheduled = 1;
   Unscheduled = 2;
   Maintenance = 3;

Type
(********** NOTE: the following type definition for commentareatype MUST be
    consistent with the definition of the comment area in
    the PACEOBJS.PAS. **************)
   commenttype = string;
   QRDatabasePtr = ^QRDatabaseType;
   QRDatabaseType = record
```

File Name: QRUTIL.PAS

```
            comment: commenttype;
            stationreliabilities: ARRAY[1..12,1..8,1..3] OF single;
            Best,Nom: ARRAY[1..12,0..255] OF single;
            end;

   QRDatabaseSubType = ARRAY[1..12,1..8,1..3] OF single;

VAR
   QRPath: string;

   Procedure ReadQRDatabase(QRfile:string;QRPtr:QRDatabasePtr);

   Procedure GetStationReliabilities(VAR stationreliabilities:QRDatabaseSubType;
                                     QRfile:string);

Implementation

   Procedure ReadQRDatabase(QRfile:string;QRPtr:QRDatabasePtr);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           This routine reads the desired QR set and stores it in        *
      *           memory for future access by the Psa processing routines.      *
      ***************************************************************************
      }
   VAR
      F: File;
      errorstring : string;
   begin
      Assign(F,QRPATH+'\'+QRFile);
      Reset(F,sizeof(QRDatabaseType));
      Blockread(F,QRPtr^,1);
      Close(F);
   end;

   Procedure GetStationReliabilities(VAR stationreliabilities:QRDatabaseSubType;
                                     QRfile:string);
      {
      ***************************************************************************
      *                                                                         *
      *        PURPOSE                                                          *
      *           To retrieve the individual station reliabilities values       *
      *           from the QR set.                                              *
      ***************************************************************************
      }
   VAR
      F: File;
      errorstring : string;
```

File Name: QRUTIL.PAS

```
      comment: commenttype;

   begin
      Assign(F,QRPATH+'\'+QRFile);
      Reset(F,1);
      Seek(F,Sizeof(comment));
      Blockread(F,stationreliabilities,sizeof(stationreliabilities));
      Close(F);
   end;

begin
   { no init section }
end.
```

APPENDIX H
GENGDOP LISTINGS

This appendix contains the Pascal code listings for the GDOP database generation program. A discussion of this program is provided in Section 5.5.

File Name: GENGDOP.SRC

The GEN_GDOP program uses the following source program files as well as other PACE units:

GEN_GDOP.PAS

File Name: GEN_GDOP.PAS

```
{*************************************************************************
*                                                                        *
*        PROGRAM NAME - Omega Performance Assessment                     *
*                                and                                     *
*                       Coverage Evaluation                              *
*                            (PACE)                                      *
*                          Workstation                                   *
*                                                                        *
*           UNIT NAME - The GENGDOP Program                              *
*                                                                        *
*************************************************************************
*                                                                        *
*      This program was prepared by                                      *
*                                                                        *
*              The Analytic Science Corporation (TASC)                   *
*                    55 Walkers Brook Drive                              *
*                Reading, Massachusetts 01867                            *
*                                                                        *
*        PACE has been developed to run on a IBM PC/AT or compatible     *
*        under MS-DOS 3.3 or higher with a minimum of 640K of main       *
*        memory and an EGA or compatible graphics adapter and color      *
*        monitor. This work was performed under contract number          *
*        DTCG23-89-C-20008, Task Order 90-0001, Task No. 5834, for       *
*        the Omega Navigation System Center (ONSCEN), Alexandria, VA.    *
*                                                                        *
*************************************************************************
*                                                                        *
*        PURPOSE                                                         *
*           This file contains the program for computing the GDOP        *
*           database. The resulting file is delivered with the PACE      *
*           software as the GDOP database.                               *
*                                                                        *
*************************************************************************}
{$N+,E+}
(
                    Generate GDOP program

   This program generates the GDOP table of values that are used
for computing PSA in PACE. The table is generated and stored by cell
number (1 to 444) and by coverage vector (0 to 255). The total GDOP
file can be interpreted as an ARRAY[1..444,0..255] OF BYTE. Each GDOP
value is stored as a one byte quantity that represents the actual
GDOP number scaled up by 10. Thus, the range of GDOP values in the
table are from 0 to 25.5. GDOP values above 25 are all assigned the
value 25.5 so that the range of values actually runs 0..25 in 0.1
increments, and 25.5. The file name that the GDOP table is stored in
is called GDOP.GDP. The expression used to calculate the GDOP
quantities is as follows:
```

File Name: GEN_GDOP.PAS

```
        sqrt((sum(i=1..q-1) (sum(j=i..q) (square(sin((Bi-Bj)/2)))/
          sum(i=1..q-2) (sum(j=i..q-1) (sum(k=j..q)
                                        (square(sin((Bk-Bj)/2))*
                                         square(sin((Bi-Bk)/2))*
                                         square(sin((Bj-Bi)/2)))))/2

}

program generate_the_GDOP_Table_for_PACE;
USES Cellutil;

CONST
   stations: array[0..7,0..1] of integer =
             ((6642,1313),
              (630,-1066),
              (2140,-15783),
              (4637,-9834),
              (-2097,5529),
              (-4305,-6519),
              (-3848,14694),
              (3461,12945));

   GDOPfile = 'GDOP.GDP';

   FUNCTION ArcCos(x : single) : single;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *            Compute the arc cosine of an angle.                             *
      *                                                                            *
      ******************************************************************************
      }
      BEGIN
         ARCCOS := Pi/2.0-ArcTan(x/Sqrt(1.0-x*x));
      END;

   FUNCTION Get_Angle(Trlat_Int, Trlon_Int, Rlat_Int, Rlon_Int : Integer) : single;
      {
      ******************************************************************************
      *                                                                            *
      *        PURPOSE                                                             *
      *           This function returns the bearing angle from                     *
      *           (Rlat_Int,Rlon_Int) to (Trlat_Int,Trlon_Int).                    *
      *                                                                            *
      *                                                                            *
      ******************************************************************************
      }
```

File Name: GEN_GDOP.PAS

```
      CONST
         RADEGR = 0.0174532925;

      VAR
         Trlat, Trlon, Rlat, Rlon : single;
         LonDiff, Distance, Per, Tm1, Sden, Beta : single;

      FUNCTION Scale(x : single) : single;
      {
      ********************************************************************************
      *                                                                              *
      *        PURPOSE                                                               *
      *           This function simply converts from degrees to radians.            *
      *                                                                              *
      *                                                                              *
      ********************************************************************************
      }
         BEGIN                              { Scale }
            Scale := x*RADEGR
         END;                               { Scale }

{*******************************************************************************}
      BEGIN                                 { Beginning of Get_Angle }

         { Convert latitudes/longitudes to their real values (divided by 10) }
         Trlat := Trlat_Int/100;
         Trlon := Trlon_Int/100;
         Rlat := Rlat_Int/100;
         Rlon := Rlon_Int/100;

         { Compute the bearing angle and return }
         Distance := ArcCos(Sin(Scale(Trlat))*Sin(Scale(rlat))+Cos(Scale(Trlat))*
         Cos(Scale(rlat))*Cos(Scale(Trlon-Rlon)))/RADEGR;
         LonDiff := Rlon-Trlon;
         IF ABS(LonDiff) >= 180
         THEN IF LonDiff < 0
            THEN LonDiff := LonDiff+360
            ELSE LonDiff := LonDiff-360;
         Per := 0.5*(180-Trlat-Rlat+Distance);
         Tm1 := Scale(Per-90);
         Sden := Sin(Scale(Per))*Sin(Tm1+Scale(rlat));
         IF Sden <= 0
         THEN Beta := 180
         ELSE BEGIN
            Beta := 2*ArcTan(Sqrt(ABS(Sin(Tm1+Scale(Trlat))*
            Sin(Scale(Per-Distance)))/Sden))/RADEGR;
            IF LonDiff = 0
            THEN IF Rlat-Trlat < 0
               THEN Beta := 180
```

```
            ELSE Beta := 360-Beta
         ELSE IF LonDiff > 0
            THEN Beta := 360-Beta
      END;                              { IF/THEN/ELSE }
      Get_Angle := Scale(360-Beta)
   END;                                 { Get_Angle }


Function Compute_Gdop(celllon,celllat:integer;coverage:byte): single;
   {
   ***************************************************************************
   *                                                                         *
   *        PURPOSE                                                          *
   *           This function computes GDOP for a given cell latitude and     *
   *           longitude for a given coverage vector.                        *
   ***************************************************************************
   }
VAR
   numbercovering,i,j,k,numberangles : integer;
   exit_flag: boolean;
   angle:array[0..7] of single;
   phi12,phi23,phi34,temp: single;
   s12,s23,s34,s1223,s1234: single;
   tempstring: string;
   result,numerator,denominator: single;
begin
   { count the number of stations covering the cell }
   numbercovering := 0;
   FOR i := 0 TO 7 DO
      IF coverage AND ($01 SHL i) > 0 THEN
         inc(numbercovering);

   CASE numbercovering OF
   0..2: Compute_Gdop := 25.5;
   3..8:
      begin
         { first get the angles for the covered stations to the cell center }
         numberangles := 0;
         FOR i := 0 TO 7 DO
            IF coverage AND ($80 SHR i) > 0 THEN
            begin
(*             angle[numberangles] := get_angle(stations[i,0],stations[i,1],
                                       celllat,celllon);
*)
               angle[numberangles] := get_angle(celllat,celllon,
                                      stations[i,0],stations[i,1]);
               inc(numberangles);
            end;
            { now compute the gdop expression for the angle combinations }
```

```
            { do the numerator first }
            numerator := 0;
            FOR i := 1 TO numberangles - 1 DO
               FOR j := i+1 TO numberangles DO
                  numerator := numerator + sqr(sin((angle[i-1]-angle[j-1])/2));
            { now do the denominator }
            denominator := 0;
            FOR i := 1 TO numberangles - 2 DO
               FOR j := i+1 TO numberangles -1 DO
                  FOR k := j+1 TO numberangles DO
                     denominator := denominator +
                                    sqr(sin((angle[k-1]-angle[j-1])/2)) *
                                    sqr(sin((angle[i-1]-angle[k-1])/2)) *
                                    sqr(sin((angle[j-1]-angle[i-1])/2));
            IF denominator <> 0 THEN
            begin
               { scale gdop results by 10 and limit between 0 and 20 in increments of .1 }
               result := sqrt(numerator/denominator)/2;
               IF result > 25 THEN
                  result := 25.5;
               compute_gdop := result;
            end
            ELSE
               compute_gdop := 25.5;
      end;

   end; { CASE }
end;

var
      tmplst: cell_defn_ptr;
      lat,lon,cellno:integer;
      coverage: byte;
      GDOParray: array[0..255] of byte;
      result: single;
      F:file;

begin
   (********* Main portion of the GENGDOP Program ************)
   { Initialize the cell grid lat/longs }
   InitCellGrid;
     { call the compute GDOP routine for each
       station combination for each cell }

   { open the GDOP file and write the results }
   assign(f,gdopfile);
   rewrite(f,sizeof(gdoparray));
   cellno := 1;
   tmplst := cell_lst;
```

File Name: GEN_GDOP.PAS

```
   WHILE tmplst <> NIL DO
   begin
      lon := (tmplst^.lon2 + tmplst^.lon1) DIV 2;
      lat := (tmplst^.lat2 + tmplst^.lat1) DIV 2;
      FOR coverage := 0 TO 255 DO
      begin
         result := 10 * compute_GDOP(lon,lat,coverage);
         GDOParray[coverage] := round(result);
         IF result > 255 THEN
         begin
            writeln('WARNING, COMPUTED GDOP > 25.5 FOR coverage vector ',coverage);
            readln;
         end
         ELSE
         begin
{           if (result > 10) AND (result < 255) THEN}
              writeln('# ',cellno,' lat ',lat,' lon ',lon,' cov ',coverage,' GDOP ',gdoparr
         end;
      end;
      { write the values out to the GDOP file }
      blockwrite(F,gdoparray,1);
      inc(cellno);
      tmplst := tmplst^.next;
   end;
   { Close the GDOP file }
   close(F);

end.
```