This is the source code for COBRA v4.04, and constitutes the COBRA programming manual. COBRA is written in Borland Turbo Pascal version 6.0, using Borland's TurboVision graphical user interface (see Borland's Turbo Pascal 6.0 Turbo Vision Guide). It is written in several separate modules, each of which addresses part of the execution, I/O, or interface used by COBRA.

Module Name:	Description:
CCAL.PAS Page 3	- Orchestrates the entire Report Execution process, and holds most of the algorithms for Appropriation Report calculations.
CCALCON.PAS Page 20	- Calculates Military Construction costs (including those used in the Appropriation Reports) and creates the Military Construction Assets Report. Also calculates Mission costs (as shown on the Mission Report) and One-Time Costs (as shown on the One-Time Cost Report).
CCALMOV.PAS Page 44	- Calculates all Moving costs (PCS, Freight, etc.) and creates the Moving Report.
CCALOTH.PAS Page 58	- Calculates all costs shown on the Other Costs Report (such as CHAMPUS, PPS, IMA, and HAP).
CCALOVR.PAS Page 71	- Calculates all costs shown on the Overhead Report (such as RPMA/BOS Change, Caretaker Costs, and Mothball Costs). Also creates the RPMA/BOS Report.
CCALPER.PAS Page 99	- Calculates all costs shown on the Personnel Costs Report (such as Salary Saved and Housing Allowance Saved). Also creates the Personnel Movement Report (with Migration Diagrams), the Personnel Realignment Report, and the Personnel Yearly Percentages Report.
CDEC.PAS Page 126	- Defines the constants, record types, and variables used by COBRA.
CFAC1.PAS Page 143	- The TurboVision object for the Personnel Standard Factors Screen.
CFAC2.PAS Page 148	- The TurboVision object for the Facilities Standard Factors Screen.
CFAC3.PAS Page 152	- The TurboVision object for the Transportation Standard Factors Screen.
CFAC4.PAS Page 156	- The TurboVision object for the Construction Standard Factors Screen.
CINIT.PAS Page 159	- Initializes data structures, Data, and Standard Factors. Also contains algorithms for printing hardcopies, deleting bases, creating base-specific and COBRA directory reports, or updating the file in use display on the status bar.
CIO.PAS Page 177	- Contains all disk I/O for loading and saving both Data files and Standard Factors files, as well as loading Old Data files and all DataBase access.

COBRA.PAS Page 224	- The main module which co-ordinates all COBRA activities and TurboVision displays (windows, menues, etc.).
CRPTIN.PAS Page 300	- Creates the Input Data Report.
CRPTOUT1.PAS Page 320	- Creates the Appropriations Detail Report, the Appropriations Detail Fixed Costs Report, and the Appropriations Detail Steady-State Costs Report.
CRPTOUT2.PAS Page 338	- Creates the Delta Chart Report, the Net Present Values Report, the COBRA Summary Report, the Appropriations Summary Report, and the Finance Report.
CSCR1.PAS Page 365	- The TurboVision object for the General Scenario Screen.
CSCR2.PAS Page 370	- The TurboVision object for the Distance Table Screen.
CSCR3.PAS Page 375	- The TurboVision object for the Movement Table Screen.
CSCR4.PAS Page 381	- The TurboVision object for the Static Base Information Screen.
CSCR5.PAS Page 389	- The TurboVision object for the Dynamic Base Information Screen.
CSCR6.PAS Page 397	- The TurboVision object for the Personnel Base Information Screen.
CSCR7.PAS Page 404	- The TurboVision object for the Military Construction Base Information Screen.
CSCR8.PAS Page 413	- The TurboVision object for the Unique Activity Base Information Screen.
CSCR9.PAS Page 420	- The TurboVision object for the Explanatory Notes Screen.
CSUP.PAS Page 422	- Contains functions used throughout COBRA (for inserting commas into numbers, for upper case conversion, centering text, checking if files or directories exist, etc.).
FINPUT.PAS Page 443	- The TurboVision object used by COBRA for input fields.
STRFMT.PAS Page 455	- Contains functions used by FINPUT.PAS.

```
UNIT CCAL;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CRT,
  CSUP,
  CCALCON,
  CCALPER,
  CCALMOV,
  CCALOTH,
  CCALOVR,
  CRPTIN,
  CRPTOUT1,
  CRPTOUT2,
  DRIVERS;

PROCEDURE ExecCalc;

IMPLEMENTATION

{*****************************************************************************
 *   Name           : CalcApproReport
 *
 *   Purpose        :
 *
 *   Notes          :
 *
 *   Invoked As     : CalcApproReport (ApproReport)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   08Apr92   AML   Streamlining (Collapsed 93 loops into 1, etc.)
 *****************************************************************************}
```

```
PROCEDURE CalcApproReport (     RevisedRPMA        : Base0Year1ToMaxRealRec;
                                RevisedBOS         : Base0YearAndBeyondRealRec;
                                SalSav             : SalSavRec;
                                PersMoves          : CalcMovesRec;
                                HousAllowSaved     : Year1ToMaxRealRec;
                                HousAllowSum       : Year1ToMaxRealRec;
                                MissionReport      : CostSaveNetByYearRec;
                                LandCost           : Year1ToMaxRealRec;
                                LandBuys           : Year1ToMaxRealRec;
                                TotActualRPMA      : REAL;
                                TotActualBOS       : Year1ToMaxRealRec;
                                TotRevisedRPMA     : Year1ToMaxRealRec;
                                TotRevisedBOS      : YearAndBeyondRealRec;
                                CalcPersAct        : CalcPersActRec;
                                NewHireCost,
                                PerDiemCiv,
                                POVMilesCiv,
                                HousPurch,
                                HHGCiv,
                                MiscCiv,
                                HousHunt,
                                RITA,
                                PPSCost,
                                Pack,
                                Frat,
                                Vehic,
                                Driv,
                                Loss,
                                UnemploymentCost,
                                CaretakerCost,
                                MaintainSpaceCost : Year1ToMaxRealRec;
                                AdminSuppCost     : Base0Year1ToMaxRealRec;
                                MothballCost      : Year1ToMaxRealRec;
                                Housing           : CostSaveNetByYearRec;
                                CostElimMil       : CostElimMilRec;
                                PerDiemMil,
                                POVMilesMil,
                                HHGMil,
                                MiscMil,
                                IMACost,
                                EnvirCost,
                                HomeownAsstCost   : Year1ToMaxRealRec;
                                Champ             : CostSaveNetByYearRec;
                                CalcFctrsPerPos   : CalcPersPosRec;
                                ApproReportMilCon : ApproReportMilConRec;
                                OneTimeMove       : Year1ToMaxRealRec;
                                DeltaChart        : DeltaChartRec;
                            Var ApproReport       : ApproReportRec);
```

```
VAR
  B, B2, Y, Y2 : INTEGER;
  BasePop,
  Savsav       : Real;

Begin { CalcApproReport }
  FillChar (ApproReport^, SizeOf (ApproReport^), 0);

  { Perform calculations }

  For B := 0 to MaxBases Do
    If CostAvoid.Base[B].Procure.Year[MaxYears] > 0
      Then ApproReport^.Savings.Procure.Beyond :=
           ApproReport^.Savings.Procure.Beyond
         + CostAvoid.Base[B].Procure.Year[MaxYears]
      Else ApproReport^.Costs.Procure.Beyond :=
           ApproReport^.Costs.Procure.Beyond
         - CostAvoid.Base[B].Procure.Year[MaxYears];
  For Y := 1 To MaxYears Do
    Begin
      For B := 0 To MaxBases Do
        Begin
          ApproReport^.Costs.OM.Other.AdminPlan.Year[Y] :=
              ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
            + AdminSuppCost.Base[B].Year[Y];
          If BudgetPropData^.Base[B].MiscRecurCost.Year[Y] < 0.0
            Then ApproReport^.Savings.MiscRecur.Year[Y] :=
                 ApproReport^.Savings.MiscRecur.Year[Y]
               - BudgetPropData^.Base[B].MiscRecurCost.Year[Y]
            Else ApproReport^.Costs.MiscRecur.Year[Y] :=
                 ApproReport^.Costs.MiscRecur.Year[Y]
               + BudgetPropData^.Base[B].MiscRecurCost.Year[Y];
          ApproReport^.Savings.FamHous.Construction.Year[Y] :=
              ApproReport^.Savings.FamHous.Construction.Year[Y]
            + CostAvoid.Base[B].FamQtr.Year[Y];
          ApproReport^.Savings.MilCon.Year[Y] :=
              ApproReport^.Savings.MilCon.Year[Y]
            + CostAvoid.Base[B].Constr.Year[Y];
          If CostAvoid.Base[B].Procure.Year[Y] > 0
            Then ApproReport^.Savings.Procure.Year[Y] :=
                 ApproReport^.Savings.Procure.Year[Y]
               + CostAvoid.Base[B].Procure.Year[Y]
            Else ApproReport^.Costs.Procure.Year[Y] :=
                 ApproReport^.Costs.Procure.Year[Y]
               - CostAvoid.Base[B].Procure.Year[Y];
          ApproReport^.Costs.EnvirMitig.Year[Y] :=
              ApproReport^.Costs.EnvirMitig.Year[Y]
            + ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y];
```

```
If IndustFundSwitch.Base[B] = 'Y' Then
  Begin
    ApproReport^.Savings.OM.OperatingIDF.Year[Y] :=
        ApproReport^.Savings.OM.OperatingIDF.Year[Y]
      + IndustFund.Base[B]^.OvhdSave.Operating.Year[Y];
    ApproReport^.Savings.OM.OtherIDF.Year[Y] :=
        ApproReport^.Savings.OM.OtherIDF.Year[Y]
      + IndustFund.Base[B]^.OvhdSave.Other.Year[Y];
    ApproReport^.Costs.OM.OperatingIDF.Year[Y] :=
        ApproReport^.Costs.OM.OperatingIDF.Year[Y]
      + IndustFund.Base[B]^.OvhdCost.Operating.Year[Y];
    ApproReport^.Costs.OM.Other.OtherIDF.Year[Y] :=
        ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
      + IndustFund.Base[B]^.OvhdCost.Other.Year[Y];
  End
Else
  Begin
    BasePop :=
        ActionEnvir^.Base[B].BaseTotOffEmploy
      + ActionEnvir^.Base[B].BaseTotEnlEmploy
      + ActionEnvir^.Base[B].BaseTotStuEmploy
      + ActionEnvir^.Base[B].BaseTotCivEmploy;
    For Y2 := 1 To Y Do
      BasePop := BasePop
        - PosReduc.Base[B].Off.Year[Y2]
        - PosReduc.Base[B].Enl.Year[Y2]
        - PosReduc.Base[B].Civ.Year[Y2];
    If (DeltaChart^.RPMABOS.Base[B].PercChange < 0) Then
      Begin
        ApproReport^.Savings.OM.BOS.Year[Y] :=
            ApproReport^.Savings.OM.BOS.Year[Y]
          +(BudgetPropData^.Base[B].CommunBudget
          + BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
                                              * ZeroCheck
           (BasePop,
            ActionEnvir^.Base[B].BaseTotOffEmploy
          + ActionEnvir^.Base[B].BaseTotEnlEmploy
          + ActionEnvir^.Base[B].BaseTotStuEmploy
          + ActionEnvir^.Base[B].BaseTotCivEmploy)
          - RevisedBOS.Base[B].Year[Y];
        ApproReport^.Savings.OM.RPMA.Year[Y] :=
            ApproReport^.Savings.OM.RPMA.Year[Y]
          + BudgetPropData^.Base[B].RPMANonPayBudget
          - RevisedRPMA.Base[B].Year[Y];
      End
```

```
          Else
            Begin
              ApproReport^.Costs.OM.BOS.Year[Y] :=
                  ApproReport^.Costs.OM.BOS.Year[Y]
                -(BudgetPropData^.Base[B].CommunBudget
                + BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
                * ZeroCheck
                 (BasePop,
                  ActionEnvir^.Base[B].BaseTotOffEmploy
                + ActionEnvir^.Base[B].BaseTotEnlEmploy
                + ActionEnvir^.Base[B].BaseTotStuEmploy
                + ActionEnvir^.Base[B].BaseTotCivEmploy)
                + RevisedBOS.Base[B].Year[Y];
              ApproReport^.Costs.OM.RPMA.Year[Y] :=
                  ApproReport^.Costs.OM.RPMA.Year[Y]
                - BudgetPropData^.Base[B].RPMANonPayBudget
                + RevisedRPMA.Base[B].Year[Y];
            End;
        End;
      For B2 := 0 To MaxBases Do
        If Distance^.Base[B].Base[B2] >= 50.0
          Then ApproReport^.Savings.MilPers.MilMoving.Year[Y] :=
               ApproReport^.Savings.MilPers.MilMoving.Year[Y]
             +(PosTrans.Base[B].Base[B2]^.Off.Year[Y]
             + PosTrans.Base[B].Base[B2]^.Enl.Year[Y])
             * StdFctrsTrans.RoutinePCSCostsPerPerson
             / StdFctrsTrans.AverageTourLength;
  End;
ApproReport^.Savings.OM.CivSalary.Year[Y] := SalSav.Civ.Year[Y];
ApproReport^.Savings.OM.Champus.Year[Y] := Champ.Save.Year[Y];
ApproReport^.Costs.OM.Champus.Year[Y] := Champ.Cost.Year[Y];
If HousAllowSaved.Year[Y] < 0.0
  Then ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y] :=
      ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y]
    - HousAllowSaved.Year[Y]
  Else ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y] :=
      ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
    + HousAllowSaved.Year[Y];
If HousAllowSum.Year[Y] < 0.0
  Then ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y] :=
      ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y]
    - HousAllowSum.Year[Y]
  Else ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y] :=
      ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
    + HousAllowSum.Year[Y];
ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
  := SalSav.Off.Year[Y];
ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
  := SalSav.Enl.Year[Y];
ApproReport^.Savings.Other.Year[Y] := MissionReport.Save.Year[Y];
ApproReport^.Savings.LandRevenue.Year[Y]
  := LandBuys.Year[Y] - LandCost.Year[Y];
```

```
ApproReport^.Savings.FamHous.Operations.Year[Y] :=
  Housing.Save.Year[Y];
ApproReport^.Savings.Total.Year[Y] :=
    ApproReport^.Savings.MilCon.Year[Y]
  + ApproReport^.Savings.FamHous.Construction.Year[Y]
  + ApproReport^.Savings.FamHous.Operations.Year[Y]
  + ApproReport^.Savings.OM.RPMA.Year[Y]
  + ApproReport^.Savings.OM.BOS.Year[Y]
  + ApproReport^.Savings.OM.OperatingIDF.Year[Y]
  + ApproReport^.Savings.OM.CivSalary.Year[Y]
  + ApproReport^.Savings.OM.Champus.Year[Y]
  + ApproReport^.Savings.OM.OtherIDF.Year[Y]
  + ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
  + ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
  + ApproReport^.Savings.MilPers.MilMoving.Year[Y]
  + ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y]
  + ApproReport^.Savings.Procure.Year[Y]
  + ApproReport^.Savings.Other.Year[Y]
  + ApproReport^.Savings.LandRevenue.Year[Y]
  + ApproReport^.Savings.MiscRecur.Year[Y];
ApproReport^.Costs.MilCon.Year[Y] :=
  ApproReportMilCon.MilCon.Year[Y];
ApproReport^.Costs.FamHous.Construction.Year[Y] :=
  ApproReportMilCon.Hous.Year[Y];
ApproReport^.Costs.FamHous.Operations.Year[Y] :=
  Housing.Cost.Year[Y];
ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y] :=
  CalcPersAct.Year[Y].RIFPay;
ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y] :=
  CalcPersAct.Year[Y].EarlyRetire;
ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y] :=
  PerDiemCiv.Year[Y];
ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y] :=
  POVMilesCiv.Year[Y];
ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y] :=
  HousPurch.Year[Y];
ApproReport^.Costs.OM.CivMoving.HHG.Year[Y] :=
  HHGCiv.Year[Y];
ApproReport^.Costs.OM.CivMoving.Misc.Year[Y] :=
  MiscCiv.Year[Y];
ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y] :=
  HousHunt.Year[Y];
ApproReport^.Costs.OM.CivMoving.PPS.Year[Y] := PPSCost.Year[Y];
ApproReport^.Costs.OM.CivMoving.RITA.Year[Y] := RITA.Year[Y];
ApproReport^.Costs.OM.Freight.Pack.Year[Y] := Pack.Year[Y];
ApproReport^.Costs.OM.Freight.Freight.Year[Y] := Frat.Year[Y];
ApproReport^.Costs.OM.Freight.Vehicles.Year[Y] := Vehic.Year[Y];
ApproReport^.Costs.OM.Freight.Driving.Year[Y] := Driv.Year[Y];
ApproReport^.Costs.OM.Freight.LossRate.Year[Y] := Loss.Year[Y];
ApproReport^.Costs.OM.Unemployment.Year[Y] :=
  UnemploymentCost.Year[Y];
```

```
ApproReport^.Costs.OM.Other.Caretaker.Year[Y] :=
  CaretakerCost.Year[Y];
ApproReport^.Costs.OM.Other.ShutDown.Year[Y] :=
  MothBallCost.Year[Y];
ApproReport^.Costs.OM.Other.Maintain.Year[Y] :=
  MaintainSpaceCost.Year[Y];
ApproReport^.Costs.OM.Other.NewHire.Year[Y] :=
  NewHireCost.Year[Y];
ApproReport^.Costs.OM.Other.OneTimeMove.Year[Y] :=
  OneTimeMove.Year[Y];
ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y] :=
  Abs (CostElimMil.Tot.Year[Y]);
ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y] :=
  PerDiemMil.Year[Y];
ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y] :=
  POVMilesMil.Year[Y];
ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y] := HHGMil.Year[Y];
ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y] := MiscMil.Year[Y];
ApproReport^.Costs.HomeownAsst.Year[Y] := HomeownAsstCost.Year[Y];
ApproReport^.Costs.InfoMgt.Year[Y] := IMACost.Year[Y];
ApproReport^.Costs.Other.Mission.Year[Y] :=
  MissionReport.Cost.Year[Y];
ApproReport^.Costs.Other.OneTime.Year[Y] := EnvirCost.Year[Y]
  - ApproReport^.Costs.EnvirMitig.Year[Y];
ApproReport^.Costs.LandPurch.Year[Y] := LandBuys.Year[Y];
ApproReport^.Costs.Total.Year[Y] :=
    ApproReport^.Costs.MilCon.Year[Y]
  + ApproReport^.Costs.Procure.Year[Y]
  + ApproReport^.Costs.FamHous.Construction.Year[Y]
  + ApproReport^.Costs.FamHous.Operations.Year[Y]
  + ApproReport^.Costs.OM.RPMA.Year[Y]
  + ApproReport^.Costs.OM.BOS.Year[Y]
  + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
  + ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
  + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
  + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
  + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
  + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
  + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
  + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
  + ApproReport^.Costs.OM.Champus.Year[Y]
  + ApproReport^.Costs.OM.Unemployment.Year[Y]
  + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
  + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
```

```
  + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
  + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
  + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
  + ApproReport^.Costs.OM.Other.OneTimeMove.Year[Y]
  + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
  + ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
  + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
  + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
  + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
  + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
  + ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
  + ApproReport^.Costs.HomeownAsst.Year[Y]
  + ApproReport^.Costs.EnvirMitig.Year[Y]
  + ApproReport^.Costs.InfoMgt.Year[Y]
  + ApproReport^.Costs.Other.Mission.Year[Y]
  + ApproReport^.Costs.Other.OneTime.Year[Y]
  + ApproReport^.Costs.LandPurch.Year[Y]
  + ApproReport^.Costs.MiscRecur.Year[Y];
ApproReport^.Net.MilCon.Year[Y] :=
    ApproReport^.Costs.MilCon.Year[Y]
  - ApproReport^.Savings.MilCon.Year[Y];
ApproReport^.Net.FamHous.Construction.Year[Y] :=
    ApproReport^.Costs.FamHous.Construction.Year[Y]
  - ApproReport^.Savings.FamHous.Construction.Year[Y];
ApproReport^.Net.FamHous.Operations.Year[Y] :=
    ApproReport^.Costs.FamHous.Operations.Year[Y]
  - ApproReport^.Savings.FamHous.Operations.Year[Y];
ApproReport^.Net.OM.RPMA.Year[Y] := ApproReport^.Costs.OM.RPMA.Year[Y]
  - ApproReport^.Savings.OM.RPMA.Year[Y];
ApproReport^.Net.OM.BOS.Year[Y] := ApproReport^.Costs.OM.BOS.Year[Y]
  - ApproReport^.Savings.OM.BOS.Year[Y];
ApproReport^.Net.OM.OperatingIDF.Year[Y] :=
    ApproReport^.Costs.OM.OperatingIDF.Year[Y]
  - ApproReport^.Savings.OM.OperatingIDF.Year[Y];
ApproReport^.Net.Om.CivSalary.Year[Y] :=
    ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
  + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
  - ApproReport^.Savings.OM.CivSalary.Year[Y];
ApproReport^.Net.OM.CivMoving.Year[Y] :=
  (ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
  + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]);
ApproReport^.Net.OM.Other.Year[Y] :=
    ApproReport^.Costs.OM.Freight.Pack.Year[Y]
  + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
  + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
  + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
```

```
        + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
        + ApproReport^.Costs.OM.Unemployment.Year[Y]
        + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
        + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
        + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
        + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
        + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
        + ApproReport^.Costs.OM.Other.OneTimeMove.Year[Y]
        + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
        - ApproReport^.Savings.OM.OtherIDF.Year[Y];
      ApproReport^.Net.MilPers.MilSalary.Year[Y] :=
        - ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
        - ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y];
      ApproReport^.Net.MilPers.MilMoving.Year[Y] :=
         (ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
        + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
        + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
        + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
        + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y])
        - ApproReport^.Savings.MilPers.MilMoving.Year[Y];
      ApproReport^.Net.MilPers.Other.Year[Y] :=
          ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
        - ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y];
      ApproReport^.Net.Procure.Year[Y] :=
          ApproReport^.Costs.Procure.Year[Y]
        - ApproReport^.Savings.Procure.Year[Y];
      ApproReport^.Net.HomeownAsst.Year[Y] :=
          ApproReport^.Costs.HomeownAsst.Year[Y];
      ApproReport^.Net.EnvirMitig.Year[Y] :=
          ApproReport^.Costs.EnvirMitig.Year[Y];
      ApproReport^.Net.InfoMgt.Year[Y] :=
          ApproReport^.Costs.InfoMgt.Year[Y];
      ApproReport^.Net.Other.Year[Y] :=
         (ApproReport^.Costs.Other.Mission.Year[Y]
        + ApproReport^.Costs.Other.OneTime.Year[Y])
        - ApproReport^.Savings.Other.Year[Y];
      ApproReport^.Net.Land.Year[Y] :=
          ApproReport^.Costs.LandPurch.Year[Y]
        - ApproReport^.Savings.LandRevenue.Year[Y];
      ApproReport^.Net.Champus.Year[Y] :=
          ApproReport^.Costs.OM.Champus.Year[Y]
        - ApproReport^.Savings.OM.Champus.Year[Y];
      ApproReport^.Net.MiscRecur.Year[Y] :=
          ApproReport^.Costs.MiscRecur.Year[Y]
        - ApproReport^.Savings.MiscRecur.Year[Y];
      ApproReport^.Net.Total.Year[Y] :=
        ApproReport^.Costs.Total.Year[Y] - ApproReport^.Savings.Total.Year[Y];
    End; {End of FOR loop}
  ApproReport^.Savings.Other.Beyond :=
    MissionReport.Save.Beyond;
  ApproReport^.Costs.Other.Mission.Beyond :=
    MissionReport.Cost.Beyond;
```

```
ApproReport^.Savings.OM.CivSalary.Beyond
  := ApproReport^.Savings.OM.CivSalary.Year[MaxYears];
ApproReport^.Savings.OM.Champus.Beyond
  := ApproReport^.Savings.OM.Champus.Year[MaxYears];
ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
  := ApproReport^.Savings.MilPers.MilSalary.Off.Year[MaxYears];
ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond
  := ApproReport^.Savings.MilPers.MilSalary.Enl.Year[MaxYears];
ApproReport^.Costs.OM.OperatingIDF.Beyond
  := ApproReport^.Costs.OM.OperatingIDF.Year[MaxYears];
ApproReport^.Savings.OM.OperatingIDF.Beyond
  := ApproReport^.Savings.OM.OperatingIDF.Year[MaxYears];
ApproReport^.Savings.OM.OtherIDF.Beyond
  := ApproReport^.Savings.OM.OtherIDF.Year[MaxYears];
ApproReport^.Savings.MilPers.Other.HousAllow.Beyond
  := ApproReport^.Savings.MilPers.Other.HousAllow.Year[MaxYears];
ApproReport^.Savings.FamHous.Operations.Beyond :=
  ApproReport^.Savings.FamHous.Operations.Year[MaxYears];
ApproReport^.Savings.OM.RPMA.Beyond :=
  ApproReport^.Savings.OM.RPMA.Year[MaxYears];
For B := 0 To MaxBases Do
  If IndustFundSwitch.Base[B] <> 'Y' Then
    Begin
      BasePop :=
                                ActionEnvir^.Base[B].BaseTotOffEmploy
                            + ActionEnvir^.Base[B].BaseTotEnlEmploy
                            + ActionEnvir^.Base[B].BaseTotStuEmploy
                            + ActionEnvir^.Base[B].BaseTotCivEmploy;
                        For Y := 1 To MaxYears Do
        BasePop := BasePop
                                - PosReduc.Base[B].Off.Year[Y]
                                - PosReduc.Base[B].Enl.Year[Y]
                                - PosReduc.Base[B].Civ.Year[Y];
      If (DeltaChart^.RPMABOS.Base[B].PercChange < 0)
        Then ApproReport^.Savings.OM.BOS.Beyond :=
             ApproReport^.Savings.OM.BOS.Beyond
                                +(BudgetPropData^.Base[B].CommunBudget
                                                                        +
BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
                                * ZeroCheck
             (BasePop,
                                     ActionEnvir^.Base[B].BaseTotOffEmploy
                                + ActionEnvir^.Base[B].BaseTotEnlEmploy
                                + ActionEnvir^.Base[B].BaseTotStuEmploy
                                + ActionEnvir^.Base[B].BaseTotCivEmploy)
           - RevisedBOS.Base[B].Beyond
        Else ApproReport^.Costs.OM.BOS.Beyond :=
                                     ApproReport^.Costs.OM.BOS.Beyond
                                -(BudgetPropData^.Base[B].CommunBudget
                                +
```

```
BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
                                        * ZeroCheck
                (BasePop,
                                            ActionEnvir^.Base[B].BaseTotOffEmploy
                                        + ActionEnvir^.Base[B].BaseTotEnlEmploy
                                        + ActionEnvir^.Base[B].BaseTotStuEmploy
                                        + ActionEnvir^.Base[B].BaseTotCivEmploy)
              + RevisedBOS.Base[B].Beyond;
        End;
    ApproReport^.Savings.MiscRecur.Beyond :=
      ApproReport^.Savings.MiscRecur.Year[MaxYears];
    ApproReport^.Savings.Total.Beyond :=
        ApproReport^.Savings.FamHous.Operations.Beyond
      + ApproReport^.Savings.OM.RPMA.Beyond
      + ApproReport^.Savings.OM.BOS.Beyond
      + ApproReport^.Savings.OM.OperatingIDF.Beyond
      + ApproReport^.Savings.OM.CivSalary.Beyond
      + ApproReport^.Savings.OM.Champus.Beyond
      + ApproReport^.Savings.OM.OtherIDF.Beyond
      + ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
      + ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond
      + ApproReport^.Savings.MilPers.Other.HousAllow.Beyond
      + ApproReport^.Savings.Procure.Beyond
      + ApproReport^.Savings.Other.Beyond
      + ApproReport^.Savings.MiscRecur.Beyond;
    ApproReport^.Costs.FamHous.Operations.Beyond :=
      Housing.Cost.Year[MaxYears];
    ApproReport^.Costs.OM.Other.Caretaker.Beyond :=
      ApproReport^.Costs.OM.Other.Caretaker.Year[MaxYears];
    ApproReport^.Costs.OM.Other.Maintain.Beyond :=
      ApproReport^.Costs.OM.Other.Maintain.Year[MaxYears];
    ApproReport^.Costs.OM.Other.OtherIDF.Beyond :=
      ApproReport^.Costs.OM.Other.OtherIDF.Year[MaxYears];
    ApproReport^.Costs.MilPers.Other.HousAllow.Beyond
      := ApproReport^.Costs.MilPers.Other.HousAllow.Year[MaxYears];
    ApproReport^.Costs.OM.Champus.Beyond
      := ApproReport^.Costs.OM.Champus.Year[MaxYears];
    ApproReport^.Costs.MiscRecur.Beyond :=
      ApproReport^.Costs.MiscRecur.Year[MaxYears];
    ApproReport^.Costs.OM.RPMA.Beyond :=
      ApproReport^.Costs.OM.RPMA.Year[MaxYears];
    ApproReport^.Costs.Total.Beyond :=
        ApproReport^.Costs.FamHous.Operations.Beyond
      + ApproReport^.Costs.OM.RPMA.Beyond
      + ApproReport^.Costs.OM.BOS.Beyond
      + ApproReport^.Costs.OM.OperatingIDF.Beyond
      + ApproReport^.Costs.OM.Champus.Beyond
      + ApproReport^.Costs.OM.Other.Caretaker.Beyond
      + ApproReport^.Costs.OM.Other.Maintain.Beyond
      + ApproReport^.Costs.OM.Other.OtherIDF.Beyond
      + ApproReport^.Costs.MilPers.Other.HousAllow.Beyond
      + ApproReport^.Costs.Other.Mission.Beyond
```

```
      + ApproReport^.Costs.MiscRecur.Beyond
      + ApproReport^.Costs.Procure.Beyond;
    ApproReport^.Net.MilCon.Beyond :=
      ApproReport^.Costs.MilCon.Beyond - ApproReport^.Savings.MilCon.Beyond;
    ApproReport^.Net.FamHous.Operations.Beyond :=
        ApproReport^.Costs.FamHous.Operations.Beyond
      - ApproReport^.Savings.FamHous.Operations.Beyond;
    ApproReport^.Net.OM.RPMA.Beyond := ApproReport^.Costs.OM.RPMA.Beyond
      - ApproReport^.Savings.OM.RPMA.Beyond;
    ApproReport^.Net.OM.BOS.Beyond := ApproReport^.Costs.OM.BOS.Beyond
      - ApproReport^.Savings.OM.BOS.Beyond;
    ApproReport^.Net.OM.OperatingIDF.Beyond :=
        ApproReport^.Costs.OM.OperatingIDF.Beyond
      - ApproReport^.Savings.OM.OperatingIDF.Beyond;
    ApproReport^.Net.Om.CivSalary.Beyond :=
      - ApproReport^.Savings.OM.CivSalary.Beyond;
    ApproReport^.Net.OM.Other.Beyond :=
        ApproReport^.Costs.OM.Other.Caretaker.Beyond
      + ApproReport^.Costs.OM.Other.Maintain.Beyond
      + ApproReport^.Costs.OM.Other.OtherIDF.Beyond
      - ApproReport^.Savings.OM.OtherIDF.Beyond;
    ApproReport^.Net.MilPers.MilSalary.Beyond :=
      - ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
      - ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond;
    ApproReport^.Net.MilPers.Other.Beyond :=
        ApproReport^.Costs.MilPers.Other.HousAllow.Beyond
      - ApproReport^.Savings.MilPers.Other.HousAllow.Beyond;
    ApproReport^.Net.Procure.Beyond :=
        ApproReport^.Costs.Procure.Beyond
      - ApproReport^.Savings.Procure.Beyond;
    ApproReport^.Net.Other.Beyond :=
        ApproReport^.Costs.Other.Mission.Beyond
      - ApproReport^.Savings.Other.Beyond;
    ApproReport^.Net.Champus.Beyond :=
        ApproReport^.Costs.OM.Champus.Beyond
      - ApproReport^.Savings.OM.Champus.Beyond;
    ApproReport^.Net.MiscRecur.Beyond :=
        ApproReport^.Costs.MiscRecur.Beyond
      - ApproReport^.Savings.MiscRecur.Beyond;
    ApproReport^.Net.Total.Beyond :=
      ApproReport^.Costs.Total.Beyond - ApproReport^.Savings.Total.Beyond;
  End;  { CalcApproReport }
```

```
{*************************************************************************
 *   Name            : Calc
 *
 *   Purpose         : Orchestrates all COBRA calculations
 *
 *   Notes           : Adjust.Year[] and Adjust1000.Year[] were used when the
 *                     user wished to inflate the reports.  This option was
 *                     disconnected (Adjust = 1, Adjust1000 = 0.001), but the
 *                     variables remain should they ever be wanted again.
 *
 *   Invoked As      : Calc
 *
 *   Return value    :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   29May92   AML   Added Adjust and Adjust1000 variables
 *************************************************************************}

PROCEDURE ExecCalc;

  Procedure Progress (Var Value : Integer);
    Begin
      GotoXY (32, 13);
      Write (Blank:Value);
      Inc (Value);
    End;

  VAR
    ApproReport        : ApproReportRec;
    PersMoves          : CalcMovesRec;
    CalcPersAct        : CalcPersActRec;
    CalcFctrsPerPos    : CalcPersPosRec;
    CostElimMil        : CostElimMilRec;
    DeltaChart         : DeltaChartRec;
    MilConFinal        : MilConFinalRec;
    MilConDetail       : MilConRec;
    MovCostAlt         : PBase0ToMaxRealRec;
    Adjust,
    Adjust1000,
    CaretakerCost,
    Driv,
    Frat,
```

```
HHGCiv,
HHGMil,
HousAllowSaved,
HousAllowSum,
HousHunt,
HousPurch,
Loss,
MaintainSpaceCost,
MilMov,
MiscCiv,
MiscMil,
MothballCost,
OneTimeMove,
Pack,
PerDiemCiv,
PerDiemMil,
POVMilesCiv,
POVMilesMil,
RITA,
TotRevisedRPMA,
Vehic,
NewHireCost,
LandCost,
LandBuys,
EnvirCost,
HomeownAsstCost,
UnemploymentCost,
IMACost,
PPSCost            : Year1ToMaxRealRec;
TotRevisedBOS,
MovCostFinal       : YearandBeyondRealRec;
Champ,
Housing,
MissionReport,
OtherCostFinal,
OvhdCostFinal,
PersFinal          : CostSaveNetByYearRec;
PersMovesSum       : PersMovesSumRec;
TotActualRPMA      : Real;
TotActualBOS       : Year1ToMaxRealRec;
TotalOneTimeCost   : REAL;
AdminSuppCost,
RevisedRPMA        : Base0Year1ToMaxRealRec;
RevisedBOS         : Base0YearAndBeyondRealRec;
SalSav             : SalSavRec;
ApproReportMilCon  : ApproReportMilConRec;
Indicator, Y       : Integer;
F                  : Text;
```

```
Begin { ExecCalc }
  For Y := 1 To MaxYears Do
    Begin
      Adjust.Year[Y] := 1;
      Adjust1000.Year[Y] := 0.001;
    End;
  New(ApproReport);
  New(DeltaChart);
  New(MovCostAlt);
  If FileExists (Info.ReportDir + 'Error.Rpt') Then
    Begin
      Assign (F, Info.ReportDir + 'Error.Rpt');
      Erase (F);
    End;

{ Draw Progress indicator -- Will be covered by COBSUM.RPT when execution
  finishes, so it won't mess up TurboVision displays }

  HideMouse;
  If ScreenMode = smCO80
    Then TextColor (15)
    Else TextColor (0);
  TextBackground (7);
  GotoXY (30,11);
  Write ('┌', Rept ('─',20), '┐');
  GotoXY (30, 12);
  Write ('│', '│':21);
  GotoXY (30, 13);
  If ScreenMode = smCO80
    Then Write ('│ ', Rept ('▒',18), ' │')
    Else Write ('│ ', Rept (' ',18), ' │');
  GotoXY (30, 14);
  Write ('│', '│':21);
  GotoXY (30,15);
  Write ('└', Rept ('─',20), '┘');
  TextColor (0);
  GotoXY (33,12);
  Write ('Report Executing');
  GotoXY (32,14);
  Write ('Progress Indicator');
  If ScreenMode = smCO80
    Then TextBackground (4)
    Else TextBackground (0);
  Indicator := 2;

  EnteredDataReport;
  Progress (Indicator);
  CalcPersFactors              (CalcFctrsPerPos, CalcPersAct, PersMoves,
                                PersMovesSum);
  Progress (Indicator);
  CalcMissionCost              (PersMovesSum, MissionReport, Adjust1000);
```

```
Progress (Indicator);
CalcConstructCost          (PersMovesSum, MilConDetail, MilConFinal,
                            Adjust);

Progress (Indicator);
CalcPersonnelCost          (CalcFctrsPerPos, PersMoves, PersFinal,
                            CostElimMil, SalSav, HousAllowSaved,
                            HousAllowSum, Adjust);

Progress (Indicator);
CalcMovingCost             (PersMoves, MovCostFinal, MovCostAlt,
                            PerDiemCiv, POVMilesCiv, HousPurch, HHGCiv,
                            MiscCiv, HousHunt, RITA, Pack, Frat, Vehic,
                            Driv, Loss, PerDiemMil, POVMilesMil, HHGMil,
                            MiscMil, MilMov, OneTimeMove, CalcPersAct,
                            Adjust1000);

Progress (Indicator);
CalcOtherCost              (PersMovesSum, MilConDetail, CalcPersAct,
                            PersMoves, CalcFctrsPerPos, OtherCostFinal,
                            NewHireCost, LandCost, LandBuys, EnvirCost,
                            HomeownAsstCost, UnemploymentCost, IMACost,
                            PPSCost, Champ, Adjust1000);

Progress (Indicator);
CalcOverheadCost           (PersMovesSum, MilConDetail, OvhdCostFinal,
                            AdminSuppCost, MothballCost, DeltaChart,
                            RevisedRPMA, RevisedBOS, TotActualRPMA,
                            TotActualBOS, TotRevisedRPMA, TotRevisedBOS,
                            CaretakerCost, MaintainSpaceCost, Housing,
                            Adjust);

Progress (Indicator);
CalcTot1TimeCost           (MilConDetail, MovCostAlt, NewHireCost,
                            PersMoves, PersMovesSum, CalcPersAct,
                            MilConFinal, MovCostFinal, IMACost,
                            AdminSuppCost, TotalOneTimeCost, Adjust);

Progress (Indicator);
SummaryReport              (MissionReport, PersFinal, MilConFinal,
                            MovCostFinal, OtherCostFinal, OvhdCostFinal,
                            CalcFctrsPerPos, TotalOneTimeCost, PersMoves,
                            MilMov, Adjust1000);

Progress (Indicator);
DeltaChartReport           (DeltaChart);
CalcApproReportMilCon      (MilConDetail, MilConFinal, PersMovesSum,
                            ApproReportMilCon);

CalcApproReport            (RevisedRPMA, RevisedBOS, SalSav, PersMoves,
                            HousAllowSaved, HousAllowSum, MissionReport,
                            LandCost, LandBuys, TotActualRPMA,
                            TotActualBOS, TotRevisedRPMA, TotRevisedBOS,
                            CalcPersAct, NewHireCost, PerDiemCiv,
                            POVMilesCiv, HousPurch, HHGCiv, MiscCiv,
                            HousHunt, RITA, PPSCost, Pack, Frat, Vehic,
                            Driv, Loss, UnemploymentCost, CaretakerCost,
                            MaintainSpaceCost, AdminSuppCost, MothballCost,
                            Housing, CostElimMil, PerDiemMil, POVMilesMil,
                            HHGMil, MiscMil, IMACost, EnvirCost,
```

```
                                    HomeownAsstCost, Champ, CalcFctrsPerPos,
                                    ApproReportMilCon, OneTimeMove, DeltaChart,
                                    ApproReport);
    Progress (Indicator);
    ApproDetailReport            (ApproReport, Adjust1000);
    Inc (Indicator);
    Progress (Indicator);
    ApproDetailFixedReport       (ApproReport, Adjust1000);
    Inc (Indicator);
    Progress (Indicator);
    ApproDetailSteadyStateReport(ApproReport, Adjust1000);
    Progress (Indicator);
    ApproSummaryReport           (ApproReport, Adjust1000);
    Progress (Indicator);
    ApproFinanceReport           (ApproReport, CalcFctrsPerPos);
    ShowMouse;
    Dispose(ApproReport);
    Dispose(DeltaChart);
    Dispose(MovCostAlt);
  END;  { ExecCalc }

END. { UNIT }
```

```
UNIT CCALCON;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE CalcApproReportMilCon (     MilConDetail       : MilConRec;
                                      MilConFinal        : MilConFinalRec;
                                      PersMovesSum       : PersMovesSumRec;
                                  VAR ApproReportMilCon  : ApproReportMilConRec);

PROCEDURE CalcConstructCost      (    PersMovesSum       : PersMovesSumRec;
                                  VAR MilConDetail       : MilConRec;
                                  VAR MilConFinal        : MilConFinalRec;
                                      Adjust             : Year1ToMaxRealRec);

PROCEDURE CalcMissionCost        (    PersMovesSum       : PersMovesSumRec;
                                  VAR MissionReport      : CostSaveNetByYearRec;
                                      Adjust1000         : Year1ToMaxRealRec);

PROCEDURE CalcTot1TimeCost       (    MilConDetail      : MilConRec;
                                      MovCostAlt        : PBase0ToMaxRealRec;
                                      NewHireCost       : Year1ToMaxRealRec;
                                      PersMoves         : CalcMovesRec;
                                      PersMovesSum      : PersMovesSumRec;
                                      CalcPersAct       : CalcPersActRec;
                                      MilConFinal       : MilConFinalRec;
                                      MovCostFinal      : YearAndBeyondRealRec;
                                      IMACost           : Year1ToMaxRealRec;
                                      AdminSuppCost     : Base0Year1ToMaxRealRec;
                                  VAR TotalOneTimeCost  : Real;
                                      Adjust            : Year1ToMaxRealRec);

IMPLEMENTATION
```

```
{*************************************************************************
 *   Name           : CalcMilConDetail
 *
 *   Purpose        : Calculates new construction costs by gaining base
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMilConDetail (MilConDetail)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   09Apr92   AML   Streamlining
 *   08May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcMilConDetail (VAR MilConDetail : MilConRec);

  VAR
    B, P              : Integer;
    UserEntryError,
    UserEntryError2 : Word;

  Begin { CalcMilConDetail }
    UserEntryError := 0;
    UserEntryError2 := 0;
    FOR B := 0 to MaxBases DO
      For P := 1 To 16 Do
        Begin
          If ((Con.Base[B].New[P]   > 0.0) Or
              (Con.Base[B].Rehab[P] > 0.0)) And
              (Con.Base[B].Cost[P]  = 0.0) And
              (StdFctrsCon.ConCost[Ord(Con.Base[B].Category[P])-64] = 0.0) And
              Not Odd(UserEntryError Div Round(YtoX(2,B)))
            Then UserEntryError := UserEntryError + Round(YtoX(2,B));
          If Con.Base[B].Category[P] <> '@'
            Then MilConDetail.Base[B].NewCost[P] :=
                StdFctrsCon.ConCost[Ord (Con.Base[B].Category[P]) - 64]
              * Con.Base[B].New[P]
              * Con.Base[B].AreaCostFactor
            Else MilConDetail.Base[B].NewCost[P] := 0.0;
```

```
        If Con.Base[B].Category[P] <> '@'
          Then MilConDetail.Base[B].RehabCost[P] :=
              StdFctrsCon.ConCost[Ord (Con.Base[B].Category[P]) - 64]
            * StdFctrsFac.RehabCostVsNewConstr
            * Con.Base[B].Rehab[P]
            * Con.Base[B].AreaCostFactor
          Else MilConDetail.Base[B].RehabCost[P] := 0.0;
        If Con.Base[B].Cost[P] = 0
          Then MilConDetail.Base[B].TotalCost[P] :=
               MilConDetail.Base[B].NewCost[P]
             + MilConDetail.Base[B].RehabCost[P]
          Else MilConDetail.Base[B].TotalCost[P] :=
               Con.Base[B].Cost[P] * 1000.0;
        If (Con.Base[B].Cost[P] < 0.001) And
          ((Con.Base[B].New[P] > 0.001) Or
           (Con.Base[B].Rehab[P] > 0.001)) And
           (Con.Base[B].Category[P] = '@') And
            Not Odd(UserEntryError2 Div Round(YtoX(2,B)))
          Then UserEntryError2 := UserEntryError2 + Round(YtoX(2,B));
      End;
  If UserEntryError > 0 Then
    Begin
      ErrorOut ('The following base(s) perform Military Construction');
      ErrorOut ('using Categories with a $0.00 cost per unit measure:');
      ErrorList (UserEntryError);
    End;
  If UserEntryError2 > 0 Then
    Begin
      ErrorOut ('The following base(s) did not specify a Total Cost value');
      ErrorOut ('for OTHER construction requirements:');
      ErrorList (UserEntryError2);
    End;
END;  { CalcMilConDetail }
```

```
{*************************************************************************
 *   Name           : CalcMilConFinal
 *
 *   Purpose        : Calculates and adds final costs to total construction
 *                    costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMilConFinal (MilConDetail, PersMovesSum, MilConFinal)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     -------   ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   08May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcMilConFinal (     MilConDetail : MilConRec;
                                PersMovesSum : PersMovesSumRec;
                            VAR MilConFinal  : MilConFinalRec);

  VAR
    B, P, Y      : INTEGER;
    Design,
    Net,
    TempPerc,
    Total        : REAL;

  Begin { CalcMilConFinal }
    FillChar (MilConFinal, SizeOf (MilConFinal), 0);

  { Perform calculation }

    Design := 0.0;
    FOR B := 0 to MaxBases DO
      For P := 1 To 16 Do
        If Con.Base[B].Cost[P] < 0.001
          Then Design := Design
            + MilConDetail.Base[B].TotalCost[P]
            *(StdFctrsFac.SitePrep
            + StdFctrsFac.SIOH
            + StdFctrsFac.Contingency
            + 1.0)
```

```
          * StdFctrsFac.Design
        Else Design := Design
          + MilConDetail.Base[B].TotalCost[P]
          /(1.0
          + StdFctrsFac.Design)
          * StdFctrsFac.Design;

  For Y := 1 To MaxYears Do
    Begin
      For B := 0 To MaxBases Do
        Begin
          MilConFinal.CostAvoidConstr.Year[Y] :=
              MilConFinal.CostAvoidConstr.Year[Y]
            + CostAvoid.Base[B].Constr.Year[Y]
            + CostAvoid.Base[B].FamQtr.Year[Y];
          If Screen1Data^.ModelDoShutDn = 'Y'
            Then TempPerc := PersMovesSum.Base[B].Perc.Year[Y]
            Else TempPerc := TimePhasShutDn.Base[B].Constr.Year[Y];
          For P := 1 To 16 Do
            If Con.Base[B].Cost[P] = 0
              Then MilConFinal.Constr.Year[Y] :=
                  MilConFinal.Constr.Year[Y]
                +(TempPerc
                * MilConDetail.Base[B].TotalCost[P]
                *(StdFctrsFac.SitePrep
                + StdFctrsFac.SIOH
                + StdFctrsFac.Contingency
                + 1.0))
              Else MilConFinal.Constr.Year[Y] :=
                  MilConFinal.Constr.Year[Y]
                +(TempPerc
                * MilConDetail.Base[B].TotalCost[P]
                /(1.0
                + StdFctrsFac.Design));
        End;
      If Y = 1
        Then MilConFinal.Result.Year[Y] :=
          MilConFinal.Constr.Year[Y] + Design
        Else MilConFinal.Result.Year[Y] :=
          MilConFinal.Constr.Year[Y];
      MilConFinal.Net.Year[Y] :=
          MilConFinal.Result.Year[Y]
        - MilConFinal.CostAvoidConstr.Year[Y]
    END;
END;  { CalcMilConFinal }
```

```
{*************************************************************************
 *   Name           : CalcApproReportMilCon
 *
 *   Purpose        :
 *
 *   Notes          :
 *
 *   Invoked As     :
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   08May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcApproReportMilCon (     MilConDetail       : MilConRec;
                                      MilConFinal        : MilConFinalRec;
                                      PersMovesSum       : PersMovesSumRec;
                                  VAR ApproReportMilCon  : ApproReportMilConRec);

  VAR
    B, P, Y : INTEGER;
    Design,
    Perc    : Real;
    FctrA,
    FctrB   : Array [0..AbsMaxBases] Of Real;

  Begin { CalcApproReportMilCon }
    Design := 0.0;

    FillChar (ApproReportMilCon, SizeOf (ApproReportMilCon), 0);
    FillChar (FctrA, SizeOf (FctrA), 0);
    FillChar (FctrB, SizeOf (FctrB), 0);

    FOR B := 0 to MaxBases DO
      Begin
        For P := 1 To 16 Do
          If Con.Base[B].Category[P] = 'I' Then {Family Quarters}
            Begin
              If Con.Base[B].Cost[P] = 0
```

```
                Then FctrA[B] := FctrA[B]
                   + MilConDetail.Base[B].TotalCost[P]
                   *(1.0
                   + StdFctrsFac.SIOH
                   + StdFctrsFac.Contingency
                   + StdFctrsFac.SitePrep)
                Else FctrB[B] := FctrB[B]
                   + MilConDetail.Base[B].TotalCost[P];
            End;
        Design := Design
          + StdFctrsFac.Design
          *(FctrA[B]
          + FctrB[B]
          /(1.0
          + StdFctrsFac.Design));
      End;

    ApproReportMilCon.Hous.Year[1] := Design;
    FOR Y := 1 to MaxYears DO
      Begin
        ApproReportMilCon.MilCon.Year[Y] := MilConFinal.Net.Year[Y];
        For B := 0 To MaxBases Do
          Begin
            If Screen1Data^.ModelDoShutDn = 'Y'
              Then Perc := PersMovesSum.Base[B].Perc.Year[Y]
              Else Perc := TimePhasShutDn.Base[B].Constr.Year[Y];
            ApproReportMilCon.Hous.Year[Y] :=
                ApproReportMilCon.Hous.Year[Y]
              +(FctrA[B]
              + FctrB[B]
              /(1.0
              + StdFctrsFac.Design))
              * Perc;
            ApproReportMilCon.MilCon.Year[Y] :=
                ApproReportMilCon.MilCon.Year[Y]
              + CostAvoid.Base[B].Constr.Year[Y]
          End;
        ApproReportMilCon.MilCon.Year[Y] :=
            ApproReportMilCon.MilCon.Year[Y]
          - ApproReportMilCon.Hous.Year[Y];
      END
  END;  { CalcApproReportMilCon }
```

```
{*****************************************************************************
 *   Name           : MilConAssetsReport
 *
 *   Purpose        : Writes detailed MilCon assets report to a file
 *
 *   Notes          :
 *
 *   Invoked As     : MilConAssets (MilConDetail)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     MILCONAS.RPT           Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   14Apr92   AML   Changed format and added information
 *****************************************************************************}

PROCEDURE MilConAssetsReport (MilConDetail : MilConRec;
                              PersMovesSum : PersMovesSumRec;
                              Adjust       : Year1ToMaxRealRec);

  VAR
    B, P, Y    : Integer;
    Buffer     : IOBuffer;
    F          : Text;
    AvdTotal,
    ConTotal,
    PurTotal,
    MarkUp,
    TmPhsInf   : Real;
    ConAvoid,
    LandPurch,
    Total      : Array [0..AbsMaxBases] Of Real;

  Begin { MilConAssetsReport }
    AvdTotal := 0;
    ConTotal := 0;
    PurTotal := 0;
    assign (F, Info.ReportDir + 'MILCONAS.RPT');
    SetTextBuf (F, Buffer);
    rewrite (F);
```

```
FOR B := 0 to MaxBases DO
  Begin
    ConAvoid[B] := 0.0;
    LandPurch[B] := 0.0;
    TmPhsInf := 0.0;

    { TmPhsInf is a multiplier to the MilCon figures which has the
      time-phasing and inflation figures in it, so as to take the
      total MilCon figure and return a figure properly inflated for the
      proper amount of construction done each year. }

    For Y := 1 To MaxYears Do
      Begin
        ConAvoid[B] := ConAvoid[B]
          +(CostAvoid.Base[B].Constr.Year[Y]
          + CostAvoid.Base[B].FamQtr.Year[Y])
          * Adjust.Year[Y];
        If RealPropProc.RealPropPur.Base[B].Cost.Year[Y] > 0
          Then LandPurch[B] := LandPurch[B]
            + RealPropProc.RealPropPur.Base[B].Cost.Year[Y]
            * Adjust.Year[Y];
        If Screen1Data^.ModelDoShutDn = 'Y'
          Then TmPhsInf := TmPhsInf
            +(PersMovesSum.Base[B].Perc.Year[Y]
            * Adjust.Year[Y])
          Else TmPhsInf := TmPhsInf
            +(TimePhasShutDn.Base[B].Constr.Year[Y]
            * Adjust.Year[Y]);
        MarkUp := (1.0
          + StdFctrsFac.SitePrep
          + StdFctrsFac.SIOH
          + StdFctrsFac.Contingency)
          *(TmPhsInf
          + StdFctrsFac.Design
          * Adjust.Year[1]);      { All design done in Year 1 }
      End;
    Total[B] := 0.0;
    Title (F, 'MILITARY CONSTRUCTION ASSETS', -B - 1);
    writeln (F, BLANK:10, 'MilCon for Base: ', BaseNState(B));
    writeln (F);
    Writeln (F, '          All Costs in $K');
    writeln (F, Blank:30,'MilCon   Using Rehab   New    New  Total');
    writeln (F, Blank:10, 'Description:',
                Blank:9,  'Categ   Rehab Cost*  MilCon Cost* Cost*');
    writeln (F, Rept('-',13):23,
                Blank:8,  '-----   ----- -----  ------ ----- -----');
    For P := 1 To 16 Do
      If (Con.Base[B].New[P] > 0.001) Or
         (Con.Base[B].Rehab[P] > 0.001) Or
         (Con.Base[B].Cost[P] > 0.001) Or
         (Con.Base[B].Category[P] > '@') Then
```

```
Begin
  If Con.Base[B].ProjectName[P] <> Nil
    Then Write (F, Blank:10, Con.Base[B].ProjectName[P]^,
         Blank:(21 - Length (Con.Base[B].ProjectName[P]^)))
    Else Write (F, Blank:10, '(none given)', Blank:9);
  Case Con.Base[B].Category[P] Of
    '@': Write (F, 'Other');
    'A': Write (F, 'Horiz');
    'B': Write (F, 'Water');
    'C': Write (F, 'AirOp');
    'D': Write (F, 'Opera');
    'E': Write (F, 'Admin');
    'F': Write (F, 'SchlB');
    'G': Write (F, 'Maint');
    'H': Write (F, 'BachQ');
    'I': Write (F, 'FamlQ');
    'J': Write (F, 'Stora');
    'K': Write (F, 'DinFc');
    'L': Write (F, 'RecFc');
    'M': Write (F, 'ComFc');
    'N': Write (F, 'Shpyd');
    'O': Write (F, 'RDT&E');
    'P': Write (F, 'POLSt');
    'Q': Write (F, 'AmmoS');
    'R': Write (F, 'MedFc');
    'S': Write (F, 'Envir');
    'T', 'U', 'V', 'W', 'X',
    'Y', 'Z', '[', '\', ']',
    '^', '_', '`', 'a', 'b', 'c':
      If StdFctrsCon.Name[Ord(Con.Base[B].Category[P])-83] = ''
        Then Write (F, 'Opt-', Chr(Ord(Con.Base[B].Category[P])-19))
        Else Write (F, Copy(
          StdFctrsCon.Name[Ord(Con.Base[B].Category[P])-83],1,5):5)
    End;
  If Con.Base[B].Cost[P] = 0 Then
    Begin
      Writeln(F, WithCommas(Con.Base[B].Rehab[P],0):8,
        WithCommas(MilConDetail.Base[B].RehabCost[P]
          * MarkUp
          / 1000.0,0):6,
        WithCommas(Con.Base[B].New[P],0):8,
        WithCommas(MilConDetail.Base[B].NewCost[P]
          * MarkUp
          / 1000.0,0):6,
        WithCommas(MilConDetail.Base[B].TotalCost[P]
          * MarkUp
          / 1000.0,0):6);
      Total[B] := Total[B]
        + MilConDetail.Base[B].TotalCost[P]
        * MarkUp
    End
```

```
          Else
            Begin
              Writeln (F, WithCommas(Con.Base[B].Rehab[P],0):8, '   n/a',
                WithCommas(Con.Base[B].New[P],0):8, '   n/a',
                WithCommas(Con.Base[B].Cost[P]
                  /(1.0
                  + StdFctrsFac.Design)
                  *(TmPhsInf
                  + StdFctrsFac.Design
                  * Adjust.Year[1]),0):6);
              Total[B] := Total[B]
                + MilConDetail.Base[B].TotalCost[P]
                /(1.0
                + StdFctrsFac.Design)
                *(TmPhsInf
                + StdFctrsFac.Design
                * Adjust.Year[1]);
            End;
          If Con.Base[B].Comment[P] <> Nil
            Then Center (F, ConCat ('(', Con.Base[B].Comment[P]^, ')'));
        End
      Else If Con.Base[B].Comment[P] <> Nil
        Then Center (F, ConCat ('(', Con.Base[B].Comment[P]^, ')'));
    writeln (F, Blank:10, Rept('-', 60));
    writeln (F, Blank:37,   'Total Construction Cost:',
      WithCommas(Total[B] / 1000.0,0):9);
    writeln (F, Blank:35, '+ Cost for Land Purchases:',
      WithCommas (LandPurch[B]/1000.0,0):9);
    writeln (F, Blank:35, '- Construction Cost Avoid:',
      WithCommas (ConAvoid[B]/1000.0,0):9);
    writeln (F, Blank:35, Rept('-', 35));
    Writeln (F, Blank:55, 'TOTAL:',
      WithCommas((Total[B] + LandPurch[B] - ConAvoid[B])/1000.0,0):9);
    AvdTotal := AvdTotal + ConAvoid[B];
    ConTotal := ConTotal + Total[B];
    PurTotal := PurTotal + LandPurch[B];
    Writeln (F);
    Writeln (F, Blank:10,
      '* MilCon Costs include Site Preparation Costs, Design Costs,');
    Writeln (F, Blank:12,
      'Contingency Planning Costs and SIOH Costs where applicable');
  END;
Title (F, 'MILITARY CONSTRUCTION ASSETS', MaxBases + 2);
Writeln (F);
Writeln (F, '          All Costs in $K');
Writeln (F, Blank:34, 'Total     Land       Cost      Total');
Writeln (F, Blank:10,
  'Base Name              MilCon  Purchase     Avoid      Cost');
Writeln (F, Blank:10,
  '---------              -------  --------    ------     -----');
```

```
    For B := 0 To MaxBases Do
      Begin
        Write (F, Blank:10, Screen1Data^.Base[B].Name,
          Blank:(21-Length (Screen1Data^.Base[B].Name)));
        Write(F, WithCommas(Total[B]/1000.0,0):9);
        Write(F, WithCommas(LandPurch[B]/1000.0,0):10);
        Write(F, WithCommas(-ConAvoid[B]/1000.0,0):10);
        Writeln(F,
          WithCommas((Total[B] + LandPurch[B] - ConAvoid[B])/1000.0,0):10);
      End;
    writeln (F, Blank:10, Rept('-', 60));
    write (F, Blank:10, 'Totals:', WithCommas(ConTotal/1000.0,0):23);
    write (F, WithCommas(PurTotal/1000.0,0):10);
    write (F, WithCommas(-AvdTotal/1000.0,0):10);
    writeln (F, WithCommas((ConTotal+ PurTotal-AvdTotal)/1000.0,0):10);
    close (F)
  END;  { MilConAssetsReport }

{*****************************************************************************
 *   Name           : CalcConstructCost
 *
 *   Purpose        : Orchestrates calculation of construction cost by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcConstructCost (PersMovesSum, MilConDetail,
 *                                       MilConFinal)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   08May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcConstructCost (     PersMovesSum : PersMovesSumRec;
                              VAR MilConDetail : MilConRec;
                              VAR MilConFinal  : MilConFinalRec;
                                  Adjust       : Year1ToMaxRealRec);
```

```
  Begin { CalcConstructCost }
    CalcMilConDetail    (MilConDetail);
    CalcMilConFinal     (MilConDetail, PersMovesSum, MilConFinal);
    MilConAssetsReport (MilConDetail, PersMovesSum, Adjust);
  END;  { CalcConstructCost }

{*************************************************************************
 *   Name           : CalcMissionCost
 *
 *   Purpose        : Calculates mission costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMissionCost (PersMovesSum, MissionCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   09Apr92   AML   Streamlining
 *   11May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcMissionCost (     PersMovesSum   : PersMovesSumRec;
                            VAR MissionReport  : CostSaveNetByYearRec;
                                Adjust1000     : Year1ToMaxRealRec);
  VAR
    B, Y        : INTEGER;
    Buffer      : IOBuffer;
    F           : TEXT;

  Begin { CalcMissionCost }
    FillChar (MissionReport, SizeOf (MissionReport), 0);

    For Y := 1 To MaxYears Do
      Begin
        For B := 0 to MaxBases DO
          If ActivMisCost^.Base[B].Year[Y] >= 0.0
            Then MissionReport.Cost.Year[Y] :=
                 MissionReport.Cost.Year[Y]
               + ActivMisCost^.Base[B].Year[Y]
```

```
          Else MissionReport.Save.Year[Y] :=
               MissionReport.Save.Year[Y]
             - ActivMisCost^.Base[B].Year[Y];
      MissionReport.Net.Year[Y] :=
        MissionReport.Cost.Year[Y] - MissionReport.Save.Year[Y];
    End;

  MissionReport.Net.Beyond := MissionReport.Net.Year[MaxYears];
  MissionReport.Cost.Beyond := MissionReport.Cost.Year[MaxYears];
  MissionReport.Save.Beyond := MissionReport.Save.Year[MaxYears];

  assign  (F, Info.ReportDir + 'MISSION.RPT');
  SetTextBuf (F, Buffer);
  rewrite (F);
  Title (F, 'MISSION COSTS', -1);
  writeln (F, Blank:10, 'Yearly Cost Breakout ($K)');
  writeln (F);
  write (F, Blank:27);
  For Y := 1 To MaxYears Do
    Write (F, FiscalYear(Y):7);
  Writeln (F, '*');
  write (F, Blank:27);
  For Y := 1 To MaxYears Do
    Write (F, '   ----');
  writeln (F);
  BeyondOut (F, 'Mission Costs    ', MissionReport.Cost, Adjust1000, 7, 0);
  BeyondOut (F, 'Mission Savings  ', MissionReport.Save, Adjust1000, 7, 0);
  writeln (F);
  BeyondOut (F, 'Net Mission Costs', MissionReport.Net, Adjust1000, 7, 0);
  Writeln (F);
  Writeln (F, Blank:10,
    '* These values also apply to Beyond Year calculations.');
  close (F)
END;  { CalcMissionCost }
```

```
{*****************************************************************************
 *   Name           : CalcBaseStats
 *
 *   Purpose        : Calculates statistics for each gaining base
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMissionCost (PersMovesSum, MissionCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     GAINBASE.RPT
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   14Apr91   AJD   Creation
 *****************************************************************************}

Procedure CalcTot1TimeCost (     MilConDetail      : MilConRec;
                                 MovCostAlt        : PBase0ToMaxRealRec;
                                 NewHireCost       : Year1ToMaxRealRec;
                                 PersMoves         : CalcMovesRec;
                                 PersMovesSum      : PersMovesSumRec;
                                 CalcPersAct       : CalcPersActRec;
                                 MilConFinal       : MilConFinalRec;
                                 MovCostFinal      : YearAndBeyondRealRec;
                                 IMACost           : Year1ToMaxRealRec;
                                 AdminSuppCost     : Base0Year1ToMaxRealRec;
                             VAR TotalOneTimeCost  : Real;
                                 Adjust            : Year1ToMaxRealRec);

  Var
    B, B2, P, Y : Integer;
    Buffer      : IOBuffer;
    F           : Text;
    Admin,
    AdminSupp,
    AvoidC,
    AvoidP,
    ConAvoid,
    EarlyRetire,
    Elim,
    ElimMil,
    Envir,
    EnvirMit,
```

```
  Hire,
  HomeAsst,
  Homeown,
  IMA,
  InfoMgt,
  Land,
  LandSales,
  MilCon,
  MilConTot,
  Mothball,
  Moving,
  NewHire,
  OneTimUnq,
  PCS,
  PPS,
  ProAvoid,
  PropCost,
  PropSave,
  Retire,
  RIF,
  RIFed,
  ShutDown,
  Temp,
  Temp2,
  Time1Unique,
  TmPhsInf,      { See MilConAssetsReport above for TmPhsInf explanation }
  Total,
  Unemp,
  Unemploy     : Real;
  CivsGone,
  CivsMoving,
  TempAdmin    : Year1ToMaxRealRec;

Begin

{ Initialize 1TIMCOST Variables }
  Admin              := 0.0;
  ConAvoid           := 0.0;
  EarlyRetire        := 0.0;
  ElimMil            := 0.0;
  Envir              := 0.0;
  Homeown            := 0.0;
  IMA                := 0.0;
  Land               := 0.0;
  LandSales          := 0.0;
  MilCon             := 0.0;
  Mothball           := 0.0;
  Moving             := 0.0;
  NewHire            := 0.0;
  PPS                := 0.0;
  ProAvoid           := 0.0;
  RIF                := 0.0;
```

```
Time1Unique      := 0.0;
TotalOneTimeCost := 0.0;
Unemploy         := 0.0;

For Y := 1 To MaxYears Do
  Begin
    CivsGone.Year[Y] := 0.0;
    CivsMoving.Year[Y] := 0.0;
    For B := 0 To MaxBases Do
      Begin
        CivsGone.Year[Y] := CivsGone.Year[Y]
          + PosElim.Base[B].Civ.Year[Y];
        For B2 := 0 To MaxBases Do
          If Distance^.Base[B].Base[B2] > 0
            Then CivsMoving.Year[Y] := CivsMoving.Year[Y]
              + PosTrans.Base[B].Base[B2]^.Civ.Year[Y];
      End;
  End;
Assign (F, Info.ReportDir + '1TIMCOST.RPT');
SetTextBuf (F, Buffer);
Rewrite (F);
For B := 0 To MaxBases Do
  Begin
    Title (F, 'BASE ONE-TIME COST REPORT', B+1);
    Writeln (F);
    Writeln (F, Blank:19, 'Base:  ', BaseNState(B));
    Writeln (F, Blank:19, '(All values in Dollars)');
    Writeln (F);

    AdminSupp := 0.0;
    AvoidC    := 0.0;
    AvoidP    := 0.0;
    Elim      := 0.0;
    EnvirMit  := 0.0;
    Hire      := 0.0;
    HomeAsst  := 0.0;
    InfoMgt   := 0.0;
    MilConTot := 0.0;
    OneTimUnq := 0.0;
    PCS       := 0.0;
    PropCost  := 0.0;
    PropSave  := 0.0;
    Retire    := 0.0;
    RIFed     := 0.0;
    ShutDown  := 0.0;
    Temp      := 0.0;
    TmPhsInf  := 0.0;
    Total     := 0.0;
    Unemp     := 0.0;
```

```
For Y := 1 To MaxYears Do
  If Screen1Data^.Model1DoShutDn = 'Y'
    Then TmPhsInf := TmPhsInf
      +(PersMovesSum.Base[B].Perc.Year[Y]
      * Adjust.Year[Y])
    Else TmPhsInf := TmPhsInf
      +(TimePhasShutDn.Base[B].Constr.Year[Y]
      * Adjust.Year[Y]);

For P := 1 To 16 Do
  Begin
    If Con.Base[B].Cost[P] = 0
      Then MilConTot := MilConTot
        + MilConDetail.Base[B].TotalCost[P]
        *(1.0
        + StdFctrsFac.SitePrep
        + StdFctrsFac.SIOH
        + StdFctrsFac.Contingency)
        *(TmPhsInf
        + StdFctrsFac.Design
        * Adjust.Year[1])
      Else MilConTot := MilConTot
        + MilConDetail.Base[B].TotalCost[P]
        /(1.0
        + StdFctrsFac.Design)
        *(TmPhsInf
        + StdFctrsFac.Design
        * Adjust.Year[1]);
    If ((Con.Base[B].Category[P]  > 'B') And
        (Con.Base[B].Category[P] <> 'P') And
        (Con.Base[B].Category[P] <  'S'))
      Then InfoMgt := InfoMgt + MilConDetail.Base[B].NewCost[P]
    Else If Con.Base[B].Category[P] > 'R'
      Then If StdFctrsCon.Units[Ord(Con.Base[B].Category[P])-83] = 'SF'
        Then InfoMgt := InfoMgt + MilConDetail.Base[B].NewCost[P];
  End;
InfoMgt := InfoMgt
  * StdFctrsFac.IMAPerc
  * TmPhsInf;
writeln (F, Blank:19, 'MilCon w/o Avoidances',
  WithCommas (MilConTot, 0):22);
MilCon := MilCon + MilConTot;
writeln (F, Blank:17, '+ Moving',
  WithCommas (MovCostAlt^.Base[B], 0):37);
Moving := Moving + MovCostAlt^.Base[B];
For Y := 1 To MaxYears Do
  Begin
    AdminSupp := AdminSupp
      + AdminSuppCost.Base[B].Year[Y];
```

```
Elim := Elim
  +((PosElim.Base[B].Off.Year[Y]
  *  StdFctrsTrans.OneTimeOffPCSCosts)
  + (PosElim.Base[B].Enl.Year[Y]
  *  StdFctrsTrans.OneTimeEnlPCSCosts))
  *  Adjust.Year[Y];
Hire := Hire
  + NewHireCost.Year[Y]
  * ZeroCheck
   (PersMoves.Civ.Into.Base[B].Year[Y],
    CivsMoving.Year[Y])
  * Adjust.Year[Y];
PCS := PCS
  +(CalcPersAct.Year[Y].TotalPPS
  - CalcPersAct.Year[Y].NumAttrited)
  * ZeroCheck
  ((PersMoves.Civ.From.Base[B].Year[Y]
  + PosElim.Base[B].Civ.Year[Y]),
   (CivsMoving.Year[Y]
  + CivsGone.Year[Y]))
  * StdFctrsPers.CivPCSCosts
  * Adjust.Year[Y];
Retire := Retire
  + CalcPersAct.Year[Y].EarlyRetire
  * ZeroCheck
  ((PersMoves.Civ.From.Base[B].Year[Y]
  + PosElim.Base[B].Civ.Year[Y]),
   (CivsMoving.Year[Y]
  + CivsGone.Year[Y]))
  * Adjust.Year[Y];
RIFed := RIFed
  + CalcPersAct.Year[Y].RIFPay
  * ZeroCheck
  ((PersMoves.Civ.From.Base[B].Year[Y]
  + PosElim.Base[B].Civ.Year[Y]),
   (CivsMoving.Year[Y]
  + CivsGone.Year[Y]))
  * Adjust.Year[Y];
EnvirMit := EnvirMit
  + ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y]
  * Adjust.Year[Y];
OneTimUnq := OneTimUnq
  + ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[Y]
  * Adjust.Year[Y];
If HapSwitch.Base[B] = 'Y' Then
  Begin
    Temp2 :=
       (PosElim.Base[B].Civ.Year[Y]
      * StdFctrsPers.CivHomeOwnRate
      + PosElim.Base[B].Off.Year[Y]
      * StdFctrsPers.PercOffMarried
      *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
```

```
        * StdFctrsPers.CivHomeOwnRate
        + PosElim.Base[B].Enl.Year[Y]
        * StdFctrsPers.PercEnlMarried
        *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
        * StdFctrsPers.CivHomeOwnRate)
        * Con.Base[B].AreaCostFactor
        * StdFctrsPers.HAPFactor
        * StdFctrsPers.HAPReceive;
      For B2 := 0 To MaxBases Do
        If Distance^.Base[B].Base[B2] >= 50.0
          Then Temp2 := Temp2
            +(PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
            * StdFctrsPers.CivHomeOwnRate
            + PosTrans.Base[B].Base[B2]^.Off.Year[Y]
            * StdFctrsPers.PercOffMarried
            *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
            * StdFctrsPers.CivHomeOwnRate
            + PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
            * StdFctrsPers.PercEnlMarried
            *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
            * StdFctrsPers.CivHomeOwnRate)
            * Con.Base[B].AreaCostFactor
            * StdFctrsPers.HAPFactor
            * StdFctrsPers.HAPReceive
  End
Else
  Begin
    Temp2 :=
          PosElim.Base[B].Civ.Year[Y]
        * StdFctrsPers.CivHomeOwnRate
        * Con.Base[B].AreaCostFactor
        * StdFctrsPers.DARSEFactor
        * StdFctrsPers.DARSEReceive;
    For B2 := 0 To MaxBases Do
      If Distance^.Base[B].Base[B2] >= 50.0
        Then Temp2 := Temp2
          + PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
          * StdFctrsPers.CivHomeOwnRate
          * Con.Base[B].AreaCostFactor
          * StdFctrsPers.DARSEFactor
          * StdFctrsPers.DARSEReceive;
  End;
HomeAsst := HomeAsst
  + StdFctrsPers.NatlAvgHomePrice
  * Temp2
  * Adjust.Year[Y];
If RealPropProc.RealPropPur.Base[B].Cost.Year[Y] > 0
  Then PropCost := PropCost
    + RealPropProc.RealPropPur.Base[B].Cost.Year[Y]
    * Adjust.Year[Y]
```

```
        Else PropSave := PropSave
          - RealPropProc.RealPropPur.Base[B].Cost.Year[Y]
          * Adjust.Year[Y];
      AvoidC := AvoidC + CostAvoid.Base[B].Constr.Year[Y]
                       + CostAvoid.Base[B].FamQtr.Year[Y];
      AvoidP := AvoidP + CostAvoid.Base[B].Procure.Year[Y];
      Unemp := Unemp
        + CalcPersAct.Year[Y].NumRIFted
        * ZeroCheck
        ((PersMoves.Civ.From.Base[B].Year[Y]
        + PosElim.Base[B].Civ.Year[Y]),
         (CivsMoving.Year[Y]
        + CivsGone.Year[Y]))
        * StdFctrsPers.AvgUnemployCostPerWeek
        * StdFctrsPers.UnemployEligible
        * Adjust.Year[Y];
    End;
  If IndustFundSwitch.Base[B] = 'Y' Then
    Begin
      If Screen1Data^.Base[B].CloseYear > 0
        Then ShutDown := IndustFund.Base[B]^.OvhdCost.MothBall.Year
            [Screen1Data^.Base[B].CloseYear]
          * Adjust.Year[Y];
    End
  Else
    Begin
      If Screen1Data^.Base[B].CloseYear > 0
        Then ShutDown := StdFctrsFac.MothballCostPerSqrFt
          * ActionEnvir^.Base[B].BaseFacilSqFt
          * Adjust.Year[Y]
    End;
  writeln (F, Blank:17, '+ Eliminated Military PCS',
    WithCommas (Elim, 0):20);
  ElimMil := ElimMil + Elim;
  writeln (F, Blank:17, '+ Administrative/Support',
    WithCommas (AdminSupp, 0):21);
  Admin := Admin + AdminSupp;
  writeln (F, Blank:17, '+ Mothball/Shutdown',
    WithCommas (ShutDown, 0):26);
  MothBall := MothBall + ShutDown;
  writeln (F, Blank:17, '+ Civilian RIF',
    WithCommas (RIFed, 0):31);
  RIF := RIF + RIFed;
  writeln (F, Blank:17, '+ Civilian Early Retirement',
    WithCommas (Retire, 0):18);
  EarlyRetire := EarlyRetire + Retire;
  writeln (F, Blank:17, '+ Civilian New Hires',
    WithCommas (Hire, 0):25);
  NewHire := NewHire + Hire;
  writeln (F, Blank:17, '+ Civilian PPS',
    WithCommas (PCS, 0):31);
  PPS := PPS + PCS;
```

```
Writeln (F, Blank:17, '+ Land Purchases',
  WithCommas (PropCost,0):29);
Land := Land + PropCost;
Writeln (F, Blank:17, '+ Environmental Mitigation',
  WithCommas (EnvirMit,0):19);
Envir := Envir + EnvirMit;
Writeln (F, Blank:17, '+ One-Time Unique Costs',
  WithCommas (OneTimUnq,0):22);
Time1Unique := Time1Unique + OneTimUnq;
writeln (F, Blank:17, '+ HAP / RSE',
  WithCommas (HomeAsst, 0):34);
HomeOwn := HomeOwn + HomeAsst;
writeln (F, Blank:17, '+ Unemployment',
  WithCommas (Unemp, 0):31);
Unemploy := Unemploy + Unemp;
writeln (F, Blank:17, '+ Info Management Account',
  WithCommas (InfoMgt, 0):20);
IMA := IMA + InfoMgt;
Total :=
    MilConTot
  + MovCostAlt^.Base[B]
  + Elim
  + AdminSupp
  + Shutdown
  + RIFed
  + Retire
  + Hire
  + PropCost
  + EnvirMit
  + OneTimUnq
  + HomeAsst
  + Unemp
  + InfoMgt
  + PCS;
TotalOneTimeCost := TotalOneTimeCost + Total;
writeln (F, Blank:17, Rept('-', 45));
writeln (F, Blank:17, '= Total One-Time Costs',
WithCommas (Total, 0):23);
writeln (F);
writeln (F);
writeln (F, Blank:19, 'Milcon Cost Avoidances',
  WithCommas (AvoidC, 0):21);
ConAvoid := ConAvoid + AvoidC;
writeln (F, Blank:17, '+ Procurement Cost Avoidances',
  WithCommas (AvoidP, 0):16);
ProAvoid := ProAvoid + AvoidP;
writeln (F, Blank:17, '+ Land Sales',
  WithCommas (PropSave, 0):33);
LandSales := LandSales + PropSave;
writeln (F, Blank:17, Rept('-', 45));
writeln (F, Blank:17, '= Total One-Time Savings',
  WithCommas (AvoidC + AvoidP + PropSave, 0):21);
```

```
    writeln (F);
    writeln (F);
    writeln (F, Blank:19, 'Total One-Time Costs',
      WithCommas (Total, 0):23);
    writeln (F, Blank:17, '- Total One-Time Savings',
      WithCommas (AvoidC + AvoidP + PropSave, 0):21);
    writeln (F, Blank:17, Rept('-', 45));
    writeln (F, Blank:17, '= Total Net One-Time Costs',
      WithCommas (Total - AvoidC - AvoidP - PropSave, 0):19);
  End;
Title (F, 'TOTAL ONE-TIME COST REPORT', 0);
writeln (F);
Writeln (F, Blank:19, '(All values in Dollars)');
Writeln (F);

writeln (F, Blank:19, 'MilCon w/o Avoidances',
  WithCommas (MilCon, 0):22);
writeln (F, Blank:17, '+ Moving',
  WithCommas (Moving, 0):37);
writeln (F, Blank:17, '+ Eliminated Military PCS',
  WithCommas (ElimMil, 0):20);
writeln (F, Blank:17, '+ Administrative/Support',
  WithCommas (Admin, 0):21);
writeln (F, Blank:17, '+ Mothball/Shutdown',
  WithCommas (Mothball, 0):26);
writeln (F, Blank:17, '+ Civilian RIF',
  WithCommas (RIF, 0):31);
writeln (F, Blank:17, '+ Civilian Early Retirement',
  WithCommas (EarlyRetire, 0):18);
writeln (F, Blank:17, '+ Civilian New Hires',
  WithCommas (NewHire, 0):25);
writeln (F, Blank:17, '+ Civilian PPS',
  WithCommas (PPS, 0):31);
writeln (F, Blank:17, '+ Land Purchases',
  WithCommas (Land, 0):29);
writeln (F, Blank:17, '+ Environmental Mitigation',
  WithCommas (Envir, 0):19);
writeln (F, Blank:17, '+ One-Time Unique Costs',
  WithCommas (Time1Unique, 0):22);
writeln (F, Blank:17, '+ HAP / RSE',
  WithCommas (Homeown, 0):34);
writeln (F, Blank:17, '+ Unemployment',
  WithCommas (Unemploy, 0):31);
writeln (F, Blank:17, '+ Info Management Account',
  WithCommas (IMA, 0):20);
writeln (F, Blank:17, Rept('-', 45));
writeln (F, Blank:17, '= Total One-Time Costs',
  WithCommas (TotalOneTimeCost, 0):23);
writeln (F);
writeln (F);
writeln (F, Blank:19, 'Milcon Cost Avoidances',
  WithCommas (ConAvoid, 0):21);
```

```
    writeln (F, Blank:17, '+ Procurement Cost Avoidances',
      WithCommas (ProAvoid, 0):16);
    writeln (F, Blank:17, '+ Land Sales',
      WithCommas (LandSales, 0):33);
    writeln (F, Blank:17, Rept('-', 45));
    writeln (F, Blank:17, '= Total One-Time Savings',
      WithCommas (ConAvoid + ProAvoid + LandSales, 0):21);
    writeln (F);
    writeln (F);
    writeln (F, Blank:19, 'Total One-Time Costs',
      WithCommas (TotalOneTimeCost, 0):23);
    writeln (F, Blank:17, '- Total One-Time Savings',
      WithCommas (ConAvoid + ProAvoid + LandSales, 0):21);
    writeln (F, Blank:17, Rept('-', 45));
    writeln (F, Blank:17, '= Total Net One-Time Costs',
      WithCommas (TotalOneTimeCost - ConAvoid - ProAvoid - LandSales, 0):19);
    Close (F);
  End;

END. { UNIT }
```

```
UNIT CCALMOV;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE CalcMovingCost (     PersMoves     : CalcMovesRec;
                           VAR MovCostFinal  : YearAndBeyondRealRec;
                           VAR MovCostAlt    : PBase0ToMaxRealRec;
                           VAR PerDiemCiv,
                               POVMilesCiv,
                               HousPurch,
                               HHGCiv,
                               MiscCiv,
                               HousHunt,
                               RITA,
                               Pack,
                               Frat,
                               Vehic,
                               Driv,
                               Loss,
                               PerDiemMil,
                               POVMilesMil,
                               HHGMil,
                               MiscMil,
                               MilMov,
                               OneTimeMove   : Year1ToMaxRealRec;
                               CalcPersAct   : CalcPersActRec;
                               Adjust1000    : Year1ToMaxRealRec);

IMPLEMENTATION
```

```
{*************************************************************************
 *   Name           : CalcPCSCosts
 *
 *   Purpose        : Calculates per diem costs, POV miles, house purchase cost,
 *                    HHG cost, Misc cost, house hunting cost, and RITA cost
 *                    by base by year
 *
 *   Notes          : Originally, all of these values were calculated in
 *                    separate procedures with separate loops, etc.
 *
 *   Notes          : Final RITA cost is not calculated here.  It is calculated
 *                    in CalcPCSCostFinal when the RITA cost in this procedure
 *                    is added to HousPurchCost and HousHuntCost and then
 *                    multiplied by a factor of 28%
 *
 *   Invoked As     : CalcPerDiemCost (PersMoves, PerDiemCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   12May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcPCSCosts (     PersMoves        : CalcMovesRec;
                         VAR PerDiemMilCost,
                             PerDiemCivCost,
                             POVMilesMilCost,
                             POVMilesCivCost,
                             HousPurchCost,
                             HHGMilCost,
                             HHGCivCost,
                             MiscMilCost,
                             MiscCivCost,
                             HousHuntCost,
                             MilMoves,
                             RITACost         : Base0Year1ToMaxRealRec;
                             CalcPersAct      : CalcPersActRec;
                         VAR PCSCostFinal     : Year1ToMaxRealRec);
```

```
VAR
  B, B2, Y,
  NumDays,
  NumPeople,
  NumTrips    : INTEGER;
  CostPerLb,
  Fctr,
  Fctr1,
  Fctr2,
  HHGOffFam,
  HHGEnlFam,
  HHGMilSin,
  HHGCiv      : REAL;

Begin { CalcPerDiemCost }
  FillChar (PerDiemMilCost, SizeOf (PerDiemMilCost), 0);
  FillChar (PerDiemCivCost, SizeOf (PerDiemCivCost), 0);
  FillChar (POVMilesMilCost, SizeOf (POVMilesMilCost), 0);
  FillChar (POVMilesCivCost, SizeOf (POVMilesCivCost), 0);
  FillChar (HousPurchCost, SizeOf (HousPurchCost), 0);
  FillChar (HHGMilCost, SizeOf (HHGMilCost), 0);
  FillChar (HHGCivCost, SizeOf (HHGCivCost), 0);
  FillChar (MiscMilCost, SizeOf (MiscMilCost), 0);
  FillChar (MiscCivCost, SizeOf (MiscCivCost), 0);
  FillChar (HousHuntCost, SizeOf (HousHuntCost), 0);
  FillChar (RITACost, SizeOf (RITACost), 0);
  FillChar (PCSCostFinal, SizeOf (PCSCostFinal), 0);
  FillChar (MilMoves, SizeOf (MilMoves), 0);

  NumPeople :=  2;
  NumTrips  :=  2;
  NumDays   := 10;
  Fctr      :=  1.75;

  FOR B := 0 to MaxBases DO
    For B2 := 0 To MaxBases Do
      If Distance^.Base[B].Base[B2] >= 50.0
        Then For Y := 1 to MaxYears DO
          Begin
            PerDiemMilCost.Base[B].Year[Y] :=
                PerDiemMilCost.Base[B].Year[Y]
              +(PosTrans.Base[B].Base[B2]^.Off.Year[Y]
              + PosTrans.Base[B].Base[B2]^.Enl.Year[Y])
              * PersCost.Base[B2].PerDiemRate
              *(Distance^.Base[B].Base[B2] / 350.0);
            PerDiemCivCost.Base[B].Year[Y] :=
                PerDiemCivCost.Base[B].Year[Y]
              +(Trunc
               (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              * CalcPersAct.Year[Y].PercOfNeed)
              * PersCost.Base[B2].PerDiemRate
              *(30.0
```

```
  + Distance^.Base[B].Base[B2]
  / 350.0));

POVMilesMilCost.Base[B].Year[Y] :=
    POVMilesMilCost.Base[B].Year[Y]
  +(PosTrans.Base[B].Base[B2]^.Off.Year[Y]
  + PosTrans.Base[B].Base[B2]^.Enl.Year[Y])
  * Distance^.Base[B].Base[B2]
  * StdFctrsTrans.POVReimbursePerMile;
POVMilesCivCost.Base[B].Year[Y] :=
    POVMilesCivCost.Base[B].Year[Y]
  + Trunc
   (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
  * CalcPersAct.Year[Y].PercOfNeed)
  * Distance^.Base[B].Base[B2]
  * StdFctrsTrans.POVReimbursePerMile;

Fctr1 :=
    StdFctrsPers.NatlAvgHomePrice
  * Con.Base[B].AreaCostFactor
  * StdFctrsPers.HomeSaleReimburseRate;
If Fctr1 > StdFctrsPers.HomeSaleReimburseMax
  Then Fctr1 := StdFctrsPers.HomeSaleReimburseMax;
Fctr2 :=
    StdFctrsPers.NatlAvgHomePrice
  * Con.Base[B2].AreaCostFactor
  * StdFctrsPers.HomePurchReimburseRate;
If Fctr2 > StdFctrsPers.HomePurchReimburseMax
  Then Fctr2 := StdFctrsPers.HomePurchReimburseMax;
If HAPSwitch.Base[B] = 'Y'
  Then Fctr1 := Fctr1 * (1 - StdFctrsPers.HAPReceive)
  Else Fctr1 := Fctr1 * (1 - StdFctrsPers.DARSEReceive);
HousPurchCost.Base[B].Year[Y] :=
    HousPurchCost.Base[B].Year[Y]
  + Trunc
   (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
  * CalcPersAct.Year[Y].PercOfNeed)
  * StdFctrsPers.CivHomeOwnRate
  *(Fctr1 + Fctr2);

MiscMilCost.Base[B].Year[Y] :=
    MiscMilCost.Base[B].Year[Y]
  +(PosTrans.Base[B].Base[B2]^.Off.Year[Y]
  + PosTrans.Base[B].Base[B2]^.Enl.Year[Y])
  * StdFctrsTrans.MiscExpPerDirectEmp;
MiscCivCost.Base[B].Year[Y] :=
    MiscCivCost.Base[B].Year[Y]
  + Trunc
   (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
  * CalcPersAct.Year[Y].PercOfNeed)
  * StdFctrsTrans.MiscExpPerDirectEmp;
```

```
HHGOffFam :=
    StdFctrsTrans.HHGWtPerOffFam
  * PosTrans.Base[B].Base[B2]^.Off.Year[Y]
  * StdFctrsPers.PercOffMarried;
HHGEnlFam :=
    StdFctrsTrans.HHGWtPerEnlFam
  * PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
  * StdFctrsPers.PercEnlMarried;
HHGMilSin :=
    StdFctrsTrans.HHGWtPerMilSing
  *(PosTrans.Base[B].Base[B2]^.Off.Year[Y]
  *(1 - StdFctrsPers.PercOffMarried)
  + PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
  *(1 - StdFctrsPers.PercEnlMarried));
HHGCiv :=
    StdFctrsTrans.HHGWtPerCiv
  * Trunc
   (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
  * CalcPersAct.Year[Y].PercOfNeed);
CostPerLb :=
    StdFctrsTrans.HHGCost
  +(FreightCost.Base[B2]
  * Distance^.Base[B].Base[B2]
  / 2000.0);
HHGMilCost.Base[B].Year[Y] :=
    HHGMilCost.Base[B].Year[Y]
  +(HHGOffFam
  + HHGEnlFam
  + HHGMilSin)
  * CostPerLb;
HHGCivCost.Base[B].Year[Y] :=
    HHGCivCost.Base[B].Year[Y]
  + HHGCiv
  * CostPerLb;

HousHuntCost.Base[B].Year[Y] :=
    HousHuntCost.Base[B].Year[Y]
  + Trunc
   (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
  * CalcPersAct.Year[Y].PercOfNeed)
  *(NumPeople
  * NumTrips
  * Distance^.Base[B].Base[B2]
  * StdFctrsTrans.AirTransPerPassMile
  +(PersCost.Base[B2].PerDiemRate
  * NumDays
  * Fctr));
```

```
            RITACost.Base[B].Year[Y] :=
                RITACost.Base[B].Year[Y]
              + Trunc
               (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              * CalcPersAct.Year[Y].PercOfNeed)
              *((PersCost.Base[B2].PerDiemRate
              *((Distance^.Base[B].Base[B2]
              / 350.0)
              + 30.0))
              +(Distance^.Base[B].Base[B2]
              * StdFctrsTrans.POVReimbursePerMile)
              + StdFctrsTrans.MiscExpPerDirectEmp);

            MilMoves.Base[B].Year[Y] :=
                MilMoves.Base[B].Year[Y]
              + PosTrans.Base[B].Base[B2]^.Off.Year[Y]
              + PosTrans.Base[B].Base[B2]^.Enl.Year[Y];
          END;

  FOR Y := 1 to MaxYears DO
    FOR B := 0 To MaxBases DO
      PCSCostFinal.Year[Y] :=
          PCSCostFinal.Year[Y]
        + PerDiemMilCost.Base[B].Year[Y]
        + PerDiemCivCost.Base[B].Year[Y]
        + POVMilesMilCost.Base[B].Year[Y]
        + POVMilesCivCost.Base[B].Year[Y]
        + HousPurchCost.Base[B].Year[Y]
        + HHGMilCost.Base[B].Year[Y]
        + HHGCivCost.Base[B].Year[Y]
        + MiscMilCost.Base[B].Year[Y]
        + MiscCivCost.Base[B].Year[Y]
        + HousHuntCost.Base[B].Year[Y]
        +((HousPurchCost.Base[B].Year[Y]
        + HousHuntCost.Base[B].Year[Y]
        + RITACost.Base[B].Year[Y])* 0.28)
        -(MilMoves.Base[B].Year[Y]
        * StdFctrsTrans.RoutinePCSCostsPerPerson
        / StdFctrsTrans.AverageTourLength);
END;  { CalcPCSCosts }
```

```
{*************************************************************************
 *   Name          : CalcFreightCosts
 *
 *   Purpose       : Calculates packing/etc costs, freight costs, vehicle and
 *                   driving costs, and loss rate by base by year
 *
 *   Notes         : Originally, all of these values were calculated in
 *                   separate procedures with separate loops, etc.
 *
 *   Invoked As    : CalcPackEtcCost (PersMoves, PackEtcCost)
 *
 *   Return value  :
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   12May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcFreightCosts (     PersMoves      : CalcMovesRec;
                             VAR PackEtcCost    : Base0Year1ToMaxRealRec;
                                 CalcPersAct    : CalcPersActRec;
                             VAR FrateCost,
                                 VehicCost,
                                 DrivCost,
                                 LossRate       : Base0Year1ToMaxRealRec;
                             VAR FrateCostFinal : Year1ToMaxRealRec);

  VAR
    B, B2, Y : INTEGER;

  Begin { CalcPackEtcCost }
    FillChar (PackEtcCost, SizeOf (PackEtcCost), 0);
    FillChar (FrateCost,  SizeOf (FrateCost), 0);
    FillChar (VehicCost, SizeOf (VehicCost), 0);
    FillChar (DrivCost, SizeOf (DrivCost), 0);
    FillChar (LossRate, SizeOf (LossRate), 0);

    FOR B := 0 To MaxBases DO
      FOR Y := 1 to MaxYears DO
        If IndustFundSwitch.Base[B] = 'Y' Then
          Begin
            PackEtcCost.Base[B].Year[Y] :=
              IndustFund.Base[B]^.MoveCost.PackUnpack.Year[Y];
```

```
      FrateCost.Base[B].Year[Y] :=
        IndustFund.Base[B]^.MoveCost.Freight.Year[Y];
      VehicCost.Base[B].Year[Y] :=
        IndustFund.Base[B]^.MoveCost.Vehic.Year[Y];
      DrivCost.Base[B].Year[Y] :=
        IndustFund.Base[B]^.MoveCost.Driv.Year[Y];
      LossRate.Base[B].Year[Y] :=
        IndustFund.Base[B]^.MoveCost.LossRate.Year[Y];
    End
  Else
    Begin
      PackEtcCost.Base[B].Year[Y] :=
        +(PersMoves.Off.From.Base[B].Year[Y]
        + PersMoves.Enl.From.Base[B].Year[Y]
        + PersMoves.Stu.From.Base[B].Year[Y]
        + Trunc
         (PersMoves.Civ.From.Base[B].Year[Y]
        * CalcPersAct.Year[Y].PercOfNeed))
        * StdFctrsTrans.MaterialPerPerson
        * StdFctrsTrans.HHGCost;
      For B2 := 0 To MaxBases Do
        If Distance^.Base[B].Base[B2] <> 0 Then
          Begin
            FrateCost.Base[B].Year[Y] :=
                FrateCost.Base[B].Year[Y]
              +(PosTrans.Base[B].Base[B2]^.Off.Year[Y]
              + PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
              + PosTrans.Base[B].Base[B2]^.Stu.Year[Y]
              + Trunc
               (PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              * CalcPersAct.Year[Y].PercOfNeed))
              * StdFctrsTrans.MaterialPerPerson
              / 2000.0
              * FreightCost.Base[B2]
              * Distance^.Base[B].Base[B2]
              +(ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y]
              + ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y])
              * FreightCost.Base[B2]
              * Distance^.Base[B].Base[B2]
              +(ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y]
              + ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y])
              * StdFctrsTrans.PackNCrate;

            VehicCost.Base[B].Year[Y] :=
                VehicCost.Base[B].Year[Y]
              + ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y]
              * StdFctrsTrans.MilVehicCostPerMile
              * Distance^.Base[B].Base[B2];

            DrivCost.Base[B].Year[Y] :=
                DrivCost.Base[B].Year[Y]
              + ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y]
```

```
                    * StdFctrsTrans.MilLightVehicCostPerMile
                    * Distance^.Base[B].Base[B2];
                End;
            End;

    FOR B := 0 To MaxBases DO
      If IndustFundSwitch.Base[B] <> 'Y'
        Then FOR Y := 1 to MaxYears DO
          LossRate.Base[B].Year[Y] :=
             (FrateCost.Base[B].Year[Y]
           + VehicCost.Base[B].Year[Y]
           + DrivCost.Base[B].Year[Y])
           * StdFctrsTrans.ShippingLossRate;

    For Y := 1 to MaxYears DO
      For B := 0 to MaxBases DO
        FrateCostFinal.Year[Y] :=
            FrateCostFinal.Year[Y]
          + PackEtcCost.Base[B].Year[Y]
          + FrateCost.Base[B].Year[Y]
          + VehicCost.Base[B].Year[Y]
          + DrivCost.Base[B].Year[Y]
          + LossRate.Base[B].Year[Y]
  END;  { CalcFreightCosts }

{*****************************************************************************
 *   Name           : CalcMovingCost
 *
 *   Purpose        : Orchestrates calculation of moving costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMovingCost (PersMoves, MovCostFinal)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   30Mar92   AML   Changed output (Commas, Titles, etc.)
 *   12May92   AML   Implemented Multi-Basing
 *****************************************************************************}
```

```
PROCEDURE CalcMovingCost (     PersMoves     : CalcMovesRec;
                          VAR MovCostFinal : YearAndBeyondRealRec;
                          VAR MovCostAlt   : PBase0ToMaxRealRec;
                          VAR PerDiemCiv,
                              POVMilesCiv,
                              HousPurch,
                              HHGCiv,
                              MiscCiv,
                              HousHunt,
                              RITA,
                              Pack,
                              Frat,
                              Vehic,
                              Driv,
                              Loss,
                              PerDiemMil,
                              POVMilesMil,
                              HHGMil,
                              MiscMil,
                              MilMov,
                              OneTimeMove  : Year1ToMaxRealRec;
                              CalcPersAct  : CalcPersActRec;
                              Adjust1000   : Year1ToMaxRealRec);

  VAR
    Buffer            : IOBuffer;
    DrivCost,
    FrateCost,
    HHGMilCost,
    HHGCivCost,
    HousHuntCost,
    HousPurchCost,
    LossRate,
    MilMoves,
    MiscMilCost,
    MiscCivCost,
    PackEtcCost,
    PerDiemMilCost,
    PerDiemCivCost,
    POVMilesMilCost,
    POVMilesCivCost,
    RITACost,
    VehicCost         : Base0Year1ToMaxRealRec;
    FrateCostFinal,
    PCSCostFinal      : Year1ToMaxRealRec;
    B, B2, Y          : INTEGER;
    F                 : TEXT;
    TotPCS,
    TotFrat           : Year1ToMaxRealRec;
```

```
Begin { CalcMovingCost }
  FillChar (MovCostAlt^, SizeOf (MovCostAlt^), 0);
  FillChar (OneTimeMove, SizeOf (OneTimeMove), 0);
  FillChar (PerDiemMil, SizeOf (PerDiemMil), 0);
  FillChar (PerDiemCiv, SizeOf (PerDiemCiv), 0);
  FillChar (POVMilesMil, SizeOf (POVMilesMil), 0);
  FillChar (POVMilesCiv, SizeOf (POVMilesCiv), 0);
  FillChar (HousPurch, SizeOf (HousPurch), 0);
  FillChar (HHGMil, SizeOf (HHGMil), 0);
  FillChar (HHGCiv, SizeOf (HHGCiv), 0);
  FillChar (MiscMil, SizeOf (MiscMil), 0);
  FillChar (MiscCiv, SizeOf (MiscCiv), 0);
  FillChar (HousHunt, SizeOf (HousHunt), 0);
  FillChar (RITA, SizeOf (RITA), 0);
  FillChar (MilMov, SizeOf (MilMov), 0);
  FillChar (TotPCS, SizeOf (TotPCS), 0);
  FillChar (Pack, SizeOf (Pack), 0);
  FillChar (Frat, SizeOf (Frat), 0);
  FillChar (Vehic, SizeOf (Vehic), 0);
  FillChar (Driv, SizeOf (Driv), 0);
  FillChar (Loss, SizeOf (Loss), 0);
  FillChar (TotFrat, SizeOf (TotFrat), 0);
  FillChar (MovCostFinal, SizeOf (MovCostFinal), 0);
  FillChar (MilMoves, SizeOf (MilMoves), 0);

  CalcPCSCosts      (PersMoves, PerDiemMilCost, PerDiemCivCost,
                     POVMilesMilCost, POVMilesCivCost, HousPurchCost,
                     HHGMilCost, HHGCivCost, MiscMilCost, MiscCivCost,
                     HousHuntCost, MilMoves, RITACost, CalcPersAct,
                     PCSCostFinal);
  CalcFreightCosts (PersMoves, PackEtcCost, CalcPersAct,
                     FrateCost, VehicCost, DrivCost, LossRate,
                     FrateCostFinal);

  { Calculate report entries }

  FOR Y := 1 to MaxYears DO
    Begin
      FOR B := 0 to MaxBases DO
        Begin
          PerDiemMil.Year[Y] :=
              PerDiemMil.Year[Y]
            + PerDiemMilCost.Base[B].Year[Y];
          PerDiemCiv.Year[Y] :=
              PerDiemCiv.Year[Y]
            + PerDiemCivCost.Base[B].Year[Y];
          POVMilesMil.Year[Y] :=
              POVMilesMil.Year[Y]
            + POVMilesMilCost.Base[B].Year[Y];
          POVMilesCiv.Year[Y] :=
              POVMilesCiv.Year[Y]
            + POVMilesCivCost.Base[B].Year[Y];
```

```
HousPurch.Year[Y] :=
    HousPurch.Year[Y]
  + HousPurchCost.Base[B].Year[Y];
HHGMil.Year[Y] :=
    HHGMil.Year[Y]
  + HHGMilCost.Base[B].Year[Y];
HHGCiv.Year[Y] :=
    HHGCiv.Year[Y]
  + HHGCivCost.Base[B].Year[Y];
MiscMil.Year[Y] :=
    MiscMil.Year[Y]
  + MiscMilCost.Base[B].Year[Y];
MiscCiv.Year[Y] :=
    MiscCiv.Year[Y]
  + MiscCivCost.Base[B].Year[Y];
HousHunt.Year[Y] :=
    HousHunt.Year[Y]
  + HousHuntCost.Base[B].Year[Y];
RITA.Year[Y] :=
    RITA.Year[Y]
  +((HousPurchCost.Base[B].Year[Y]
  + HousHuntCost.Base[B].Year[Y]
  + RITACost.Base[B].Year[Y]) * 0.28);
MilMov.Year[Y] :=
    MilMov.Year[Y]
  -(MilMoves.Base[B].Year[Y]
  * StdFctrsTrans.RoutinePCSCostsPerPerson
  / StdFctrsTrans.AverageTourLength);
Pack.Year[Y]  := Pack.Year[Y]  + PackEtcCost.Base[B].Year[Y];
Frat.Year[Y]  := Frat.Year[Y]  + FrateCost.Base[B].Year[Y];
Vehic.Year[Y] := Vehic.Year[Y] + VehicCost.Base[B].Year[Y];
Driv.Year[Y]  := Driv.Year[Y]  + DrivCost.Base[B].Year[Y];
Loss.Year[Y]  := Loss.Year[Y]  + LossRate.Base[B].Year[Y];
MovCostAlt^.Base[B] :=
    MovCostAlt^.Base[B]
  +(PerDiemMilCost.Base[B].Year[Y]
  + PerDiemCivCost.Base[B].Year[Y]
  + POVMilesMilCost.Base[B].Year[Y]
  + POVMilesCivCost.Base[B].Year[Y]
  + HousPurchCost.Base[B].Year[Y]
  + HHGMilCost.Base[B].Year[Y]
  + HHGCivCost.Base[B].Year[Y]
  + MiscMilCost.Base[B].Year[Y]
  + MiscCivCost.Base[B].Year[Y]
  + HousHuntCost.Base[B].Year[Y]
  +((HousPurchCost.Base[B].Year[Y]
  + HousHuntCost.Base[B].Year[Y]
  + RITACost.Base[B].Year[Y]) * 0.28)
  -(MilMoves.Base[B].Year[Y]
  * StdFctrsTrans.RoutinePCSCostsPerPerson
  / StdFctrsTrans.AverageTourLength)
  + PackEtcCost.Base[B].Year[Y]
```

```
            + FrateCost.Base[B].Year[Y]
            + VehicCost.Base[B].Year[Y]
            + DrivCost.Base[B].Year[Y]
            + LossRate.Base[B].Year[Y])
            * Adjust1000.Year[Y]
            * 1000.0
            + ActionEnvir^.Base[B].OneTimeMovingCosts.Year[Y];
          OneTimeMove.Year[Y] := OneTimeMove.Year[Y]
            + ActionEnvir^.Base[B].OneTimeMovingCosts.Year[Y];
        END;
      TotPCS.Year[Y] :=
          TotPCS.Year[Y]
        + PerDiemMil.Year[Y]
        + PerDiemCiv.Year[Y]
        + POVMilesMil.Year[Y]
        + POVMilesCiv.Year[Y]
        + HousPurch.Year[Y]
        + HHGMil.Year[Y]
        + HHGCiv.Year[Y]
        + MiscMil.Year[Y]
        + MiscCiv.Year[Y]
        + HousHunt.Year[Y]
        + RITA.Year[Y]
        + MilMov.Year[Y];
      TotFrat.Year[Y]    :=
          TotFrat.Year[Y]
        + Pack.Year[Y]
        + Frat.Year[Y]
        + Vehic.Year[Y]
        + Driv.Year[Y]
        + Loss.Year[Y];
      MovCostFinal.Year[Y] :=
          TotPCS.Year[Y]
        + TotFrat.Year[Y]
        + OneTimeMove.Year[Y];
    END;

  assign (F, Info.ReportDir + 'MOVING.RPT');
  SetTextBuf (F, Buffer);
  rewrite (F);
  Title (F, 'MOVING COSTS', -1);
  writeln (F, Blank:10, 'Yearly Cost Breakout ($K)');
  writeln (F);
  FiscOut (F, 22, 8, 0);
  writeln (F, BLANK:10,
    'PCS COSTS      -----   -----   -----   -----   -----   -----');
  YearOut (F, 'Mil Per Diem', PerDiemMil, Adjust1000, 8, 0);
  YearOut (F, 'Civ Per Diem', PerDiemCiv, Adjust1000, 8, 0);
  YearOut (F, 'Mil POV mi  ', POVMilesMil, Adjust1000, 8, 0);
  YearOut (F, 'Civ POV mi  ', POVMilesCiv, Adjust1000, 8, 0);
  YearOut (F, 'House Purch ', HousPurch, Adjust1000, 8, 0);
  YearOut (F, 'Mil HHG Cost', HHGMil, Adjust1000, 8, 0);
```

```
    YearOut (F, 'Civ HHG Cost', HHGCiv, Adjust1000, 8, 0);
    YearOut (F, 'Mil Misc    ', MiscMil, Adjust1000, 8, 0);
    YearOut (F, 'Civ Misc    ', MiscCiv, Adjust1000, 8, 0);
    YearOut (F, 'House Hunt  ', HousHunt, Adjust1000, 8, 0);
    YearOut (F, 'RITA Costs  ', RITA, Adjust1000, 8, 0);
    YearOut (F, 'Mil Moves   ', MilMov, Adjust1000, 8, 0);
    writeln (F);
    YearOut (F, 'Subtotal    ', TotPCS, Adjust1000, 8, 0);
    writeln (F);
    writeln (F, Blank:10, 'FREIGHT COSTS');
    YearOut (F, 'Packing     ', Pack, Adjust1000, 8, 0);
    YearOut (F, 'Freight     ', Frat, Adjust1000, 8, 0);
    YearOut (F, 'Vehicle     ', Vehic, Adjust1000, 8, 0);
    YearOut (F, 'Driving     ', Driv, Adjust1000, 8, 0);
    YearOut (F, 'Loss Rate   ', Loss, Adjust1000, 8, 0);
    writeln (F);
    YearOut (F, 'Subtotal    ', TotFrat, Adjust1000, 8, 0);
    writeln (F);
    writeln (F, Blank:10, 'OTHER COSTS');
    YearOut (F, '1Time Moving', OneTimeMove, Adjust1000, 8, 0);
    writeln (F);
    BeyondOut (F, 'TOTAL       ', MovCostFinal, Adjust1000, 8, 0);
    close (F)
   END;  { CalcMovingCost }
END. { UNIT }
```

```
UNIT CCALOTH;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE CalcOtherCost (     PersMovesSum      : PersMovesSumRec;
                              MilConDetail      : MilConRec;
                              CalcPersAct       : CalcPersActRec;
                              PersMoves         : CalcMovesRec;
                              CalcFctrsPerPos   : CalcPersPosRec;
                          VAR OtherCostFinal    : CostSaveNetByYearRec;
                          VAR NewHireCost       : Year1ToMaxRealRec;
                          VAR LandCost          : Year1ToMaxRealRec;
                          VAR LandBuys          : Year1ToMaxRealRec;
                          VAR EnvirCost         : Year1ToMaxRealRec;
                          VAR HomeownAsstCost   : Year1ToMaxRealRec;
                          VAR UnemploymentCost  : Year1ToMaxRealRec;
                          VAR IMACost           : Year1ToMaxRealRec;
                          VAR PPSCost           : Year1ToMaxRealRec;
                          VAR Champ             : CostSaveNetByYearRec;
                              Adjust1000        : Year1ToMaxRealRec);

IMPLEMENTATION
{*****************************************************************************
 *   Name          : CalcChampusCost
 *
 *   Purpose       : Calculates CHAMPUS costs by year
 *
 *   Notes         :
 *
 *   Invoked As    : CalcChampusCost (ChampusCost)
 *
 *   Return value  :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   14May92   AML   Implemented Multi-Basing
 *****************************************************************************}
```

```
PROCEDURE CalcChampusCost (VAR Champ : CostSaveNetByYearRec);

  VAR
    B, Y : INTEGER;

  Begin { CalcChampusCost }
    FillChar (Champ, SizeOf (Champ), 0);
    For B := 0 To MaxBases Do
      For Y := 1 To MaxYears Do
        Begin
          If Champus.Base[B].OnBaseInPatVisits.Year[Y] > 0
            Then Champ.Save.Year[Y] :=
                 Champ.Save.Year[Y]
               +(Champus.Base[B].OnBaseInPatVisits.Year[Y]
               * Champus.Base[B].ChampInPatCostVisit)
               *(1.0
               - ActionEnvir^.Base[B].ChampusToMedicare)
            Else Champ.Cost.Year[Y] :=
                 Champ.Cost.Year[Y]
               -(Champus.Base[B].OnBaseInPatVisits.Year[Y]
               * Champus.Base[B].ChampInPatCostVisit)
               *(1.0
               - ActionEnvir^.Base[B].ChampusToMedicare);
          If Champus.Base[B].OnBaseOutPatVisits.Year[Y] > 0
            Then Champ.Save.Year[Y] :=
                 Champ.Save.Year[Y]
               +(Champus.Base[B].OnBaseOutPatVisits.Year[Y]
               * Champus.Base[B].ChampOutPatCostVisit)
               *(1.0
               - ActionEnvir^.Base[B].ChampusToMedicare)
            Else Champ.Cost.Year[Y] :=
                 Champ.Cost.Year[Y]
               -(Champus.Base[B].OnBaseOutPatVisits.Year[Y]
               * Champus.Base[B].ChampOutPatCostVisit)
               *(1.0
               - ActionEnvir^.Base[B].ChampusToMedicare);
        End;
    For Y := 1 To MaxYears Do
      Champ.Net.Year[Y] :=
          Champ.Cost.Year[Y]
        - Champ.Save.Year[Y];
  END;  { CalcChampusCost }
```

```
{*************************************************************************
 *   Name           : CalcNewHireCost
 *
 *   Purpose        : Calculates new hire costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcNewHireCost (CalcPersAct, NewHireCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcNewHireCost (     CalcPersAct : CalcPersActRec;
                            VAR NewHireCost : Year1ToMaxRealRec);

  VAR
    Y : INTEGER;

  Begin { CalcNewHireCost }
    FillChar (NewHireCost, SizeOf (NewHireCost), 0);

    FOR Y := 1 to MaxYears DO
      Begin
        NewHireCost.Year[Y] :=
           (CalcPersAct.Year[Y].NumNeeded
         - CalcPersAct.Year[Y].NumMoving)
         * StdFctrsPers.NewHireCost;
      END
  END;  { CalcNewHireCost }
```

```
{*****************************************************************************
 *   Name           : CalcLandCost
 *
 *   Purpose        : Calculates land costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcLandCost (LandCost, LandBuys)
 *
 *   Return values :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     -----------------      -------   ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcLandCost (VAR LandCost : Year1ToMaxRealRec;
                        VAR LandBuys : Year1ToMaxRealRec);

  VAR
    B, Y       : INTEGER;
    LandSales : Year1ToMaxRealRec;

  Begin { CalcLandCost }
    FillChar (LandCost, SizeOf (LandCost), 0);
    FillChar (LandBuys, SizeOf (LandBuys), 0);
    FillChar (LandSales, SizeOf (LandSales), 0);
    For B := 0 To MaxBases Do
      For Y := 1 To MaxYears Do
        If RealPropProc.RealPropPur.Base[B].Cost.Year[Y] < 0
          Then LandSales.Year[Y] :=
               LandSales.Year[Y]
             + RealPropProc.RealPropPur.Base[B].Cost.Year[Y]
          Else LandBuys.Year[Y] :=
               LandBuys.Year[Y]
             + RealPropProc.RealPropPur.Base[B].Cost.Year[Y];
    FOR Y := 1 to MaxYears DO
      LandCost.Year[Y] := LandSales.Year[Y] + LandBuys.Year[Y]
  END;  { CalcLandCost }
```

```
{*****************************************************************************
 *   Name           : CalcEnvirCost
 *
 *   Purpose        : Calculates environmental costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcEnvirCost (EnvirCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcEnvirCost (VAR EnvirCost : Year1ToMaxRealRec);

  VAR
    B, Y : INTEGER;

  Begin { CalcEnvirCost }
    FillChar (EnvirCost, SizeOf (EnvirCost), 0);
    For Y := 1 To MaxYears Do
      For B := 0 To MaxBases Do
        EnvirCost.Year[Y] := EnvirCost.Year[Y]
          + ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y]
          + ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[Y];
   END;  { CalcEnvirCost }
```

```
{*************************************************************************
 *   Name           : CalcHomeownAsstCost
 *
 *   Purpose        : Calculates homeowners assistance costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcHomeownAsstCost (CalcPersAct, HomeownAsstCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   15May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcHomeownAsstCost (     PersMoves        : CalcMovesRec;
                                VAR HomeownAsstCost : Year1ToMaxRealRec);

  VAR
    B, B2, Y : INTEGER;
    Off,
    Enl,
    Civ      : REAL;

  Begin { CalcHomeownAsstCost }
    FillChar (HomeOwnAsstCost, SizeOf (HomeOwnAsstCost), 0);
    FOR Y := 1 to MaxYears DO
      Begin
        Off := 0.0;
        Enl := 0.0;
        Civ := 0.0;
        For B := 0 to MaxBases Do
          If HapSwitch.Base[B] = 'Y' Then
            Begin
              Off := Off
                + PosElim.Base[B].Off.Year[Y]
                * StdFctrsPers.PercOffMarried
                *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
                * StdFctrsPers.CivHomeOwnRate
                * Con.Base[B].AreaCostFactor
                * StdFctrsPers.HAPFactor
                * StdFctrsPers.HAPReceive
```

```
        * StdFctrsPers.NatlAvgHomePrice;
      Enl := Enl
        + PosElim.Base[B].Enl.Year[Y]
        * StdFctrsPers.PercEnlMarried
        *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
        * StdFctrsPers.CivHomeOwnRate
        * Con.Base[B].AreaCostFactor
        * StdFctrsPers.HAPFactor
        * StdFctrsPers.HAPReceive
        * StdFctrsPers.NatlAvgHomePrice;
      Civ := Civ
        + PosElim.Base[B].Civ.Year[Y]
        * StdFctrsPers.CivHomeOwnRate
        * Con.Base[B].AreaCostFactor
        * StdFctrsPers.HAPFactor
        * StdFctrsPers.HAPReceive
        * StdFctrsPers.NatlAvgHomePrice;
      For B2 := 0 To MaxBases Do
        If Distance^.Base[B].Base[B2] >= 50.0 Then
          Begin
            Off := Off
              + PosTrans.Base[B].Base[B2]^.Off.Year[Y]
              * StdFctrsPers.PercOffMarried
              *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
              * StdFctrsPers.CivHomeOwnRate
              * Con.Base[B].AreaCostFactor
              * StdFctrsPers.HAPFactor
              * StdFctrsPers.HAPReceive
              * StdFctrsPers.NatlAvgHomePrice;
            Enl := Enl
              + PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
              * StdFctrsPers.PercEnlMarried
              *(1.0 - ActionEnvir^.Base[B].PercFamLivOnBase)
              * StdFctrsPers.CivHomeOwnRate
              * Con.Base[B].AreaCostFactor
              * StdFctrsPers.HAPFactor
              * StdFctrsPers.HAPReceive
              * StdFctrsPers.NatlAvgHomePrice;
            Civ := Civ
              + PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              * StdFctrsPers.CivHomeOwnRate
              * Con.Base[B].AreaCostFactor
              * StdFctrsPers.HAPFactor
              * StdFctrsPers.HAPReceive
              * StdFctrsPers.NatlAvgHomePrice;
          End
    End
  Else
    Begin
      Civ := Civ
        + PosElim.Base[B].Civ.Year[Y]
        * StdFctrsPers.CivHomeOwnRate
```

```
                * Con.Base[B].AreaCostFactor
                * StdFctrsPers.DARSEFactor
                * StdFctrsPers.DARSEReceive
                * StdFctrsPers.NatlAvgHomePrice;
              For B2 := 0 To MaxBases Do
                If Distance^.Base[B].Base[B2] >= 50.0
                  Then Civ := Civ
                    + PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
                    * StdFctrsPers.CivHomeOwnRate
                    * Con.Base[B].AreaCostFactor
                    * StdFctrsPers.DARSEFactor
                    * StdFctrsPers.DARSEReceive
                    * StdFctrsPers.NatlAvgHomePrice;
            End;
        HomeOwnAsstCost.Year[Y] := Off + Enl + Civ;
      END;
  END;  { CalcHomeownAsstCost }

{*****************************************************************************
 *   Name          : CalcUnemploymentCost
 *
 *   Purpose       : Calculates unemployment costs by year
 *
 *   Notes         :
 *
 *   Invoked As    : CalcUnemploymentCost (CalcFctrsPerPos, CalcPersAct,
 *                                         UnemploymentCost)
 *
 *   Return value  :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   15May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcUnemploymentCost (    CalcFctrsPerPos  : CalcPersPosRec;
                                    CalcPersAct      : CalcPersActRec;
                                VAR UnemploymentCost : Year1ToMaxRealRec);

  VAR
    Y : INTEGER;
```

```
  Begin { CalcUnemploymentCost }
    FillChar (UnemploymentCost, SizeOf (UnemploymentCost), 0);
    FOR Y := 1 to MaxYears DO
      UnemploymentCost.Year[Y] :=
          CalcPersAct.Year[Y].NumRIFted
        * StdFctrsPers.AvgUnemployCostPerWeek
        * StdFctrsPers.UnemployEligible;
  END;  { CalcUnemploymentCost }

{*****************************************************************************
 *   Name           : CalcPPSCost
 *
 *   Purpose        : Calculates PPS costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcPPSCost (CalcPersAct, PPSCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcPPSCost (    CalcPersAct : CalcPersActRec;
                       VAR PPSCost     : Year1ToMaxRealRec);

  VAR
    Y : INTEGER;

  Begin { CalcPPSCost }
    FillChar (PPSCost, SizeOf (PPSCost), 0);
    FOR Y := 1 to MaxYears DO
      PPSCost.Year[Y] :=
         (CalcPersAct.Year[Y].TotalPPS
        - CalcPersAct.Year[Y].NumAttrited)
        * StdFctrsPers.CivPCSCosts
  END;  { CalcPPSCost }
```

```
{*****************************************************************************
 *   Name           : CalcIMACost
 *
 *   Purpose        : Calculates IMA costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcIMACost (PersMovesSum, MilConDetail, IMACost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcIMACost (     PersMovesSum : PersMovesSumRec;
                            MilConDetail : MilConRec;
                        VAR IMACost      : Year1ToMaxRealRec);

  VAR
    B, P, Y : INTEGER;
    Ima     : REAL;

  Begin { CalcIMACost }
    FillChar (IMACost, SizeOf (IMACost), 0);
    IMA := 0.0;
    FOR B := 0 to MaxBases DO
      Begin
        IMA := 0.0;
        For P := 1 to 16 Do
          If ((Con.Base[B].Category[P]  > 'B') And
              (Con.Base[B].Category[P] <> 'P') And
              (Con.Base[B].Category[P] <  'S'))
            Then IMA := IMA + MilConDetail.Base[B].NewCost[P]
          Else If Con.Base[B].Category[P] > 'R'
            Then If StdFctrsCon.Units[Ord (Con.Base[B].Category[P]) - 83] = 'SF'
              Then IMA := IMA + MilConDetail.Base[B].NewCost[P];
        For Y := 1 To MaxYears Do
          If Screen1Data^.ModelDoShutDn = 'Y'
            Then IMACost.Year[Y] :=
                 IMACost.Year[Y]
               +(IMA
```

```
                * PersMovesSum.Base[B].Perc.Year[Y]
                * StdFctrsFac.IMAPerc)
            Else IMACost.Year[Y] :=
                 IMACost.Year[Y]
               +(IMA
                * TimePhasShutDn.Base[B].Constr.Year[Y]
                * StdFctrsFac.IMAPerc)
      End;
  END;  { CalcIMACost }

{*************************************************************************
 *   Name           : CalcOtherCost
 *
 *   Purpose        : Orchestrates calculation of other costs by year
 *
 *   Notes          :
 *
 *   Invoked As     :
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   30Mar92  AML   Changed output (Commas, Titles, etc.)
 *   15May92  AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcOtherCost (     PersMovesSum      : PersMovesSumRec;
                              MilConDetail      : MilConRec;
                              CalcPersAct       : CalcPersActRec;
                              PersMoves         : CalcMovesRec;
                              CalcFctrsPerPos   : CalcPersPosRec;
                          VAR OtherCostFinal    : CostSaveNetByYearRec;
                          VAR NewHireCost       : Year1ToMaxRealRec;
                          VAR LandCost          : Year1ToMaxRealRec;
                          VAR LandBuys          : Year1ToMaxRealRec;
                          VAR EnvirCost         : Year1ToMaxRealRec;
                          VAR HomeownAsstCost   : Year1ToMaxRealRec;
                          VAR UnemploymentCost  : Year1ToMaxRealRec;
                          VAR IMACost           : Year1ToMaxRealRec;
                          VAR PPSCost           : Year1ToMaxRealRec;
                          VAR Champ             : CostSaveNetByYearRec;
                              Adjust1000        : Year1ToMaxRealRec);
```

```
VAR
  Buffer : IOBuffer;
  B, Y   : INTEGER;
  F      : TEXT;
  Proc   : Year1ToMaxRealRec;

Begin { CalcOtherCost }
  FillChar (OtherCostFinal, SizeOf (OtherCostFinal), 0);
  FillChar (Proc, SizeOf (Proc), 0);
  CalcChampusCost       (Champ);
  CalcNewHireCost       (CalcPersAct, NewHireCost);
  CalcLandCost          (LandCost, LandBuys);
  CalcEnvirCost         (EnvirCost);
  CalcHomeownAsstCost   (PersMoves, HomeownAsstCost);
  CalcUnemploymentCost  (CalcFctrsPerPos, CalcPersAct, UnemploymentCost);
  CalcPPSCost           (CalcPersAct, PPSCost);
  CalcIMACost           (PersMovesSum, MilConDetail, IMACost);
  For B := 0 To MaxBases Do
    For Y := 1 To MaxYears Do
      Proc.Year[Y] := Proc.Year[Y] + CostAvoid.Base[B].Procure.Year[Y];
  FOR Y := 1 to MaxYears DO
    Begin
      OtherCostFinal.Cost.Year[Y] :=
          Champ.Net.Year[Y]
        + NewHireCost.Year[Y]
        + CalcPersAct.Year[Y].RIFPay
        + CalcPersAct.Year[Y].EarlyRetire
        + LandBuys.Year[Y]
        + EnvirCost.Year[Y]
        + HomeownAsstCost.Year[Y]
        + UnemploymentCost.Year[Y]
        + PPSCost.Year[Y]
        + IMACost.Year[Y];
      OtherCostFinal.Save.Year[Y] :=
          LandBuys.Year[Y]
        - LandCost.Year[Y]
        + Proc.Year[Y];
      OtherCostFinal.Net.Year[Y] :=
          OtherCostFinal.Cost.Year[Y]
        - OtherCostFinal.Save.Year[Y];
    End;
  OtherCostFinal.Cost.Beyond :=
      Champ.Cost.Year[MaxYears];
  OtherCostFinal.Save.Beyond :=
      Champ.Save.Year[MaxYears]
    + Proc.Year[MaxYears];
  OtherCostFinal.Net.Beyond :=
      OtherCostFinal.Cost.Beyond
    - OtherCostFinal.Save.Beyond;

  assign (F, Info.ReportDir + 'OTHER.RPT');
  SetTextBuf (F, Buffer);
```

```
    rewrite (F);
    Title (F, 'OTHER COSTS', -1);
    writeln (F, BLANK:10, 'Yearly Cost Breakout ($K)');
    writeln (F);
    FiscOut (F, 22, 8, 0);
    writeln (F, BLANK:22, Rept('   -----', 6));
    BeyondOut (F, 'CHAMPUS*    ', Champ.Net, Adjust1000, 8, 0);
    YearOut (F, 'New Hires   ', NewHireCost, Adjust1000, 8, 0);
    Write (F, Blank:10, 'RIF Pay     ');
    FOR Y := 1 to MaxYears DO
       write (F, WithCommas(CalcPersAct.Year[Y].RIFPay
                            * Adjust1000.Year[Y],0):8);
    writeln (F);
    write (F, BLANK:10, 'Early Retire');
    FOR Y := 1 to MaxYears DO
       write (F, WithCommas(CalcPersAct.Year[Y].EarlyRetire
                            * Adjust1000.Year[Y],0):8);
    writeln (F);
    write (F, BLANK:10, 'Land Sales  ');
    FOR Y := 1 to MaxYears DO
       write (F, WithCommas((LandCost.Year[Y] - LandBuys.Year[Y])
                             * Adjust1000.Year[Y],0):8);
    writeln (F);
    YearOut (F, 'Land Buys   ', LandBuys, Adjust1000, 8, 0);
    YearOut (F, 'Environmentl', EnvirCost, Adjust1000, 8, 0);
    YearOut (F, 'HAP / RSE   ', HomeownAsstCost, Adjust1000, 8, 0);
    YearOut (F, 'Unemployment', UnemploymentCost, Adjust1000, 8, 0);
    YearOut (F, 'PPS         ', PPSCost, Adjust1000, 8, 0);
    YearOut (F, 'IMA         ', IMACost, Adjust1000, 8, 0);
    write (F, BLANK:10, 'Proc Avoid* ');
    FOR Y := 1 to MaxYears DO
       write (F, WithCommas(-Proc.Year[Y] * Adjust1000.Year[Y],0):8);
    writeln (F);
    writeln (F, Rept('-', 60):70);
    BeyondOut (F, 'TOTAL       ', OtherCostFinal.Net, Adjust1000, 8, 0);
    writeln (F);
    writeln (F, Blank:10,
      '* Procurement Cost Avoidances and CHAMPUS costs continue');
    writeln (F, Blank:12, 'into the Beyond years.');
    close (F)
  END;  { CalcOtherCost }

END. { UNIT }
```

```
UNIT CCALOVR;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE CalcOverheadCost (     PersMovesSum       : PersMovesSumRec;
                                 MilConDetail       : MilConRec;
                             VAR OvhdCostFinal      : CostSaveNetByYearRec;
                                                                    V   A   R
AdminSuppCost        : Base0Year1ToMaxRealRec;
                             VAR MothballCost       : Year1ToMaxRealRec;
                             VAR DeltaChart         : DeltaChartRec;
                             VAR RevisedRPMA        : Base0Year1ToMaxRealRec;
                             VAR RevisedBOS         : Base0YearAndBeyondRealRec;
                             VAR TotActualRPMA      : REAL;
                             VAR TotActualBOS       : Year1ToMaxRealRec;
                             VAR TotRevisedRPMA     : Year1ToMaxRealRec;
                             VAR TotRevisedBOS      : YearAndBeyondRealRec;
                             VAR CaretakerCost      : Year1ToMaxRealRec;
                             VAR MaintainSpaceCost  : Year1ToMaxRealRec;
                             VAR Housing            : CostSaveNetByYearRec;
                                 Adjust             : Year1ToMaxRealRec);

IMPLEMENTATION
```

```
{*****************************************************************************
 *   Name           : CalcAdminSuppCost
 *
 *   Purpose        : Calculates administrative support costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcAdminSuppCost (AdminSuppCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcAdminSuppCost (VAR AdminSuppCost : Base0Year1ToMaxRealRec);

  VAR
    B, B2, Y, Y2    : Integer;
    TempCost,
    TempMove        : Real;
    UserEntryError  : Word;

  Begin { CalcAdminSuppCost }
    FillChar (AdminSuppCost, SizeOf (AdminSuppCost), 0);
    UserEntryError := 0;

    For B := 0 To MaxBases Do
      If IndustFundSwitch.Base[B] = 'Y'
        Then For Y := 1 To MaxYears Do
                         AdminSuppCost.Base[B].Year[Y] :=
                              IndustFund.Base[B]^.OvhdCost.AdminPlan.Year[Y]
      Else
        Begin
          Y2 := 0;
          For Y := 1 To MaxYears Do
            Begin
              TempMove := 0.0;
              For B2 := 0 To MaxBases Do
                If Distance^.Base[B].Base[B2] > 0 Then
```

```
                    TempMove := TempMove
                      + PosTrans.Base[B].Base[B2]^.Off.Year[Y]
                      + PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
                      + PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
                      + PosTrans.Base[B].Base[B2]^.Stu.Year[Y]
                      + PosElim.Base[B].Off.Year[Y]
                      + PosElim.Base[B].Enl.Year[Y]
                      + PosElim.Base[B].Civ.Year[Y]
                      + ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y]
                      + ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y]
                      + ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y]
                      + ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y];
                  If Abs(TempMove) > 0.001
                    Then Y2 := Y;
                End;
              If (Screen1Data^.Base[B].CloseYear < Y2) And
                 (Screen1Data^.Base[B].CloseYear > 0) And
                 Not Odd(UserEntryError Div Round(YtoX(2,B)))
                Then UserEntryError := UserEntryError + Round(YtoX(2,B));
              If Y2 > 0 Then
                Begin
                  TempCost :=
                     (BudgetPropData^.Base[B].BaseOpnsNonPayBudget
                    + BudgetPropData^.Base[B].BaseOpnsPay)
                    * StdFctrsFac.SuppToMoveFctr;
                  For Y := 1 To Y2 Do
                                                AdminSuppCost.Base[B].Year[Y] := TempCost
* YtoX (0.75, Y - 1);
                End;
            End;
      If UserEntryError > 0 Then
        Begin
          ErrorOut ('The following base(s) moved out personnel, equipment,');
          ErrorOut ('and/or vehicles after being closed or deactivated:');
          ErrorList (UserEntryError);
        End;
    END;  { CalcAdminSuppCost }
```

```
{*****************************************************************************
 *   Name           : CalcRPMACostFctr
 *
 *   Purpose        : Calculates RPMA cost factors by losing and gaining bases
 *
 *   Notes          :
 *
 *   Invoked As     : CalcRPMACostFctr (RPMACostFctr)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcRPMACostFctr (VAR RPMACostFctr : Base0ToMaxRealRec);

  VAR
    B : INTEGER;

  Begin { CalcRPMACostFctr }
    FillChar (RPMACostFctr, SizeOf (RPMACostFctr), 0);

    FOR B := 0 to MaxBases DO
        RPMACostFctr.Base[B] := ZeroCheck(
                          BudgetPropData^.Base[B].RPMANonPayBudget,
                          YtoX (ActionEnvir^.Base[B].BaseFacilSqFt,
                   StdFctrsFac.RPMACostBldgPower));
   END;  { CalcRPMACostFctr }
```

```
{*****************************************************************************
 *   Name           : CalcRevisedRPMA
 *
 *   Purpose        : Calculates revised RPMA costs by losing and gaining bases
 *                    by year
 *
 *    Notes                     : Even though acreage is no longer considered in
RPMA
 *                                                calculations, this procedure still
adds up acreage bought
 *                                                and sold for the error checking.
 *                  : Only SF MilCon figures used, excluding FamQtrs
 *
 *   Invoked As     : CalcRevisedRPMA (PersMovesSum, MilConDetail, RPMACostFctr,
 *                                     RevisedRPMA)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcRevisedRPMA (     PersMovesSum : PersMovesSumRec;
                                MilConDetail : MilConRec;
                                RPMACostFctr : Base0ToMaxRealRec;
                            VAR RevisedRPMA  : Base0Year1ToMaxRealRec);

  VAR
    B, P, Y, Y2        : INTEGER;
          A1Base, A2Base,
          CumulConPerc,
    CumulShutPerc      : Real;
    UserEntryError,
    UserEntryError2,
    UserEntryError3    : Word;

  Begin { CalcRevisedRPMA }
    FillChar (RevisedRPMA, SizeOf (RevisedRPMA), 0);
    UserEntryError := 0;
    UserEntryError2 := 0;
    UserEntryError3 := 0;
```

```
FOR B := 0 to MaxBases Do
  Begin
    A1Base := ActionEnvir^.Base[B].AcreOnBase;
    CumulConPerc := 0.0;
    CumulShutPerc := 0.0;
    FOR Y := 1 to MaxYears Do
      Begin
        CumulConPerc := 0.0;
        CumulShutPerc := 0.0;
        For Y2 := 1 To Y Do
          If Screen1Data^.ModelDoShutDn = 'Y' Then
            Begin
              CumulConPerc := CumulConPerc
                + PersMovesSum.Base[B].Perc.Year[Y2];
              CumulShutPerc := CumulShutPerc
                + PersMovesSum.Base[B].ShutDn.Year[Y2];
            End
          Else
            Begin
              CumulConPerc := CumulConPerc
                + TimePhasShutDn.Base[B].Constr.Year[Y2];
              CumulShutPerc := CumulShutPerc
                + TimePhasShutDn.Base[B].ShutDn.Year[Y2];
            End;
        If Screen1Data^.ModelDoShutDn = 'Y'
          Then CumulShutPerc := CumulShutPerc
            - (PersMovesSum.Base[B].ShutDn.Year[Y] * 0.5)
          Else CumulShutPerc := CumulShutPerc
            - (TimePhasShutDn.Base[B].ShutDn.Year[Y] * 0.5);
        A1Base := A1Base + RealPropProc.RealPropPur.Base[B].Acre.Year[Y];
        If (A1Base < 0) And Not Odd(UserEntryError Div Round(YtoX(2,B)))
          Then UserEntryError := UserEntryError + Round(YtoX(2,B));
        A2Base := ActionEnvir^.Base[B].BaseFacilSqFt
          -(RealPropProc.Base[B].FacilSqFtShutDown
          * CumulShutPerc);
        If Y > 2
          Then For P := 1 to 16 Do
            If ((Con.Base[B].Category[P]  > 'B') And
                (Con.Base[B].Category[P] <> 'H') And
                (Con.Base[B].Category[P] <> 'I') And
                (Con.Base[B].Category[P] <> 'P') And
                (Con.Base[B].Category[P] <  'S'))
              Then A2Base := A2Base + Con.Base[B].New[P] * CumulConPerc
            Else If (Con.Base[B].Category[P] = 'H') And
                    (StdFctrsCon.Units[-2] = 'SF')
              Then A2Base := A2Base + Con.Base[B].New[P] * CumulConPerc
            Else If (Con.Base[B].Category[P] = 'H') And
                    (StdFctrsCon.Units[-2] <> 'SF')
              Then A2Base := A2Base + Con.Base[B].New[P] * CumulConPerc
                * StdFctrsFac.AvgBachQtrs
```

```
            Else If Con.Base[B].Category[P] > 'R'
              Then If StdFctrsCon.Units[Ord (Con.Base[B].Category[P]) - 83]
                = 'SF' Then
                  A2Base := A2Base + Con.Base[B].New[P] * CumulConPerc;
          If (A2Base < 0) And Not Odd(UserEntryError2 Div Round(YtoX(2,B)))
            Then UserEntryError2 := UserEntryError2 + Round(YtoX(2,B));
          If (A2Base > 0.01) And
             (Y > Screen1Data^.Base[B].CloseYear) And
             (Screen1Data^.Base[B].CloseYear > 0) And
             (Screen1Data^.Base[B].Deactivate <> 'Y') And
              Not Odd(UserEntryError3 Div Round(YtoX(2,B)))
            Then UserEntryError3 := UserEntryError3 + Round(YtoX(2,B));
                          RevisedRPMA.Base[B].Year[Y] := RPMACostFctr.Base[B]
                               * YtoX (A2Base, StdFctrsFac.RPMACostBldgPower);
        END
    END;
  If UserEntryError > 0 Then
    Begin
      ErrorOut ('The following base(s) sold more acreage than they had:');
      ErrorList (UserEntryError);
    End;
  If UserEntryError2 > 0 Then
    Begin
      ErrorOut ('The following base(s) shut down more SF than they had:');
      ErrorList (UserEntryError2);
    End;
  If UserEntryError3 > 0 Then
    Begin
      ErrorOut
        ('The following base(s) still had square footage after they closed:');
      ErrorList (UserEntryError3);
    End;
END;  { CalcRevisedRPMA }
```

```
{*****************************************************************************
 *   Name           : CalcRevisedBOS
 *
 *   Purpose        : Calculates revised BOS by losing and gaining bases by year
 *
 *   Notes          : RevisedBOS for Beyond years was previously assumed to be
 *                    the same as Year 6 BOS.  BOS now double-counts troops in
 *                    the year they move, so Beyond-year BOS must be calculated
 *                    separately (in case there are any Year 6 moves).  All
 *                    other BOS functions adjusted accordingly.
 *
 *   Invoked As     : CalcRevisedBOS (RevisedBOS)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *   27Jul92   AML   BOS Phase-Out adjustments
 *****************************************************************************}

PROCEDURE CalcRevisedBOS (VAR RevisedBOS     : Base0YearAndBeyondRealRec);

  VAR
    B, B2, Y          : INTEGER;
    BasePopulation,
    TotMilAndCiv      : REAL;

  Begin { CalcRevisedBOS }
    FillChar (RevisedBOS, SizeOf (RevisedBOS), 0);

    FOR B := 0 to MaxBases Do
      Begin
        TotMilAndCiv := ActionEnvir^.Base[B].BaseTotOffEmploy
                      + ActionEnvir^.Base[B].BaseTotEnlEmploy
                      + ActionEnvir^.Base[B].BaseTotStuEmploy
                      + ActionEnvir^.Base[B].BaseTotCivEmploy;
        BasePopulation := TotMilAndCiv;
        FOR Y := 1 to MaxYears Do
          Begin
            BasePopulation :=
                BasePopulation
              - PosReduc.Base[B].Off.Year[Y]
```

```
        - PosReduc.Base[B].Enl.Year[Y]
        - PosReduc.Base[B].Civ.Year[Y];
      TotMilAndCiv :=
          TotMilAndCiv
        - PosReduc.Base[B].Off.Year[Y]
        - PosReduc.Base[B].Enl.Year[Y]
        - PosReduc.Base[B].Civ.Year[Y]
        - PosElim.Base[B].Off.Year[Y]
        - PosElim.Base[B].Enl.Year[Y]
        - PosElim.Base[B].Civ.Year[Y];
      For B2 := 0 To MaxBases Do
        If Distance^.Base[B].Base[B2] <> 0 Then
          Begin
            TotMilAndCiv :=
                TotMilAndCiv
              + PosTrans.Base[B2].Base[B]^.Off.Year[Y]
              + PosTrans.Base[B2].Base[B]^.Enl.Year[Y]
              + PosTrans.Base[B2].Base[B]^.Civ.Year[Y]
              + PosTrans.Base[B2].Base[B]^.Stu.Year[Y];
            If Y > 1 Then
              TotMilAndCiv :=
                  TotMilAndCiv
                - PosTrans.Base[B].Base[B2]^.Off.Year[Y-1]
                - PosTrans.Base[B].Base[B2]^.Enl.Year[Y-1]
                - PosTrans.Base[B].Base[B2]^.Civ.Year[Y-1]
                - PosTrans.Base[B].Base[B2]^.Stu.Year[Y-1];
          End;
      RevisedBOS.Base[B].Year[Y] :=
         (BudgetPropData^.Base[B].CommunBudget
        + BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
        * ZeroCheck
         (BasePopulation,
          ActionEnvir^.Base[B].BaseTotOffEmploy
        + ActionEnvir^.Base[B].BaseTotEnlEmploy
        + ActionEnvir^.Base[B].BaseTotStuEmploy
        + ActionEnvir^.Base[B].BaseTotCivEmploy)
        * YtoX(ZeroCheck(
          TotMilAndCiv,
          BasePopulation),
          StdFctrsFac.BOSIndex);
    END;
  For B2 := 0 To MaxBases Do
    If Distance^.Base[B].Base[B2] <> 0 Then
      TotMilAndCiv :=
          TotMilAndCiv
        - PosTrans.Base[B].Base[B2]^.Off.Year[MaxYears]
        - PosTrans.Base[B].Base[B2]^.Enl.Year[MaxYears]
        - PosTrans.Base[B].Base[B2]^.Civ.Year[MaxYears]
        - PosTrans.Base[B].Base[B2]^.Stu.Year[MaxYears];
```

```
        RevisedBOS.Base[B].Beyond :=
           (BudgetPropData^.Base[B].CommunBudget
          + BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
          * ZeroCheck
           (BasePopulation,
            ActionEnvir^.Base[B].BaseTotOffEmploy
          + ActionEnvir^.Base[B].BaseTotEnlEmploy
          + ActionEnvir^.Base[B].BaseTotStuEmploy
          + ActionEnvir^.Base[B].BaseTotCivEmploy)
          * YtoX(ZeroCheck(
            TotMilAndCiv,
            BasePopulation),
            StdFctrsFac.BOSIndex);
      END;
  END;  { CalcRevisedBOS }

{*****************************************************************************
 *   Name           : CalcTotActualRPMABOS
 *
 *   Purpose        : Calculates totals of actual RPMA and BOS costs
 *
 *   Notes          :
 *
 *   Invoked As     : CalcTotActualRPMABOS (TotActualRPMA, TotActualBOS)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcTotActualRPMABOS (VAR TotActualRPMA : Real;
                                VAR TotActualBOS  : Year1ToMaxRealRec);

  VAR
    B, Y    : Integer;
    BasePop : Real;

  Begin { CalcTotActualRPMABOS }
    TotActualRPMA := 0.00;
    FillChar (TotActualBOS, SizeOf (TotActualBOS), 0);
```

```
  FOR B := 0 to MaxBases DO
    If IndustFundSwitch.Base[B] <> 'Y' Then
      Begin
        TotActualRPMA := TotActualRPMA
          + BudgetPropData^.Base[B].RPMANonPayBudget;

        BasePop :=
           (ActionEnvir^.Base[B].BaseTotOffEmploy
          + ActionEnvir^.Base[B].BaseTotEnlEmploy
          + ActionEnvir^.Base[B].BaseTotStuEmploy
          + ActionEnvir^.Base[B].BaseTotCivEmploy);

        For Y := 1 To MaxYears Do
          Begin
            BasePop := BasePop
              - PosReduc.Base[B].Off.Year[Y]
              - PosReduc.Base[B].Enl.Year[Y]
              - PosReduc.Base[B].Civ.Year[Y];
            TotActualBOS.Year[Y] := TotActualBOS.Year[Y]
              +(BudgetPropData^.Base[B].CommunBudget
              + BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
              * ZeroCheck
               (BasePop,
                ActionEnvir^.Base[B].BaseTotOffEmploy
              + ActionEnvir^.Base[B].BaseTotEnlEmploy
              + ActionEnvir^.Base[B].BaseTotStuEmploy
              + ActionEnvir^.Base[B].BaseTotCivEmploy);
          End
      End
End;  { CalcTotActualRPMABOS }
```

```
{*************************************************************************
 *   Name           : CalcTotRevisedRPMABOS
 *
 *   Purpose        : Calculates totals of revised RPMA and BOS costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcTotRevisedRPMABOS (RevisedRPMA, RevisedBOS,
 *                                           TotRevisedRPMA, TotRevisedBOS)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcTotRevisedRPMABOS (     RevisedRPMA     : Base0Year1ToMaxRealRec;
                                      RevisedBOS      : Base0YearAndBeyondRealRec;
                                  VAR TotRevisedRPMA  : Year1ToMaxRealRec;
                                  VAR TotRevisedBOS   : YearAndBeyondRealRec);

  VAR
    B, Y : INTEGER;

  Begin { CalcTotRevisedRPMABOS }
    FillChar (TotRevisedRPMA, SizeOf(TotRevisedRPMA), 0);
    FillChar (TotRevisedBOS, SizeOf(TotRevisedBOS), 0);

    FOR B := 0 to MaxBases DO
      If IndustFundSwitch.Base[B] <> 'Y' Then
        Begin
          FOR Y := 1 to MaxYears DO
            Begin
              TotRevisedRPMA.Year[Y] :=
                  TotRevisedRPMA.Year[Y]
                + RevisedRPMA.Base[B].Year[Y];
              TotRevisedBOS.Year[Y]  :=
                  TotRevisedBOS.Year[Y]
                + RevisedBOS.Base[B].Year[Y]
            END;
```

```
          TotRevisedBOS.Beyond :=
              TotRevisedBOS.Beyond
            + RevisedBOS.Base[B].Beyond;
        End;
    END;  { CalcTotRevisedRPMABOS }

{*****************************************************************************
 *   Name           : CalcHousingChange
 *
 *   Purpose        : Calculates housing changes by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcRPMABOSChang (RevisedRPMA, RevisedBOS, TotActualRPMA,
 *                                      TotActualBOS, TotRevisedRPMA,
 *                                      TotRevisedBOS, RPMABOSChang)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   04Jun92   AML   Creation
 *****************************************************************************}

Procedure CalcHousingChang (     PersMovesSum : PersMovesSumRec;
                            VAR Housing       : CostSaveNetByYearRec);

  Var
    B, P, Y, Y2    : Integer;
    CumulConPerc,
    CumulShutPerc,
    FamHousBase,
    FamQtrsCon,
    NewFamQtrs     : Real;

  Begin
    FillChar (Housing, SizeOf (Housing), 0);

    For B := 0 To MaxBases Do
      Begin
        FamQtrsCon := 0.0;

        FamHousBase := { Total Occupied & Vacant Fam Qtrs at Start }
          ((ActionEnvir^.Base[B].BaseTotOffEmploy
          * StdFctrsPers.PercOffMarried)
          +(ActionEnvir^.Base[B].BaseTotEnlEmploy
          * StdFctrsPers.PercEnlMarried))
          * ActionEnvir^.Base[B].PercFamLivOnBase
          + ActionEnvir^.Base[B].OffHousUnitsVacant
          + ActionEnvir^.Base[B].EnlHousUnitsVacant;
```

```
    For P := 1 to 16 Do
      If Con.Base[B].Category[P] = 'I' {Family Quarters}
        Then FamQtrsCon := FamQtrsCon
          + Con.Base[B].New[P]
          + Con.Base[B].Rehab[P];
    If StdFctrsCon.Units[-1] = 'SF'
      Then FamQtrsCon := ZeroCheck (FamQtrsCon, StdFctrsFac.AvgFamQtrs);

    For Y := 1 To MaxYears Do
      If (Y < Screen1Data^.Base[B].CloseYear) Or
         (Screen1Data^.Base[B].CloseYear = 0) Then
        Begin
          CumulConPerc  := 0.0;
          CumulShutPerc  := 0.0;
          For Y2 := 1 To Y Do
            If Screen1Data^.ModelDoShutDn = 'Y' Then
              Begin
                CumulConPerc := CumulConPerc
                  + PersMovesSum.Base[B].Perc.Year[Y2];
                CumulShutPerc := CumulShutPerc
                  + PersMovesSum.Base[B].ShutDn.Year[Y2];
              End
            Else
              Begin
                CumulConPerc := CumulConPerc
                  + TimePhasShutDn.Base[B].Constr.Year[Y2];
                CumulShutPerc := CumulShutPerc
                  + TimePhasShutDn.Base[B].ShutDn.Year[Y2];
              End;
          If Screen1Data^.ModelDoShutDn = 'Y'
            Then CumulShutPerc := CumulShutPerc
              - (PersMovesSum.Base[B].ShutDn.Year[Y] * 0.5)
            Else CumulShutPerc := CumulShutPerc
              - (TimePhasShutDn.Base[B].ShutDn.Year[Y] * 0.5);
          NewFamQtrs := { New Fam Qtrs Constructed So Far }
            Round (FamQtrsCon * CumulConPerc);
          Housing.Cost.Year[Y] := { Budget * % Change of Fam Qtrs }
              Housing.Cost.Year[Y]
            + BudgetPropData^.Base[B].FamHousBudget
            * ZeroCheck(
              NewFamQtrs,
              FamHousBase);
          Housing.Save.Year[Y] :=
              Housing.Save.Year[Y]
            +(BudgetPropData^.Base[B].FamHousBudget
            * ActionEnvir^.Base[B].FamHousShutDn
            * CumulShutPerc);
        End
      Else Housing.Save.Year[Y] :=
           Housing.Save.Year[Y]
         + BudgetPropData^.Base[B].FamHousBudget;
End;
```

```
    For Y := 1 To MaxYears Do
      Begin
        Housing.Net.Year[Y] :=
            Housing.Cost.Year[Y]
          - Housing.Save.Year[Y];
      End;
  End;

{*************************************************************************
 *   Name           : CalcRPMABOSChang
 *
 *   Purpose        : Calculates RPMA and BOS changes by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcRPMABOSChang (RevisedRPMA, RevisedBOS, TotActualRPMA,
 *                                      TotActualBOS, TotRevisedRPMA,
 *                                      TotRevisedBOS, RPMABOSChang)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     RPMABOS.RPT          Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   27Mar92  AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   13May92  AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcRPMABOSChang (     RevisedRPMA     : Base0Year1ToMaxRealRec;
                                 RevisedBOS      : Base0YearAndBeyondRealRec;
                             VAR TotActualRPMA   : REAL;
                             VAR TotActualBOS    : Year1ToMaxRealRec;
                             VAR TotRevisedRPMA  : Year1ToMaxRealRec;
                             VAR TotRevisedBOS   : YearAndBeyondRealRec;
                             VAR RPMABOSChang    : YearAndBeyondRealRec;
                                 Housing         : CostSaveNetByYearRec;
                                 Adjust          : Year1ToMaxRealRec);
```

```
VAR
  B, Y       : INTEGER;
  BOSChang,
  RPMAChang  : Real;
  Buffer     : IOBuffer;
  F          : TEXT;

Begin { CalcRPMABOSChang }
  FillChar (RPMABOSChang, SizeOf (RPMABOSChang), 0);

  assign  (F, Info.ReportDir + 'RPMABOS.RPT');
  SetTextBuf (F, Buffer);
  rewrite (F);
  Title(F,'RPMA/BOS CHANGE REPORT',1);
  writeln (F);
  writeln (F, Blank:21, '(All values in Dollars)');

  FOR Y := 1 to MaxYears DO
    Begin
      RPMAChang := TotRevisedRPMA.Year[Y] - TotActualRPMA;
      BOSChang  := TotRevisedBOS.Year[Y]  - TotActualBOS.Year[Y];

      RPMABOSChang.Year[Y] :=
          RPMAChang
        + BOSChang
        + Housing.Net.Year[Y];

      { Write variables to output file }

      writeln (F);
      writeln (F, FiscalYear(Y):16, '      RPMA Changes',
        WithCommas(RPMAChang * Adjust.Year[Y],0):15);
      writeln (F, BLANK:20, '+ BOS Changes',
        WithCommas(BOSChang * Adjust.Year[Y],0):16);
      writeln (F, BLANK:20, '+ Housing Changes',
        WithCommas(Housing.Net.Year[Y] * Adjust.Year[Y],0):12);
      writeln (F, BLANK:20, Rept('-', 29));
      writeln (F, BLANK:22, 'Total Changes',
        WithCommas(RPMABOSChang.Year[Y] * Adjust.Year[Y],0):14);
    END;
  BOSChang  := TotRevisedBOS.Beyond - TotActualBOS.Year[MaxYears];

  RPMABOSChang.Beyond :=
      RPMAChang
    + BOSChang
    + Housing.Net.Year[MaxYears];

  writeln (F);
  writeln (F, 'Beyond':17, '     RPMA Changes',
    WithCommas(RPMAChang * Adjust.Year[MaxYears],0):15);
  writeln (F, BLANK:20, '+ BOS  Changes',
    WithCommas(BOSChang * Adjust.Year[MaxYears],0):15);
```

```
    writeln (F, BLANK:20, '+ Housing Changes',
      WithCommas(Housing.Net.Year[MaxYears] * Adjust.Year[MaxYears],0):12);
    writeln (F, BLANK:20, Rept('-', 29));
    writeln (F, BLANK:22, 'Total Changes',
      WithCommas(RPMABOSChang.Beyond * Adjust.Year[MaxYears],0):14);
    close (F)
  END;  { CalcRPMABOSChang }

{*************************************************************************
 *   Name          : CalcDeltaChart
 *
 *   Purpose       : Calculates various components of Delta chart
 *
 *   Notes         : Was originally eight separate procedures
 *
 *   Invoked As    : CalcDeltaChart (MilConDetail, RevisedBOS, RevisedRPMA
 *                                   DeltaChart, Adjust)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   13May92  AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcDeltaChart (    MilConDetail : MilConRec;
                              RevisedBOS   : Base0YearAndBeyondRealRec;
                              RevisedRPMA  : Base0Year1ToMaxRealRec;
                          VAR DeltaChart   : DeltaChartRec;
                              Adjust       : Year1ToMaxRealRec);

  VAR
    B, B2, P, Y : Integer;
    TotalPop    : Real;

  Begin { CalcDeltaChartBOS }
    FillChar (DeltaChart^, SizeOf (DeltaChart^), 0);

    FOR B := 0 to MaxBases DO
      Begin
        DeltaChart^.RPMA.Base[B].Delta :=
           (RevisedRPMA.Base[B].Year[MaxYears]
          - BudgetPropData^.Base[B].RPMANonPayBudget)
          * Adjust.Year[MaxYears];
        DeltaChart^.RPMA.Base[B].PercChange := ZeroCheck
          (DeltaChart^.RPMA.Base[B].Delta,
           BudgetPropData^.Base[B].RPMANonPayBudget);
```

```
DeltaChart^.SF.Base[B].Delta :=
  - RealPropProc.Base[B].FacilSqFtShutDown;

TotalPop :=
   (ActionEnvir^.Base[B].BaseTotOffEmploy
  + ActionEnvir^.Base[B].BaseTotEnlEmploy
  + ActionEnvir^.Base[B].BaseTotStuEmploy
  + ActionEnvir^.Base[B].BaseTotCivEmploy);

For Y := 1 to MaxYears Do
  Begin
    DeltaChart^.Land.Base[B].Delta :=
        DeltaChart^.Land.Base[B].Delta
      + RealPropProc.RealPropPur.Base[B].Acre.Year[Y];

    DeltaChart^.Pers.Base[B].Delta :=
        DeltaChart^.Pers.Base[B].Delta
      + PosTrans.Base[B].CaretakeCiv.Year[Y]
      + PosTrans.Base[B].CaretakeMil.Year[Y]
      - PosElim.Base[B].Off.Year[Y]
      - PosElim.Base[B].Enl.Year[Y]
      - PosElim.Base[B].Civ.Year[Y];
    TotalPop := TotalPop
      - PosReduc.Base[B].Off.Year[Y]
      - PosReduc.Base[B].Enl.Year[Y]
      - PosReduc.Base[B].Civ.Year[Y];
    For B2 := 0 To MaxBases Do
      If Distance^.Base[B].Base[B2] <> 0 Then
        DeltaChart^.Pers.Base[B].Delta :=
            DeltaChart^.Pers.Base[B].Delta
          + PosTrans.Base[B2].Base[B]^.Civ.Year[Y]
          + PosTrans.Base[B2].Base[B]^.Stu.Year[Y]
          + PosTrans.Base[B2].Base[B]^.Enl.Year[Y]
          + PosTrans.Base[B2].Base[B]^.Off.Year[Y]
          - PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
          - PosTrans.Base[B].Base[B2]^.Stu.Year[Y]
          - PosTrans.Base[B].Base[B2]^.Enl.Year[Y]
          - PosTrans.Base[B].Base[B2]^.Off.Year[Y];
  END;

For P := 1 to 16 Do
  If ((Con.Base[B].Category[P]  > 'B') And
      (Con.Base[B].Category[P] <> 'P') And
      (Con.Base[B].Category[P] <  'S')) Then
    Begin
      If (Con.Base[B].Category[P] = 'H') And
         (StdFctrsCon.Units[-2] <> 'SF')
        Then DeltaChart^.SF.Base[B].Delta :=
            DeltaChart^.SF.Base[B].Delta
          + Con.Base[B].New[P]
          * StdFctrsFac.AvgBachQtrs
```

```
        Else If (Con.Base[B].Category[P] = 'I') And
           (StdFctrsCon.Units[-1] <> 'SF')
          Then DeltaChart^.SF.Base[B].Delta :=
              DeltaChart^.SF.Base[B].Delta
            + Con.Base[B].New[P]
            * StdFctrsFac.AvgFamQtrs
        Else DeltaChart^.SF.Base[B].Delta :=
            DeltaChart^.SF.Base[B].Delta
          + Con.Base[B].New[P]
      End
    Else If Con.Base[B].Category[P] > 'R'
      Then If StdFctrsCon.Units[Ord (Con.Base[B].Category[P]) - 83] = 'SF'
        Then DeltaChart^.SF.Base[B].Delta :=
             DeltaChart^.SF.Base[B].Delta
           + Con.Base[B].New[P];
DeltaChart^.SF.Base[B].PercChange := ZeroCheck
  (DeltaChart^.SF.Base[B].Delta, ActionEnvir^.Base[B].BaseFacilSqFt);

DeltaChart^.BOS.Base[B].Delta :=
    (RevisedBOS.Base[B].Beyond
   -(BudgetPropData^.Base[B].BaseOpnsNonPayBudget
   + BudgetPropData^.Base[B].CommunBudget)
   * ZeroCheck
    (TotalPop,
     ActionEnvir^.Base[B].BaseTotOffEmploy
   + ActionEnvir^.Base[B].BaseTotEnlEmploy
   + ActionEnvir^.Base[B].BaseTotStuEmploy
   + ActionEnvir^.Base[B].BaseTotCivEmploy))
   * Adjust.Year[MaxYears];
DeltaChart^.BOS.Base[B].PercChange := ZeroCheck(
     DeltaChart^.BOS.Base[B].Delta,
    (BudgetPropData^.Base[B].BaseOpnsNonPayBudget
   + BudgetPropData^.Base[B].CommunBudget)
   * ZeroCheck
    (TotalPop,
     ActionEnvir^.Base[B].BaseTotOffEmploy
   + ActionEnvir^.Base[B].BaseTotEnlEmploy
   + ActionEnvir^.Base[B].BaseTotStuEmploy
   + ActionEnvir^.Base[B].BaseTotCivEmploy));

DeltaChart^.RPMABOS.Base[B].Delta :=
    DeltaChart^.RPMA.Base[B].Delta
  + DeltaChart^.BOS.Base[B].Delta;
DeltaChart^.RPMABOS.Base[B].PercChange := ZeroCheck(
    DeltaChart^.RPMABOS.Base[B].Delta,
    BudgetPropData^.Base[B].RPMANonPayBudget
  +(BudgetPropData^.Base[B].BaseOpnsNonPayBudget
  + BudgetPropData^.Base[B].CommunBudget)
  * ZeroCheck
   (TotalPop,
    ActionEnvir^.Base[B].BaseTotOffEmploy
  + ActionEnvir^.Base[B].BaseTotEnlEmploy
```

```
          + ActionEnvir^.Base[B].BaseTotStuEmploy
          + ActionEnvir^.Base[B].BaseTotCivEmploy));

        DeltaChart^.Land.Base[B].PercChange := ZeroCheck
          (DeltaChart^.Land.Base[B].Delta, ActionEnvir^.Base[B].AcreOnBase);

        DeltaChart^.Pers.Base[B].PercChange := ZeroCheck
          (DeltaChart^.Pers.Base[B].Delta, TotalPop);
      END
  END;  { CalcDeltaChart }

{*****************************************************************************
 *   Name           : CalcMothballCost
 *
 *   Purpose        : Calculates mothball (shutdown) costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMothballCost (MothballCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcMothballCost (     PersMovesSum : PersMovesSumRec;
                            VAR MothballCost : Year1ToMaxRealRec);

  VAR
    B, Y : INTEGER;
    Perc : Real;

  Begin { CalcMothballCost }
    FillChar (MothballCost, SizeOf (MothballCost), 0);
    For B := 0 To MaxBases Do
      IF (Screen1Data^.Base[B].CloseYear > 0) Then
        Begin
          If IndustFundSwitch.Base[B] <> 'Y'
            THEN For Y := 1 To MaxYears Do
```

```
            Begin
              IF Screen1Data^.ModelDoShutDn = 'Y'
                Then Perc := PersMovesSum.Base[B].ShutDn.Year[Y]
                Else Perc := TimePhasShutDn.Base[B].ShutDn.Year[Y];
              MothballCost.Year[Y] := MothballCost.Year[Y]
                +(StdFctrsFac.MothballCostPerSqrFt
                * RealPropProc.Base[B].FacilSqFtShutDown
                * Perc)
            End
          Else For Y := 1 To MaxYears Do
            MothballCost.Year[Y] := MothballCost.Year[Y]
             + IndustFund.Base[B]^.OvhdCost.MothBall.Year[Y];
      End;
  END;  { CalcMothballCost }

{*****************************************************************************
 *   Name           : CalcCaretakerCost
 *
 *   Purpose        : Calculates caretaker costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcCaretakerCost (RPMACostFctr, CaretakerCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcCaretakerCost (    RPMACostFctr   : Base0ToMaxRealRec;
                             VAR CaretakerCost : Year1ToMaxRealRec);

  VAR
    B, Y    : INTEGER;
    Base1,
    Base2,
    Fctr1,
    Fctr2,
    Pop     : REAL;
```

```
Begin { CalcCaretakerCost }
  FillChar (CaretakerCost, SizeOf (CaretakerCost), 0);
  For B := 0 To MaxBases Do
    Begin
      Base1 := 0.0;
      Base2 := 0.0;
      Pop := ActionEnvir^.Base[B].BaseTotCivEmploy
           + ActionEnvir^.Base[B].BaseTotOffEmploy
           + ActionEnvir^.Base[B].BaseTotEnlEmploy
           + ActionEnvir^.Base[B].BaseTotStuEmploy;
      For Y := 1 to MaxYears DO
        If IndustFundSwitch.Base[B] = 'Y'
          Then CaretakerCost.Year[Y] :=
               CaretakerCost.Year[Y]
             + IndustFund.Base[B]^.OvhdCost.Caretaker.Year[Y]
        Else
          Begin
            Base1 := Base1
              + PosTrans.Base[B].CaretakeCiv.Year[Y];
            Base2 := Base2
              + PosTrans.Base[B].CaretakeMil.Year[Y];
            Pop := Pop
              - PosReduc.Base[B].Off.Year[Y]
              - PosReduc.Base[B].Enl.Year[Y]
              - PosReduc.Base[B].Civ.Year[Y];
            Fctr1 := ZeroCheck(
               (BudgetPropData^.Base[B].CommunBudget
              + BudgetPropData^.Base[B].BaseOpnsNonPayBudget)
              * ZeroCheck
               (Pop,
                ActionEnvir^.Base[B].BaseTotCivEmploy
              + ActionEnvir^.Base[B].BaseTotOffEmploy
              + ActionEnvir^.Base[B].BaseTotEnlEmploy
              + ActionEnvir^.Base[B].BaseTotStuEmploy),
                YtoX (Pop, StdFctrsFac.BOSIndex))
              * YtoX (Base1 + Base2, StdFctrsFac.BOSIndex);
            Fctr2 := RPMACostFctr.Base[B]
                   * YtoX ((StdFctrsFac.CaretakerAdminSpace
                          *(Base1 + Base2)),
                            StdFctrsFac.RPMACostBldgPower);
            IF Base1 + Base2 > 0
              Then CaretakerCost.Year[Y] :=
                   CaretakerCost.Year[Y]
                 + Fctr1
                 + Fctr2
                 +(StdFctrsPers.EnlSal
                 * Base2)
                 +(StdFctrsPers.CivSal
                 * Base1)
          End;
    END;
END;  { CalcCaretakerCost }
```

```
{*************************************************************************
 *   Name           : CalcMaintainSpaceCost
 *
 *   Purpose        : Calculates cost of maintaining space by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMaintainSpaceCost (MaintainSpaceCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     -----------------      --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcMaintainSpaceCost (VAR MaintainSpaceCost : Year1ToMaxRealRec);

  VAR
    B, Y       : INTEGER;
    CumulPerc  : REAL;

  Begin { CalcMaintainSpaceCost }
    FillChar (MaintainSpaceCost, SizeOf (MaintainSpaceCost), 0);
    For B := 0 to MaxBases Do
      If IndustFundSwitch.Base[B] = 'Y'
        Then For Y := 1 To MaxYears Do
            MaintainSpaceCost.Year[Y] :=
            MaintainSpaceCost.Year[Y]
          + IndustFund.Base[B]^.OvhdCost.MaintSpace.Year[Y]
        Else
          Begin
            CumulPerc := 0.0;
            IF (Screen1Data^.Base[B].CloseYear = 0) Or
               (Screen1Data^.Base[B].Deactivate = 'Y')
              THEN FOR Y := 1 to MaxYears DO
                Begin
                  IF Screen1Data^.ModelDoShutDn = 'Y' Then
                    Begin
                      IF Y < Screen1Data^.Base[B].CloseYear
                        THEN CumulPerc := 0.0
                        ELSE CumulPerc := 1.0;
                    End
```

```
                    ELSE CumulPerc := CumulPerc
                      + TimePhasShutDn.Base[B].ShutDn.Year[Y];
                    MaintainSpaceCost.Year[Y] :=
                        MaintainSpaceCost.Year[Y]
                      + CumulPerc
                      * ZeroCheck
                       (RealPropProc.Base[B].FacilSqFtShutDown,
                        ActionEnvir^.Base[B].BaseFacilSqFt)
                      * BudgetPropData^.Base[B].RPMANonPayBudget
                      * StdFctrsFac.CaretakerPercOrigRPMACost
                  End;
            END
  END;  { CalcMaintainSpaceCost }

{*****************************************************************************
 *   Name           : CalcMiscRecurring
 *
 *   Purpose        : Calculates Misc Recurring Cost for Gaining Bases
 *
 *   Notes          :
 *
 *   Invoked As     : CalcMiscRecurring (MiscRecurring)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   21Apr92   AML   Creation
 *   13May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcMiscRecurring (Var MiscRecurring : Year1ToMaxRealRec);

  Var
    B, Y : Integer;

  Begin
    FillChar (MiscRecurring, SizeOf (MiscRecurring), 0);
    For B := 0 To MaxBases Do
      For Y := 1 To MaxYears Do
        MiscRecurring.Year[Y] := MiscRecurring.Year[Y]
          + BudgetPropData^.Base[B].MiscRecurCost.Year[Y]
  End;
```

```
{*************************************************************************
 *   Name           : CalcOverheadCost
 *
 *   Purpose        : Orchestrates calculation of overhead cost by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcOverheadCost (PersMovesSum, MilConDetail,
 *                                      OvhdCostFinal, AdminSuppCost,
 *                                      MothballCost)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   13May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcOverheadCost (     PersMovesSum       : PersMovesSumRec;
                                 MilConDetail       : MilConRec;
                             VAR OvhdCostFinal      : CostSaveNetByYearRec;
                                                                    V   A   R
AdminSuppCost        : Base0Year1ToMaxRealRec;
                                                                    V   A   R
MothballCost              : Year1ToMaxRealRec;
                             VAR DeltaChart         : DeltaChartRec;
                             VAR RevisedRPMA        : Base0Year1ToMaxRealRec;
                                                                    V   A   R
RevisedBOS                     : Base0YearAndBeyondRealRec;
                             VAR TotActualRPMA      : REAL;
                             VAR TotActualBOS       : Year1ToMaxRealRec;
                             VAR TotRevisedRPMA     : Year1ToMaxRealRec;
                                                                    V   A   R
TotRevisedBOS        : YearAndBeyondRealRec;
                             VAR CaretakerCost      : Year1ToMaxRealRec;
                             VAR MaintainSpaceCost  : Year1ToMaxRealRec;
                             VAR Housing            : CostSaveNetByYearRec;
                                 Adjust             : Year1ToMaxRealRec);
```

```
VAR
  RPMACostFctr  : Base0ToMaxRealRec;
  Buffer        : IOBuffer;
  F             : TEXT;
        AdminCost,
  MiscRecurring,
  OperatingCost,
  OperatingSave,
  OtherCost,
        OtherSave           : Year1ToMaxRealRec;
        RPMABOSChang    : YearAndBeyondRealRec;
  B, Y          : INTEGER;

Begin { CalcOverheadCost }
  FillChar (AdminCost, SizeOf (AdminCost), 0);
  FillChar (OperatingCost, SizeOf (OperatingCost), 0);
  FillChar (OtherCost, SizeOf (OtherCost), 0);
  FillChar (OperatingSave, SizeOf (OperatingSave), 0);
  FillChar (OtherSave, SizeOf (OtherSave), 0);
  FillChar (OvhdCostFinal, SizeOf (OvhdCostFinal), 0);

  CalcAdminSuppCost      (AdminSuppCost);
  CalcRPMACostFctr       (RPMACostFctr);
  CalcRevisedRPMA        (PersMovesSum, MilConDetail, RPMACostFctr,
                          RevisedRPMA);
  CalcRevisedBOS         (RevisedBOS);
  CalcTotActualRPMABOS   (TotActualRPMA, TotActualBOS);
  CalcTotRevisedRPMABOS  (RevisedRPMA, RevisedBOS, TotRevisedRPMA,
                          TotRevisedBOS);
  CalcHousingChang       (PersMovesSum, Housing);
  CalcRPMABOSChang       (RevisedRPMA, RevisedBOS, TotActualRPMA,
                          TotActualBOS, TotRevisedRPMA, TotRevisedBOS,
                          RPMABOSChang, Housing, Adjust);
  CalcMothballCost       (PersMovesSum, MothballCost);
  CalcCaretakerCost      (RPMACostFctr, CaretakerCost);
  CalcMaintainSpaceCost  (MaintainSpaceCost);
  CalcMiscRecurring      (MiscRecurring);
  CalcDeltaChart         (MilConDetail, RevisedBOS, RevisedRPMA, DeltaChart,
                          Adjust);

  For B := 0 To MaxBases Do
            For Y := 1 To MaxYears Do
      Begin
                        If IndustFundSwitch.Base[B] = 'Y' Then
          Begin
                                  OperatingCost.Year[Y] := OperatingCost.Year[Y]
                                       +
IndustFund.Base[B]^.OvhdCost.Operating.Year[Y];
                                  OperatingSave.Year[Y] := OperatingSave.Year[Y]
                                       +
IndustFund.Base[B]^.OvhdSave.Operating.Year[Y];
                                  OtherCost.Year[Y] := OtherCost.Year[Y]
```

```
                                                +
IndustFund.Base[B]^.OvhdCost.Other.Year[Y];
                                        OtherSave.Year[Y] := OtherSave.Year[Y]
                                         + IndustFund.Base[B]^.OvhdSave.Other.Year[Y];
                                  End;
                            AdminCost.Year[Y] := AdminCost.Year[Y]
                                  + AdminSuppCost.Base[B].Year[Y];
                      End;

    assign  (F, Info.ReportDir + 'OVERHEAD.RPT');
    SetTextBuf (F, Buffer);
    rewrite (F);
    Title (F,'OVERHEAD COSTS REPORT',1);
    Writeln (F);
    Writeln (F, Blank:26, '(All values in Dollars)');

    For Y := 1 to MaxYears DO
      Begin
        OvhdCostFinal.Cost.Year[Y] :=
                                  AdminCost.Year[Y]
          + OperatingCost.Year[Y]
          + RPMABOSChang.Year[Y]
          + MothBallCost.Year[Y]
          + CaretakerCost.Year[Y]
          + MaintainSpaceCost.Year[Y]
          + MiscRecurring.Year[Y]
          + OtherCost.Year[Y];
        OvhdCostFinal.Save.Year[Y] :=
            OperatingSave.Year[Y]
          + OtherSave.Year[Y];
        Writeln (F);
        writeln (F, FiscalYear(Y):16,
          Blank:11, 'Admin/Supp Cost',
                        WithCommas (AdminCost.Year[Y] * Adjust.Year[Y],0):16);
        writeln (F, BLANK:25, '+ Uniq Operating Cost',
          WithCommas (OperatingCost.Year[Y] * Adjust.Year[Y],0):12);
        writeln (F, BLANK:25, '+ RPMABOS Change',
          WithCommas (RPMABOSChang.Year[Y] * Adjust.Year[Y],0):17);
        writeln (F, BLANK:25, '+ Mothball Cost',
          WithCommas (MothballCost.Year[Y] * Adjust.Year[Y],0):18);
        writeln (F, BLANK:25, '+ Caretaker Cost',
          WithCommas (CaretakerCost.Year[Y] * Adjust.Year[Y],0):17);
        writeln (F, BLANK:25, '+ Maintain Space',
          WithCommas (MaintainSpaceCost.Year[Y] * Adjust.Year[Y],0):17);
        writeln (F, BLANK:25, '+ Misc Recur Cost',
          WithCommas (MiscRecurring.Year[Y] * Adjust.Year[Y],0):16);
        writeln (F, BLANK:25, '+ Uniq Other Cost',
          WithCommas (OtherCost.Year[Y] * Adjust.Year[Y],0):16);
        writeln (F, BLANK:25, '- Uniq Operating Savings',
          WithCommas (OperatingSave.Year[Y] * Adjust.Year[Y],0):9);
        writeln (F, BLANK:25, '- Uniq Other Savings',
          WithCommas (OtherSave.Year[Y] * Adjust.Year[Y],0):13);
```

```
        writeln (F, BLANK:25, Rept('-', 33));
        OvhdCostFinal.Net.Year[Y] :=
            OvhdCostFinal.Cost.Year[Y]
          - OvhdCostFinal.Save.Year[Y];
        writeln (F, BLANK:25, 'Total Overhead Cost',
          WithCommas (OvhdCostFinal.Net.Year[Y] * Adjust.Year[Y],0):14);
        If (Y Div 3) * 3 = Y Then
          Begin
            Title (F, 'OVERHEAD COSTS REPORT', Y Div 3 + 1);
            Writeln (F);
            Writeln (F, Blank:26, '(All values in Dollars)');
          End;
      END;

    OvhdCostFinal.Cost.Beyond :=
        OperatingCost.Year[MaxYears]
      + CaretakerCost.Year[MaxYears]
      + MaintainSpaceCost.Year[MaxYears]
      + RPMABOSChang.Beyond
      + MiscRecurring.Year[MaxYears]
      + OtherCost.Year[MaxYears];
    OvhdCostFinal.Save.Beyond :=
        OperatingSave.Year[MaxYears]
      + OtherSave.Year[MaxYears];
    writeln (F);
    writeln (F, BLANK:10, 'Beyond Years     Uniq Operating Cost',
      WithCommas (OperatingCost.Year[MaxYears] * Adjust.Year[MaxYears],0):12);
    writeln (F, BLANK:25, '+ RPMABOS Change',
      WithCommas (RPMABOSChang.Beyond * Adjust.Year[MaxYears],0):17);
    writeln (F, BLANK:25, '+ Caretaker Cost',
      WithCommas (CaretakerCost.Year[MaxYears] * Adjust.Year[MaxYears],0):17);
    writeln (F, BLANK:25, '+ Maintain Space', WithCommas
       (MaintainSpaceCost.Year[MaxYears]
      * Adjust.Year[MaxYears],0):17);
    writeln (F, BLANK:25, '+ Misc Recur Cost',
      WithCommas (MiscRecurring.Year[MaxYears] * Adjust.Year[MaxYears],0):16);
    writeln (F, BLANK:25, '+ Uniq Other Cost',
      WithCommas (OtherCost.Year[MaxYears] * Adjust.Year[MaxYears],0):16);
    writeln (F, BLANK:25, '- Uniq Operating Savings',
      WithCommas (OperatingSave.Year[MaxYears] * Adjust.Year[MaxYears],0):9);
    writeln (F, BLANK:25, '- Uniq Other Savings',
      WithCommas (OtherSave.Year[MaxYears] * Adjust.Year[MaxYears],0):13);
    writeln (F, BLANK:25, Rept('-', 33));
    OvhdCostFinal.Net.Beyond :=
        OvhdCostFinal.Cost.Beyond
      - OvhdCostFinal.Save.Beyond;
    writeln (F, BLANK:25, 'Total Overhead Cost',
      WithCommas (OvhdCostFinal.Net.Beyond * Adjust.Year[MaxYears],0):14);
    close (F)
  END;  { CalcOverheadCost }

END. { UNIT }
```

```
UNIT CCALPER;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE CalcPersonnelCost (     CalcFctrsPerPos : CalcPersPosRec;
                                  PersMoves       : CalcMovesRec;
                              VAR PersFinal       : CostSaveNetByYearRec;
                              VAR CostElimMil     : CostElimMilRec;
                              VAR SalSav          : SalSavRec;
                              VAR HousAllowSaved  : Year1ToMaxRealRec;
                              VAR HousAllowSum    : Year1ToMaxRealRec;
                                  Adjust          : Year1ToMaxRealRec);

PROCEDURE CalcPersFactors    (VAR CalcFctrsPerPos     : CalcPersPosRec;
                              VAR CalcPersAct         : CalcPersActRec;
                              VAR PersMoves           : CalcMovesRec;
                              VAR PersMovesSum        : PersMovesSumRec);

IMPLEMENTATION

{*****************************************************************************
 *   Name           : CalcSalSav
 *
 *   Purpose        : Calculates salary savings by year for Off, Enl, and Civ
 *
 *   Notes          :
 *
 *   Invoked As     : CalcSalSav (CalcFctrsPerPos, SalSav)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     -------   ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   11May92   AML   Implemented Multi-Basing
 *****************************************************************************}
```

```
PROCEDURE CalcSalSav (    CalcFctrsPerPos : CalcPersPosRec;
                      VAR SalSav          : SalSavRec);

  VAR
    Y, Y2        : INTEGER;
    Diff,
    NewCivElim,
    NewEnlElim,
    NewOffElim   : REAL;

  Begin { CalcSalSav }
    FillChar (SalSav, SizeOf (SalSav), 0);
    FOR Y := 1 to MaxYears DO
      Begin
        SalSav.Off.Beyond :=
            SalSav.Off.Beyond
          + CalcFctrsPerPos.OffPosElim.Year[Y]
          * StdFctrsPers.OffSal;
        SalSav.Enl.Beyond :=
            SalSav.Enl.Beyond
          + CalcFctrsPerPos.EnlPosElim.Year[Y]
          * StdFctrsPers.EnlSal;
        SalSav.Civ.Beyond :=
            SalSav.Civ.Beyond
          + CalcFctrsPerPos.CivPosElim.Year[Y]
          * StdFctrsPers.CivSal;
        NewOffElim := CalcFctrsPerPos.OffPosElim.Year[Y] / 2.0;
        NewEnlElim := CalcFctrsPerPos.EnlPosElim.Year[Y] / 2.0;
        NewCivElim := CalcFctrsPerPos.CivPosElim.Year[Y] / 2.0;
        IF Y > 1
          Then For Y2 := 1 To Y-1 Do
            Begin
              NewOffElim := NewOffElim + CalcFctrsPerPos.OffPosElim.Year[Y2];
              NewEnlElim := NewEnlElim + CalcFctrsPerPos.EnlPosElim.Year[Y2];
              NewCivElim := NewCivElim + CalcFctrsPerPos.CivPosElim.Year[Y2];
            End;
        SalSav.Civ.Year[Y] := NewCivElim * StdFctrsPers.CivSal;
        SalSav.Enl.Year[Y] := NewEnlElim * StdFctrsPers.EnlSal;
        SalSav.Off.Year[Y] := NewOffElim * StdFctrsPers.OffSal;
        SalSav.Tot.Year[Y] :=
            SalSav.Civ.Year[Y]
          + SalSav.Enl.Year[Y]
          + SalSav.Off.Year[Y]
      END;
    SalSav.Tot.Beyond :=
        SalSav.Civ.Beyond
      + SalSav.Enl.Beyond
      + SalSav.Off.Beyond;
  END;  { CalcSalSav }
```

```
{*************************************************************************
 *   Name           : CalcCostElimMil
 *
 *   Purpose        : Calculates cost of eliminating military persons by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcCostElimMil (CalcFctrsPerPos, CostElimMil)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   11May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcCostElimMil (    CalcFctrsPerPos : CalcPersPosRec;
                           VAR CostElimMil     : CostElimMilRec);

  VAR
    Y : INTEGER;

  Begin { CalcCostElimMil }
    FillChar (CostElimMil, SizeOf (CostElimMil), 0);
    FOR Y := 1 to MaxYears DO
      Begin
        CostElimMil.Enl.Year[Y] :=
          - CalcFctrsPerPos.EnlPosElim.Year[Y]
          * StdFctrsTrans.OneTimeEnlPCSCosts;
        CostElimMil.Off.Year[Y] :=
          - CalcFctrsPerPos.OffPosElim.Year[Y]
          * StdFctrsTrans.OneTimeOffPCSCosts;
        CostElimMil.Tot.Year[Y] :=
            CostElimMil.Enl.Year[Y]
          + CostElimMil.Off.Year[Y]
      END;
  END;  { CalcCostElimMil }
```

```
{*************************************************************************
 *   Name           : CalcHousAllow
 *
 *   Purpose        : Calculates cost of housing allowances by gaining base by
 *                    year for officers and enlisted
 *
 *   Notes          :
 *
 *   Invoked As     : CalcHousAllow (HousAllow)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   11May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcHousAllow (VAR HousAllow : HousAllowRec);

  VAR
    AddlCostFamOff,
    AddlCostFamEnl,
    AddlSaveFamOff,
    AddlSaveFamEnl,
    AnnVHADeltaOff,
    AnnVHADeltaEnl,
    BachOffVHA,
    BachEnlVHA,
    ChgSingVHARate,
    BachQtrsCon,
    FamQtrsCon,
    TempOffTrans,
    TempEnlTrans     : REAL;
    B, B2, P, Y, Y2 : INTEGER;

  Begin { CalcHousAllow }
    FillChar (HousAllow, SizeOf (HousAllow), 0);
    FOR B := 0 to MaxBases DO
      Begin
        BachQtrsCon := 0.0;
        FamQtrsCon := 0.0;
```

```
For P := 1 to 16 Do
  Begin
    If Con.Base[B].Category[P] = 'H' {Bachelor Quarters}
      Then BachQtrsCon := BachQtrsCon
        + Con.Base[B].New[P]
        + Con.Base[B].Rehab[P];
    If Con.Base[B].Category[P] = 'I' {Family Quarters}
      Then FamQtrsCon := FamQtrsCon
        + Con.Base[B].New[P]
        + Con.Base[B].Rehab[P];
  End;
If StdFctrsCon.Units[-1] = 'SF'
  Then FamQtrsCon := ZeroCheck (FamQtrsCon, StdFctrsFac.AvgFamQtrs);
If StdFctrsCon.Units[-2] = 'SF'
  Then BachQtrsCon := ZeroCheck (BachQtrsCon, StdFctrsFac.AvgBachQtrs);
FOR Y := 1 to MaxYears DO
  Begin
    AnnVHADeltaOff := 0.0;
    AnnVHADeltaEnl := 0.0;
    AddlCostFamOff := 0.0;
    AddlCostFamEnl := 0.0;
    AddlSaveFamOff := 0.0;
    AddlSaveFamEnl := 0.0;
    BachOffVHA     := 0.0;
    BachEnlVHA     := 0.0;
    ChgSingVHARate := 0.0;
    For B2 := 0 To MaxBases Do
      If Distance^.Base[B2].Base[B] > 0 Then
        Begin
          TempOffTrans := 0.0;
          TempEnlTrans := 0.0;
          For Y2 := 1 To Y Do
            Begin
              TempOffTrans :=
                  TempOffTrans
                + PosTrans.Base[B2].Base[B]^.Off.Year[Y2];
              TempEnlTrans :=
                  TempEnlTrans
                + PosTrans.Base[B2].Base[B]^.Enl.Year[Y2];
            End;
          AnnVHADeltaOff :=
              AnnVHADeltaOff
            +(TempOffTrans
            * StdFctrsPers.PercOffMarried
            *(1
            - ActionEnvir^.Base[B2].PercFamLivOnBase))
            *(PersCost.Base[B].OffVHA
            - PersCost.Base[B2].OffVHA)
            * 12.0;
```

```
AddlCostFamOff :=
    AddlCostFamOff
  +(TempOffTrans
  * StdFctrsPers.PercOffMarried
  * ActionEnvir^.Base[B2].PercFamLivOnBase);
AddlSaveFamOff :=
    AddlSaveFamOff
  +(TempOffTrans
  * StdFctrsPers.PercOffMarried
  *(1.0
  - ActionEnvir^.Base[B2].PercFamLivOnBase));
ChgSingVHARate :=
    ChgSingVHARate
  + TempOffTrans
  *(1
  - StdFctrsPers.PercOffMarried)
  *(PersCost.Base[B].OffVHA
  - PersCost.Base[B2].OffVHA)
  * 12.0;
BachOffVHA := BachOffVHA
  + Min
   (BachQtrsCon,
    TempOffTrans
  *(1.0
  - StdFctrsPers.PercOffMarried))
  *(PersCost.Base[B].OffVHA
  * 12.0
  *(1.0
  - StdFctrsPers.PercEnlHous));
BachEnlVHA := BachEnlVHA
  + Min
   (BachQtrsCon,
    TempEnlTrans
  *(1.0
  - StdFctrsPers.PercEnlMarried))
  *(PersCost.Base[B].EnlVHA
  * 12.0
  * StdFctrsPers.PercEnlHous);
AnnVHADeltaEnl :=
    AnnVHADeltaEnl
  +(TempEnlTrans
  * StdFctrsPers.PercEnlMarried
  *(1
  - ActionEnvir^.Base[B2].PercFamLivOnBase))
  *(PersCost.Base[B].EnlVHA
  - PersCost.Base[B2].EnlVHA)
  * 12.0;
AddlCostFamEnl :=
    AddlCostFamEnl
  +(TempEnlTrans
  * StdFctrsPers.PercEnlMarried
  * ActionEnvir^.Base[B2].PercFamLivOnBase);
```

```
        AddlSaveFamEnl :=
            AddlCostFamEnl
          +(TempEnlTrans
          * StdFctrsPers.PercEnlMarried
          * ActionEnvir^.Base[B2].PercFamLivOnBase);
      End;
    AddlCostFamOff :=
        AddlCostFamOff
      - ActionEnvir^.Base[B].OffHousUnitsVacant
      -((1.0
      - StdFctrsPers.PercEnlHous)
      * FamQtrsCon);
    AddlCostFamEnl :=
        AddlCostFamEnl
      - ActionEnvir^.Base[B].EnlHousUnitsVacant
      -(StdFctrsPers.PercEnlHous
      * FamQtrsCon);
    AddlSaveFamOff := AtLeastZero
       (Min
      (-AddlCostFamOff,
        AddlSaveFamOff)
      *(StdFctrsPers.OffBAQDepENDs
      + PersCost.Base[B].OffVHA
      * 12.0));
    AddlSaveFamEnl := AtLeastZero
       (Min
      (-AddlCostFamEnl,
        AddlSaveFamEnl)
      *(StdFctrsPers.EnlBAQDepENDs
      + PersCost.Base[B].EnlVHA
      * 12.0));
    AddlCostFamOff := AtLeastZero
       (AddlCostFamOff
      *(StdFctrsPers.OffBAQDepENDs
      + PersCost.Base[B].OffVHA
      * 12.0));
    AddlCostFamEnl := AtLeastZero
       (AddlCostFamEnl
      *(StdFctrsPers.EnlBAQDepENDs
      + PersCost.Base[B].EnlVHA
      * 12.0));
  HousAllow.Off.Base[B].Year[Y] :=
      AnnVHADeltaOff
    + AddlCostFamOff
    - AddlSaveFamOff
    + ChgSingVHARate
    - BachOffVHA;
  HousAllow.Enl.Base[B].Year[Y] :=
      AnnVHADeltaEnl
    + AddlCostFamEnl
    - AddlSaveFamEnl
    - BachEnlVHA;
```

```
        End;
      End;
  END;  { CalcHousAllow }

{*****************************************************************************
 *   Name           : CalcPersonnelCost
 *
 *   Purpose        : Orchestrates calculation of personnel costs by year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcPersonnelCost
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   11May92   AML   Implemented Multi-Basing
 *****************************************************************************}

PROCEDURE CalcPersonnelCost (     CalcFctrsPerPos : CalcPersPosRec;
                                  PersMoves       : CalcMovesRec;
                              VAR PersFinal       : CostSaveNetByYearRec;
                              VAR CostElimMil     : CostElimMilRec;
                              VAR SalSav          : SalSavRec;
                              VAR HousAllowSaved  : Year1ToMaxRealRec;
                              VAR HousAllowSum    : Year1ToMaxRealRec;
                                  Adjust          : Year1ToMaxRealRec);

  VAR
    B, B2, Y,
    Temp, Y2    : INTEGER;
    Buffer      : IOBuffer;
    Enl, Off,
    TempAdjust  : REAL;
    F           : TEXT;
    HousAllow   : HousAllowRec;
```

```
Begin { CalcPersonnelCost }
  FillChar (HousAllowSaved, SizeOf (HousAllowSaved), 0);
  FillChar (HousAllowSum, SizeOf (HousAllowSum), 0);
  FillChar (PersFinal, SizeOf (PersFinal), 0);
  Off := 0.00;
  Enl := 0.00;
  CalcSalSav        (CalcFctrsPerPos, SalSav);
  CalcCostElimMil   (CalcFctrsPerPos, CostElimMil);
  CalcHousAllow     (HousAllow);
  FOR Y := 1 to MaxYears DO
    FOR B := 0 to MaxBases DO
      HousAllowSum.Year[Y] := HousAllowSum.Year[Y]
        + HousAllow.Off.Base[B].Year[Y]
        + HousAllow.Enl.Base[B].Year[Y];
  FOR Y := 1 to MaxYears DO
    Begin
      For B := 0 to MaxBases Do
        Begin
          Off := Off -
              (PosElim.Base[B].Off.Year[Y]
            + PersMoves.Off.From.Base[B].Year[Y])
            * StdFctrsPers.PercOffMarried
            *(PersCost.Base[B].OffVHA
            + StdFctrsPers.OffBAQDepends)
            * ActionEnvir^.Base[B].PercFamLivOnBase;
          Enl := Enl -
              (PosElim.Base[B].Enl.Year[Y]
            + PersMoves.Enl.From.Base[B].Year[Y])
            * StdFctrsPers.PercEnlMarried
            *(PersCost.Base[B].EnlVHA
            + StdFctrsPers.EnlBAQDepENDs)
            * ActionEnvir^.Base[B].PercFamLivOnBase;
        End;
      HousAllowSaved.Year[Y] := Off + Enl
    END;
  assign  (F, Info.ReportDir + 'PERSONNE.RPT');
  SetTextBuf (F, Buffer);
  rewrite (F);
  Temp := 0;
  For B := 0 To MaxBases Do
    Temp := Temp + Screen1Data^.Base[B].CloseYear;
  FOR Y := 1 to MaxYears + 1 DO
    Begin
      If (Y = 1) Or (Y = 5) Then
        Begin
          Title (F, 'PERSONNEL COSTS REPORT [PERSONNE.RPT]', Y Div 5 + 1);
          Writeln (F);
          Writeln (F, Blank:23, '(All values in Dollars)');
        End;
      IF Temp > 0
        Then HousAllowSaved.Year[Y] := 0.0;
```

```
PersFinal.Cost.Year[Y] :=
    HousAllowSum.Year[Y]
  + HousAllowSaved.Year[Y];
PersFinal.Save.Year[Y] :=
    SalSav.Tot.Year[Y]
  + CostElimMil.Tot.Year[Y];
PersFinal.Net.Year[Y] :=
    PersFinal.Cost.Year[Y]
  - PersFinal.Save.Year[Y];
writeln (F);
If Y <= MaxYears Then
  Begin
    Y2 := Y;
    TempAdjust := Adjust.Year[Y];
    Write (F, FiscalYear(Y):16);
  End
Else
  Begin
    Y2 := MaxYears;
    TempAdjust := Adjust.Year[MaxYears];
    Write (F, 'Beyond':16);
  End;
writeln (F, Blank:8, 'Housing Allowance',
  WithCommas((HousAllowSum.Year[Y2] + HousAllowSaved.Year[Y2])
    * TempAdjust,0):20);
If Y <= MaxYears Then
  Begin
    writeln (F, BLANK:22, '- Officer Salary Saved',
      WithCommas(SalSav.Off.Year[Y2] * TempAdjust,0):17);
    writeln (F, BLANK:22, '- Enlisted Salary Saved',
      WithCommas(SalSav.Enl.Year[Y2] * TempAdjust,0):16);
    writeln (F, BLANK:22, '- Civilian Salary Saved',
      WithCommas(SalSav.Civ.Year[Y2] * TempAdjust,0):16);
    writeln (F, BLANK:22, '- Eliminated Military',
      WithCommas(CostElimMil.Tot.Year[Y2] * TempAdjust,0):18);
    writeln (F, BLANK:22, Rept('-', 39));
    writeln (F, BLANK:24, 'Total Personnel Cost',
      WithCommas(PersFinal.Net.Year[Y2] * TempAdjust,0):17)
  End
Else
  Begin
    writeln (F, BLANK:22, '- Officer Salary Saved',
      WithCommas(SalSav.Off.Beyond * TempAdjust,0):17);
    writeln (F, BLANK:22, '- Enlisted Salary Saved',
      WithCommas(SalSav.Enl.Beyond * TempAdjust,0):16);
    writeln (F, BLANK:22, '- Civilian Salary Saved',
      WithCommas(SalSav.Civ.Beyond * TempAdjust,0):16);
    writeln (F, BLANK:22, Rept('-', 39));
    writeln (F, BLANK:24, 'Total Personnel Cost',
      WithCommas(
         (PersFinal.Cost.Year[Y2]
       - SalSav.Tot.Beyond)
```

```
                * TempAdjust,0):17)
            End;
        End;
      PersFinal.Cost.Beyond := PersFinal.Cost.Year[MaxYears];
      PersFinal.Save.Beyond := SalSav.Tot.Beyond;
      PersFinal.Net.Beyond  :=
          PersFinal.Cost.Beyond
        - PersFinal.Save.Beyond;
      close (F)
    END;  { CalcPersonnelCost }

{*****************************************************************************
 *   Name          : CalcFactors2
 *
 *   Purpose       : Calculates numbers of Officers, Enlisted, and Civilian
 *                   persons moving each year into a base and from a base.
 *
 *   Invoked As    : CalcFactors2 (PersMoves)
 *
 *   Return value  :
 *
 *   Files/Tables
 *     Names                 Access    Purpose
 *     ------------------    --------  ----------------------------------------
 *     PERSMOVE.RPT          Write     Writes variables that make up personnel
 *                                     movements (see below)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   11May92   AML   Implemented Multi-Basing
 *   14May92   AML   Increased info in PERSMOVE.RPT
 *****************************************************************************}

PROCEDURE CalcFactors2 (VAR PersMoves     : CalcMovesRec);

  Const
    MigrationDiagramBox : Array [0..2] Of String[10] =
      ('++-¦¦++¦- ', '└-┼┐¦- ', '╚=╫╗‖- ');

  VAR
    B, B2, I, Y,
    TempCloseYear    : INTEGER;
    Buffer           : IOBuffer;
    F                : TEXT;
    MigWin           : String10;
    NetOff,
    NetEnl,
```

```
    NetStu,
    NetCiv            : Base0ToMaxRealRec;
    RunningOff,
    RunningEnl,
    RunningCiv,
    RunningStu,
    TempMil,
    TempTotalInto,
    TempTotalFrom,
    TotalInto,
    TotalFrom,
    TotCivInto,
    TotCivFrom,
    TotStuInto,
    TotStuFrom,
    TotEnlInto,
    TotEnlFrom,
    TotOffInto,
    TotOffFrom        : Real;
    UserEntryError,
    UserEntryError2 : Word;

  Begin { CalcFactors4 }

    { Initialize variables }

    UserEntryError := 0;
    UserEntryError2 := 0;
    FillChar (PersMoves, SizeOf (PersMoves), 0);
{
  Migration Diagram characters are held in a single string.  The order of the
  elements is:  BottomLeft, BottomRight, Horizontal, LeftBreak, RightBreak,
                TopLeft, TopRight, Vertical, and HorizontalBreak)
}
    MigWin := MigrationDiagramBox [Info.HighASCII];

    assign  (F, Info.ReportDir + 'PERSMOVE.RPT');
    SetTextBuf (F, Buffer);
    rewrite (F);

    { Perform calculations }

    FOR B := 0 to MaxBases DO
      Begin
        TotalInto := 0.0;
        TotalFrom := 0.0;
        TotCivInto := 0.0;
        TotCivFrom := 0.0;
        TotStuInto := 0.0;
        TotStuFrom := 0.0;
        TotEnlInto := 0.0;
        TotEnlFrom := 0.0;
```

```
TotOffInto := 0.0;
TotOffFrom := 0.0;
RunningOff := ActionEnvir^.Base[B].BaseTotOffEmploy;
RunningEnl := ActionEnvir^.Base[B].BaseTotEnlEmploy;
RunningCiv := ActionEnvir^.Base[B].BaseTotCivEmploy;
RunningStu := ActionEnvir^.Base[B].BaseTotStuEmploy;

Title (F, 'PERSONNEL MOVEMENT REPORT', B + 1);
Writeln (F, Blank:10, BaseNState(B),
  Blank:(29 - Length (BaseNState(B))),
  'Gains       Losses    Net Gains');
Writeln (F, Blank:39, '-----       ------    ---------');
FOR Y := 1 to MaxYears DO
  Begin
    For B2 := 0 to MaxBases Do
      If Distance^.Base[B].Base[B2] > 0 Then
        Begin
          PersMoves.Civ.Into.Base[B].Year[Y] :=
              PersMoves.Civ.Into.Base[B].Year[Y]
            + PosTrans.Base[B2].Base[B]^.Civ.Year[Y];
          PersMoves.Off.Into.Base[B].Year[Y] :=
              PersMoves.Off.Into.Base[B].Year[Y]
            + PosTrans.Base[B2].Base[B]^.Off.Year[Y];
          PersMoves.Enl.Into.Base[B].Year[Y] :=
              PersMoves.Enl.Into.Base[B].Year[Y]
            + PosTrans.Base[B2].Base[B]^.Enl.Year[Y];
          PersMoves.Stu.Into.Base[B].Year[Y] :=
              PersMoves.Stu.Into.Base[B].Year[Y]
            + PosTrans.Base[B2].Base[B]^.Stu.Year[Y];
          PersMoves.Civ.From.Base[B].Year[Y] :=
              PersMoves.Civ.From.Base[B].Year[Y]
            + PosTrans.Base[B].Base[B2]^.Civ.Year[Y];
          PersMoves.Off.From.Base[B].Year[Y] :=
              PersMoves.Off.From.Base[B].Year[Y]
            + PosTrans.Base[B].Base[B2]^.Off.Year[Y];
          PersMoves.Enl.From.Base[B].Year[Y] :=
              PersMoves.Enl.From.Base[B].Year[Y]
            + PosTrans.Base[B].Base[B2]^.Enl.Year[Y];
          PersMoves.Stu.From.Base[B].Year[Y] :=
              PersMoves.Stu.From.Base[B].Year[Y]
            + PosTrans.Base[B].Base[B2]^.Stu.Year[Y];
        End;
    TempTotalInto :=
        PersMoves.Off.Into.Base[B].Year[Y]
      + PersMoves.Enl.Into.Base[B].Year[Y]
      + PersMoves.Stu.Into.Base[B].Year[Y]
      + PersMoves.Civ.Into.Base[B].Year[Y];
    TotalInto := TotalInto + TempTotalInto;
    TotOffInto := TotOffInto + PersMoves.Off.Into.Base[B].Year[Y];
    TotEnlInto := TotEnlInto + PersMoves.Enl.Into.Base[B].Year[Y];
    TotStuInto := TotStuInto + PersMoves.Stu.Into.Base[B].Year[Y];
    TotCivInto := TotCivInto + PersMoves.Civ.Into.Base[B].Year[Y];
```

```
TempTotalFrom :=
    PersMoves.Off.From.Base[B].Year[Y]
  + PersMoves.Enl.From.Base[B].Year[Y]
  + PersMoves.Stu.From.Base[B].Year[Y]
  + PersMoves.Civ.From.Base[B].Year[Y];
TotalFrom := TotalFrom + TempTotalFrom;
TotOffFrom := TotOffFrom + PersMoves.Off.From.Base[B].Year[Y];
TotEnlFrom := TotEnlFrom + PersMoves.Enl.From.Base[B].Year[Y];
TotStuFrom := TotStuFrom + PersMoves.Stu.From.Base[B].Year[Y];
TotCivFrom := TotCivFrom + PersMoves.Civ.From.Base[B].Year[Y];
writeln (F, Blank:10, FiscalYear(Y):4, ':',
  BLANK:7, 'Civilians',
  WithCommas(PersMoves.Civ.Into.Base[B].Year[Y],0):13,
  WithCommas(PersMoves.Civ.From.Base[B].Year[Y],0):13,
  WithCommas(PersMoves.Civ.Into.Base[B].Year[Y]
           - PersMoves.Civ.From.Base[B].Year[Y],0):13);
writeln (F, BLANK:20, '+ Students',
  WithCommas(PersMoves.Stu.Into.Base[B].Year[Y],0):14,
  WithCommas(PersMoves.Stu.From.Base[B].Year[Y],0):13,
  WithCommas(PersMoves.Stu.Into.Base[B].Year[Y]
           - PersMoves.Stu.From.Base[B].Year[Y],0):13);
writeln (F, BLANK:20, '+ Enlisted',
  WithCommas(PersMoves.Enl.Into.Base[B].Year[Y],0):14,
  WithCommas(PersMoves.Enl.From.Base[B].Year[Y],0):13,
  WithCommas(PersMoves.Enl.Into.Base[B].Year[Y]
           - PersMoves.Enl.From.Base[B].Year[Y],0):13);
writeln (F, BLANK:20, '+ Officers',
  WithCommas(PersMoves.Off.Into.Base[B].Year[Y],0):14,
  WithCommas(PersMoves.Off.From.Base[B].Year[Y],0):13,
  WithCommas(PersMoves.Off.Into.Base[B].Year[Y]
           - PersMoves.Off.From.Base[B].Year[Y],0):13);
writeln (F, BLANK:20, Rept('-', 50));
writeln (F, BLANK:22,   'Total',
  WithCommas(TempTotalInto,0):17,
  WithCommas(TempTotalFrom,0):13,
  WithCommas(TempTotalInto - TempTotalFrom,0):13);
writeln (F);
RunningOff := RunningOff
  + PersMoves.Off.Into.Base[B].Year[Y]
  - PersMoves.Off.From.Base[B].Year[Y]
  - PosElim.Base[B].Off.Year[Y]
  - PosReduc.Base[B].Off.Year[Y];
RunningEnl := RunningEnl
  + PosTrans.Base[B].CaretakeMil.Year[Y]
  + PersMoves.Enl.Into.Base[B].Year[Y]
  - PersMoves.Enl.From.Base[B].Year[Y]
  - PosElim.Base[B].Enl.Year[Y]
  - PosReduc.Base[B].Enl.Year[Y];
RunningStu := RunningStu
  + PersMoves.Stu.Into.Base[B].Year[Y]
  - PersMoves.Stu.From.Base[B].Year[Y];
```

```
          RunningCiv := RunningCiv
            + PosTrans.Base[B].CaretakeCiv.Year[Y]
            + PersMoves.Civ.Into.Base[B].Year[Y]
            - PersMoves.Civ.From.Base[B].Year[Y]
            - PosElim.Base[B].Civ.Year[Y]
            - PosReduc.Base[B].Civ.Year[Y];
          If ((RunningOff < 0) Or (RunningEnl < 0) Or
              (RunningStu < 0) Or (RunningCiv < 0)) And
              Not Odd(UserEntryError Div Round(YtoX(2,B)))
            Then UserEntryError := UserEntryError + Round(YtoX(2,B));
          TempCloseYear := Screen1Data^.Base[B].CloseYear;
          If (TempCloseYear = 0) Or (Screen1Data^.Base[B].Deactivate = 'Y')
            Then TempCloseYear := AbsMaxYears + 1;
          If  (Y > TempCloseYear) And
             ((RunningOff > 0) Or
              (RunningEnl > 0) Or
              (RunningStu > 0) Or
              (RunningCiv > 0)) And
              Not Odd(UserEntryError2 Div Round(YtoX(2,B)))
            Then UserEntryError2 := UserEntryError2 + Round(YtoX(2,B));
        End;
      NetOff.Base[B] := TotOffInto - TotOffFrom;
      NetEnl.Base[B] := TotEnlInto - TotEnlFrom;
      NetStu.Base[B] := TotStuInto - TotStuFrom;
      NetCiv.Base[B] := TotCivInto - TotCivFrom;
      Writeln (F, Blank:10, 'TOTAL:      Civilians',
        WithCommas(TotCivInto,0):13,
        WithCommas(TotCivFrom,0):13,
        WithCommas(NetCiv.Base[B],0):13);
      writeln (F, BLANK:20, '+ Students',
        WithCommas(TotStuInto,0):14,
        WithCommas(TotStuFrom,0):13,
        WithCommas(NetStu.Base[B],0):13);
      writeln (F, BLANK:20, '+ Enlisted',
        WithCommas(TotEnlInto,0):14,
        WithCommas(TotEnlFrom,0):13,
        WithCommas(NetEnl.Base[B],0):13);
      writeln (F, BLANK:20, '+ Officers',
        WithCommas(TotOffInto,0):14,
        WithCommas(TotOffFrom,0):13,
        WithCommas(NetOff.Base[B],0):13);
      writeln (F, BLANK:20, Rept('-', 50));
      writeln (F, BLANK:22,   'Total',
        WithCommas(TotalInto,0):17,
        WithCommas(TotalFrom,0):13,
        WithCommas(TotalInto - TotalFrom,0):13);
    End;
  For B := 0 To MaxBases Do
    Begin
      For Y := 1 To MaxYears Do
        Begin
```

```
      NetOff.Base[B] := NetOff.Base[B]
        - PosReduc.Base[B].Off.Year[Y]
        - PosElim.Base[B].Off.Year[Y];
      NetEnl.Base[B] := NetEnl.Base[B]
        + PosTrans.Base[B].CaretakeMil.Year[Y]
        - PosReduc.Base[B].Enl.Year[Y]
        - PosElim.Base[B].Enl.Year[Y];
      NetCiv.Base[B] := NetCiv.Base[B]
        + PosTrans.Base[B].CaretakeCiv.Year[Y]
        - PosReduc.Base[B].Civ.Year[Y]
        - PosElim.Base[B].Civ.Year[Y];
    End;
  If (B Div 3) * 3 = B Then
    Title (F, 'PERSONNEL MOVEMENT REPORT', MaxBases + 2 + (B Div 3));
  Writeln (F);
  Writeln (F, Blank:10, MigWin[6],
    Rept(MigWin[3],33), MigWin[7]);
  Writeln (F, Blank:10, MigWin[8],
    BaseNState(B),
    Blank:(33-Length(BaseNState(B))),
    MigWin[8]);
  Writeln (F, Blank:10, MigWin[4],
    Rept(MigWin[9],33), MigWin[5]);
  Writeln (F, Blank:10, MigWin[8], Blank:12,
    'START     END  CHANGE', MigWin[8]);
  Writeln (F, Blank:10, MigWin[8], 'Officers',
    WithCommas(ActionEnvir^.Base[B].BaseTotOffEmploy,0):9,
    WithCommas(ActionEnvir^.Base[B].BaseTotOffEmploy
      + NetOff.Base[B],0):8,
    WithCommas(NetOff.Base[B],-1):8, MigWin[8]);
  Writeln (F, Blank:10, MigWin[8], 'Enlisted',
    WithCommas(ActionEnvir^.Base[B].BaseTotEnlEmploy,0):9,
    WithCommas(ActionEnvir^.Base[B].BaseTotEnlEmploy
      + NetEnl.Base[B],0):8,
    WithCommas(NetEnl.Base[B],-1):8, MigWin[8]);
  Writeln (F, Blank:10, MigWin[8], 'Students',
    WithCommas(ActionEnvir^.Base[B].BaseTotStuEmploy,0):9,
    WithCommas(ActionEnvir^.Base[B].BaseTotStuEmploy
      + NetStu.Base[B],0):8,
    WithCommas(NetStu.Base[B],-1):8, MigWin[8]);
  TempMil := ActionEnvir^.Base[B].BaseTotOffEmploy
           + ActionEnvir^.Base[B].BaseTotEnlEmploy
           + ActionEnvir^.Base[B].BaseTotStuEmploy;
  Writeln (F, Blank:10, MigWin[8], 'TOTAL MIL',
    WithCommas(TempMil,0):8,
    WithCommas(TempMil
             + NetOff.Base[B]
             + NetEnl.Base[B]
             + NetStu.Base[B],0):8,
    WithCommas(NetOff.Base[B]
        + NetEnl.Base[B]
        + NetStu.Base[B],-1):8, MigWin[8]);
```

```
      Writeln (F, Blank:10, MigWin[8], 'Civilians',
        WithCommas(ActionEnvir^.Base[B].BaseTotCivEmploy,0):8,
        WithCommas(ActionEnvir^.Base[B].BaseTotCivEmploy
          + NetCiv.Base[B],0):8,
        WithCommas(NetCiv.Base[B],-1):8, MigWin[8]);
      Writeln (F, Blank:10, MigWin[8], 'TOTAL',
        WithCommas(ActionEnvir^.Base[B].BaseTotCivEmploy + TempMil,0):12,
        WithCommas(ActionEnvir^.Base[B].BaseTotCivEmploy
                 + TempMil
                 + NetOff.Base[B]
                 + NetEnl.Base[B]
                 + NetStu.Base[B]
                 + NetCiv.Base[B],0):8,
        WithCommas(NetOff.Base[B]
            + NetEnl.Base[B]
            + NetStu.Base[B]
            + NetCiv.Base[B],-1):8, MigWin[8]);
      Writeln (F, Blank:10, MigWin[1],
        Rept(MigWin[3],33), MigWin[2]);
    End;
  close (F);
  If UserEntryError > 0 Then
    Begin
      ErrorOut
        ('The following base(s), at some point, have a negative number');
      ErrorOut ('of Officers, Enlisted, Civilians, and/or Students:');
      ErrorList (UserEntryError);
    End;
  If UserEntryError2 > 0 Then
    Begin
      ErrorOut ('The following base(s) still have Officers, Enlisted,');
      ErrorOut ('Civilians, and/or Students present after Closing:');
      ErrorList (UserEntryError2);
    End;
END;  { CalcFactors2 }
```

```
{*************************************************************************
 *   Name           : CalcFactors3
 *   Purpose        : Calculates number of positions eliminated for each year
 *                    and civilian personnel actions.
 *   Invoked As     : CalcFactors3 (CalcFctrsPerPos, CalcPersAct)
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     PERSONEL.RPT           Write     Writes variables that make up personnel
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   11May92   AML   Implemented Multi-Basing
 *************************************************************************}

Procedure CalcFactors3 (     PersMoves        : CalcMovesRec;
                        Var CalcFctrsPerPos : CalcPersPosRec;
                        Var CalcPersAct      : CalcPersActRec);

  Const
    XXXXX = 0;
{   XXXXX was previously a by-base percentage of civilians not entitled to PPS }

  Var
    B, B2, Y          : Integer;
    Buffer            : IOBuffer;
    F                 : Text;
    ConsistCheck,
    NumAffectOver50,
    TempPersVolRIFs,
    TempPPSAllowed,
    TempVolRIFPPS     : Real;
    UserEntryError    : Word;

  Begin { CalcFactors3 }

    { Initialize variables }

    UserEntryError := 0;
    ConsistCheck :=
        StdFctrsPers.CivEarlyRetire
      + StdFctrsPers.CivTurnover
      + StdFctrsPers.PercCivsQuitting;
    FillChar (CalcFctrsPerPos, SizeOf (CalcFctrsPerPos), 0);
    FillChar (CalcPersAct, SizeOf (CalcPersAct), 0);

    { Perform calculations }
```

```
For Y := 1 to MaxYears Do
  Begin
    NumAffectOver50 := 0.0;
    TempPersVolRIFs := 0.0;
    TempPPSAllowed  := 0.0;
    TempVolRIFPPS   := 0.0;

    For B := 0 To MaxBases Do
      Begin
        If (ConsistCheck + ActionEnvir^.Base[B].VolRIFPerc > 1) And (Y = 1)
          Then UserEntryError := UserEntryError + Round(YtoX(2,B));
        CalcFctrsPerPos.OffPosReduc.Year[Y] :=
            CalcFctrsPerPos.OffPosReduc.Year[Y]
          + PosReduc.Base[B].Off.Year[Y];
        CalcFctrsPerPos.EnlPosReduc.Year[Y] :=
            CalcFctrsPerPos.EnlPosReduc.Year[Y]
          + PosReduc.Base[B].Enl.Year[Y];
        CalcFctrsPerPos.CivPosReduc.Year[Y] :=
            CalcFctrsPerPos.CivPosReduc.Year[Y]
          + PosReduc.Base[B].Civ.Year[Y];
        CalcFctrsPerPos.OffPosElim.Year[Y] :=
            CalcFctrsPerPos.OffPosElim.Year[Y]
          + PosElim.Base[B].Off.Year[Y];
        CalcFctrsPerPos.EnlPosElim.Year[Y] :=
            CalcFctrsPerPos.EnlPosElim.Year[Y]
          + PosElim.Base[B].Enl.Year[Y];
        CalcFctrsPerPos.CivPosElim.Year[Y] :=
            CalcFctrsPerPos.CivPosElim.Year[Y]
          + PosElim.Base[B].Civ.Year[Y];
        CalcPersAct.Year[Y].NumNeeded :=
            CalcPersAct.Year[Y].NumNeeded
          + PersMoves.Civ.From.Base[B].Year[Y];
        TempPersVolRIFs :=
            TempPersVolRIFs
          +(PosElim.Base[B].Civ.Year[Y]
          * ActionEnvir^.Base[B].VolRIFPerc);
        TempPPSAllowed :=
            TempPPSAllowed
          + PosElim.Base[B].Civ.Year[Y]
          *(1 - XXXXX);
        TempVolRIFPPS :=
            TempVolRIFPPS
          + PosElim.Base[B].Civ.Year[Y]
          *(1 - XXXXX)
          * ActionEnvir^.Base[B].VolRIFPerc;
        For B2 := 0 To MaxBases Do
          If Distance^.Base[B].Base[B2] >= 50.0 Then
            Begin
              NumAffectOver50 :=
                  NumAffectOver50
                + PosTrans.Base[B].Base[B2]^.Civ.Year[Y];
```

```
            TempPersVolRIFs :=
                TempPersVolRIFs
              +(PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              * ActionEnvir^.Base[B].VolRIFPerc);
            TempPPSAllowed :=
                TempPPSAllowed
              + PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              *(1 - XXXXX);
            TempVolRIFPPS :=
                TempVolRIFPPS
              + PosTrans.Base[B].Base[B2]^.Civ.Year[Y]
              *(1 - XXXXX)
              * ActionEnvir^.Base[B].VolRIFPerc;
          End;
  End;
NumAffectOver50 :=
    NumAffectOver50
  + Round(CalcFctrsPerPos.CivPosElim.Year[Y]);
CalcPersAct.Year[Y].NumAffected :=
    Round(CalcFctrsPerPos.CivPosElim.Year[Y])
  + CalcPersAct.Year[Y].NumNeeded;
CalcPersAct.Year[Y].NumRetiring :=
  Round (NumAffectOver50 * StdFctrsPers.CivEarlyRetire);
CalcPersAct.Year[Y].NumAttrited :=
  Round (NumAffectOver50 * StdFctrsPers.CivTurnover);
CalcPersAct.Year[Y].CivsQuitting := Round
   (NumAffectOver50
  * StdFctrsPers.PercCivsQuitting);
CalcPersAct.Year[Y].PersVolRIFs := Round (TempPersVolRIFs);
CalcPersAct.Year[Y].NumRemain :=
    CalcPersAct.Year[Y].NumAffected
  - CalcPersAct.Year[Y].NumRetiring
  - CalcPersAct.Year[Y].NumAttrited
  - CalcPersAct.Year[Y].CivsQuitting
  - CalcPersAct.Year[Y].PersVolRIFs;
CalcPersAct.Year[Y].ShortOverage :=
    CalcPersAct.Year[Y].NumRemain
  - CalcPersAct.Year[Y].NumNeeded;
CalcPersAct.Year[Y].PersNewHires :=
  Round(AtLeastZero (-CalcPersAct.Year[Y].ShortOverage));
CalcPersAct.Year[Y].PriPlacement := Round(Min
   (Round(StdFctrsPers.PriPlacementPerc
  * TempPPSAllowed),
    CalcPersAct.Year[Y].ShortOverage))
  + CalcPersAct.Year[Y].PersNewHires;
CalcPersAct.Year[Y].PersInvolRIFs := Round (AtLeastZero
   (CalcPersAct.Year[Y].ShortOverage
  - CalcPersAct.Year[Y].PriPlacement));
CalcPersAct.Year[Y].NumMoving := Round (Min
  (CalcPersAct.Year[Y].NumRemain, CalcPersAct.Year[Y].NumNeeded));
```

```
    CalcPersAct.Year[Y].NumRIFTed :=
        CalcPersAct.Year[Y].PersVolRIFs
      + CalcPersAct.Year[Y].PersInvolRIFs;
    CalcPersAct.Year[Y].TotalPPS :=
        CalcPersAct.Year[Y].PriPlacement
      + CalcPersAct.Year[Y].NumAttrited;
    CalcPersAct.Year[Y].Total :=
        CalcPersAct.Year[Y].NumRetiring
      + CalcPersAct.Year[Y].NumRIFted
      + CalcPersAct.Year[Y].TotalPPS
      + CalcPersAct.Year[Y].NumNeeded;
    CalcPersAct.Year[Y].PercOfNeed :=
      Zerocheck (CalcPersAct.Year[Y].NumMoving,
                 CalcPersAct.Year[Y].NumNeeded);
    CalcPersAct.Year[Y].RIFPay := (Round
       (TempVolRIFPPS)
      + CalcPersAct.Year[Y].PersInvolRIFs)
      * StdFctrsPers.CivSal
      * StdFctrsPers.CivRIFPayFctr;
    CalcPersAct.Year[Y].EarlyRetire := Round
       (TempPPSAllowed
      * StdFctrsPers.CivEarlyRetire)
      *(StdFctrsPers.CivSal
      * StdFctrsPers.CivRetirePayFctr)
  End;

{ Write output file }

assign  (F, Info.ReportDir + 'PERSONEL.RPT');
SetTextBuf (F, Buffer);
rewrite (F);
Title(F, 'PERSONNEL REALIGNMENTS [PERSONEL.RPT]', -1);
write (F, BLANK:30, 'Rate');
For Y := 1 To MaxYears Do
  Write (F, FiscalYear(Y):6);
Writeln (F);
write (F, BLANK:30);
For Y := 0 To MaxYears Do
  Write (F, '----  ');
writeln (F);
write   (F, Blank:10, 'Civs Eliminated         ');
FOR Y := 1 to MaxYears DO
   write (F, CalcPersAct.Year[Y].NumAffected:6);
writeln (F);
write   (F, Blank:10, 'Civs Early Retir*');
write (F, (StdFctrsPers.CivEarlyRetire*100.0):6:2, '%');
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumRetiring:6);
writeln (F);
write   (F, Blank:10, 'Civ Turnover*    ');
write (F, (StdFctrsPers.CivTurnover*100.0):6:2, '%');
```

```
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumAttrited:6);
writeln (F);
write   (F, Blank:10, 'Civs Quitting*   ');
write (F, (StdFctrsPers.PercCivsQuitting*100.0):6:2, '%');
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].CivsQuitting:6);
writeln (F);
write   (F, Blank:10, 'Civs Not Moving*+       ');
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].PersVolRIFs:6);
writeln (F);
writeln (F);
write   (F, Blank:10, 'Civs Avail to Move      ');
FOR Y := 1 To MaxYears DO
  write (F, CalcPersAct.Year[Y].NumRemain:6);
writeln (F);
write   (F, Blank:10, 'Civs Required', Blank:11);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumNeeded:6);
writeln (F);
write   (F, Blank:10, 'Shortfall(-)/Overage(+) ');
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].ShortOverage:6);
writeln (F);
write   (F, Blank:10, 'New Civs Hired', Blank:10);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].PersNewHires:6);
writeln (F);
writeln (F);
write   (F, Blank:10, 'Prio Placement#  ');
write (F, (StdFctrsPers.PriPlacementPerc*100.0):6:2, '%');
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].PriPlacement:6);
writeln (F);
write   (F, Blank:10, 'Civs Invol RIFed        ');
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].PersInvolRIFs:6);
writeln (F);
write   (F, Blank:10, 'Civs Moved', Blank:14);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumMoving:6);
writeln (F);
writeln (F);
write   (F, Blank:10, 'Total Retired', Blank:11);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumRetiring:6);
writeln (F);
write   (F, Blank:10, 'Total RIFed', Blank:13);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumRIFted:6);
writeln (F);
```

```
write   (F, Blank:10, 'Total PPS#', Blank:14);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].TotalPPS
            - CalcPersAct.Year[Y].NumAttrited:6);
writeln (F);
write   (F, Blank:10, 'Total Hired', Blank:13);
FOR Y := 1 To MaxYears DO
   write (F, CalcPersAct.Year[Y].NumNeeded:6);
writeln (F);
writeln (F);
writeln (F, Blank:10,
  '* In moves of less than 50 miles Early Retirements, Civilian');
writeln (F, Blank:12,
  'Turnover, Civilians Quitting, and Civilians Not Willing to');
writeln (F, Blank:12, 'Move are not calculated.');
writeln (F);
writeln (F, Blank:10,
  '+ The rate of Civilians Not Willing to Move varies from base');
writeln (F, Blank:12,
  'to base.');
writeln (F);
writeln (F, Blank:10,
  '# Not all Priority Placements involve a PCS. The rate of PPS');
writeln (F, Blank:12, 'placements involving PCS is',
  (StdFctrsPers.PPSInvolvePCS * 100.0):6:2, '%');
close (F);

{ CalcPersAct.Year[Y].TotalPPS is now being multiplied by
  StdFctrsPers.PPSInvolvePCS to find out how many people applying for PPS
  require PCS (Before we needed to know how many applied, now we're going to
  find out how many we have to pay for). }

For Y := 1 TO MaxYears DO
  CalcPersAct.Year[Y].TotalPPS := Round(
     (CalcPersAct.Year[Y].TotalPPS
    - CalcPersAct.Year[Y].NumAttrited)
    * StdFctrsPers.PPSInvolvePCS)
    + CalcPersAct.Year[Y].NumAttrited;

If UserEntryError > 0
  Then If ConsistCheck > 1 Then
    Begin
      ErrorOut ('(%Retiring + %Turnover + %Quitting) > 100%');
      ErrorOut ('Verify that the Personnel Standard Factors are correct.');
      ErrorOut ('');
    End
  Else
    Begin
      ErrorOut ('The following base(s) have');
      ErrorOut ('(%Retiring + %Turnover + %Quitting + %VolRIF) > 100% :');
      ErrorList (UserEntryError);
    End;
```

```
  END;  { CalcFactors3 }

{*************************************************************************
 *   Name          : CalcFactors4
 *
 *   Purpose       : Calculates summary of people moving each year and
 *                   percentage of them each year in respect to total number
 *                   moving over all years BY BASE.
 *
 *   Invoked As    : CalcFactors4 (PersMoves, PersMovesSum)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   11May92  AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcFactors4 (    PersMoves    : CalcMovesRec;
                        VAR PersMovesSum : PersMovesSumRec);

  VAR
    B, Y,
    Temp   : INTEGER;
    Buffer : IOBuffer;
    F      : TEXT;
    MovIn,
    MovOut : Base0Year1ToMaxRealRec;
    TotIn,
    TotOut : Base0ToMaxRealRec;

  Begin { CALCFACTORS4B }

    { Initialize variables }
    FillChar (PersMovesSum, SizeOf (PersMovesSum), 0);
    FillChar (MovIn, SizeOf (MovIn), 0);
    FillChar (MovOut, SizeOf (MovOut), 0);
    FillChar (TotIn, SizeOf (TotIn), 0);
    FillChar (TotOut, SizeOf (TotOut), 0);

    { Perform calculations }

    FOR B := 0 to MaxBases DO
      Begin
        FOR Y := 1 to MaxYears DO
          Begin
            MovIn.Base[B].Year[Y] :=
                PersMoves.Off.Into.Base[B].Year[Y]
              + PersMoves.Enl.Into.Base[B].Year[Y]
```

```
          + PersMoves.Civ.Into.Base[B].Year[Y]
          + PersMoves.Stu.Into.Base[B].Year[Y];
        TotIn.Base[B] := TotIn.Base[B] + MovIn.Base[B].Year[Y];
        MovOut.Base[B].Year[Y] :=
            PersMoves.Off.From.Base[B].Year[Y]
          + PersMoves.Enl.From.Base[B].Year[Y]
          + PersMoves.Civ.From.Base[B].Year[Y]
          + PersMoves.Stu.From.Base[B].Year[Y]
          + PosElim.Base[B].Off.Year[Y]
          + PosElim.Base[B].Enl.Year[Y]
          + PosElim.Base[B].Civ.Year[Y];
        TotOut.Base[B] := TotOut.Base[B] + MovOut.Base[B].Year[Y];
      End;
    If Screen1Data^.Base[B].CloseYear > 0
      Then Temp := Screen1Data^.Base[B].CloseYear
      Else Temp := LastYear;
    For Y := 1 To MaxYears Do
      If TotOut.Base[B] > 0
        Then PersMovesSum.Base[B].ShutDn.Year[Y] := ZeroCheck
          (MovOut.Base[B].Year[Y], TotOut.Base[B])
        Else If Y <= Temp
          Then PersMovesSum.Base[B].ShutDn.Year[Y] := 1.0 / Temp;
    If Temp = MaxYears
      Then Dec (Temp);
    For Y := 1 To MaxYears Do
      If TotIn.Base[B] > 0
        Then PersMovesSum.Base[B].Perc.Year[Y] := ZeroCheck
          (MovIn.Base[B].Year[Y], TotIn.Base[B])
        Else If Y <= Temp
          Then PersMovesSum.Base[B].Perc.Year[Y] := 1.0 / Temp;
    PersMovesSum.Base[B].Perc.Year[1] :=
        PersMovesSum.Base[B].Perc.Year[1]
      + PersMovesSum.Base[B].Perc.Year[2];
    If Temp > 1
      Then For Y := 2 To Temp Do
        PersMovesSum.Base[B].Perc.Year[Y] :=
          PersMovesSum.Base[B].Perc.Year[Y+1];
    If Temp < MaxYears - 1 Then
      For Y := Temp + 2 To MaxYears Do
        PersMovesSum.Base[B].Perc.Year[Temp] :=
            PersMovesSum.Base[B].Perc.Year[Temp]
          + PersMovesSum.Base[B].Perc.Year[Y];
    For Y := Temp + 1 To MaxYears Do
      PersMovesSum.Base[B].Perc.Year[Y] := 0.0;
  End;

assign  (F, Info.ReportDir + 'PERSPERC.RPT');
SetTextBuf (F, Buffer);
rewrite (F);
```

```
  For B := 0 To MaxBases Do
    Begin
      If (B Div 3) * 3 = B
        Then Title (F, 'PERSONNEL YEARLY PERCENTAGES',B Div 3 + 1)
        Else Writeln (F);
      writeln (F);
      writeln (F, Blank:10, 'Base:  ', BaseNState(B));
      writeln (F);
      writeln (F, BLANK:20,
        'Moving In       MilCon    Move Out/Elim     ShutDn');
      writeln (F, BLANK:10,
        'Year   Total   Percent   TimPhas   Total   Percent   TimPhas');
      writeln (F, Blank:10,
        '----   -----   -------   -------   -----   -------   -------');

      FOR Y := 1 to MaxYears DO
        writeln (F, FiscalYear(Y):14,
          WithCommas(MovIn.Base[B].Year[Y],0):8,
          (ZeroCheck(MovIn.Base[B].Year[Y],TotIn.Base[B]) * 100.0):9:2, '%',
          (PersMovesSum.Base[B].Perc.Year[Y] * 100.0):9:2, '%',
          WithCommas(MovOut.Base[B].Year[Y],0):8,
          (ZeroCheck(MovOut.Base[B].Year[Y],TotOut.Base[B]) * 100.0):9:2, '%',
          (PersMovesSum.Base[B].ShutDn.Year[Y] * 100.0):9:2, '%');
      writeln (F, BLANK:17,
        '-----   -------   -------   -----   -------   -------');
      writeln (F, BLANK:10, 'TOTALS', TotIn.Base[B]:6:0,
        ZeroCheck(TotIn.Base[B]*100.0,TotIn.Base[B]):9:2,
        '%   100.00%', TotOut.Base[B]:8:0,
        ZeroCheck(TotOut.Base[B]*100.0,TotOut.Base[B]):9:2, '%   100.00%');
    End;
  close (F)

END;  { CalcFactors4 }
```

```
{*************************************************************************
 *   Name           : CalcPersFactors
 *
 *   Purpose        : Orchestrates calculations of all personnel movement
 *                    numbers and factors.
 *
 *   Notes          :
 *
 *   Invoked As     : CalcPersFactors (CalcFctrsPerPos, CalcPersAct, PersMoves,
 *                                     PersMovesSum)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   11May92   AML   Implemented Multi-Basing
 *************************************************************************}

PROCEDURE CalcPersFactors (VAR CalcFctrsPerPos : CalcPersPosRec;
                           VAR CalcPersAct     : CalcPersActRec;
                           VAR PersMoves       : CalcMovesRec;
                           VAR PersMovesSum    : PersMovesSumRec);

  Begin { CalcPersFactors }
    CalcFactors2 (PersMoves);
    CalcFactors3 (PersMoves, CalcFctrsPerPos, CalcPersAct);
    CalcFactors4 (PersMoves, PersMovesSum);
  END;  { CalcPersFactors }

END. { UNIT }
```

```
UNIT CDEC;

{$F+,O+,S-,D+,L+}

Interface

Uses
  Dos;

Const
  Version          = '4.04';
  GroupVersion     = '1.20';
  SetUpVersion     = '1.10';
  Blank            = ' ';

  AbsMaxBases      = 14;
  AbsMaxYears      = 6;
  NumYears         = 20;

  cmNextScr        = 9901;
  cmPrevScr        = 9902;

  cmAbout          = 101; { Help / About COBRA                    }
  cmViewHelp       = 102; { Help / View Help                      }
  cmPrintHelp      = 103; { Help / Print Help                     }
  cmViewUsed       = 104; { Help / View Used                      }
  cmCalculator     = 105; { Help / Calculator                     }
  cmCalendar       = 106; { Help / Calendar                       }

  cmLoadPrev       = 111; { File / Load Previous Data Set         }
  cmSaveData       = 112; { File / Save Current Data Set          }
  cmNewData        = 113; { File / Clear Data                     }
  cmDeleteData     = 114; { File / Delete Saved Data Set          }
  cmLoadSF         = 115; { File / Load Standard Factors          }
  cmSaveSF         = 116; { File / Save Standard Factors          }
  cmNewSF          = 117; { File / Clear Standard Factors         }
  cmCobraFiles     = 118; { File / File Directory                 }
  cmExitCobra      = 119; { File / Exit COBRA                     }

  cmLoadDB         = 121; { DataBase / Load Base(s)               }
  cmSaveDB         = 122; { DataBase / Save Base(s)               }
  cmDistanceDB     = 123; { DataBase / Distance                   }

  cmScreenOne      = 131; { Enter Data / General Scenario         }
  cmScreenTwo      = 132; { Enter Data / Distance Table           }
  cmScreenThree    = 133; { Enter Data / Transfer Table           }
  cmScreenFour     = 134; { Enter Data / Base Info (Static)       }
  cmScreenFive     = 135; { Enter Data / Base Info (Dynamic)      }
  cmScreenSix      = 136; { Enter Data / Base Info (Personnel)    }
  cmScreenSeven    = 137; { Enter Data / Base Info (MilCon)       }
  cmScreenEight    = 138; { Enter Data / Base Info (IndFund)      }
  cmScreenNine     = 139; { Enter Data / General Footnotes        }
```

```
cmStdFactors1  = 141; { Enter Data / Personnel Factors      }
cmStdFactors2  = 142; { Enter Data / Facility Factors       }
cmStdFactors3  = 143; { Enter Data / Transportation Factors }
cmStdFactors4  = 144; { Enter Data / Construction Factors   }
cmDeleteBase   = 145; { Enter Data / Delete Base            }

cmExecute      = 151; { Reports / Execute                   }
cmMakeReport   = 152; { Reports / Base Report(s)            }
cmViewReport   = 153; { Reports / View Report               }
cmPrintReport  = 154; { Reports / Print Report              }
cmDeleteReport = 155; { Reports / Delete Report             }
cmViewGroup    = 156; { Reports / View Group                }
cmPrintGroup   = 157; { Reports / Print Group               }
cmDeleteGroup  = 158; { Reports / Delete Group              }

cmDosShell     = 161; { Utilities / DOS Shell               }
cmChangeDir    = 162; { Utilities / Change Dir              }
hlChangeDir    = 162; { Utilities / Change Dir [hist list]  }
cmPrinterSetup = 163; { Utilities / Printer Setup           }
cmLoadOld      = 164; { Utilities / Load Old Data           }

cmAutoLoad     = 171; { (filename passed from command line) }
cmRedraw       = 172; { redraw screen (in case of problem)  }
cmVideoToggle  = 173; { switch B&W/Mono/Color display       }

mdAutoLoad =  0;
mdDataBase =  1;
mdDataFile =  2;
mdDelBase  =  3;
mdDistance =  4;
mdGroup    =  5;
mdHelp     =  6;
mdNewSF    =  7;
mdNull     =  8;
mdOldData  =  9;
mdReport   = 10;
mdStdFctrs = 11;
mdView     = 12;

hcAboutCobra       = 1006;
hcApproReports     = 1071;
hcAutoStart        = 1063;
hcCalculator       = 1010;
hcCalendar         = 1011;
hcCascadeTile      = 1037;
hcCobraFiles       = 1014;
hcCOBSUM           = 1067;
hcConCategories    = 1057;
hcCreateNew        = 1015;
hcDBaseMenu        = 1058;
hcDeleteBase       = 1033;
hcDeleteData       = 1016;
```

```
hcDeleteGroup       = 1029;
hcDeleteReport      = 1028;
hcDescription       = 1052;
hcDistanceDB        = 1061;
hcENTER             = 1062;
hcEnterData         = 1021;
hcError             = 1068;
hcExecute           = 1024;
hcExitCobra         = 1020;
hcFileMenu          = 1002;
hcFilesUsed         = 1009;
hcHelpMenu          = 1001;
hcHelpNames         = 1066;
hcINPUTDAT          = 1070;
hcInputScreens      = 1022;
hcLoadDB            = 1059;
hcLoadOld           = 1032;
hcLoadPrevious      = 1012;
hcLoadSF            = 1017;
hcLoadWindow        = 1055;
hcMakeReport        = 1034;
hcMILCONAS          = 1075;
hcNewSF             = 1019;
hcNext              = 1038;
hcNPV               = 1069;
hcNullDataEight     = 1054;
hcNullDataThree     = 1053;
hcONETIMECOST       = 1072;
hcOtherReports      = 1076;
hcOverheadReports   = 1074;
hcPersonnelReports  = 1073;
hcPrinterSetup      = 1031;
hcPrintGroup        = 1027;
hcPrintHelp         = 1008;
hcPrintReport       = 1026;
hcReportMenu        = 1003;
hcReportNames       = 1065;
hcSaveCurrent       = 1013;
hcSaveDB            = 1060;
hcSaveSF            = 1018;
hcSavewindow        = 1056;
hcScreenEight       = 1046;
hcScreenFive        = 1043;
hcScreenFour        = 1042;
hcScreenNine        = 1047;
hcScreenOne         = 1039;
hcScreenSeven       = 1045;
hcScreenSix         = 1044;
hcScreenThree       = 1041;
hcScreenTwo         = 1040;
hcSFScreens         = 1023;
hcShellCD           = 1030;
```

```
  hcSizeMove          = 1035;
  hcStdFctrsFour      = 1051;
  hcStdFctrsOne       = 1048;
  hcStdFctrsThree     = 1050;
  hcStdFctrsTwo       = 1049;
  hcUtilityMenu       = 1004;
  hcViewGroup         = 1064;
  hcViewHelp          = 1007;
  hcViewReport        = 1025;
  hcWindowMenu        = 1005;
  hcZoom              = 1036;

TYPE
  String02 = String[2];
  String04 = String[4];
  String05 = String[5];
  String10 = String[10];
  String17 = String[17];
  String20 = String[20];
  String60 = String[60];

  Year0ToMaxInteger = Array [0..AbsMaxYears] of Integer;
  Year1ToMaxInteger = Array [1..AbsMaxYears] of Integer;
  Year0ToMaxReal    = Array [0..AbsMaxYears] of Real;
  Year1ToMaxReal    = Array [1..AbsMaxYears] of Real;
  Year1ToNumReal    = Array [1..NumYears]    of Real;
  Year0ToMaxIntegerRec      = Record
                                 Year   : Year0ToMaxInteger
                              End;
  Year1ToMaxIntegerRec      = Record
                                 Year   : Year1ToMaxInteger
                              End;
  Year0ToMaxRealRec         = Record
                                 Year   : Year0ToMaxReal
                              End;
  Year1ToMaxRealRec         = Record
                                 Year   : Year1ToMaxReal
                              End;
  Year1ToNumRealRec         = Record
                                 Year   : Year1ToNumReal
                              End;
  YearAndBeyondRealRec      = Record
                                 Year   : Year1ToMaxReal;
                                 Beyond : Real
                              End;

  Base0ToMaxReal            = Array [0..AbsMaxBases] of Real;
  Base0ToMaxRealRec         = Record
                                 Base : Base0ToMaxReal
                              End;
  PBase0ToMaxRealRec        = ^Base0ToMaxRealRec;
```

```
Base0Year1ToMaxRealRec     = Record
                                Base : Array [0..AbsMaxBases]
                                       Of Year1ToMaxRealRec
                             End;
Base0YearAndBeyondRealRec = Record
                                Base : Array [0..AbsMaxBases]
                                       Of YearAndBeyondRealRec
                             End;
BaseArray                  = Array [0..14] of String20;


CostSaveNetByYearRec = Record
                          Cost,
                          Save,
                          Net    : YearAndBeyondRealRec;
                       End;

ActionEnvirSubRec2 = Record
                        HeavyOrSpecVehic,
                        MilLightVehic,
                        MoveMissionEqpmt,
                        MoveSupportEqpmt    : Year1ToMaxRealRec;
                     End;
ActionEnvirSubRec  = Record
                        AcreOnBase,
                        BaseFacilSqFt,
                        BaseTotCivEmploy,
                        BaseTotOffEmploy,
                        BaseTotEnlEmploy,
                        BaseTotStuEmploy,
                        EnlHousUnitsVacant : Real;
                        EnvirMitigReqd     : Year1ToMaxRealRec;
                        OffHousUnitsVacant : Real;
                        OneTimeMovingCosts,
                        OneTimeUniqueCosts : Year1ToMaxRealRec;
                        FamHousShutDn,
                        PercFamLivOnBase,
                        VolRIFPerc,
                        ChampusToMedicare  : Real;
                        Base               : Array [0..AbsMaxBases]
                                             Of ^ActionEnvirSubRec2;
                     End;
ActionEnvirRec     = Record
                        Base               : Array [0..AbsMaxBases]
                                             Of ActionEnvirSubRec;
                     End;

ApproReportMilConRec = Record
                          Hous,
                          MilCon         : Year1ToMaxRealRec
                       End;
```

```
ApproReportSubRec33  = Record
                          MilSalary,
                          MilMoving,
                          Other          : YearAndBeyondRealRec
                       End;
ApproReportSubRec32  = Record
                          RPMA,
                          BOS,
                          OperatingIDF,
                          CivSalary,
                          CivMoving,
                          Other          : YearAndBeyondRealRec
                       End;
ApproReportSubRec11  = Record
                          Construction,
                          Operations     : YearAndBeyondRealRec
                       End;
ApproReportSubRec3   = Record
                          MilCon         : YearAndBeyondRealRec;
                          FamHous        : ApproReportSubRec11;
                          OM             : ApproReportSubRec32;
                          MilPers        : ApproReportSubRec33;
                          Procure,
                          HomeownAsst,
                          EnvirMitig,
                          InfoMgt,
                          Other,
                          Land,
                          Champus,
                          MiscRecur,
                          Total          : YearAndBeyondRealRec
                       End;
ApproReportSubRec24  = Record
                          Mission,
                          OneTime        : YearAndBeyondRealRec
                       End;
ApproReportSubRec232 = Record
                          ElimMilPCS,
                          PerDiem,
                          POVMiles,
                          HHG,
                          Misc           : YearAndBeyondRealRec
                       End;
ApproReportSubRec132 = Record
                          HousAllow      : YearAndBeyondRealRec
                       End;
ApproReportSubRec23  = Record
                          MilMoving      : ApproReportSubRec232;
                          Other          : ApproReportSubRec132
                       End;
```

```
ApproReportSubRec224 = Record
                          Caretaker,
                          AdminPlan,
                          Shutdown,
                          Maintain,
                          NewHire,
                          OneTimeMove,
                          OtherIDF       : YearAndBeyondRealRec
                       End;
ApproReportSubRec223 = Record
                          Pack,
                          Freight,
                          Vehicles,
                          Driving,
                          LossRate       : YearAndBeyondRealRec
                       End;
ApproReportSubRec222 = Record
                          PerDiem,
                          POVMiles,
                          HomePurchSale,
                          HHG,
                          Misc,
                          HousHunt,
                          PPS,
                          RITA           : YearAndBeyondRealRec
                       End;
ApproReportSubRec221 = Record
                          CivRIF,
                          CivRetire      : YearAndBeyondRealRec
                       End;
ApproReportSubRec22  = Record
                          RPMA,
                          BOS,
                          OperatingIDF   : YearAndBeyondRealRec;
                          CivSalary      : ApproReportSubRec221;
                          CivMoving      : ApproReportSubRec222;
                          Freight        : ApproReportSubRec223;
                          Champus,
                          Unemployment   : YearAndBeyondRealRec;
                          Other          : ApproReportSubRec224
                       End;
ApproReportSubRec2   = Record
                          MilCon         : YearAndBeyondRealRec;
                          FamHous        : ApproReportSubRec11;
                          OM             : ApproReportSubRec22;
                          MilPers        : ApproReportSubRec23;
                          Procure,
                          HomeownAsst,
                          EnvirMitig,
                          InfoMgt        : YearAndBeyondRealRec;
                          Other          : ApproReportSubRec24;
                          LandPurch,
```

```
                                MiscRecur,
                                Total          : YearAndBeyondRealRec
                             End;
ApproReportSubRec131 = Record
                                Off,
                                Enl            : YearAndBeyondRealRec
                             End;
ApproReportSubRec13  = Record
                                MilSalary      : ApproReportSubRec131;
                                MilMoving      : YearAndBeyondRealRec;
                                Other          : ApproReportSubRec132
                             End;
ApproReportSubRec12  = Record
                                RPMA,
                                BOS,
                                OperatingIDF,
                                CivSalary,
                                CivMoving,
                                Freight,
                                Champus,
                                OtherIDF       : YearAndBeyondRealRec
                             End;
ApproReportSubRec1   = Record
                                MilCon         : YearAndBeyondRealRec;
                                FamHous        : ApproReportSubRec11;
                                OM             : ApproReportSubRec12;
                                MilPers        : ApproReportSubRec13;
                                Procure,
                                HomeownAsst,
                                EnvirMitig,
                                InfoMgt,
                                Other,
                                LandRevenue,
                                MiscRecur,
                                Total          : YearAndBeyondRealRec;
                             End;
ApproReportSubRec    = Record
                                Savings        : ApproReportSubRec1;
                                Costs          : ApproReportSubRec2;
                                Net            : ApproReportSubRec3;
                             End;
ApproReportRec       = ^ApproReportSubRec;

AutoRec = Record
             Execute,
             ExitCobra : Boolean;
             FileName,
             ReportDir : PathStr;
          End;
```

```
BudgetPropDataSubRec = Record
                          BaseOpnsNonPayBudget,
                          BaseOpnsPay,
                          CommunBudget,
                          FamHousBudget,
                          RPMANonPayBudget,
                          RPMAPayBudget      : Real;
                          MiscRecurCost      : Year1ToMaxRealRec;
                       End;
BudgetPropDataRec    = Record
                          Base               : Array [0..AbsMaxBases]
                                               Of BudgetPropDataSubRec
                       End;

CalcMovesSubRec2 = Record
                      Base : Array [0..AbsMaxBases] of Year0ToMaxIntegerRec
                   End;
CalcMovesSubRec1 = Record
                      Into,
                      From : CalcMovesSubRec2;
                   End;
CalcMovesRec     = Record
                      Civ,
                      Enl,
                      Off,
                      Stu  : CalcMovesSubRec1;
                   End;

CalcPersActSubRec = Record
                       CivsQuitting,
                       NumAffected,
                       NumAttrited,
                       NumMoving,
                       NumNeeded,
                       NumRetiring,
                       NumRemain,
                       NumRIFTed,
                       PersInvolRIFs,
                       PersNewHires,
                       PersVolRIFs,
                       PriPlacement,
                       ShortOverage,
                       TotalPPS,
                       Total        : Integer;
                       EarlyRetire,
                       PercOfNeed,
                       RIFPay       : Real
                    End;
CalcPersActRec    = Record
                       Year         : Array [1..AbsMaxYears]
                                      of CalcPersActSubRec;
                    End;
```

```
CalcPersPosRec = Record
                    CivPosReduc,
                    EnlPosReduc,
                    OffPosReduc,
                    CivPosElim,
                    EnlPosElim,
                    OffPosElim : Year0ToMaxRealRec
                 End;

ChampusSubRec = Record
                   OnBaseInPatVisits,
                   OnBaseOutPatVisits   : Year1ToMaxRealRec;
                   ChampInPatCostVisit,
                   ChampOutPatCostVisit : Real
                End;
ChampusRec    = Record
                   Base : Array [0..AbsMaxBases] of ChampusSubRec
                End;

ConSubRec   = Record
                AreaCostFactor : Real;
                ProjectName    : Array [1..16] of ^String20;
                Category       : Array [1..16] of Char;
                New,
                Rehab,
                Cost           : Array [1..16] of Real;
                Comment        : Array [1..16] of ^String60;
             End;
ConRec      = Record
                Base           : Array [0..AbsMaxBases] of ConSubRec;
             End;

CostAvoidSubRec = Record
                     Constr,
                     FamQtr,
                     Procure : Year1ToMaxRealRec;
                  End;
CostAvoidRec    = Record
                     Base    : Array [0..AbsMaxBases] Of CostAvoidSubRec;
                  End;

CostElimMilRec = Record
                    Enl,
                    Off,
                    Tot  : Year1ToMaxRealRec
                 End;

PDBaseRec = ^DBaseRec;
DBaseRec  = Record
               BaseName : String20;
               Next     : PDBaseRec;
            End;
```

```
DBaseSearchRec = Record
                    Selected   : Word;
                    SearchName : String20;
                 End;

DeltaChartSubRec2 = Record
                       Delta,
                       PercChange : Real
                    End;
DeltaChartSubRec1 = Record
                       Base       : Array [0..AbsMaxBases]
                                    Of DeltaChartSubRec2;
                    End;
DeltaChartSubRec  = Record
                       BOS,
                       Land,
                       Pers,
                       RPMA,
                       RPMABOS,
                       SF         : DeltaChartSubRec1
                    End;
DeltaChartRec     = ^DeltaChartSubRec;

DeviceRec  = Record
               DeviceName,
               ReportDir  : PathStr;
               HighASCII  : Word;
             End;

DistanceRec    = Record
                    Base : Array [0..AbsMaxBases] Of Base0ToMaxRealRec;
                 End;

HousAllowRec    = Record
                     Enl,
                     Off  : Base0Year1ToMaxRealRec
                  End;

IndustFundSubRec1 = Record
                       AdminPlan,
                       Operating,
                       Mothball,
                       Caretaker,
                       MaintSpace,
                       Other      : Year1ToMaxRealRec
                    End;
IndustFundSubRec2 = Record
                       Operating,
                       Other      : Year1ToMaxRealRec
                    End;
```

```
IndustFundSubRec3 = Record
                       PackUnpack,
                       Freight,
                       Vehic,
                       Driv,
                       LossRate    : Year1ToMaxRealRec
                    End;
IndustFundSub     = Record
                       OvhdCost    : IndustFundSubRec1;
                       OvhdSave    : IndustFundSubRec2;
                       MoveCost    : IndustFundSubRec3
                    End;
IndustFundRec     = Record
                       Base        : Array [0..AbsMaxBases] of ^IndustFundSub;
                    End;

IndustFundSwitchRec = Record
                         Base : Array[0..AbsMaxBases] Of Char;
                      End;

IOBuffer = Array [0..2047] of Byte;

MilConFinalRec = Record
                    Constr,
                    Result,
                    CostAvoidConstr,
                    Net                : Year1ToMaxRealRec
                 End;

MilConSubRec  = Record
                   NewCost,
                   RehabCost,
                   TotalCost       : Array [1..18] Of Real;
                End;
MilConRec     = Record
                   Base            : Array [0..AbsMaxBases] of MilConSubRec;
                End;

NPVTableRec = Record
                 Cost,
                 Inflated,
                 NPV       : Year1ToNumRealRec
              End;

PersMovesSubRec = Record
                     Perc,
                     ShutDn : Year1ToMaxRealRec
                  End;
PersMovesSumRec = Record
                     Base    : Array [0..AbsMaxBases] Of PersMovesSubRec;
                  End;
```

```
PersCostSubRec = Record
                    EnlVHA,
                    OffVHA,
                    PerDiemRate : Real
                 End;
PersCostRec    = Record
                    Base        : Array [0..AbsMaxBases] of PersCostSubRec;
                 End;

PosElimOrReducSubRec = Record
                         Civ,
                         Enl,
                         Off  : Year0ToMaxIntegerRec;
                       End;
PosElimOrReducRec =    Record
                         Base : Array [0..AbsMaxBases]
                                Of PosElimOrReducSubRec;
                       End;

PosTransSubRec2 = Record
                     Civ,
                     Enl,
                     Off,
                     Stu          : Year0ToMaxIntegerRec;
                  End;
PosTransSubRec  = Record
                     CaretakeCiv,
                     CaretakeMil  : Year0ToMaxIntegerRec;
                     Base         : Array [0..AbsMaxBases]
                                    Of ^PosTransSubRec2;
                  End;
PosTransRec      = Record
                     Base         : Array [0..AbsMaxBases]
                                    Of PosTransSubRec
                  End;

RealPropPurSubRec     = Record
                          Acre,
                          Cost                : Year1ToMaxRealRec;
                        End;
RealPropPurRec        = Record
                          Base                : Array [0..AbsMaxBases]
                                                of RealPropPurSubRec;
                        End;
RealPropProcSchedRec = Record
                          FacilSqFtShutDown : Real;
                        End;
RealPropProcRec       = Record
                          Base                : Array [0..AbsMaxBases]
                                                Of RealPropProcSchedRec;
                          RealPropPur         : RealPropPurRec
                        End;
```

```
SalSavRec = Record
               Civ,
               Enl,
               Off,
               Tot : YearAndBeyondRealRec
            End;

Screen1DataSubRec = Record
                       Name           : String20;
                       State          : String02;
                       CloseYear      : Integer;
                       Deactivate     : Char;
                    End;
Screen1DataRec    = Record
                       OptionPkgName,
                       Service,
                       Group          : String20;
                       SFFile         : PathStr;
                       FY1            : Integer;
                       ModelDoShutDn  : Char;
                       Base           : Array[0..AbsMaxBases]
                                        Of Screen1DataSubRec;
                       Summary        : Array[1..6] Of String60;
                       DateLine       : String20;
                       SetDateFlag    : Word;
                    End;

Screen9DataRec    = Record
                       Line  : Array [1..15] of String60;
                       Pages : Word;
                    End;

StdFctrsConRec    = Record
                       ConCost : Array [1..35] of Real;
                       Name    : Array [1..16] of String20;
                       Units   : Array [-2..16] of String02;
                    End;

StdFctrsFacRec = Record
                    RPMACostBldgPower,
                    BOSIndex,
                    SuppToMoveFctr,
                    CaretakerAdminSpace,
                    CaretakerPercOrigRPMACost,
                    MothballCostPerSqrFt,
                    DiscountNPV,
                    InflationNPV                : Real;
                    InflationRate               : Year1ToMaxRealRec;
                    AvgBachQtrs,
                    AvgFamQtrs,
                    RehabCostVsNewConstr,
                    IMAPerc,
```

```
                    Design,
                    SIOH,
                    Contingency,
                    SitePrep                    : Real;
                End;

StdFctrsPersRec     = Record
                        PercOffMarried,
                        PercEnlMarried,
                        PercEnlHous,
                        OffSal,
                        OffBAQDepEnds,
                        EnlSal,
                        EnlBAQDepEnds,
                        AvgUnemployCostPerWeek,
                        UnemployEligible,
                        CivSal,
                        CivTurnover,
                        CivEarlyRetire,
                        PercCivsQuitting,
                        CivRifPayFctr,
                        CivRetirePayFctr,
                        PriPlacementPerc,
                        PPSInvolvePCS,
                        CivPCSCosts,
                        NewHireCost,
                        NatlAvgHomePrice,
                        HomeSaleReimburseRate,
                        HomeSaleReimburseMax,
                        HomePurchReimburseRate,
                        HomePurchReimburseMax,
                        CivHomeOwnRate,
                        HAPFactor,
                        HAPReceive,
                        DARSEFactor,
                        DARSEReceive            : Real;
                        FileDescription         : String20;
                    End;

StdFctrsTransRec    = Record
                        MaterialPerPerson,
                        HHGWtPerOffFam,
                        HHGWtPerEnlFam,
                        HHGWtPerMilSing,
                        HHGWtPerCiv,
                        HHGCost,
                        ShippingLossRate,
                        PackNCrate,
                        MilLightVehicCostPerMile,
                        MilVehicCostPerMile,
                        POVReimbursePerMile,
                        AirTransPerPassMile,
```

```
                                MiscExpPerDirectEmp,
                                AverageTourLength,
                                RoutinePCSCostsPerPerson,
                                OneTimeOffPCSCosts,
                                OneTimeEnlPCSCosts        : Real;
                             End;

  TimePhasShutDnSubRec2 = Record
                                Year     : Year1ToMaxReal
                              End;
  TimePhasShutDnSubRec  = Record
                                Constr,
                                Shutdn   : TimePhasShutDnSubRec2
                              End;
  TimePhasShutDnRec     = Record
                                Base     : Array [0..AbsMaxBases]
                                           Of TimePhasShutDnSubRec
                              End;

  TwoWords = Record
               First, Second : Word
             End;

VAR
  ActionEnvir              : ^ActionEnvirRec;
  ActivMisCost             : ^Base0Year1ToMaxRealRec;
  AutoStart                : AutoRec;
  BackGroundChar           : Char;
  BudgetPropData           : ^BudgetPropDataRec;
  Champus                  : ChampusRec;
  ChangeFlag               : Boolean;
  Con                      : ConRec;
  CostAvoid                : CostAvoidRec;
  DataFileUsed             : PathStr;
  Distance                 : ^DistanceRec;
  FreightCost              : Base0ToMaxRealRec;
  HAPSwitch                : IndustFundSwitchRec;
  IndustFund               : IndustFundRec;
  IndustFundSwitch         : IndustFundSwitchRec;
  Info                     : DeviceRec;
  LastYear,
  MaxBases,
  MaxYears                 : Integer;
  PersCost                 : PersCostRec;
  PosElim,
  PosReduc                 : PosElimOrReducRec;
  PosTrans                 : PosTransRec;
  RealPropProc             : RealPropProcRec;
  SavedExitProc            : Pointer;
  SaverCounter             : LongInt;
  Screen1Data              : ^Screen1DataRec;
  Screen9Data              : Screen9DataRec;
```

```
  SFFileUsed              : PathStr;
  StdFctrsCon             : StdFctrsConRec;
  StdFctrsFac             : StdFctrsFacRec;
  StdFctrsPers            : StdFctrsPersRec;
  StdFctrsTrans           : StdFctrsTransRec;
  TimePhasShutDn          : TimePhasShutDnRec;

IMPLEMENTATION

End.
```

```
UNIT CFAC1;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT;

TYPE
  PFact1Dlg = ^TFact1Dlg;
  TFact1Dlg = OBJECT (TDialog)
                CONSTRUCTOR Init;
              END; {Object}

IMPLEMENTATION

{*************************************************************************
 *  Name          : FactOne
 *
 *  Purpose       : Handle the first Standard Factor entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : FactOne(info)
 *
 *  Return value  : status of exit (a cmXXXX value)
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     initial/entered values for the screen
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  23Jan91   ASB   Creation
 *************************************************************************}

CONSTRUCTOR TFact1Dlg.Init;

  VAR
    P    : PView;
    rect : TRect;

  Begin
    {*********************************************************************
     * Create a dialog box to display data choices in
     *********************************************************************}
```

```
rect.Assign(0,0,78,22);
TDialog.Init (rect, 'Table One - Standard Personnel Factors');
Options := Options OR ofCentered;

{*****************************************************************
 * Data fields
 *****************************************************************}

rect.Assign (38,2,39,3);
Insert(New(PStaticText,Init(rect,'%')));
rect.Assign (28,2,38,3);
P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0));
Insert (P);
rect.Assign (10,2,28,3);
Insert(New(PLabel,Init(rect,'~O~fficers Married:',P)));

rect.Assign (11,3,39,4);
Insert(New(PStaticText,Init(rect,'Enlisted Married:          %')));
rect.Assign (28,3,38,4);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(4,4,39,5);
Insert(New(PStaticText,Init(rect,'Enlisted Housing MilCon:          %')));
rect.Assign(28,4,38,5);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(4,6,28,7);
Insert(New(PStaticText,Init(rect,'Officer Salary ($/Year):')));
rect.Assign(28,6,38,7);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 99999.99)));

rect.Assign(3,7,28,8);
Insert(New(PStaticText,Init(rect,'Officer BAQ w/Depdts ($):')));
rect.Assign(28,7,38,8);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 20000.0)));

rect.Assign(3,8,28,9);
Insert(New(PStaticText,Init(rect,'Enlisted Salary ($/Year):')));
rect.Assign(28,8,38,9);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 99999.99)));

rect.Assign(2,9,28,10);
Insert(New(PStaticText,Init(rect,'Enlisted BAQ w/Depdts ($):')));
rect.Assign(28,9,38,10);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 20000.0)));

rect.Assign(2,10,28,11);
Insert(New(PStaticText,Init(rect,'Avg Unemploy Cost($/Week):')));
rect.Assign(28,10,38,11);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 2000.0)));
```

```
rect.Assign(2,11,28,12);
Insert(New(PStaticText,Init(rect,'Unemploy Eligible (Weeks):')));
rect.Assign(28,11,38,12);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0, 0.0, 52.0)));

rect.Assign(3,12,28,13);
Insert(New(PStaticText,Init(rect,'Civilian Salary ($/Year):')));
rect.Assign(28,12,38,13);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 99999.99)));

rect.Assign(10,14,39,15);
Insert(New(PStaticText,Init(rect,'Civilian Turnover:          %')));
rect.Assign(28,14,38,15);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(2,15,39,16);
Insert(New(PStaticText,Init(rect,'Civilian Early Retirement:          %')));
rect.Assign(28,15,38,16);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(9,16,39,17);
Insert(New(PStaticText,Init(rect,'Civilians Quitting:          %')));
rect.Assign(28,16,38,17);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(4,17,39,18);
Insert(New(PStaticText,Init(rect,'Civilian RIF Pay Factor:          %')));
rect.Assign(28,17,38,18);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(76,2,77,3);
Insert(New(PStaticText,Init(rect,'%')));
rect.Assign(66,2,76,3);
P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0));
Insert (P);
rect.Assign(42,2,66,3);
Insert(New(PLabel,Init(rect,'~C~iv Retired Pay Factor:',P)));

rect.Assign(47,3,77,4);
Insert(New(PStaticText,Init(rect,'Priority Placement:          %')));
rect.Assign(66,3,76,4);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(45,4,77,5);
Insert(New(PStaticText,Init(rect,'PPS Place Involv PCS:          %')));
rect.assign(66,4,76,5);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0)));

rect.Assign(49,5,66,6);
Insert(New(PStaticText,Init(rect,'Civ PCS Cost ($):')));
rect.Assign(66,5,76,6);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 99999.99)));
```

```
rect.Assign(48,6,66,7);
Insert(New(PStaticText,Init(rect,'New Hire Cost ($):')));
rect.Assign(66,6,76,7);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 10000.0)));

rect.Assign(40,8,66,9);
Insert(New(PStaticText,Init(rect,'Nat Median Home Price($K):')));
rect.Assign(66,8,76,9);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 2500.0)));

rect.Assign(41,9,77,10);
Insert(New(PStaticText,Init(rect,'Home Sale Reimburse Rate:          %')));
rect.Assign(66,9,76,10);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(40,10,66,11);
Insert(New(PStaticText,Init(rect,'Max Home Sale Reimburs($):')));
rect.Assign(66,10,76,11);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 25000.0)));

rect.Assign(40,11,77,12);
Insert(New(PStaticText,Init(rect,'Home Purch Reimburse Rate:          %')));
rect.Assign(66,11,76,12);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(40,12,66,13);
Insert(New(PStaticText,Init(rect,'Max Home Purch Reimbur($):')));
rect.Assign(66,12,76,13);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 25000.0)));

rect.Assign(46,13,77,14);
Insert(New(PStaticText,Init(rect, 'Home Ownership Rate:          %')));
rect.Assign(66,13,76,14);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2, 0.0, 100.0)));

rect.Assign (46,14,77,15);
Insert(New(PStaticText,Init(rect, 'HAP Home Value Rate:          %')));
rect.Assign (66,14,76,15);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0)));

rect.Assign (47,15,77,16);
Insert(New(PStaticText,Init(rect, 'HAP Receiving Rate:          %')));
rect.Assign (66,15,76,16);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0)));

rect.Assign (46,16,77,17);
Insert(New(PStaticText,Init(rect, 'RSE Home Value Rate:          %')));
rect.Assign (66,16,76,17);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0)));

rect.Assign (47,17,77,18);
Insert(New(PStaticText,Init(rect, 'RSE Receiving Rate:          %')));
```

```
    rect.Assign (66,17,76,18);
    Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0)));

    rect.Assign (6,19,27,20);
    Insert(New(PStaticText,Init(rect, 'Standard Factors File')));
    rect.Assign (4,20,16,21);
    Insert(New(PStaticText,Init(rect, 'Description:')));
    rect.Assign (16,20,38,21);
    Insert(New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0)));

    {*****************************************************************
     * "OK" and "Cancel" buttons
     *****************************************************************}

    rect.Assign (48,19,60,21);
    Insert(New(PButton,Init(rect,'~N~ext',cmOK,bfNormal)));

    rect.Assign (63,19,75,21);
    Insert(New(PButton,Init(rect,'~D~one',cmNo,bfNormal)));
  END;  { TFact1Dlg.Init }

END.  { UNIT }
```

```
UNIT CFAC2;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT,
  CDEC,
  CSUP;

TYPE
  PFact2Dlg = ^TFact2Dlg;
  TFact2Dlg = OBJECT (TDialog)
                CONSTRUCTOR Init;
              END; {Object}

IMPLEMENTATION

{*************************************************************************
 *  Name          : FactTwo
 *
 *  Purpose       : Handle the first Standard Factor entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : FactTwo(info)
 *
 *  Return value : status of exit (a cmXXXX value)
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     initial/entered values for the screen
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  23Jan91   ASB   Creation
 *************************************************************************}

CONSTRUCTOR TFact2Dlg.Init;

  VAR
    P    : PView;
    rect : TRect;
    X, Y : Integer;
```

```
Begin
  {*********************************************************************
   * Create a dialog box to display data choices in
   *********************************************************************}

  rect.Assign(0,0,78,22);
  TDialog.Init (rect, 'Table Two - Standard Facilities Factors');
  Options := Options OR ofCentered;

  {*****************************************************************
   * Data fields
   *****************************************************************}

  rect.Assign(26,2,38,3);
  P := New(PFInputLine,Init(rect,10,DRealSet,DReal,2, 0.0, 5.0));
  Insert(P);
  rect.Assign(1,2,26,3);
  Insert(New(PLabel,Init(rect,'RPMA ~B~uilding SF Index*:',P)));
  rect.Assign(4,3,26,4);
  Insert(New(PStaticText,Init(rect,'BOS Population Index*:')));
  rect.Assign(26,3,38,4);
  Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,2, 0.0, 10.0)));
  rect.Assign(2,20,32,21);
  Insert(New(PStaticText,Init(rect,'*Indices are used as exponents')));
  rect.Assign(2,4,39,5);
  Insert(New(PStaticText,Init(rect,'Support for Move Factor:           %')));
  rect.Assign(26,4,38,5);
  Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,1, 0.0, 100.0)));

  rect.Assign(2,6,18,7);
  Insert(New(PStaticText,Init(rect,'CARETAKER COSTS')));
  rect.Assign(4,7,26,8);
  Insert(New(PStaticText,Init(rect,'Admin Space (SF/Pers):')));
  rect.Assign(26,7,38,8);
  Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,1, 0.0, 1.0E6)));
  rect.Assign(4,8,39,9);
  Insert(New(PStaticText,Init(rect,'Percent Original RPMA:           %')));
  rect.Assign(26,8,38,9);
  Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,1, 0.0, 100.0)));
  rect.Assign(5,9,26,10);
  Insert(New(PStaticText,Init(rect,'Mothball Cost ($/SF):')));
  rect.Assign(26,9,38,10);
  Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,2, 0.0, 100.0)));

  rect.Assign(4,11,26,12);
  Insert(New(PStaticText,Init(rect,'NPV/ROI Discount Rate:')));
  rect.Assign(38,11,39,12);
  Insert(New(PStaticText,Init(rect,'%')));
  rect.Assign(26,11,38,12);
  Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,1, 0.0, 100.0)));
  rect.Assign(3,12,26,13);
  Insert(New(PStaticText,Init(rect,'NPV/ROI Inflation Rate:')));
```

```
rect.Assign(38,12,39,13);
Insert(New(PStaticText,Init(rect,'%')));
rect.Assign(26,12,38,13);
Insert(New(PFInputLine,Init(rect,10,DRealSet,DReal,1, 0.0, 100.0)));

rect.Assign(2,14,34,15);
Insert(New(PStaticText,Init(rect, 'FINANCE.RPT INFLATION RATES')));
For X := 0 To 2 Do
  For Y := 0 To 1 Do
    Begin
      rect.Assign(4 + Y * 19, 15 + X, 20 + Y * 19, 16 + X);
      Insert(New(PStaticText,Init
        (rect, FiscalYear(X * 2 + Y + 1) + ':          %')));
      rect.Assign(9 + Y * 19, 15 + X, 19 + Y * 19, 16 + X);
      Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));
    End;

rect.Assign(66,2,76,3);
P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,500.0));
Insert (P);
rect.Assign(41,2,66,3);
Insert(New(PLabel,Init(rect,'~A~vg Bach Qtrs Size (SF):', P)));
rect.Assign(43,3,66,4);
Insert(New(PStaticText,Init(rect,'Avg Fam Qtrs Size (SF):')));
rect.Assign(66,3,76,4);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,2000.0)));

rect.Assign(40,5,76,6);
Insert(New(PStaticText,Init(rect, 'MILCON PERCENTAGES')));
rect.Assign(42,6,77,7);
Insert(New(PStaticText,Init(rect,'Rehab vs. New Construct:          %')));
rect.Assign(66,6,76,7);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));
rect.Assign(42,7,77,8);
Insert(New(PStaticText,Init(rect,'Info Management Account:          %')));
rect.Assign(66,7,76,8);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));

rect.Assign(48,8,77,9);
Insert(New(PStaticText,Init(rect,'Design Percentage:          %')));
rect.Assign(50,9,77,10);
Insert(New(PStaticText,Init(rect,'SIOH Percentage:          %')));
rect.Assign(43,10,77,11);
Insert(New(PStaticText,Init(rect,'Contingency Percentage:          %')));
rect.Assign(45,11,77,12);
Insert(New(PStaticText,Init(rect,'Site Prep Percentage:          %')));
For X := 0 To 3 Do
  Begin
    rect.Assign(66,8 + X,76,9 + X);
    Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));
  End;
```

```
    {*****************************************************************
     * "OK" and "Cancel" buttons
     *****************************************************************}

    rect.Assign(33,19,45,21);
    Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

    rect.Assign(48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign(63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; { TFact2Dlg.Init }

END.  { UNIT }
```

```
UNIT CFAC3;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT;

TYPE
  PFact3Dlg = ^TFact3Dlg;
  TFact3Dlg = OBJECT (TDialog)
                CONSTRUCTOR Init;
              END; { Object }

IMPLEMENTATION

{*************************************************************************
 *  Name          : FactThree
 *
 *  Purpose       : Handle the first Standard Factor entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : FactThree(info)
 *
 *  Return value : status of exit (a cmXXXX value)
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     initial/entered values for the screen
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  23Jan91   ASB   Creation
 *************************************************************************}

CONSTRUCTOR TFact3Dlg.Init;

  VAR
    P    : PView;
    rect : TRect;

  Begin
    {*********************************************************************
     * Create a dialog box to display data choices in
     *********************************************************************}
```

```
rect.Assign(0,0,78,22);
TDialog.Init (rect, 'Table Three - Standard Transportation Factors');
Options := Options OR ofCentered;

{*****************************************************************
 * Data fields
 *****************************************************************}

rect.Assign(28,2,38,3);
P := New(PFInputLine,Init(rect,8,DRealSet,DReal,0, 0.0, 10000.0));
Insert(P);
rect.Assign(1,2,28,3);
Insert(New(PLabel,Init(rect,'~M~aterial/Assignd Pers(Lb):',P)));

rect.Assign(5,4,28,5);
Insert(New(PStaticText,Init(rect,  'Officer HHG(Lb/Family):')));
rect.Assign(28,4,38,5);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,0, 0.0, 100000.0)));
rect.Assign(4,5,28,6);
Insert(New(PStaticText,Init(rect, 'Enlisted HHG(Lb/Family):')));
rect.Assign(28,5,38,6);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,0, 0.0, 100000.0)));
rect.Assign(4,6,28,7);
Insert(New(PStaticText,Init(rect, 'Military HHG(Lb/Single):')));
rect.Assign(28,6,38,7);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,0, 0.0, 100000.0)));
rect.Assign(4,7,28,8);
Insert(New(PStaticText,Init(rect, 'Civilian HHG(Lb/Person):')));
rect.Assign(28,7,38,8);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,0, 0.0, 100000.0)));

rect.Assign(3,9,28,10);
Insert(New(PStaticText,Init(rect,'Total HHG Costs($/100Lb):')));
rect.Assign(28,9,38,10);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100.0)));

rect.Assign(9,11,28,12);
Insert(New(PStaticText,Init(rect,       'Shipping Loss Rate:')));
rect.Assign(28,11,38,12);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,1, 0.0, 100.0)));
rect.Assign(38,11,39,12);
Insert(New(PStaticText,Init(rect,'%')));

rect.Assign(66,2,76,3);
P := New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0));
Insert (P);
rect.Assign(39,2,66,3);
Insert(New(PLabel,Init(rect,'~E~quip Pack & Crate($/Ton):',P)));
```

```
rect.Assign(42,3,66,4);
Insert(New(PStaticText,Init(rect, 'Milit Light Vehic($/Mi):')));
rect.Assign(66,3,76,4);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 1000.0)));

rect.Assign(43,4,66,5);
Insert(New(PStaticText,Init(rect, 'Heavy/Spec Vehic($/Mi):')));
rect.Assign(66,4,76,5);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 1000.0)));

rect.Assign(66,5,76,6);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0)));
rect.Assign(42,5,66,6);
Insert(New(PStaticText,Init(rect,     'POV Reimbursement($/Mi):')));

rect.Assign(41,6,66,7);
Insert(New(PStaticText,Init(rect,'Air Transport($/Pass Mi):')));
rect.Assign(66,6,76,7);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0)));

rect.Assign(42,7,66,8);
Insert(New(PStaticText,Init(rect, 'Misc($/Direct Employee):')));
rect.Assign(66,7,76,8);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0)));

rect.Assign(41,9,66,10);
Insert(New(PStaticText,Init(rect,'Avg Mil Tour Length(Yrs):')));
rect.Assign(66,9,76,10);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 1.0, 20.0)));

rect.Assign(41,10,66,11);
Insert(New(PStaticText,Init(rect,'Routine PCS($/Pers/Tour):')));
rect.Assign(66,10,76,11);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0)));

rect.Assign(41,11,66,12);
Insert(New(PStaticText,Init(rect,'One-Time Off PCS Cost($):')));
rect.Assign(66,11,76,12);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0)));

rect.Assign(41,12,66,13);
Insert(New(PStaticText,Init(rect,'One-Time Enl PCS Cost($):')));
rect.Assign(66,12,76,13);
Insert(New(PFInputLine,Init(rect,8,DRealSet,DReal,2, 0.0, 100000.0)));

{*****************************************************************
 * "OK" and "Cancel" buttons
 *****************************************************************}

rect.Assign(33,19,45,21);
Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));
```

```
    rect.Assign(48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign(63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; { TFact3Dlg.Init }

END.  { UNIT }
```

```
UNIT CFAC4;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT;

TYPE
  PFact4Dlg = ^TFact4Dlg;
  TFact4Dlg = OBJECT (TDialog)
                CONSTRUCTOR Init;
              END;  { Object }

IMPLEMENTATION

{*********************************************************************
 *  Name          : FactFour
 *
 *  Purpose       : Handle the first Standard Factor entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : FactFour(info)
 *
 *  Return value  : status of exit (a cmXXXX value)
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ------------------------------------------
 *    info              VAR     initial/entered values for the screen
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  23Jan91   ASB   Creation
 *********************************************************************}

CONSTRUCTOR TFact4Dlg.Init;

  VAR
    C, L : Integer;
    P    : PView;
    rect : TRect;

  Begin
    {*******************************************************************
     * Create a dialog box to display data choices in
     *******************************************************************}
```

```
rect.Assign(0,0,78,22);
TDialog.Init (rect, 'Table Four - Standard Construction Factors');
Options := Options OR ofCentered;

{*****************************************************************
 * Data fields
 *****************************************************************}

rect.Assign(2, 2,28,3);
Insert(New(PStaticText,Init(rect,'Horizontal           (SY)')));
rect.Assign(2, 3,28,4);
Insert(New(PStaticText,Init(rect,'Waterfront           (LF)')));
rect.Assign(2, 4,28,5);
Insert(New(PStaticText,Init(rect,'Air Operations       (SF)')));
rect.Assign(2, 5,28,6);
Insert(New(PStaticText,Init(rect,'Operational          (SF)')));
rect.Assign(2, 6,28,7);
Insert(New(PStaticText,Init(rect,'Administrative       (SF)')));
rect.Assign(2, 7,28,8);
Insert(New(PStaticText,Init(rect,'School Buildings     (SF)')));
rect.Assign(2, 8,28,9);
Insert(New(PStaticText,Init(rect,'Maintenance Shops    (SF)')));
rect.Assign(2, 9,21,10);
Insert(New(PStaticText,Init(rect,'Bachelor Quarters*')));
rect.Assign(2,10,19,11);
Insert(New(PStaticText,Init(rect,'Family Quarters*')));
rect.Assign(2,11,28,12);
Insert(New(PStaticText,Init(rect,'Storage Facilities   (SF)')));
rect.Assign(2,12,28,13);
Insert(New(PStaticText,Init(rect,'Dining Facilities    (SF)')));
rect.Assign(2,13,28,14);
Insert(New(PStaticText,Init(rect,'Recreation Facil     (SF)')));
rect.Assign(2,14,28,15);
Insert(New(PStaticText,Init(rect,'Communications Facil (SF)')));
rect.Assign(2,15,28,16);
Insert(New(PStaticText,Init(rect,'Shipyard Maintenance (SF)')));
rect.Assign(2,16,28,17);
Insert(New(PStaticText,Init(rect,'RDT&E Facilities     (SF)')));
rect.Assign(2,17,28,18);
Insert(New(PStaticText,Init(rect,'POL Storage          (BL)')));
rect.Assign(2,18,28,19);
Insert(New(PStaticText,Init(rect,'Ammunition Storage   (SF)')));
rect.Assign(2,19,28,20);
Insert(New(PStaticText,Init(rect,'Medical Facilities   (SF)')));
rect.Assign(2,20,28,21);
Insert(New(PStaticText,Init(rect,'Environmental')));

For C := 0 To 1 Do
  For L := 0 To 18 Do
    If (L < 16) Or (C = 0) Then
```

```
        Begin
          rect.Assign(28 + C * 38, 2 + L, 38 + C * 38, 3 + L);
          P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,999999));
          Insert(P);
          If L + C = 0 Then
            Begin
              rect.Assign(1,1,40,2);
              Insert(New(PLabel,Init(rect,
                '~M~ilCon Categories   Units   Cost/UM  ', P)));
            End
        End;
    For L := 0 To 15 Do
      Begin
        rect.Assign(39, 2 + L, 60, 3 + L);
        P := New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0));
        Insert (P);
        If L = 0 Then
          Begin
            rect.Assign(38,1,76,2);
            Insert(New(PLabel,Init(rect,
              '~O~ptional Categories  Units   Cost/UM', P)));
          End
      End;
    rect.Assign(23,  9, 27, 10);
    Insert(New(PFInputLine,Init(rect,2,DCharSet,DString,0,0.0,0.0)));
    rect.Assign(23, 10, 27, 11);
    Insert(New(PFInputLine,Init(rect,2,DCharSet,DString,0,0.0,0.0)));
    rect.Assign(23, 20, 27, 21);
    Insert(New(PFInputLine,Init(rect,2,DCharSet,DString,0,0.0,0.0)));
    For L := 0 To 15 Do
      Begin
        rect.Assign(61, 2 + L, 65, 3 + L);
        Insert(New(PFInputLine,Init(rect,2,DCharSet,DString,0,0.0,0.0)));
      End;

    rect.Assign(39,19,46,20);
    Insert(New(PStaticText,Init(rect, '*Units:')));
    rect.Assign(40,20,48,21);
    Insert(New(PStaticText,Init(rect, 'SF or EA')));

    {*****************************************************************
     * "OK" and "Cancel" buttons
     *****************************************************************}

    rect.Assign(48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign(63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; { TFact4Dlg.Init }

END.  { UNIT }
```

```
UNIT CINIT;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP,
  DOS;

{ InitGlobalVariables split into two procedures (IGV & ISF) by AML 3/26/92 }

{ InitPointers created by AML 5/12/92 ... TurboPascal allows only 64K for
  Global Variables; with 15 bases the 64K limit was breached, so some data
  had to be handled dynamically.  InitPointers is only called once.        }

PROCEDURE InitGlobalVariables;

PROCEDURE InitPointers;

PROCEDURE InitStandardFactors;

Procedure CapCheck       (Var BaseName   : String20);

Function  HardCopy           (FileName   : PathStr) : Boolean;

PROCEDURE KillBase           (WhichBase  : Integer);

Function  MakeReport         (WhichBase  : Integer;
                              FileName   : PathStr) : Boolean;

Procedure OptionSearch;

Procedure UpdateDataFile (Var FileName   : String);

IMPLEMENTATION

PROCEDURE InitPointers;

  Var
    B, B2, P : Integer;

  Begin
    New (ActionEnvir);
    New (ActivMisCost);
    New (BudgetPropData);
    New (Distance);
    New (Screen1Data);
    For B := 0 To AbsMaxBases Do
      Begin
        For B2 := 0 To AbsMaxBases Do
```

```
          Begin
            PosTrans.Base[B].Base[B2]      := Nil;
            ActionEnvir^.Base[B].Base[B2] := Nil;
          End;
        For P := 1 To 16 Do
          Begin
            Con.Base[B].ProjectName[P]     := Nil;
            Con.Base[B].Comment[P]         := Nil;
          End;
        Industfund.Base[B]                 := Nil;
      End;
  End;

PROCEDURE InitGlobalVariables;

  VAR
    B, B2, P, Y : Integer;

  Begin
    ChangeFlag := True;
    DataFileUsed := '(none)';

    FillChar (ActionEnvir^, SizeOf (ActionEnvir^), 0);
    FillChar (ActivMisCost^, SizeOf (ActivMisCost^), 0);
    FillChar (BudgetPropData^, SizeOf (BudgetPropData^), 0);
    FillChar (Champus, SizeOf (Champus), 0);
    FillChar (CostAvoid, SizeOf (CostAvoid), 0);
    FillChar (Distance^, SizeOf (Distance^), 0);
    FillChar (FreightCost, SizeOf (FreightCost), 0);
    FillChar (PersCost, SizeOf (PersCost), 0);
    FillChar (PosElim, SizeOf (PosElim), 0);
    FillChar (PosReduc, SizeOf (PosReduc), 0);
    FillChar (PosTrans, SizeOf (PosTrans), 0);
    FillChar (RealPropProc, SizeOf (RealPropProc), 0);
    FillChar (Screen9Data, SizeOf (Screen9Data), 0);
    FillChar (TimePhasShutDn, SizeOf (TimePhasShutDn), 0);

    LastYear := 0;
    MaxBases := 0;
    MaxYears := 6;

    Screen1Data^.FY1                          := 1994;
    Screen1Data^.ModelDoShutDn                := 'Y';
    Screen1Data^.OptionPkgName                := '';
    Screen1Data^.Service                      := '';
    Screen1Data^.Group                        := '';
    For B := 0 To AbsMaxBases Do
      Begin
        Screen1Data^.Base[B].Name             := '';
        Screen1Data^.Base[B].State            := '';
        Screen1Data^.Base[B].CloseYear        := 0;
        Screen1Data^.Base[B].Deactivate       := 'N';
```

```
    End;
  For P := 1 To 6 Do
    Screen1Data^.Summary[P]                          := '';
  Screen1Data^.DateLine                              := AsOfDateTime;

  For B := 0 To AbsMaxBases Do
    For B2 := 0 To AbsMaxBases Do
      Begin
        If PosTrans.Base[B].Base[B2] <> Nil Then
          Begin
            Dispose (PosTrans.Base[B].Base[B2]);
            PosTrans.Base[B].Base[B2] := Nil;
          End;
        If ActionEnvir^.Base[B].Base[B2] <> Nil Then
          Begin
            Dispose (ActionEnvir^.Base[B].Base[B2]);
            ActionEnvir^.Base[B].Base[B2] := Nil;
          End;
      End;

  For B := 0 To AbsMaxBases Do
    Begin
      ActionEnvir^.Base[B].ChampusToMedicare := 0.209;
      HAPSwitch.Base[B]                      := 'N';
      IndustFundSwitch.Base[B]               := 'N';
      Con.Base[B].AreaCostFactor             := 1.0;
      For B2 := 0 To AbsMaxBases Do
        PosTrans.Base[B].Base[B2]            := Nil;
      For P := 1 To 16 Do
        Begin
          If Con.Base[B].ProjectName[P] <> Nil Then
            Begin
              Dispose (Con.Base[B].ProjectName[P]);
              Con.Base[B].ProjectName[P] := Nil;
            End;
          Con.Base[B].Category[P] := '@';
          Con.Base[B].New[P]      := 0.0;
          Con.Base[B].Rehab[P]    := 0.0;
          Con.Base[B].Cost[P]     := 0.0;
          If Con.Base[B].Comment[P] <> Nil Then
            Begin
              Dispose (Con.Base[B].Comment[P]);
              Con.Base[B].Comment[P] := Nil;
            End;
        End;
      If IndustFund.Base[B] <> Nil Then
        Begin
          Dispose (IndustFund.Base[B]);
          IndustFund.Base[B] := Nil;
        End;
    End;
END;  {InitGlobalVariables}
```

```
PROCEDURE InitStandardFactors;

  VAR
    P, Y : Integer;

  Begin
    ChangeFlag := True;
    SFFileUsed := '(none)';

    FillChar (StdFctrsCon, SizeOf (StdFctrsCon), 0);
    FillChar (StdFctrsFac, SizeOf (StdFctrsFac), 0);
    FillChar (StdFctrsPers, SizeOf (StdFctrsPers), 0);
    FillChar (StdFctrsTrans, SizeOf (StdFctrsTrans), 0);

    StdFctrsCon.Units[-1] := 'SF';
    StdFctrsCon.Units[-2] := 'SF';
    StdFctrsPers.AvgUnemployCostPerWeek := 216;
    StdFctrsPers.HAPFactor              := 0.37;
    StdFctrsPers.HAPReceive             := 0.2;
    StdFctrsPers.DARSEFactor            := 0.23;
    StdFctrsPers.DARSEReceive           := 0.15;
    StdFctrsPers.UnemployEligible       := 26;
    StdFctrsTrans.AverageTourLength     := 3;

  END; {InitStandardFactors}

{*************************************************************************
 *  Name          : HardCopy
 *
 *  Purpose       : Output report file to printer
 *
 *  Invoked As    : HardCopy (FileName)
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  06Apr92   AML   Creation
 *************************************************************************}

Function HardCopy (FileName : PathStr) : Boolean;

  Var
    C          : Char;
    FIn, FOut  : Text;
    InBuffer,
    OutBuffer  : IOBuffer;
    PrinterOK  : Boolean;

{$I-}
  Begin
    If FileExists (FileName) Then
      Begin
        Assign (FIn, FileName);
```

```
        SetTextBuf (FIn, InBuffer);
        Reset (FIn);
        Assign (FOut, Info.DeviceName);
        SetTextBuf (FOut, OutBuffer);
        If FileExists (Info.DeviceName)
          Then Append (FOut)
          Else ReWrite (FOut);
        PrinterOK := (IOResult = 0);
        While (Not EOF (FIn)) And PrinterOK Do
          Begin
            Read (FIn, C);
            Write (FOut, C);
            PrinterOK := (IOResult = 0);
          End;
        Write (FOut, Chr(12));
        Close (FOut);
        Close (FIn);
        HardCopy := PrinterOK;
      End
    Else HardCopy := False;
  End;
{$I-}

{*****************************************************************************
 *   Name          : KillBase
 *
 *   Purpose       : Deletes a base by moving all higher numbered bases down,
 *                   initializing the top base records, and decrementing
 *                   MaxBases.
 *
 *   Invoked As    : KillBase (WhichBase)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   01Apr92  AML   Creation
 *****************************************************************************}

Procedure KillBase (WhichBase : Integer);

  Var
    B, B2, P, Y : Integer;

  Begin
    If WhichBase < MaxBases
      Then For B := WhichBase To MaxBases - 1 Do
        Begin
          Screen1Data^.Base[B].Name       := Screen1Data^.Base[B+1].Name;
          Screen1Data^.Base[B].State      := Screen1Data^.Base[B+1].State;
```

```
Screen1Data^.Base[B].CloseYear  := Screen1Data^.Base[B+1].CloseYear;
Screen1Data^.Base[B].Deactivate := Screen1Data^.Base[B+1].Deactivate;

For B2 := 0 To MaxBases Do
  Begin
    Distance^.Base[B].Base[B2] := Distance^.Base[B+1].Base[B2];
    Distance^.Base[B2].Base[B] := Distance^.Base[B2].Base[B+1];
  End;
Distance^.Base[B].Base[B] := 0.0;

For B2 := 0 To MaxBases Do
  If Distance^.Base[B+1].Base[B2] > 0.0 Then
    Begin
      If PosTrans.Base[B].Base[B2] = Nil
        Then New (PosTrans.Base[B].Base[B2]);
      If PosTrans.Base[B2].Base[B] = Nil
        Then New (PosTrans.Base[B2].Base[B]);
      FillChar (PosTrans.Base[B].Base[B2]^,
        SizeOf (PosTrans.Base[B].Base[B2]^), 0);
      FillChar (PosTrans.Base[B2].Base[B]^,
        SizeOf (PosTrans.Base[B2].Base[B]^), 0);
      If ActionEnvir^.Base[B].Base[B2] = Nil
        Then New (ActionEnvir^.Base[B].Base[B2]);
      If ActionEnvir^.Base[B2].Base[B] = Nil
        Then New (ActionEnvir^.Base[B2].Base[B]);
      FillChar (ActionEnvir^.Base[B].Base[B2]^,
        SizeOf (ActionEnvir^.Base[B].Base[B2]^), 0);
      FillChar (ActionEnvir^.Base[B2].Base[B]^,
        SizeOf (ActionEnvir^.Base[B2].Base[B]^), 0);
      For Y := 1 To MaxYears Do
        Begin
          PosTrans.Base[B].Base[B2]^.Off.Year[Y] :=
            PosTrans.Base[B+1].Base[B2]^.Off.Year[Y];
          PosTrans.Base[B].Base[B2]^.Enl.Year[Y] :=
            PosTrans.Base[B+1].Base[B2]^.Enl.Year[Y];
          PosTrans.Base[B].Base[B2]^.Civ.Year[Y] :=
            PosTrans.Base[B+1].Base[B2]^.Civ.Year[Y];
          PosTrans.Base[B].Base[B2]^.Stu.Year[Y] :=
            PosTrans.Base[B+1].Base[B2]^.Stu.Year[Y];
          ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y] :=
           ActionEnvir^.Base[B+1].Base[B2]^.MoveMissionEqpmt.Year[Y];
          ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y] :=
           ActionEnvir^.Base[B+1].Base[B2]^.MoveSupportEqpmt.Year[Y];
          ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y] :=
            ActionEnvir^.Base[B+1].Base[B2]^.MilLightVehic.Year[Y];
          ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y] :=
           ActionEnvir^.Base[B+1].Base[B2]^.HeavyOrSpecVehic.Year[Y];
          PosTrans.Base[B2].Base[B]^.Off.Year[Y] :=
            PosTrans.Base[B2].Base[B+1]^.Off.Year[Y];
          PosTrans.Base[B2].Base[B]^.Enl.Year[Y] :=
            PosTrans.Base[B2].Base[B+1]^.Enl.Year[Y];
          PosTrans.Base[B2].Base[B]^.Civ.Year[Y] :=
```

```
            PosTrans.Base[B2].Base[B+1]^.Civ.Year[Y];
          PosTrans.Base[B2].Base[B]^.Stu.Year[Y] :=
            PosTrans.Base[B2].Base[B+1]^.Stu.Year[Y];
          ActionEnvir^.Base[B2].Base[B]^.MoveMissionEqpmt.Year[Y] :=
            ActionEnvir^.Base[B2].Base[B+1]^.MoveMissionEqpmt.Year[Y];
          ActionEnvir^.Base[B2].Base[B]^.MoveSupportEqpmt.Year[Y] :=
            ActionEnvir^.Base[B2].Base[B+1]^.MoveSupportEqpmt.Year[Y];
          ActionEnvir^.Base[B2].Base[B]^.MilLightVehic.Year[Y] :=
            ActionEnvir^.Base[B2].Base[B+1]^.MilLightVehic.Year[Y];
          ActionEnvir^.Base[B2].Base[B]^.HeavyOrSpecVehic.Year[Y] :=
            ActionEnvir^.Base[B2].Base[B+1]^.HeavyOrSpecVehic.Year[Y];
        End;
    End
  Else
    Begin
      If PosTrans.Base[B].Base[B2] <> Nil Then
        Begin
          Dispose (PosTrans.Base[B].Base[B2]);
          PosTrans.Base[B].Base[B2] := Nil;
        End;
      If PosTrans.Base[B2].Base[B] <> Nil Then
        Begin
          Dispose (PosTrans.Base[B2].Base[B]);
          PosTrans.Base[B2].Base[B] := Nil;
        End;
      If ActionEnvir^.Base[B].Base[B2] <> Nil Then
        Begin
          Dispose (ActionEnvir^.Base[B].Base[B2]);
          ActionEnvir^.Base[B].Base[B2] := Nil;
        End;
      If ActionEnvir^.Base[B2].Base[B] <> Nil Then
        Begin
          Dispose (ActionEnvir^.Base[B2].Base[B]);
          ActionEnvir^.Base[B2].Base[B] := Nil;
        End;
    End;

ActionEnvir^.Base[B].BaseTotOffEmploy :=
  ActionEnvir^.Base[B+1].BaseTotOffEmploy;
ActionEnvir^.Base[B].BaseTotEnlEmploy :=
  ActionEnvir^.Base[B+1].BaseTotEnlEmploy;
ActionEnvir^.Base[B].BaseTotStuEmploy :=
  ActionEnvir^.Base[B+1].BaseTotStuEmploy;
ActionEnvir^.Base[B].PercFamLivOnBase :=
  ActionEnvir^.Base[B+1].PercFamLivOnBase;
ActionEnvir^.Base[B].BaseTotCivEmploy :=
  ActionEnvir^.Base[B+1].BaseTotCivEmploy;
ActionEnvir^.Base[B].VolRIFPerc :=
  ActionEnvir^.Base[B+1].VolRIFPerc;
ActionEnvir^.Base[B].OffHousUnitsVacant :=
  ActionEnvir^.Base[B+1].OffHousUnitsVacant;
ActionEnvir^.Base[B].EnlHousUnitsVacant :=
```

```
  ActionEnvir^.Base[B+1].EnlHousUnitsVacant;
ActionEnvir^.Base[B].BaseFacilSqFt :=
  ActionEnvir^.Base[B+1].BaseFacilSqFt;
ActionEnvir^.Base[B].AcreOnBase :=
  ActionEnvir^.Base[B+1].AcreOnBase;
HAPSwitch.Base[B] :=
  HAPSwitch.Base[B+1];
IndustFundSwitch.Base[B] :=
  IndustFundSwitch.Base[B+1];
PersCost.Base[B].OffVHA :=
  PersCost.Base[B+1].OffVHA;
PersCost.Base[B].EnlVHA :=
  PersCost.Base[B+1].EnlVHA;
PersCost.Base[B].PerDiemRate :=
  PersCost.Base[B+1].PerDiemRate;
FreightCost.Base[B] :=
  FreightCost.Base[B+1];
Con.Base[B].AreaCostFactor :=
  Con.Base[B+1].AreaCostFactor;
BudgetPropData^.Base[B].RPMANonPayBudget :=
  BudgetPropData^.Base[B+1].RPMANonPayBudget;
BudgetPropData^.Base[B].RPMAPayBudget :=
  BudgetPropData^.Base[B+1].RPMAPayBudget;
BudgetPropData^.Base[B].CommunBudget :=
  BudgetPropData^.Base[B+1].CommunBudget;
BudgetPropData^.Base[B].BaseOpnsNonPayBudget :=
  BudgetPropData^.Base[B+1].BaseOpnsNonPayBudget;
BudgetPropData^.Base[B].BaseOpnsPay :=
  BudgetPropData^.Base[B+1].BaseOpnsPay;
BudgetPropData^.Base[B].FamHousBudget :=
  BudgetPropData^.Base[B+1].FamHousBudget;
Champus.Base[B].ChampInPatCostVisit :=
  Champus.Base[B+1].ChampInPatCostVisit;
Champus.Base[B].ChampOutPatCostVisit :=
  Champus.Base[B+1].ChampOutPatCostVisit;
ActionEnvir^.Base[B].ChampusToMedicare :=
  ActionEnvir^.Base[B+1].ChampusToMedicare;

For Y := 1 To MaxYears Do
  Begin
    PosReduc.Base[B].Off.Year[Y] :=
      PosReduc.Base[B+1].Off.Year[Y];
    PosReduc.Base[B].Enl.Year[Y] :=
      PosReduc.Base[B+1].Enl.Year[Y];
    PosReduc.Base[B].Civ.Year[Y] :=
      PosReduc.Base[B+1].Civ.Year[Y];
    PosElim.Base[B].Off.Year[Y] :=
      PosElim.Base[B+1].Off.Year[Y];
    PosElim.Base[B].Enl.Year[Y] :=
      PosElim.Base[B+1].Enl.Year[Y];
    PosElim.Base[B].Civ.Year[Y] :=
      PosElim.Base[B+1].Civ.Year[Y];
```

```
      PosTrans.Base[B].CareTakeMil.Year[Y] :=
        PosTrans.Base[B+1].CareTakeMil.Year[Y];
      PosTrans.Base[B].CareTakeCiv.Year[Y] :=
        PosTrans.Base[B+1].CareTakeCiv.Year[Y];
      CostAvoid.Base[B].Constr.Year[Y] :=
        CostAvoid.Base[B+1].Constr.Year[Y];
      CostAvoid.Base[B].FamQtr.Year[Y] :=
        CostAvoid.Base[B+1].FamQtr.Year[Y];
      CostAvoid.Base[B].Procure.Year[Y] :=
        CostAvoid.Base[B+1].Procure.Year[Y];
      ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[Y] :=
        ActionEnvir^.Base[B+1].OneTimeUniqueCosts.Year[Y];
      ActionEnvir^.Base[B].OneTimeMovingCosts.Year[Y] :=
        ActionEnvir^.Base[B+1].OneTimeMovingCosts.Year[Y];
      ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y] :=
        ActionEnvir^.Base[B+1].EnvirMitigReqd.Year[Y];
      ActivMisCost^.Base[B].Year[Y] :=
        ActivMisCost^.Base[B+1].Year[Y];
      BudgetPropData^.Base[B].MiscRecurCost.Year[Y] :=
        BudgetPropData^.Base[B+1].MiscRecurCost.Year[Y];
      RealPropProc.RealPropPur.Base[B].Acre.Year[Y] :=
        RealPropProc.RealPropPur.Base[B+1].Acre.Year[Y];
      RealPropProc.RealPropPur.Base[B].Cost.Year[Y] :=
        RealPropProc.RealPropPur.Base[B+1].Cost.Year[Y];
      TimePhasShutDn.Base[B].Constr.Year[Y] :=
        TimePhasShutDn.Base[B+1].Constr.Year[Y];
      TimePhasShutDn.Base[B].ShutDn.Year[Y] :=
        TimePhasShutDn.Base[B+1].ShutDn.Year[Y];
      Champus.Base[B].OnBaseInPatVisits.Year[Y] :=
        Champus.Base[B+1].OnBaseInPatVisits.Year[Y];
      Champus.Base[B].OnBaseOutPatVisits.Year[Y] :=
        Champus.Base[B+1].OnBaseOutPatVisits.Year[Y];
    End;
  ActionEnvir^.Base[B].FamHousShutDn :=
    ActionEnvir^.Base[B+1].FamHousShutDn;
  RealPropProc.Base[B].FacilSqFtShutDown :=
    RealPropProc.Base[B+1].FacilSqFtShutDown;

  For P := 1 To 16 Do
    Begin
      If Con.Base[B].ProjectName[P] <> Nil
        Then Dispose (Con.Base[B].ProjectName[P]);
      Con.Base[B].ProjectName[P] := Con.Base[B+1].ProjectName[P];
      Con.Base[B+1].ProjectName[P] := Nil;
      Con.Base[B].Category[P] :=
        Con.Base[B+1].Category[P];
      Con.Base[B].New[P] :=
        Con.Base[B+1].New[P];
      Con.Base[B].Rehab[P] :=
        Con.Base[B+1].Rehab[P];
      Con.Base[B].Cost[P] :=
        Con.Base[B+1].Cost[P];
```

```
            If Con.Base[B].Comment[P] <> Nil
              Then Dispose (Con.Base[B].Comment[P]);
            Con.Base[B].Comment[P] := Con.Base[B+1].Comment[P];
            Con.Base[B+1].Comment[P] := Nil;
          End;

        If IndustFund.Base[B] <> Nil
          Then Dispose (IndustFund.Base[B]);
        IndustFund.Base[B] := IndustFund.Base[B+1];
        IndustFund.Base[B+1] := Nil;
      End;

  Screen1Data^.Base[MaxBases].Name := '';
  Screen1Data^.Base[MaxBases].State := '';
  Screen1Data^.Base[MaxBases].CloseYear := 0;
  Screen1Data^.Base[MaxBases].Deactivate := 'N';

  For B := 0 To MaxBases Do
    Begin
      Distance^.Base[B].Base[MaxBases] := 0.0;
      Distance^.Base[MaxBases].Base[B] := 0.0;
    End;

  For B := 0 To MaxBases Do
    Begin
      If PosTrans.Base[MaxBases].Base[B] <> Nil Then
        Begin
          Dispose (PosTrans.Base[MaxBases].Base[B]);
          PosTrans.Base[MaxBases].Base[B] := Nil;
        End;
      If PosTrans.Base[B].Base[MaxBases] <> Nil Then
        Begin
          Dispose (PosTrans.Base[B].Base[MaxBases]);
          PosTrans.Base[B].Base[MaxBases] := Nil;
        End;
      If ActionEnvir^.Base[MaxBases].Base[B] <> Nil Then
        Begin
          Dispose (ActionEnvir^.Base[MaxBases].Base[B]);
          ActionEnvir^.Base[MaxBases].Base[B] := Nil;
        End;
      If ActionEnvir^.Base[B].Base[MaxBases] <> Nil Then
        Begin
          Dispose (ActionEnvir^.Base[B].Base[MaxBases]);
          ActionEnvir^.Base[B].Base[MaxBases] := Nil;
        End;
    End;

  ActionEnvir^.Base[MaxBases].BaseTotOffEmploy := 0.0;
  ActionEnvir^.Base[MaxBases].BaseTotEnlEmploy := 0.0;
  ActionEnvir^.Base[MaxBases].BaseTotStuEmploy := 0.0;
  ActionEnvir^.Base[MaxBases].PercFamLivOnBase := 0.0;
  ActionEnvir^.Base[MaxBases].BaseTotCivEmploy := 0.0;
```

```
ActionEnvir^.Base[MaxBases].VolRIFPerc := 0.0;
ActionEnvir^.Base[MaxBases].OffHousUnitsVacant := 0.0;
ActionEnvir^.Base[MaxBases].EnlHousUnitsVacant := 0.0;
ActionEnvir^.Base[MaxBases].BaseFacilSqFt := 0.0;
ActionEnvir^.Base[MaxBases].AcreOnBase := 0.0;
HAPSwitch.Base[MaxBases] := 'N';
IndustFundSwitch.Base[MaxBases] := 'N';
PersCost.Base[MaxBases].OffVHA := 0.0;
PersCost.Base[MaxBases].EnlVHA := 0.0;
PersCost.Base[MaxBases].PerDiemRate := 0.0;
FreightCost.Base[MaxBases] := 0.0;
Con.Base[MaxBases].AreaCostFactor := 1.0;
BudgetPropData^.Base[MaxBases].RPMANonPayBudget := 0.0;
BudgetPropData^.Base[MaxBases].RPMAPayBudget := 0.0;
BudgetPropData^.Base[MaxBases].CommunBudget := 0.0;
BudgetPropData^.Base[MaxBases].BaseOpnsNonPayBudget := 0.0;
BudgetPropData^.Base[MaxBases].BaseOpnsPay := 0.0;
BudgetPropData^.Base[MaxBases].FamHousBudget := 0.0;
Champus.Base[MaxBases].ChampInPatCostVisit := 0.0;
Champus.Base[MaxBases].ChampOutPatCostVisit := 0.0;
ActionEnvir^.Base[MaxBases].ChampusToMedicare := 0.0;

For Y := 1 To MaxYears Do
  Begin
    PosReduc.Base[MaxBases].Off.Year[Y] := 0;
    PosReduc.Base[MaxBases].Enl.Year[Y] := 0;
    PosReduc.Base[MaxBases].Civ.Year[Y] := 0;
    PosElim.Base[MaxBases].Off.Year[Y] := 0;
    PosElim.Base[MaxBases].Enl.Year[Y] := 0;
    PosElim.Base[MaxBases].Civ.Year[Y] := 0;
    PosTrans.Base[MaxBases].CareTakeMil.Year[Y] := 0;
    PosTrans.Base[MaxBases].CareTakeCiv.Year[Y] := 0;
    CostAvoid.Base[MaxBases].Constr.Year[Y] := 0.0;
    CostAvoid.Base[MaxBases].FamQtr.Year[Y] := 0.0;
    CostAvoid.Base[MaxBases].Procure.Year[Y] := 0.0;
    ActionEnvir^.Base[MaxBases].OneTimeUniqueCosts.Year[Y] := 0.0;
    ActionEnvir^.Base[MaxBases].OneTimeMovingCosts.Year[Y] := 0.0;
    ActionEnvir^.Base[MaxBases].EnvirMitigReqd.Year[Y] := 0.0;
    ActivMisCost^.Base[MaxBases].Year[Y] := 0.0;
    BudgetPropData^.Base[MaxBases].MiscRecurCost.Year[Y] := 0.0;
    RealPropProc.RealPropPur.Base[MaxBases].Acre.Year[Y] := 0.0;
    RealPropProc.RealPropPur.Base[MaxBases].Cost.Year[Y] := 0.0;
    TimePhasShutDn.Base[MaxBases].Constr.Year[Y] := 0.0;
    TimePhasShutDn.Base[MaxBases].ShutDn.Year[Y] := 0.0;
    Champus.Base[MaxBases].OnBaseInPatVisits.Year[Y] := 0.0;
    Champus.Base[MaxBases].OnBaseOutPatVisits.Year[Y] := 0.0;
  End;
RealPropProc.Base[MaxBases].FacilSqFtShutDown := 0.0;
ActionEnvir^.Base[MaxBases].FamHousShutDn := 0.0;
```

```
    For P := 1 To 16 Do
      Begin
        If Con.Base[MaxBases].ProjectName[P] <> Nil Then
          Begin
            Dispose (Con.Base[MaxBases].ProjectName[P]);
            Con.Base[MaxBases].ProjectName[P] := Nil;
          End;
        Con.Base[MaxBases].Category[P] := '@';
        Con.Base[MaxBases].New[P] := 0.0;
        Con.Base[MaxBases].Rehab[P] := 0.0;
        Con.Base[MaxBases].Cost[P] := 0.0;
        If Con.Base[MaxBases].Comment[P] <> Nil Then
          Begin
            Dispose (Con.Base[MaxBases].Comment[P]);
            Con.Base[MaxBases].Comment[P] := Nil;
          End;
      End;

    If IndustFund.Base[MaxBases] <> Nil Then
      Begin
        Dispose (IndustFund.Base[MaxBases]);
        IndustFund.Base[MaxBases] := Nil;
      End;

    If MaxBases > 0
      Then Dec (MaxBases);
  End;

{*****************************************************************************
 *   Name          : MakeReport
 *
 *   Purpose       : Makes a Base-Specific Report
 *
 *   Notes         : Uses pages from PERSMOVE, 1TIMCOST, and MILCONAS.
 *                 : Chr(12) is the FORMFEED Character.
 *
 *   Invoked As    : MakeReport (WhichBase, FileName)
 *
 *   Return value  : Word
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   21May92  AML   Creation
 *****************************************************************************}

Function MakeReport (WhichBase: Integer; FileName: PathStr):Boolean;
```

```
Var
  InBuffer,
  OutBuffer : IOBuffer;
  Counter,
  Page      : Integer;
  InFil,
  OutFil    : Text;
  NoProblem : Boolean;
  ReportData,
  TempFile  : PathStr;

Begin
  {$I-}
  Assign (OutFil, ConCat(FileName, '.RPT'));
  SetTextBuf (OutFil, OutBuffer);
  ReWrite (OutFil);
  {$I+}
  NoProblem := (IOResult = 0);
  If NoProblem
    Then For Counter := 1 To 3 Do
      Begin
        Case Counter Of
          1: TempFile := Info.ReportDir + '1TIMCOST.RPT';
          2: TempFile := Info.ReportDir + 'MILCONAS.RPT';
          3: TempFile := Info.ReportDir + 'PERSMOVE.RPT';
          End;
        If FileExists (TempFile) Then
          Begin
            Assign (InFil, TempFile);
            SetTextBuf (InFil, InBuffer);
            Reset (InFil);
            Page := 0;
            While Not Eof (InFil) Do
              Begin
                Readln (InFil, ReportData);
                If (Pos(Chr(12), ReportData) > 0) And
                   (Page <= MaxBases)
                  Then Inc (Page)
                Else If (Page = WhichBase)
                  Then Writeln (OutFil, ReportData);
              End;
            If Counter < 3
              Then Writeln (OutFil, Chr(12));
            Close (InFil);
          End
        Else NoProblem := False;
      End;
  Close (OutFil);
  If Not NoProblem
    Then Erase (OutFil);
  MakeReport := NoProblem;
End;
```

```
{*************************************************************************
 *   Name          : OptionSearch
 *
 *   Purpose       : Displays a list of COBRA files in the current directory,
 *                   along with Option Package Name and Summary lines (if any)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   08Jul92  AML   Creation
 *************************************************************************}

Procedure OptionSearch;

Var
  InBuffer,
  OutBuffer : IOBuffer;
  C         : Char;
  Count,
  Count2,
  Filler,
  TotalCOB,
  TotalSF   : Integer;
  Dir       : PathStr;
  Fin, Fout : Text;
  OptPkg    : String20;
  RowName   : String10;
  SR        : SearchRec;
  Str       : String60;

  Function Pad20 (Stuff : String): String;
    Begin
      Stuff := StripSpace (Stuff);
      If Length (Stuff) > 20
        Then Stuff := Copy (Stuff, 1, 20)
      Else While Length (Stuff) < 20 Do
        Stuff := Concat (Stuff, ' ');
      Pad20 := Stuff;
    End;

Begin
  Dir := FExpand ('');
  Assign (FOut, Info.ReportDir + 'COBFILES.RPT');
  SetTextBuf (FOut, OutBuffer);
  ReWrite (FOut);
  For Count := 1 To 6 Do
    Writeln (FOut);
  Center (FOut, 'COBRA FILES IN ' + Dir);
  Center (FOut, '(As of ' + AsOfDateTime + ')');
```

```
Writeln (FOut);
Writeln (FOut);
TotalCOB := 0;
TotalSF := 0;
{$I-}
FindFirst ('*.CBR', 0, SR);
{$I+}
While (DosError = 0) Do
  Begin
    Inc (TotalCOB);
    If TotalCOB = 1 Then
      Begin
        Writeln (FOut, Blank:10, 'Description:                File Name:');
        Writeln (FOut, Blank:10, '------------                ----------');
      End;
    Assign (FIn, SR.Name);
    SetTextBuf (Fin, InBuffer);
    Reset (FIn);
    Readln (FIn, RowName, C, Count, Filler, Filler);
    Readln (FIn, Str);
    If StripSpace (Copy (Str,11,20)) = ''
      Then Writeln
        (FOut, Blank:10, '(Scenario Data File)     ', Dir + SR.Name)
      Else Writeln
        (FOut, Blank:10, Pad20 (Copy (Str, 11, 20)), '     ', Dir + SR.Name);
    For Count2 := 0 To Count + 2 Do
      Readln (FIn, Str);
    For Count2 := 1 To 6 Do
      Begin
        Readln (FIn, Str);
        If StripSpace(Str) <> ''
          Then Writeln (FOut, Blank:15, Str);
      End;
    Writeln (FOut);
    Close (FIn);
    FindNext (SR);
  End;
{$I-}
FindFirst ('*.COB', 0, SR);
{$I+}
While DosError = 0 Do
  Begin
    Inc (TotalCOB);
    If TotalCOB = 1 Then
      Begin
        Writeln (FOut, Blank:10, 'Description:                File Name:');
        Writeln (FOut, Blank:10, '------------                ----------');
      End;
    Assign (FIn, SR.Name);
    SetTextBuf (FIn, InBuffer);
    Reset (FIn);
    Readln (FIn, Count, Filler, Filler, Filler, Filler, Filler);
```

```
    For Count2 := 0 To Count + 1 Do
      Readln (FIn, Str);
    If StripSpace (Copy (Str, 31, 20)) = ''
      Then Writeln
        (FOut, Blank:10, '(Old Data File)          ', Dir + SR.Name)
      Else Writeln
        (FOut, Blank:10, Pad20 (Copy (Str, 31, 20)), '     ', Dir + SR.Name);
    Writeln (FOut);
    Close (FIn);
    FindNext (SR);
  End;
{$I-}
FindFirst ('*.SFF', 0, SR);
{$I+}
While (DosError = 0) Do
  Begin
    Inc (TotalSF);
    If TotalCOB + TotalSF = 1 Then
      Begin
        Writeln (FOut, Blank:10, 'Description:             File Name:');
        Writeln (FOut, Blank:10, '------------             ----------');
      End;
    Assign (FIn, Dir + SR.Name);
    SetTextBuf (FIn, InBuffer);
    Reset (FIn);
    For Count2 := 1 To 11 Do
      If EOF (FIn)
        Then Str := ''
        Else Readln (FIn, Str);
    If StripSpace (Copy (Str, 11, 20)) = ''
      Then Writeln
        (FOut, Blank:10, '(Standard Factors)       ', Dir + SR.Name)
      Else Writeln
        (FOut, Blank:10, Pad20 (Copy (Str, 11, 20)), '     ', Dir + SR.Name);
    Writeln (FOut);
    Close (FIn);
    FindNext (SR);
  End;
Writeln (Fout);
If TotalCOB + TotalSF > 0 Then
  Begin
    Write (FOut, Blank:10, 'There ');
    If TotalCob + TotalSF = 1
      Then Write (FOut, 'is')
      Else Write (FOut, 'are');
    If TotalCob > 10
      Then Write (FOut, TotalCob:3)
    Else If TotalCob > 0
      Then Write (FOut, TotalCob:2);
    If TotalCob > 0
      Then Write (FOut, ' COBRA data file');
    If TotalCob > 1
```

```
        Then Write (FOut, 's');
      If TotalCob * TotalSF > 0
        Then Write (FOut, ' and');
      If TotalSF > 10
        Then Write (FOut, TotalSF:3)
      Else If TotalSF > 0
        Then Write (FOut, TotalSF:2);
      If TotalSF > 0
        Then Write (FOut, ' Standard Factors file');
      If TotalSF > 1
        Then Write (FOut, 's');
      Writeln (FOut, '.');
    End;
  Close (FOut);
  If TotalCOB + TotalSF = 0
    Then Erase (FOut);
End;

{*************************************************************************
 *   Name          : UpdateDataFile
 *
 *   Purpose       : Places a stripped (no path or extension) and padded
 *                   (with spaces to 9 characters) version of DataFileUsed
 *                   on the Status Line (with "*" for ".COB" files)
 *
 *   Invoked As    : UpdateDataFile (FileName)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   07May92  AML   Creation
 *************************************************************************}

Procedure UpdateDataFile (Var FileName : String);

  Var
    Temp : String;

  Begin
    Temp := StripFile (DataFileUsed);
    If (Pos('.COB',DataFileUsed) > 0) Or (Pos('.COB',SFFileUsed) > 0)
      Then Temp := Concat (Temp, '*');
    While Length (Temp) < 8 Do
      Temp := Concat (Blank, Temp, Blank);
    If Length (Temp) < 9
      Then Temp := Concat (Temp, Blank);
    FileName := Temp;
  End;
```

```
Procedure CapCheck (Var BaseName : String20);

  Var
    C     : Integer;
    Cap,
    Stuck : Boolean;

  Begin
    Stuck := False;
    If (Length(BaseName) > 2) And (BaseName[1] = '@') Then
      Begin
        BaseName := Copy (BaseName, 2, Length(BaseName) - 1);
        Cap := True;
        For C := 1 To Length(BaseName) Do
          If (BaseName[C] > '''') And (BaseName[C] < '{') And Cap Then
            Begin
              BaseName[C] := Chr(Ord(BaseName[C])-32);
              If Not Stuck
                Then Cap := False;
            End
          Else If (BaseName[C] = '@') Then
            Begin
              Stuck := Not Stuck;
              If Stuck
                Then Cap := True;
            End
          Else If (BaseName[C] > '@') And (BaseName[C] < '[') And Not Cap
            Then BaseName[C] := Chr(Ord(BaseName[C])+32)
          Else If (BaseName[C] > '@') And (BaseName[C] < '[')
                   And Cap And Not Stuck
            Then Cap := False
          Else If (BaseName[C] = ' ') Or (BaseName[C] = ',') Or
                  (BaseName[C] = '-') Or (BaseName[C] = '.') Or
                  (BaseName[C] = '(') Or (BaseName[C] = ')') Or
                  (BaseName[C] = '[') Or (BaseName[C] = ']') Or
                  (BaseName[C] = '/') Or (BaseName[C] = ':')
            Then Cap := True;
      End;
    While Pos ('@', BaseName) > 0 Do
      Delete (BaseName, Pos ('@', BaseName), 1)
  End;

END.
```

```
Unit CIO;

{$F+,O+,S-,D+,L+}

Interface

Uses
  DOS,
  CDEC,
  CSUP;

Function  InputData     (Fname      : PathStr) : Boolean;

Function  InputSF       (Fname      : PathStr) : Boolean;

Function  InputOld      (Fname      : PathStr) : Boolean;

Function  OutputData    (Fname      : PathStr) : Boolean;

Function  OutputSF      (Fname      : PathStr) : Boolean;

Function  DBCheck       (WhichBase : Integer;
                         FName      : PathStr) : Boolean;

Procedure DBLoad        (BName      : String20;
                         WhichBase : Integer;
                         FName      : PathStr);

Procedure DBNames       (FName      : PathStr;
                     Var Names      : PDBaseRec;
                     Var Number     : Integer);

Procedure DBSave        (WhichBase : Integer;
                         FName      : PathStr);

Procedure DBDistSave    (FName      : PathStr;
                         Base1,
                         Base2      : String20;
                         DistDB,
                         DistMem    : Real);

Procedure RightJust(Var Fil         : Text;
                        Str         : String;
                        Width       : Integer);

Implementation
```

```
{*************************************************************************
 *   Name           : RightJust
 *
 *   Purpose        : Writes a String out to a file padded with spaces
 *
 *   Invoked As     : RightJust (Fil,str,width)
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 3.30+
 *     Production : MS-DOS 3.30+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   30JAN91   GJG   Creation
 *************************************************************************}

Procedure RightJust (Var Fil   : Text;
                         Str   : String;
                         Width : Integer);

  Begin
    Write (Fil, Str:0, Copy (Rept (Blank,30), 1, Width - Length (Str)));
  End;

{*************************************************************************
 *   Name           : input
 *
 *   Purpose        : Opens and reads COBRA application data from file specified.
 *
 *   Notes          : Originally included Standard Factors
 *
 *   Invoked As     : InputData (Fname)
 *
 *   Files/Tables
 *     Names                 Access    Purpose
 *     ------------------    --------  ----------------------------------------
 *     fname                 String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 3.30+
 *     Production : MS-DOS 3.30+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   17JAN90   GJG   Creation
 *   27Mar92   AML   Separated Standard Factors
 *************************************************************************}


Function InputData (Fname : PathStr) : Boolean;
```

```
Var
  BigName    : String60;
  Buffer     : IOBuffer;
  B, B2,
  Fil, P, Y  : Integer;
  F          : Text;
  Filler     : Real;
  Garbage    : String;
  RowName    : String10;
  TempStr    : String20;

Begin
  InputData := FileExists(FName);
  If FileExists (FName) Then
    Begin
      Assign (F, FName);
      SetTextBuf (F, Buffer);
      Reset (F);

      { General Environment - Screen 1 }

      Readln (F, RowName, Screen1Data^.ModelDoShutDn, MaxBases, Fil,
        Screen1Data^.FY1);

      Readln (F, RowName, Screen1Data^.OptionPkgName,
        Screen1Data^.Service, Screen1Data^.Group);
      Screen1Data^.OptionPkgName := StripSpace(Screen1Data^.OptionPkgName);
      Screen1Data^.Service := StripSpace(Screen1Data^.Service);
      Screen1Data^.Group := StripSpace(Screen1Data^.Group);

      For B := 0 TO MaxBases DO
        Begin
          Readln (F, RowName,
            Screen1Data^.Base[B].Name,
            Screen1Data^.Base[B].Deactivate,
            Screen1Data^.Base[B].CloseYear);
          Screen1Data^.Base[B].Name := StripSpace(Screen1Data^.Base[B].Name);
          If LastYear < Screen1Data^.Base[B].CloseYear
            Then LastYear := Screen1Data^.Base[B].CloseYear;
        End;
      If LastYear = 0
        Then LastYear := MaxYears;

      Readln (F, RowName, Screen1Data^.SFFile);
      Screen1Data^.SFFile := StripSpace (Screen1Data^.SFFile);

      Readln (F, RowName);
      For P := 1 To 6 Do
        Begin
          Readln (F, Screen1Data^.Summary[P]);
          Screen1Data^.Summary[P] := StripSpace(Screen1Data^.Summary[P]);
        End;
```

```
Readln (F, RowName, Screen1Data^.DateLine);
Screen1Data^.DateLine := StripSpace(Screen1Data^.DateLine);

{ Distance Table - Screen 2 }

For B := 0 To MaxBases Do
  Begin
    Read (F, RowName);
    For B2 := 0 To MaxBases Do
      Read (F, Distance^.Base[B].Base[B2]);
    Readln (F);
  End;

{ Transfer Table - Screen 3 }

For B := 0 To MaxBases Do
  For B2 := 0 To MaxBases Do
    If (Distance^.Base[B].Base[B2] > 0) Then
      Begin
        New (PosTrans.Base[B].Base[B2]);
        FillChar (PosTrans.Base[B].Base[B2]^,
          SizeOf (PosTrans.Base[B].Base[B2]^), 0);
        New (ActionEnvir^.Base[B].Base[B2]);
        FillChar (ActionEnvir^.Base[B].Base[B2]^,
          SizeOf (ActionEnvir^.Base[B].Base[B2]^), 0);
        Read (F, RowName);
        For Y := 1 To MaxYears Do
          Read (F, PosTrans.Base[B].Base[B2]^.Off.Year[Y]);
        Readln (F);
        Read (F, RowName);
        For Y := 1 To MaxYears Do
          Read (F, PosTrans.Base[B].Base[B2]^.Enl.Year[Y]);
        Readln (F);
        Read (F, RowName);
        For Y := 1 To MaxYears Do
          Read (F, PosTrans.Base[B].Base[B2]^.Civ.Year[Y]);
        Readln (F);
        Read (F, RowName);
        For Y := 1 To MaxYears Do
          Read (F, PosTrans.Base[B].Base[B2]^.Stu.Year[Y]);
        Readln (F);
        Read (F, RowName);
        For Y := 1 To MaxYears Do
          Read (F,
            ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y]);
        Readln (F);
        Read (F, RowName);
        For Y := 1 To MaxYears Do
          Read (F,
            ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y]);
        Readln (F);
        Read (F, RowName);
```

```
          For Y := 1 To MaxYears Do
           Read (F, ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y]);
          Readln (F);
          Read (F, RowName);
          For Y := 1 To MaxYears Do
            Read (F,
              ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y]);
          Readln (F);
        End
      Else If RowName[10] <> '2'
        Then For Fil := 1 To 8 Do
          Readln (F, Garbage);

{ Static Base Info - Screen 4 }

For B := 0 To MaxBases Do
  Begin
    Readln (F, RowName,
      ActionEnvir^.Base[B].BaseTotOffEmploy,
      ActionEnvir^.Base[B].BaseTotEnlEmploy,
      ActionEnvir^.Base[B].BaseTotStuEmploy,
      ActionEnvir^.Base[B].PercFamLivOnBase,
      ActionEnvir^.Base[B].BaseTotCivEmploy,
      ActionEnvir^.Base[B].VolRIFPerc, Filler,
      ActionEnvir^.Base[B].OffHousUnitsVacant,
      ActionEnvir^.Base[B].EnlHousUnitsVacant,
      ActionEnvir^.Base[B].BaseFacilSqFt,
      ActionEnvir^.Base[B].AcreOnBase, Filler);
    Readln (F, RowName,
      IndustFundSwitch.Base[B],
      PersCost.Base[B].OffVHA,
      PersCost.Base[B].EnlVHA,
      PersCost.Base[B].PerDiemRate,
      FreightCost.Base[B],
      Con.Base[B].AreaCostFactor);
    If StripSpace(RowName) <> 'CostInfo2' Then
      Begin
        PersCost.Base[B].OffVHA :=
          Round(PersCost.Base[B].OffVHA / 12.0);
        PersCost.Base[B].EnlVHA :=
          Round(PersCost.Base[B].EnlVHA / 12.0);
      End;
    Readln (F, RowName,
      BudgetPropData^.Base[B].RPMANonPayBudget,
      BudgetPropData^.Base[B].RPMAPayBudget,
      BudgetPropData^.Base[B].CommunBudget,
      BudgetPropData^.Base[B].BaseOpnsNonPayBudget,
      BudgetPropData^.Base[B].BaseOpnsPay,
      BudgetPropData^.Base[B].FamHousBudget, Filler);
    Readln (F, RowName, Filler, Filler,
      Champus.Base[B].ChampInPatCostVisit,
      Champus.Base[B].ChampOutPatCostVisit);
```

```
  End;
If IndustFundSwitch.Base[B] <> 'Y'
  Then IndustFundSwitch.Base[B] := 'N';

{ Dynamic Base Info - Screen 5 }

For B := 0 To MaxBases Do
  Begin
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, CostAvoid.Base[B].Constr.Year[Y]);
    Readln (F);
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, CostAvoid.Base[B].Procure.Year[Y]);
    Readln (F, Filler, ActionEnvir^.Base[B].FamHousShutDn);
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[Y]);
    Readln (F);
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, ActionEnvir^.Base[B].OneTimeMovingCosts.Year[Y]);
    Readln (F);
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y]);
    Readln (F);
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Begin
        Read (F, ActivMisCost^.Base[B].Year[Y]);
        If (Screen1Data^.Base[B].CloseYear > 0) And
           (StripSpace(RowName) = 'ActivMis')
          Then ActivMisCost^.Base[B].Year[Y] :=
             - ActivMisCost^.Base[B].Year[Y];
      End;
    Readln (F);
    If StripSpace(RowName) = 'ActivMis'
      Then For Y := 2 To MaxYears Do
        ActivMisCost^.Base[B].Year[Y] :=
            ActivMisCost^.Base[B].Year[Y]
          + ActivMisCost^.Base[B].Year[Y-1];
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, RealPropProc.RealPropPur.Base[B].Acre.Year[Y]);
    Readln (F);
    Read (F, RowName);
    For Y := 1 To MaxYears Do
      Read (F, RealPropProc.RealPropPur.Base[B].Cost.Year[Y]);
    Readln (F);
    Read (F, RowName);
```

```
      For Y := 1 To MaxYears Do
        Begin
          Read (F, Filler);
          RealPropProc.Base[B].FacilSqFtShutDown :=
            RealPropProc.Base[B].FacilSqFtShutDown + Filler;
        End;
      Readln (F);
      Readln (F, RowName, Filler, Filler, Filler, Filler, Filler, Filler);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, TimePhasShutDn.Base[B].Constr.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, TimePhasShutDn.Base[B].ShutDn.Year[Y]);
      Readln (F);
    End;

  { Base Personnel Info - Screen 6 }

  For B := 0 To MaxBases Do
    Begin
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosReduc.Base[B].Off.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosReduc.Base[B].Enl.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosReduc.Base[B].Civ.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosElim.Base[B].Off.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosElim.Base[B].Enl.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosElim.Base[B].Civ.Year[Y]);
      Readln (F);
      Read (F, RowName);
      For Y := 1 To MaxYears Do
        Read (F, PosTrans.Base[B].CareTakeMil.Year[Y]);
      Readln (F);
      Read (F, RowName);
```

```
    For Y := 1 To MaxYears Do
      Read (F, PosTrans.Base[B].CareTakeCiv.Year[Y]);
    Readln (F);
  End;

{ Military Construction Data - Screen 7 }

For B := 0 To MaxBases Do
  Begin
    Read (F, RowName);
    For P := 1 To 16 Do
      Read (F, Con.Base[B].Category[P]);
    Readln (F);
    Read (F, RowName);
    For P := 1 To 16 Do
      Begin
        Read (F, TempStr);
        TempStr := StripSpace (TempStr);
        If TempStr <> '' Then
          Begin
            New (Con.Base[B].ProjectName[P]);
            Con.Base[B].ProjectName[P]^ := TempStr;
          End;
      End;
    Readln (F);
    Read (F, RowName);
    For P := 1 To 16 Do
      Read (F, Con.Base[B].New[P]);
    Readln (F);
    Read (F, RowName);
    For P := 1 To 16 Do
      Read (F, Con.Base[B].Rehab[P]);
    Readln (F);
    Read (F, RowName);
    For P := 1 To 16 Do
      Read (F, Con.Base[B].Cost[P]);
    Readln (F);
    Readln (F, RowName);
    For P := 1 To 16 Do
      Begin
        Readln (F, BigName);
        BigName := StripSpace (BigName);
        If BigName <> '' Then
          Begin
            New (Con.Base[B].Comment[P]);
            Con.Base[B].Comment[P]^ := BigName;
          End;
      End;
  End;

{ Industrial Funded Facilities data - Screen 8 }
```

```
For B := 0 To MaxBases Do
  If IndustFundSwitch.Base[B] = 'Y' Then
    Begin
      New (IndustFund.Base[B]);
      For Y := 1 To MaxYears Do
        Readln (F, RowName,
          IndustFund.Base[B]^.OvhdCost.AdminPlan.Year[Y],
          IndustFund.Base[B]^.OvhdCost.Operating.Year[Y],
          IndustFund.Base[B]^.OvhdCost.Mothball.Year[Y],
          IndustFund.Base[B]^.OvhdCost.Caretaker.Year[Y],
          IndustFund.Base[B]^.OvhdCost.MaintSpace.Year[Y],
          IndustFund.Base[B]^.OvhdCost.Other.Year[Y],
          IndustFund.Base[B]^.OvhdSave.Operating.Year[Y],
          IndustFund.Base[B]^.OvhdSave.Other.Year[Y],
          IndustFund.Base[B]^.MoveCost.PackUnpack.Year[Y],
          IndustFund.Base[B]^.MoveCost.Freight.Year[Y],
          IndustFund.Base[B]^.MoveCost.Vehic.Year[Y],
          IndustFund.Base[B]^.MoveCost.Driv.Year[Y],
          IndustFund.Base[B]^.MoveCost.LossRate.Year[Y]);
    End;

{ Footnotes/Comments - Screen 9 }

Readln (F, RowName, Screen9Data.Pages);
For P := 1 To 15 Do
  Readln (F, Screen9Data.Line[P]);

{ Stuff added after v3.73 Prototype delivery }

If Not EOF (F) Then
  Begin
    Read (F, RowName);
    For B := 0 To MaxBases Do
      Begin
        Read (F, Screen1Data^.Base[B].State);
        Screen1Data^.Base[B].State :=
          StripSpace(UpperCase(Screen1Data^.Base[B].State));
      End;
    For B := 0 To MaxBases Do
      Begin
        Read (F, HapSwitch.Base[B]);
        If HapSwitch.Base[B] <> 'Y'
          Then HapSwitch.Base[B] := 'N'
      End;
    Readln (F);
  End;

If Not EOF (F)
  Then For B := 0 To MaxBases Do
    Begin
      Read (F, RowName);
```

```
            For Y := 1 To MaxYears Do
              Read (F, CostAvoid.Base[B].FamQtr.Year[Y]);
            Readln(F);
          End;

      If Not EOF (F) Then
        Begin
          Read (F, RowName);
          For B := 0 To MaxBases Do
            Read (F, ActionEnvir^.Base[B].ChampusToMedicare);
          Readln (F);
        End;

      If Not EOF (F)
        Then For B := 0 To MaxBases Do
          Begin
            Read (F, RowName);
            For Y := 1 To MaxYears Do
              Read (F, BudgetPropData^.Base[B].MiscRecurCost.Year[Y]);
            Readln(F);
          End;

      If Not EOF (F)
        Then For B := 0 To MaxBases Do
          Begin
            Read (F, RowName);
            For Y := 1 To MaxYears Do
              Read (F,
                Champus.Base[B].OnBaseInPatVisits.Year[Y],
                Champus.Base[B].OnBaseOutPatVisits.Year[Y]);
            Readln (F);
          End;

      Close (F)
    End
End;
```

```
{*************************************************************************
 *   Name           : InputSF
 *
 *   Purpose        : Opens and reads Standard Factors data from file specified.
 *
 *   Note           : Originally part of Input
 *
 *   Invoked As     : InputSF (Fname)
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     fname                  String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 3.30+
 *     Production : MS-DOS 3.30+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   17JAN90   GJG   Creation
 *   27Mar92   AML   Separated from input
 *************************************************************************}

Function InputSF (Fname : PathStr) : Boolean;

  Var
    Buffer        : IOBuffer;
    F             : Text;
    Filler        : Real;
    P, Y          : Integer;
    RowName       : String[10];

  Begin
    InputSF := FileExists (FName);
    If FileExists (FName) Then
      Begin
        Assign (F, FName);
        SetTextBuf (F, Buffer);
        Reset (F);

        { Read Standard Personnel Factors data-SFS1 }

        Readln (F, RowName,
                   StdFctrsPers.PercOffMarried,
                   StdFctrsPers.PercEnlMarried,
                   StdFctrsPers.PercEnlHous,
                   StdFctrsPers.OffSal,
                   StdFctrsPers.OffBAQDepends,
                   StdFctrsPers.EnlSal,
                   StdFctrsPers.EnlBAQDepends,
                   StdFctrsPers.CivSal,
```

```
            StdFctrsPers.CivTurnover,
            StdFctrsPers.CivEarlyRetire,
            StdFctrsPers.CivRifPayFctr,
            StdFctrsPers.CivRetirePayFctr,
            StdFctrsPers.PercCivsQuitting,
            StdFctrsPers.NewHireCost,
            StdFctrsPers.NatlAvgHomePrice,
            StdFctrsPers.HomeSaleReimburseRate,
            StdFctrsPers.HomeSaleReimburseMax,
            StdFctrsPers.HomePurchReimburseRate,
            StdFctrsPers.HomePurchReimburseMax,
            StdFctrsPers.CivHomeOwnRate,
            StdFctrsPers.HAPFactor,
            StdFctrsPers.AvgUnemployCostPerWeek,
            StdFctrsPers.PriPlacementPerc,
            StdFctrsPers.CivPCSCosts,
            Filler, Filler,
            StdFctrsPers.PPSInvolvePCS);

{ Read Standard Facility Factors data-SFS2 }

              Readln (F, RowName, Filler,
            StdFctrsFac.RPMACostBldgPower, Filler,
            StdFctrsFac.BOSIndex, Filler,
            StdFctrsFac.SuppToMoveFctr,
            StdFctrsFac.CaretakerAdminSpace, Filler,
            StdFctrsFac.CaretakerPercOrigRPMACost,
            StdFctrsFac.MothballCostPerSqrFt);
Read (F, RowName, StdFctrsFac.DiscountNPV);
For Y := 1 To 6 Do
  Read (F, StdFctrsFac.InflationRate.Year[Y]);
Readln (F, StdFctrsFac.InflationNPV,
            Filler);

{ Read Standard Transportation Factors data-SFS3 }

Readln (F, RowName,
            StdFctrsTrans.MaterialPerPerson,
            StdFctrsTrans.MilLightVehicCostPerMile,
            StdFctrsTrans.MilVehicCostPerMile,
            StdFctrsTrans.ShippingLossRate,
            StdFctrsTrans.HHGCost,
            Filler, Filler, Filler, Filler,
            StdFctrsTrans.PackNCrate,
            StdFctrsTrans.HHGWtPerOffFam,
            StdFctrsTrans.HHGWtPerEnlFam,
            StdFctrsTrans.HHGWtPerMilSing,
            StdFctrsTrans.HHGWtPerCiv,
            StdFctrsTrans.POVReimbursePerMile,
            StdFctrsTrans.AirTransPerPassMile,
            StdFctrsTrans.MiscExpPerDirectEmp,
            StdFctrsTrans.RoutinePCSCostsPerPerson,
```

```
            StdFctrsTrans.AverageTourLength,
            StdFctrsTrans.OneTimeOffPCSCosts,
            StdFctrsTrans.OneTimeEnlPCSCosts);

{ Read Standard Construction Type Factors data-SFS4 }

Read (F, RowName);
For P := 1 To 24 Do
  Read (F, StdFctrsCon.ConCost[P]);
Readln (F);
Read (F, RowName);
For P := 1 To 5 Do
  Begin
    Read (F, StdFctrsCon.Name[P]);
    StdFctrsCon.Name[P] := StripSpace (StdFctrsCon.Name[P]);
  End;
Readln (F);
Read (F, RowName);
For P := 0 To 5 Do
  Begin
    Read (F, StdFctrsCon.Units[P]);
    StdFctrsCon.Units[P] := StripSpace (StdFctrsCon.Units[P]);
  End;
Readln (F);
Readln (F, RowName,
           StdFctrsFac.AvgBachQtrs,
           StdFctrsFac.AvgFamQtrs);
Readln (F, RowName,
           StdFctrsFac.RehabCostVsNewConstr,
           StdFctrsFac.IMAPerc,
           StdFctrsFac.Design,
           StdFctrsFac.SIOH,
           StdFctrsFac.Contingency,
           StdFctrsFac.SitePrep);

{ Stuff added after v3.73 Prototype delivery }

If Not EOF (F)
  Then Readln (F, RowName, StdFctrsPers.UnemployEligible);

If Not EOF (F) Then
  Begin
    Readln (F, RowName, StdFctrsPers.FileDescription);
    StdFctrsPers.FileDescription :=
      StripSpace (StdFctrsPers.FileDescription);
  End;

If Not EOF (F)
  Then Readln (F, RowName, StdFctrsPers.DARSEFactor);
```

```
      If Not EOF (F) Then
        Begin
          Read (F, RowName);
          For P := 25 To 35 Do
            Read (F, StdFctrsCon.ConCost[P]);
          Readln (F);
          Read (F, RowName);
          For P := 6 To 16 Do
            Begin
              Read (F, StdFctrsCon.Name[P]);
              StdFctrsCon.Name[P] := StripSpace (StdFctrsCon.Name[P]);
            End;
          Readln (F);
          Read (F, RowName);
          For P := 6 To 16 Do
            Begin
              Read (F, StdFctrsCon.Units[P]);
              StdFctrsCon.Units[P] := StripSpace (StdFctrsCon.Units[P]);
            End;
          Readln (F);
        End;

      If Not EOF (F)
        Then Readln (F, RowName, StdFctrsCon.Units[-1],
          StdFctrsCon.Units[-2]);

      If Not EOF (F)
        Then Readln (F, RowName, StdFctrsPers.HAPReceive,
          StdFctrsPers.DARSEReceive)
        Else
          Begin
            StdFctrsPers.HAPFactor :=
                StdFctrsPers.HAPFactor / StdFctrsPers.HAPReceive;
            StdFctrsPers.DARSEFactor :=
                StdFctrsPers.DARSEFactor / StdFctrsPers.DARSEReceive;
          End;

      Close (F)
    End
End;
```

```
{*************************************************************************
 *   Name           : Output
 *
 *   Purpose        : Opens and writes COBRA application data from file
 *                    specified.
 *
 *   Notes          : Originally wrote all COBRA data,
 *                    now writes screens but NOT standard factors
 *
 *   Invoked As     : OutputData (Fname)
 *
 *   Return value   : -none-
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     fname                String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 3.30+
 *     Production : MS-DOS 3.30+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   17JAN90   GJG   Creation
 *   26Mar92   AML   Separated Standard Factors
 *************************************************************************}

Function OutputData (Fname : PathStr) : Boolean;

  Var
    F      : Text;
    B, B2,
    P, Y   : Integer;
    Buffer : IOBuffer;

  Begin
    OutputData := TRUE;
    Assign (F, Fname);
    SetTextBuf (F, Buffer);
    {$I-}
    Rewrite (F);
    {$I+}

    IF (NOT IOResult = 103) AND (NOT IOResult = 0) THEN
      Begin
        OutputData := FALSE;
      End
    ELSE
      Begin
```

```
{ General Environment - Screen 1 }

RightJust (F, 'GenData', 10);
Writeln (F, Screen1Data^.ModelDoShutDn, ' ',
  MaxBases, ' 6 ', Screen1Data^.FY1);

RightJust (F, 'Scenario', 10);
RightJust (F, Screen1Data^.OptionPkgName, 20);
RightJust (F, Screen1Data^.Service, 20);
RightJust (F, Screen1Data^.Group, 20);
Writeln (F);

For B := 0 TO MaxBases DO
  Begin
    RightJust (F, 'BaseInfo', 10);
    RightJust (F, Screen1Data^.Base[B].Name, 20);
    Writeln (F, Screen1Data^.Base[B].Deactivate,
                Screen1Data^.Base[B].CloseYear:2);
  End;

RightJust (F, 'SFFile',10);
RightJust (F, Screen1Data^.SFFile, 62);
Writeln (F);

Writeln (F, 'Summary Lines');
For P := 1 To 6 Do
  Begin
    RightJust (F, Screen1Data^.Summary[P], 62);
    Writeln (F);
  End;

RightJust (F, 'DateLine',10);
RightJust (F, Screen1Data^.DateLine, 20);
Writeln (F);

{ Distances Table - Screen 2 }

For B := 0 To MaxBases Do
  Begin
    Write (F, 'DistTable2');
    For B2 := 0 To MaxBases Do
      Write (F, Distance^.Base[B].Base[B2], ' ');
    Writeln (F);
  End;

{ Write Transfer Table - Screen 3 }

For B := 0 To MaxBases Do
  For B2 := 0 To MaxBases Do
    If Distance^.Base[B].Base[B2] > 0 Then
```

```
        Begin
          Write (F, 'TransOff-2');
          For Y := 1 To MaxYears Do
            Write (F, PosTrans.Base[B].Base[B2]^.Off.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransEnl-2');
          For Y := 1 To MaxYears Do
            Write (F, PosTrans.Base[B].Base[B2]^.Enl.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransCiv-2');
          For Y := 1 To MaxYears Do
            Write (F, PosTrans.Base[B].Base[B2]^.Civ.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransStu-2');
          For Y := 1 To MaxYears Do
            Write (F, PosTrans.Base[B].Base[B2]^.Stu.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransMis-2');
          For Y := 1 To MaxYears Do
           Write (F,
            ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransSup-2');
          For Y := 1 To MaxYears Do
           Write (F,
            ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransLit-2');
          For Y := 1 To MaxYears Do
            Write (F,
              ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y], ' ');
          Writeln (F);
          Write (F, 'TransHev-2');
          For Y := 1 To MaxYears Do
           Write (F,
            ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y], ' ');
          Writeln (F);
        End;

{ Write Static Base Info - Screen 4 }

For B := 0 To MaxBases Do
  Begin
    Writeln (F, 'PersInfo  ',
      ActionEnvir^.Base[B].BaseTotOffEmploy, ' ',
      ActionEnvir^.Base[B].BaseTotEnlEmploy, ' ',
      ActionEnvir^.Base[B].BaseTotStuEmploy, ' ',
      ActionEnvir^.Base[B].PercFamLivOnBase, ' ',
      ActionEnvir^.Base[B].BaseTotCivEmploy, ' ',
      ActionEnvir^.Base[B].VolRIFPerc, ' 0 ',
      ActionEnvir^.Base[B].OffHousUnitsVacant, ' ',
      ActionEnvir^.Base[B].EnlHousUnitsVacant, ' ',
```

```
        ActionEnvir^.Base[B].BaseFacilSqFt, ' ',
        ActionEnvir^.Base[B].AcreOnBase, ' 0');
      Writeln (F, 'CostInfo2 ',
        IndustFundSwitch.Base[B],
        PersCost.Base[B].OffVHA, ' ',
        PersCost.Base[B].EnlVHA, ' ',
        PersCost.Base[B].PerDiemRate, ' ',
        FreightCost.Base[B], ' ',
        Con.Base[B].AreaCostFactor);
      Writeln (F, 'BdgtInfo  ',
        BudgetPropData^.Base[B].RPMANonPayBudget, ' ',
        BudgetPropData^.Base[B].RPMAPayBudget, ' ',
        BudgetPropData^.Base[B].CommunBudget, ' ',
        BudgetPropData^.Base[B].BaseOpnsNonPayBudget, ' ',
        BudgetPropData^.Base[B].BaseOpnsPay, ' ',
        BudgetPropData^.Base[B].FamHousBudget, ' 0');
      Writeln (F, 'Champus   0 0 ',
        Champus.Base[B].ChampInPatCostVisit, ' ',
        Champus.Base[B].ChampOutPatCostVisit);
    End;

  { Write Dynamic Base Info - Screen 5 }

  For B := 0 To MaxBases Do
    Begin
      Write (F, 'ConAvoid  ');
      For Y := 1 To MaxYears Do
        Write (F, CostAvoid.Base[B].Constr.Year[Y], ' ');
      Writeln (F);
      Write (F, 'ProAvoid  ');
      For Y := 1 To MaxYears Do
        Write (F, CostAvoid.Base[B].Procure.Year[Y], ' ');
      Writeln (F, '0 ', ActionEnvir^.Base[B].FamHousShutDn);
      Write (F, '1TimUniq  ');
      For Y := 1 To MaxYears Do
        Write (F, ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[Y], ' ');
      Writeln (F);
      Write (F, '1TimMovg  ');
      For Y := 1 To MaxYears Do
        Write (F, ActionEnvir^.Base[B].OneTimeMovingCosts.Year[Y], ' ');
      Writeln (F);
      Write (F, 'EnvirMit  ');
      For Y := 1 To MaxYears Do
        Write (F, ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y], ' ');
      Writeln (F);
      Write (F, 'ActivMis2 ');
      For Y := 1 To MaxYears Do
        Write (F, ActivMisCost^.Base[B].Year[Y], ' ');
      Writeln (F);
      Write (F, 'PropAcre  ');
      For Y := 1 To MaxYears Do
        Write (F, RealPropProc.RealPropPur.Base[B].Acre.Year[Y], ' ');
```

```
      Writeln (F);
      Write (F, 'PropCost   ');
      For Y := 1 To MaxYears Do
        Write (F, RealPropProc.RealPropPur.Base[B].Cost.Year[Y], ' ');
      Writeln (F);
      Write (F, 'ShtDnSqFt ', RealPropProc.Base[B].FacilSqFtShutDown);
      For Y := 1 To MaxYears - 1 Do
        Write (F, ' 0');
      Writeln (F);
      Write (F, 'ShtDnCPV   ');
      For Y := 1 To MaxYears Do
        Write (F, ' 0');
      Writeln (F);
      Write (F, 'ConSched   ');
      For Y := 1 To MaxYears Do
        Write (F, TimePhasShutDn.Base[B].Constr.Year[Y], ' ');
      Writeln (F);
      Write (F, 'ShutSched ');
      For Y := 1 To MaxYears Do
        Write (F, TimePhasShutDn.Base[B].ShutDn.Year[Y], ' ');
      Writeln (F);
    End;

  { Write Base Personnel Info - Screen 6 }

  For B := 0 To MaxBases Do
    Begin
      Write (F, 'OffReduc   ');
      For Y := 1 To MaxYears Do
        Write (F, PosReduc.Base[B].Off.Year[Y], ' ');
      Writeln (F);
      Write (F, 'EnlReduc   ');
      For Y := 1 To MaxYears Do
        Write (F, PosReduc.Base[B].Enl.Year[Y], ' ');
      Writeln (F);
      Write (F, 'CivReduc   ');
      For Y := 1 To MaxYears Do
        Write (F, PosReduc.Base[B].Civ.Year[Y], ' ');
      Writeln (F);
      Write (F, 'OffElim    ');
      For Y := 1 To MaxYears Do
        Write (F, PosElim.Base[B].Off.Year[Y], ' ');
      Writeln (F);
      Write (F, 'EnlElim    ');
      For Y := 1 To MaxYears Do
        Write (F, PosElim.Base[B].Enl.Year[Y], ' ');
      Writeln (F);
      Write (F, 'CivElim    ');
      For Y := 1 To MaxYears Do
        Write (F, PosElim.Base[B].Civ.Year[Y], ' ');
      Writeln (F);
      Write (F, 'CaretkMil ');
```

```
      For Y := 1 To MaxYears Do
        Write (F, PosTrans.Base[B].CaretakeMil.Year[Y], ' ');
      Writeln (F);
      Write (F, 'CaretkCiv ');
      For Y := 1 To MaxYears Do
        Write (F, PosTrans.Base[B].CaretakeCiv.Year[Y], ' ');
      Writeln (F);
    End;

  { Write Construction Data - Screen 7 }

  For B := 0 TO MaxBases DO
    Begin
      Write (F, 'ConCatCF  ');
      For P := 1 To 16 Do
        Write (F, Con.Base[B].Category[P]);
      Writeln (F);
      Write (F, 'ProjNames ');
      For P := 1 To 16 Do
        If Con.Base[B].ProjectName[P] = Nil
          Then Write (F, Blank:20)
          Else RightJust (F, Con.Base[B].ProjectName[P]^, 20);
      Writeln (F);
      RightJust (F, 'ConNew', 10);
      For P := 1 To 16 Do
        Write (F, Con.Base[B].New[P], ' ');
      Writeln (F);
      RightJust (F, 'ConRehab', 10);
      For P := 1 To 16 Do
        Write (F, Con.Base[B].Rehab[P], ' ');
      Writeln (F);
      RightJust (F, 'ConCost', 10);
      For P := 1 To 16 Do
        Write (F, Con.Base[B].Cost[P], ' ');
      Writeln (F);
      Writeln (F, 'ProjComm  ');
      For P := 1 To 16 Do
        If Con.Base[B].Comment[P] = Nil
          Then Writeln (F, Blank:60)
          Else
            Begin
              RightJust (F, Con.Base[B].Comment[P]^, 60);
              Writeln (F);
            End;
    End;

  { Write Industrial Funded Facilities data - Screen 8 }

  For B := 0 To MaxBases Do
    If IndustFundSwitch.Base[B] = 'Y' Then
      Begin
```

```
      For Y := 1 To MaxYears DO
        Writeln (F, 'IndustFund ',
          IndustFund.Base[B]^.OvhdCost.AdminPlan.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdCost.Operating.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdCost.Mothball.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdCost.Caretaker.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdCost.MaintSpace.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdCost.Other.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdSave.Operating.Year[Y], ' ',
          IndustFund.Base[B]^.OvhdSave.Other.Year[Y], ' ',
          IndustFund.Base[B]^.MoveCost.PackUnpack.Year[Y], ' ',
          IndustFund.Base[B]^.MoveCost.Freight.Year[Y], ' ',
          IndustFund.Base[B]^.MoveCost.Vehic.Year[Y], ' ',
          IndustFund.Base[B]^.MoveCost.Driv.Year[Y], ' ',
          IndustFund.Base[B]^.MoveCost.LossRate.Year[Y]);
    End;

{ Write Footnotes / Comments - Screen 9 }

Writeln (F, 'Footnotes ', Screen9Data.Pages);
For P := 1 To 15 Do
  Writeln (F, Screen9Data.Line[P]);

{ Stuff added after v3.73 Prototype delivery }

Write (F, 'NewStuff  ');
For B := 0 To MaxBases Do
  RightJust (F, Screen1Data^.Base[B].State, 2);
For B := 0 To MaxBases Do
  Write (F, HAPSwitch.Base[B]);
Writeln (F);

For B := 0 To MaxBases Do
  Begin
    Write (F, 'FamQtrAvd ');
    For Y := 1 To MaxYears Do
      Write (F, CostAvoid.Base[B].FamQtr.Year[Y], ' ');
    Writeln (F);
  End;

Write(F, 'CHAMPshft ');
For B := 0 To MaxBases Do
  Write (F, ActionEnvir^.Base[B].ChampusToMedicare, ' ');
Writeln(F);

For B := 0 To MaxBases Do
  Begin
    Write (F, 'MiscRecur ');
    For Y := 1 To MaxYears Do
      Write (F, BudgetPropData^.Base[B].MiscRecurCost.Year[Y], ' ');
    Writeln (F);
  End;
```

```
      For B := 0 To MaxBases Do
        Begin
          Write (F, 'CHAMPVis  ');
          For Y := 1 To MaxYears Do
            Write (F,
              Champus.Base[B].OnBaseInPatVisits.Year[Y], ' ',
              Champus.Base[B].OnBaseOutPatVisits.Year[Y], ' ');
          Writeln (F);
        End;

      Close (F);
    End;
  End;

{*****************************************************************************
 *   Name           : OutputSF
 *
 *   Purpose        : Opens and writes Standard Factors data to file specified.
 *
 *   Notes          : Originally part of Output
 *
 *   Invoked As     : OutputSF (Fname)
 *
 *   Return value   : -none-
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     fname                  String    Standard Factors data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 3.30+
 *     Production : MS-DOS 3.30+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   17JAN90   GJG   Creation
 *   26Mar92   AML   Separated from COBRA applications data
 *****************************************************************************}

Function OutputSF (Fname : PathStr) : Boolean;

  Var
    Buffer : IOBuffer;
    F      : Text;
    P, Y   : Integer;

  Begin
    OutputSF := TRUE;
    Assign (F, Fname);
    SetTextBuf (F, Buffer);
    {$I-}
```

```
Rewrite (F);
{$I+}

IF (NOT IOResult = 103) AND (NOT IOResult = 0) THEN
   Begin
      OutputSF := FALSE;
   End
ELSE
   Begin

      { Write Standard Personnel Factors data-SFS1 }

      Writeln (F, 'StPersonel ',
                StdFctrsPers.PercOffMarried, ' ',
                StdFctrsPers.PercEnlMarried, ' ',
                StdFctrsPers.PercEnlHous, ' ',
                StdFctrsPers.OffSal, ' ',
                StdFctrsPers.OffBAQDepends, ' ',
                StdFctrsPers.EnlSal, ' ',
                StdFctrsPers.EnlBAQDepends, ' ',
                StdFctrsPers.CivSal, ' ',
                StdFctrsPers.CivTurnover, ' ',
                StdFctrsPers.CivEarlyRetire, ' ',
                StdFctrsPers.CivRifPayFctr, ' ',
                StdFctrsPers.CivRetirePayFctr, ' ',
                StdFctrsPers.PercCivsQuitting, ' ',
                StdFctrsPers.NewHireCost, ' ',
                StdFctrsPers.NatlAvgHomePrice, ' ',
                StdFctrsPers.HomeSaleReimburseRate, ' ',
                StdFctrsPers.HomeSaleReimburseMax, ' ',
                StdFctrsPers.HomePurchReimburseRate, ' ',
                StdFctrsPers.HomePurchReimburseMax, ' ',
                StdFctrsPers.CivHomeOwnRate, ' ',
                StdFctrsPers.HAPFactor, ' ',
                StdFctrsPers.AvgUnemployCostPerWeek, ' ',
                StdFctrsPers.PriPlacementPerc, ' ',
                StdFctrsPers.CivPCSCosts, ' 0 0 ',
                StdFctrsPers.PPSInvolvePCS);

      { Write Standard Facility Factors data-SFS2 }

                     Writeln (F, 'StFacFact  0 ',
                StdFctrsFac.RPMACostBldgPower, ' 0 ',
                StdFctrsFac.BOSIndex, ' 0 ',
                StdFctrsFac.SuppToMoveFctr, ' ',
                StdFctrsFac.CaretakerAdminSpace, ' 0 ',
                StdFctrsFac.CaretakerPercOrigRPMACost, ' ',
                StdFctrsFac.MothballCostPerSqrFt);
      Write (F, 'Disc/Infla ', StdFctrsFac.DiscountNPV, ' ');
      For Y := 1 To 6 Do
        Write (F, StdFctrsFac.InflationRate.Year[Y], ' ');
      Writeln (F, StdFctrsFac.InflationNPV, ' 0');
```

```
{ Write Standard Transportation Factors data-SFS3 }

Writeln (F, 'StTrans    ',
          StdFctrsTrans.MaterialPerPerson, ' ',
          StdFctrsTrans.MilLightVehicCostPerMile, ' ',
          StdFctrsTrans.MilVehicCostPerMile, ' ',
          StdFctrsTrans.ShippingLossRate, ' ',
          StdFctrsTrans.HHGCost, ' 0 0 0 0 ',
          StdFctrsTrans.PackNCrate, ' ',
          StdFctrsTrans.HHGWtPerOffFam, ' ',
          StdFctrsTrans.HHGWtPerEnlFam, ' ',
          StdFctrsTrans.HHGWtPerMilSing, ' ',
          StdFctrsTrans.HHGWtPerCiv, ' ',
          StdFctrsTrans.POVReimbursePerMile, ' ',
          StdFctrsTrans.AirTransPerPassMile, ' ',
          StdFctrsTrans.MiscExpPerDirectEmp, ' ',
          StdFctrsTrans.RoutinePCSCostsPerPerson, ' ',
          StdFctrsTrans.AverageTourLength, ' ',
          StdFctrsTrans.OneTimeOffPCSCosts, ' ',
          StdFctrsTrans.OneTimeEnlPCSCosts);

{ Write Standard Construction Type Factors data-SFS4 }

Write (F, 'StConst   ');
For P := 1 To 24 Do
  Write (F, StdFctrsCon.ConCost[P], ' ');
Writeln (F);
Write (F, 'OptNames  ');
For P := 1 To 5 Do
  RightJust (F, StdFctrsCon.Name[P], 20);
Writeln (F);
Write (F, 'OptUnits  ');
For P := 0 To 5 Do
  RightJust (F, StdFctrsCon.Units[P], 2);
Writelln(F);
Writeln (F, 'QtrSize   ',
  StdFctrsFac.AvgBachQtrs, ' ',
  StdFctrsFac.AvgFamQtrs);
Writeln (F, 'ConPercs  ',
  StdFctrsFac.RehabCostVsNewConstr, ' ',
  StdFctrsFac.IMAPerc, ' ',
  StdFctrsFac.Design, ' ',
  StdFctrsFac.SIOH, ' ',
  StdFctrsFac.Contingency, ' ',
  StdFctrsFac.SitePrep);

{ Stuff added after v3.73 Prototype delivery }

Writeln (F, 'UnempElig ', StdFctrsPers.UnemployEligible);

Write (F, 'FileDesc  ');
RightJust (F, StdFctrsPers.FileDescription, 20);
```

```
        Writeln (F);

        Writeln (F, 'DARSEFctr ', StdFctrsPers.DARSEFactor);

        { Stuff added after v4.00 "final" delivery }

        Write (F, 'StConst+  ');
        For P := 25 To 35 Do
          Write (F, StdFctrsCon.ConCost[P], ' ');
        Writeln (F);
        Write (F, 'OptNames+ ');
        For P := 6 To 16 Do
          RightJust (F, StdFctrsCon.Name[P], 20);
        Writeln (F);
        Write (F, 'OptUnits+ ');
        For P := 6 To 16 Do
          RightJust (F, StdFctrsCon.Units[P], 2);
        Writeln (F);

        Write (F, 'Qtr SF/EA:');
        RightJust (F, StdFctrsCon.Units[-1], 2);
        RightJust (F, StdFctrsCon.Units[-2], 2);
        Writeln (F);

        Writeln (F, 'HAP/DARSE+',
          StdFctrsPers.HAPReceive,
          StdFctrsPers.DARSEReceive);

        Close (F);
      End;
 End;
{*****************************************************************************
 *   Name          : InputOld
 *
 *   Purpose       : Opens and reads old (pre-v4.00) COBRA data files
 *
 *   Invoked As    : InputOld(Fname)
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     fname                String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 3.30+
 *     Production : MS-DOS 3.30+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   17JAN90   GJG   Creation
 *   28Apr92   AML   Data names and structures updated
 *****************************************************************************}
```

```
Function InputOld (Fname : PathStr) : Boolean;

  Var
    A, C, D        : Array [0..AbsMaxBases,1..18] of Real;
    Buffer         : IOBuffer;
    DateNTime      : DateTime;
    F              : Text;
    B, B2,
    P, Y, Fil,
    MaxYearsTemp   : Integer;
    Filler         : Real;
    FilStr         : String20;
    RowName        : String10;
    Time           : LongInt;

  Begin
    FillChar (A, SizeOf(A), 0);
    FillChar (C, SizeOf(C), 0);
    FillChar (D, SizeOf(D), 0);
    InputOld := FileExists (FName);
    If FileExists (FName) Then
      Begin
        Assign(F, Fname);
        SetTextBuf (F, Buffer);
        Reset(F);
        GetFTime (F, Time);
        UnPackTime (Time, DateNTime);
        Screen1Data^.DateLine := TimeStr (DateNTime.Year, DateNTime.Month,
          DateNTime.Day, DateNTime.Hour, DateNTime.Min);

        Readln(F, MaxBases, MaxYearsTemp, Y, Fil,
          Screen1Data^.Base[0].CloseYear, Screen1Data^.FY1);
        LastYear := Screen1Data^.Base[0].CloseYear;
        If Y = 3
          Then Screen1Data^.Base[0].Deactivate := 'Y';
        If Y = 1
          Then Screen1Data^.Base[0].CloseYear := 0;

        Readln(F, RowName,
          Screen1Data^.Base[0].Name,
          Screen1Data^.Service,
          Screen1Data^.Group);
        Screen1Data^.Base[0].Name := StripSpace(Screen1Data^.Base[0].Name);
        Screen1Data^.Service := StripSpace(Screen1Data^.Service);
        Screen1Data^.Group := StripSpace(Screen1Data^.Group);

        For B := 1 To MaxBases Do
          Begin
            Readln(F, RowName,
              Screen1Data^.Base[B].Name,
              FilStr,
              Distance^.Base[0].Base[B]);
```

```
      If Distance^.Base[0].Base[B] = 0
        Then Distance^.Base[0].Base[B] := 0.1;
      Screen1Data^.Base[B].Name := StripSpace(Screen1Data^.Base[B].Name);
      Distance^.Base[B].Base[0] := Distance^.Base[0].Base[B];
    End;

  Readln(F, RowName, FilStr,
    Screen1Data^.OptionPkgName);
  Screen1Data^.OptionPkgName := StripSpace(Screen1Data^.OptionPkgName);

  Readln(F, RowName,
    StdFctrsFac.DiscountNPV,
    StdFctrsFac.InflationNPV);
  For Y := 1 To MaxYearsTemp Do
    StdFctrsFac.InflationRate.Year[Y]
      := StdFctrsFac.InflationNPV;

  Readln(F, RowName,
    ActionEnvir^.Base[0].BaseTotOffEmploy,
    ActionEnvir^.Base[0].BaseTotCivEmploy,
    ActionEnvir^.Base[0].BaseFacilSqFt,
    ActionEnvir^.Base[0].AcreOnBase,
    Filler, Filler, Filler, Filler,
    Filler, Filler, Filler,
    ActionEnvir^.Base[0].PercFamLivOnBase,
    ActionEnvir^.Base[0].EnvirMitigReqd.Year[1],
    ActionEnvir^.Base[0].OneTimeUniqueCosts.Year[1]);

  For B := 1 To MaxBases Do
    Begin
      New (ActionEnvir^.Base[0].Base[B]);
      FillChar (ActionEnvir^.Base[0].Base[B]^,
        SizeOf (ActionEnvir^.Base[0].Base[B]^), 0);
      New (ActionEnvir^.Base[B].Base[0]);
      FillChar (ActionEnvir^.Base[B].Base[0]^,
        SizeOf (ActionEnvir^.Base[B].Base[0]^), 0);
      Readln(F, RowName,
        ActionEnvir^.Base[B].BaseTotOffEmploy,
        ActionEnvir^.Base[B].BaseTotCivEmploy,
        ActionEnvir^.Base[B].BaseFacilSqFt,
        ActionEnvir^.Base[B].AcreOnBase,
        ActionEnvir^.Base[B].OffHousUnitsVacant,
        ActionEnvir^.Base[B].EnlHousUnitsVacant,
        ActionEnvir^.Base[0].Base[B]^.MoveMissionEqpmt.Year[1],
        ActionEnvir^.Base[0].Base[B]^.MoveSupportEqpmt.Year[1], Filler,
        ActionEnvir^.Base[0].Base[B]^.MilLightVehic.Year[1],
        ActionEnvir^.Base[0].Base[B]^.HeavyOrSpecVehic.Year[1],
        Filler,
        ActionEnvir^.Base[B].EnvirMitigReqd.Year[1],
        ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[1]);
    End;
```

```
For B := 0 To MaxBases Do
  Begin
    Filler := Round
      (ActionEnvir^.Base[B].EnvirMitigReqd.Year[1]/3000.0)*1000.0;
    For Y := 1 To 3 Do
      ActionEnvir^.Base[B].EnvirMitigReqd.Year[Y] := Filler;
    Filler := Round
      (ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[1]/3000.0)*1000.0;
    For Y := 1 To 3 Do
      ActionEnvir^.Base[B].OneTimeUniqueCosts.Year[Y] := Filler;
    ActionEnvir^.Base[B].BaseTotEnlEmploy :=
      Round(ActionEnvir^.Base[B].BaseTotOffEmploy * 0.5);
    ActionEnvir^.Base[B].BaseTotOffEmploy :=
      ActionEnvir^.Base[B].BaseTotEnlEmploy;
  End;

For Y := 0 To MaxYearsTemp Do
  Readln(F, RowName,
    PosElim.Base[0].Off.Year[Y],
    PosElim.Base[0].Enl.Year[Y],
    PosElim.Base[0].Civ.Year[Y],
    PosTrans.Base[0].CaretakeMil.Year[Y],
    PosTrans.Base[0].CaretakeCiv.Year[Y]);
For Y := MaxYearsTemp DownTo 1 Do
  Begin
    PosElim.Base[0].Off.Year[Y] := Round(AtLeastZero(
        PosElim.Base[0].Off.Year[Y - 1]
      - PosElim.Base[0].Off.Year[Y]));
    PosElim.Base[0].Enl.Year[Y] := Round(AtLeastZero(
        PosElim.Base[0].Enl.Year[Y - 1]
      - PosElim.Base[0].Enl.Year[Y]));
    PosElim.Base[0].Civ.Year[Y] := Round(AtLeastZero(
        PosElim.Base[0].Civ.Year[Y - 1]
      - PosElim.Base[0].Civ.Year[Y]));
    PosTrans.Base[0].CaretakeMil.Year[Y] :=
        PosTrans.Base[0].CaretakeMil.Year[Y]
      - PosTrans.Base[0].CaretakeMil.Year[Y - 1];
    PosTrans.Base[0].CaretakeCiv.Year[Y] :=
        PosTrans.Base[0].CaretakeCiv.Year[Y]
      - PosTrans.Base[0].CaretakeCiv.Year[Y - 1]
  End;
PosElim.Base[0].Off.Year[0] := 0;
PosElim.Base[0].Enl.Year[0] := 0;
PosElim.Base[0].Civ.Year[0] := 0;
PosTrans.Base[0].CaretakeMil.Year[0] := 0;
PosTrans.Base[0].CaretakeCiv.Year[0] := 0;

For B := 1 To MaxBases Do
  Begin
    New (PosTrans.Base[0].Base[B]);
    FillChar (PosTrans.Base[0].Base[B]^,
      SizeOf (PosTrans.Base[0].Base[B]^), 0);
```

```
      New (PosTrans.Base[B].Base[0]);
      FillChar (PosTrans.Base[B].Base[0]^,
        SizeOf (PosTrans.Base[B].Base[0]^), 0);
      For Y := 0 To MaxYearsTemp Do
        Readln(F, RowName,
          PosTrans.Base[0].Base[B]^.Off.Year[Y],
          PosTrans.Base[0].Base[B]^.Enl.Year[Y],
          PosTrans.Base[0].Base[B]^.Civ.Year[Y]);
      For Y := MaxYearsTemp DownTo 1 Do
        Begin
          PosTrans.Base[0].Base[B]^.Off.Year[Y] := Round(AtLeastZero(
              PosTrans.Base[0].Base[B]^.Off.Year[Y]
            - PosTrans.Base[0].Base[B]^.Off.Year[Y - 1]));
          PosElim.Base[0].Off.Year[Y] := Round(AtLeastZero(
              PosElim.Base[0].Off.Year[Y]
            - PosTrans.Base[0].Base[B]^.Off.Year[Y]));
          PosTrans.Base[0].Base[B]^.Enl.Year[Y] := Round(AtLeastZero(
              PosTrans.Base[0].Base[B]^.Enl.Year[Y]
            - PosTrans.Base[0].Base[B]^.Enl.Year[Y - 1]));
          PosElim.Base[0].Enl.Year[Y] := Round(AtLeastZero(
              PosElim.Base[0].Enl.Year[Y]
            - PosTrans.Base[0].Base[B]^.Enl.Year[Y]));
          PosTrans.Base[0].Base[B]^.Civ.Year[Y] := Round(AtLeastZero(
              PosTrans.Base[0].Base[B]^.Civ.Year[Y]
            - PosTrans.Base[0].Base[B]^.Civ.Year[Y - 1]));
          PosElim.Base[0].Civ.Year[Y] := Round(AtLeastZero(
              PosElim.Base[0].Civ.Year[Y]
            - PosTrans.Base[0].Base[B]^.Civ.Year[Y]));
        End;
      PosTrans.Base[0].Base[B]^.Off.Year[0] := 0;
      PosTrans.Base[0].Base[B]^.Enl.Year[0] := 0;
      PosTrans.Base[0].Base[B]^.Civ.Year[0] := 0;
    End;

  For B := 0 To MaxBases Do
    Begin
      Readln(F, RowName,
        PersCost.Base[B].OffVHA,
        PersCost.Base[B].EnlVHA,
        PersCost.Base[B].PerDiemRate);
      PersCost.Base[B].OffVHA := Round(PersCost.Base[B].OffVHA / 12.0);
      PersCost.Base[B].EnlVHA := Round(PersCost.Base[B].EnlVHA / 12.0);
    End;

  For Y := 1 To MaxYearsTemp Do
    Readln(F, RowName,
      CostAvoid.Base[1].Constr.Year[Y],
      CostAvoid.Base[1].Procure.Year[Y]);

  Readln(F, RowName);
```

```
For B := 0 TO MaxBases DO
  Begin
    Readln(F, RowName,
      BudgetPropData^.Base[B].RPMANonPayBudget,
      BudgetPropData^.Base[B].RPMAPayBudget,
      BudgetPropData^.Base[B].CommunBudget,
      BudgetPropData^.Base[B].BaseOpnsNonPayBudget,
      BudgetPropData^.Base[B].BaseOpnsPay,
      BudgetPropData^.Base[B].FamHousBudget);
    BudgetPropData^.Base[B].RPMANonPayBudget :=
      Round(BudgetPropData^.Base[B].RPMANonPayBudget/1000.0)*1000.0;
    BudgetPropData^.Base[B].RPMAPayBudget :=
      Round(BudgetPropData^.Base[B].RPMAPayBudget/1000.0)*1000.0;
    BudgetPropData^.Base[B].CommunBudget :=
      Round(BudgetPropData^.Base[B].CommunBudget/1000.0)*1000.0;
    BudgetPropData^.Base[B].BaseOpnsNonPayBudget :=
     Round(BudgetPropData^.Base[B].BaseOpnsNonPayBudget/1000.0)*1000.0;
    BudgetPropData^.Base[B].BaseOpnsPay :=
      Round(BudgetPropData^.Base[B].BaseOpnsPay/1000.0)*1000.0;
    BudgetPropData^.Base[B].FamHousBudget :=
      Round(BudgetPropData^.Base[B].FamHousBudget/1000.0)*1000.0;
  End;

For B := 0 TO MaxBases DO
  Readln(F, RowName,
    ActivMisCost^.Base[B].Year[MaxYearsTemp]);
ActivMisCost^.Base[0].Year[MaxYearsTemp] :=
  -ActivMisCost^.Base[0].Year[MaxYearsTemp];

Readln(F, RowName,
  RealPropProc.Base[0].FacilSqFtShutDown,
  RealPropProc.RealPropPur.Base[0].Acre.Year[LastYear],
  RealPropProc.RealPropPur.Base[0].Cost.Year[LastYear], Fil);
RealPropProc.RealPropPur.Base[0].Acre.Year[LastYear] :=
  - RealPropProc.RealPropPur.Base[0].Acre.Year[LastYear];
RealPropProc.RealPropPur.Base[0].Cost.Year[LastYear] := - Round(
  RealPropProc.RealPropPur.Base[0].Cost.Year[LastYear]/1000.0)*1000.0;

For B := 1 TO MaxBases DO
  Begin
    Readln(F, RowName,
      RealPropProc.RealPropPur.Base[B].Acre.Year[1],
      RealPropProc.RealPropPur.Base[B].Cost.Year[1]);
    RealPropProc.RealPropPur.Base[B].Cost.Year[1] := Round(
      RealPropProc.RealPropPur.Base[B].Cost.Year[1]/1000.0)*1000.0;
  End;

For B := 0 TO MaxBases DO
  Begin
    Read(F, RowName, Con.Base[B].AreaCostFactor);
    For P := 1 To 12 Do
      Read(F, A[B, P]);
```

```
    Readln(F, Filler, Filler);
  End;

For B := 0 TO MaxBases DO
  Begin
    Read(F, RowName, Filler);
    For P := 1 To 12 Do
      Read(F, D[B, P]);
    Readln(F, Filler, Filler);
  End;

For B := 0 TO MaxBases DO
  Begin
    Read(F, RowName, Filler);
    For P := 1 TO 12 Do
      Read(F, C[B, P]);
    Readln (F, Filler, Filler);
  End;

For B := 0 TO MaxBases DO
  Readln(F, RowName,
    FreightCost.Base[B]);

Readln(F, RowName, Fil);
If Fil = 1
  Then Screen1Data^.ModelDoShutDn := 'Y'
  Else Screen1Data^.ModelDoShutDn := 'N';

For B := 0 TO MaxBases DO
  Begin
    Readln(F, RowName,
      Champus.Base[B].OnBaseInPatVisits.Year[LastYear],
      Champus.Base[B].OnBaseOutPatVisits.Year[LastYear],
      Champus.Base[B].ChampInPatCostVisit,
      Champus.Base[B].ChampOutPatCostVisit);
    If B > 0 Then
      Begin
        Champus.Base[B].OnBaseInPatVisits.Year[LastYear] :=
          - Champus.Base[B].OnBaseInPatVisits.Year[LastYear];
        Champus.Base[B].OnBaseOutPatVisits.Year[LastYear] :=
          - Champus.Base[B].OnBaseOutPatVisits.Year[LastYear];
      End;
    For Y := LastYear + 1 To MaxYearsTemp Do
      Begin
        Champus.Base[B].OnBaseInPatVisits.Year[Y] :=
          Champus.Base[B].OnBaseInPatVisits.Year[LastYear];
        Champus.Base[B].OnBaseOutPatVisits.Year[Y] :=
          Champus.Base[B].OnBaseOutPatVisits.Year[LastYear];
      End;
  End;
```

```
For Y := 1 To MaxYearsTemp DO
  Begin
    Readln(F, RowName,
      TimePhasShutDn.Base[0].Constr.Year[Y],
      TimePhasShutDn.Base[0].Shutdn.Year[Y]);
    For B := 1 To MaxBases Do
      Begin
        TimePhasShutDn.Base[B].Constr.Year[Y]
          := TimePhasShutDn.Base[0].Constr.Year[Y];
        TimePhasShutDn.Base[B].Shutdn.Year[Y]
          := TimePhasShutDn.Base[0].Shutdn.Year[Y];
      End;
  End;
For B := 0 To MaxBases Do
  Begin
    If Screen1Data^.ModelDoShutDn = 'N'
      Then For Y := 1 To MaxYearsTemp Do
        ActivMisCost^.Base[B].Year[Y] :=
            ActivMisCost^.Base[B].Year[MaxYearsTemp]
          * TimePhasShutDn.Base[B].Constr.Year[Y];
    For Y := 2 To MaxYearsTemp Do
      ActivMisCost^.Base[B].Year[Y] :=
          ActivMisCost^.Base[B].Year[Y]
        + ActivMisCost^.Base[B].Year[Y-1];
  End;

For B := 1 To MaxBases Do
  If Screen1Data^.ModelDoShutDn = 'N'
    Then For Fil := MaxYearsTemp DownTo 1 Do
      Begin
        ActionEnvir^.Base[0].Base[B]^.MoveMissionEqpmt.Year[Fil] :=
            ActionEnvir^.Base[0].Base[B]^.MoveMissionEqpmt.Year[1]
          * TimePhasShutDn.Base[B].Constr.Year[Fil];
        ActionEnvir^.Base[0].Base[B]^.MoveSupportEqpmt.Year[Fil] :=
            ActionEnvir^.Base[0].Base[B]^.MoveSupportEqpmt.Year[1]
          * TimePhasShutDn.Base[B].Constr.Year[Fil];
        ActionEnvir^.Base[0].Base[B]^.MilLightVehic.Year[Fil] :=
            ActionEnvir^.Base[0].Base[B]^.MilLightVehic.Year[1]
          * TimePhasShutDn.Base[B].Constr.Year[Fil];
        ActionEnvir^.Base[0].Base[B]^.HeavyOrSpecVehic.Year[Fil] :=
            ActionEnvir^.Base[0].Base[B]^.HeavyOrSpecVehic.Year[1]
          * TimePhasShutDn.Base[B].Constr.Year[Fil];
      End;

Readln(F, RowName, Fil, Filler, Filler,
  StdFctrsPers.PercOffMarried,
  StdFctrsPers.PercEnlMarried,
  StdFctrsPers.PercEnlHous,
  StdFctrsPers.OffSal,
  StdFctrsPers.OffBAQDepends,
  StdFctrsPers.EnlSal,
  StdFctrsPers.EnlBAQDepends,
```

```
    StdFctrsPers.CivSal,
    StdFctrsPers.CivTurnover,
    StdFctrsPers.CivEarlyRetire,
    StdFctrsPers.CivRifPayFctr,
    Filler,
    StdFctrsPers.CivRetirePayFctr,
    StdFctrsPers.PercCivsQuitting,
    StdFctrsPers.NewHireCost,
    StdFctrsPers.NatlAvgHomePrice,
    StdFctrsPers.HomeSaleReimburseRate,
    StdFctrsPers.HomeSaleReimburseMax,
    StdFctrsPers.HomePurchReimburseRate,
    StdFctrsPers.HomePurchReimburseMax);

  StdFctrsPers.PPSInvolvePCS := 0.5;

  StdFctrsPers.NatlAvgHomePrice :=
    Round (StdFctrsPers.NatlAvgHomePrice / 10.0) * 10.0;

              Readln(F, RowName, Filler,
    StdFctrsFac.RPMACostBldgPower, Filler,
    StdFctrsFac.BOSIndex, Filler,
    StdFctrsFac.SuppToMoveFctr,
    StdFctrsFac.CaretakerAdminSpace, Filler,
    StdFctrsFac.CaretakerPercOrigRPMACost,
    StdFctrsFac.MothballCostPerSqrFt);

  Readln(F, RowName,
    StdFctrsTrans.MaterialPerPerson,
    StdFctrsTrans.MilLightVehicCostPerMile,
    StdFctrsTrans.MilVehicCostPerMile,
    StdFctrsTrans.ShippingLossRate,
    StdFctrsTrans.HHGCost,
    Filler, Filler, Filler, Filler,
    StdFctrsTrans.HHGWtPerOffFam,
    StdFctrsTrans.HHGWtPerEnlFam,
    StdFctrsTrans.HHGWtPerMilSing,
    StdFctrsTrans.HHGWtPerCiv,
    StdFctrsTrans.POVReimbursePerMile,
    StdFctrsTrans.AirTransPerPassMile,
    StdFctrsTrans.MiscExpPerDirectEmp,
    StdFctrsPers.CivHomeOwnRate,
    StdFctrsTrans.RoutinePCSCostsPerPerson,
    StdFctrsTrans.OneTimeOffPCSCosts,
    StdFctrsTrans.OneTimeEnlPCSCosts);

  Readln (F, RowName,
    StdFctrsCon.ConCost[1],
    StdFctrsCon.ConCost[2],
    StdFctrsCon.ConCost[3],
    StdFctrsCon.ConCost[4],
    StdFctrsFac.RehabCostVsNewConstr,
```

```
    StdFctrsCon.ConCost[5],
    StdFctrsCon.ConCost[6],
    StdFctrsCon.ConCost[7],
    StdFctrsCon.ConCost[8],
    StdFctrsFac.AvgBachQtrs,
    StdFctrsCon.ConCost[9],
    StdFctrsFac.AvgFamQtrs,
    StdFctrsCon.ConCost[10],
    StdFctrsCon.ConCost[11],
    StdFctrsCon.ConCost[12],
    StdFctrsFac.Design,
    StdFctrsFac.SIOH,
    StdFctrsFac.Contingency,
    StdFctrsFac.SitePrep);

  IF not eof (F) THEN
    Begin
      New (IndustFund.Base[0]);
      For Y := 1 to MaxYearsTemp DO
        Readln (F, RowName,
          IndustFund.Base[0]^.OvhdCost.AdminPlan.Year[Y],
          IndustFund.Base[0]^.OvhdCost.Operating.Year[Y],
          IndustFund.Base[0]^.OvhdCost.Mothball.Year[Y],
          IndustFund.Base[0]^.OvhdCost.Caretaker.Year[Y],
          IndustFund.Base[0]^.OvhdCost.MaintSpace.Year[Y],
          IndustFund.Base[0]^.OvhdCost.Other.Year[Y],
          IndustFund.Base[0]^.OvhdSave.Operating.Year[Y],
          IndustFund.Base[0]^.OvhdSave.Other.Year[Y],
          IndustFund.Base[0]^.MoveCost.PackUnpack.Year[Y],
          IndustFund.Base[0]^.MoveCost.Freight.Year[Y],
          IndustFund.Base[0]^.MoveCost.Vehic.Year[Y],
          IndustFund.Base[0]^.MoveCost.Driv.Year[Y],
          IndustFund.Base[0]^.MoveCost.LossRate.Year[Y]);
    End;

  IF not eof (F) THEN
    Readln (F, RowName, IndustFundSwitch.Base[0]);
  IF IndustFundSwitch.Base[0] = 'y'
    Then IndustFundSwitch.Base[0] := 'Y'
  Else If IndustFundSwitch.Base[0] <> 'Y'
    Then IndustFundSwitch.Base[0] := 'N';
  If (IndustFundSwitch.Base[0] = 'N') And
     (IndustFund.Base[0] <> Nil) Then
    Begin
      Dispose (IndustFund.Base[0]);
      IndustFund.Base[0] := Nil;
    End;

  If Not EOF (F)
    Then For B := 0 TO MaxBases DO
      Begin
        Read(F, RowName);
```

```
        For P := 13 To 18 Do
          Read(F, A[B, P]);
        Readln(F);
      End;

IF not eof (F) THEN
  Begin
    For P := 1 To 5 Do
      Begin
        Readln (F, RowName,
          StdFctrsCon.Name[P],
          StdFctrsCon.ConCost[P+19]);
        StdFctrsCon.Name[P] := StripSpace(StdFctrsCon.Name[P]);
        If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'AMM' Then
          Begin
            StdFctrsCon.ConCost[17] :=
              StdFctrsCon.ConCost[P+19];
            StdFctrsCon.ConCost[P+19] := 0;
          End
        Else If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'COM' Then
          Begin
            StdFctrsCon.ConCost[13] :=
              StdFctrsCon.ConCost[P+19];
            StdFctrsCon.ConCost[P+19] := 0;
          End
        Else If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'ENV' Then
          Begin
            StdFctrsCon.ConCost[19] :=
              StdFctrsCon.ConCost[P+19];
            StdFctrsCon.ConCost[P+19] := 0;
          End
        Else If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'MED' Then
          Begin
            StdFctrsCon.ConCost[18] :=
              StdFctrsCon.ConCost[P+19];
            StdFctrsCon.ConCost[P+19] := 0;
          End
        Else If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'POL' Then
          Begin
            StdFctrsCon.ConCost[16] :=
              StdFctrsCon.ConCost[P+19];
            StdFctrsCon.ConCost[P+19] := 0;
          End
        Else If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'RDT' Then
          Begin
            StdFctrsCon.ConCost[15] :=
              StdFctrsCon.ConCost[P+19];
            StdFctrsCon.ConCost[P+19] := 0;
          End
        Else If UpperCase(Copy(StdFctrsCon.Name[P],1,3)) = 'SHI' Then
```

```
            Begin
              StdFctrsCon.ConCost[14] :=
                StdFctrsCon.ConCost[P+19];
              StdFctrsCon.ConCost[P+19] := 0;
            End;
        End;
       Readln (F, RowName, FilStr, Filler);
    End;

IF not eof (F) THEN
  Readln (F, RowName, Filler);

IF not eof (F) THEN
  Readln (F, RowName, StdFctrsPers.AvgUnemployCostPerWeek);

IF not eof (F) THEN
  Readln (F, RowName, Filler);

IF not eof (F) THEN
  Readln (F, RowName, ActionEnvir^.Base[0].VolRIFPerc);
For B := 1 To MaxBases Do
  Begin
    ActionEnvir^.Base[B].PercFamLivOnBase :=
      ActionEnvir^.Base[0].PercFamLivOnBase;
    ActionEnvir^.Base[B].VolRIFPerc :=
      ActionEnvir^.Base[0].VolRIFPerc;
  End;

IF not eof (F) THEN
  Readln (F, RowName, StdFctrsPers.PriPlacementPerc);

IF not eof (F) THEN
  Readln (F, RowName, StdFctrsPers.CivPCSCosts);

IF not eof (F) THEN
  Readln (F, RowName, Filler);

IF not eof (F) THEN
  Readln (F, RowName, StdFctrsFac.IMAPerc);

If Not EOF (F)
  Then For B := 0 TO MaxBases DO
    Begin
      Read(F, RowName);
      For P := 13 To 18 Do
        Read(F, D[B, P]);
      Readln(F);
    End;

If Not EOF (F)
  Then For B := 0 TO MaxBases DO
```

```
      Begin
        Read(F, RowName);
        For P := 13 To 18 Do
          Read(F, C[B, P]);
        Readln(F);
      End;

  For B := 0 To MaxBases Do
    Begin
      Fil := 1;
      For P := 1 To 18 Do
        If Fil < 17 Then
          Begin
            Con.Base[B].Rehab[Fil] :=
              Min (AtLeastZero (A[B,P]-D[B,P]), C[B,P]);
            Con.Base[B].New[Fil] :=
              AtLeastZero (A[B,P] - D[B,P]) - Con.Base[B].Rehab[Fil];
            If Con.Base[B].Rehab[Fil] + Con.Base[B].New[Fil] > 0 Then
              Begin
                If P < 13
                  Then Con.Base[B].Category[Fil] := Chr (64 + P)
                Else If P < 18 Then
                  Begin
                    If UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'AMM'
                      Then Con.Base[B].Category[Fil] := 'Q'
                    Else If UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'COM'
                      Then Con.Base[B].Category[Fil] := 'M'
                    Else If UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'ENV'
                      Then Con.Base[B].Category[Fil] := 'S'
                    Else If UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'MED'
                      Then Con.Base[B].Category[Fil] := 'R'
                    Else If UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'POL'
                      Then Con.Base[B].Category[Fil] := 'P'
                    Else If (UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'R&D') Or
                            (UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'RDT')
                      Then Con.Base[B].Category[Fil] := 'O'
                    Else If (UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'SHI') Or
                            (UpperCase(Copy(StdFctrsCon.Name[P-12],1,3))
                        = 'SHP')
                      Then Con.Base[B].Category[Fil] := 'N'
                    Else Con.Base[B].Category[Fil] := Chr (70 + P);
                  End
```

```
                    Else
                      Begin
                        If UpperCase(Copy(FilStr,1,3)) = 'AMM' Then
                          Begin
                            Con.Base[B].Category[Fil] := 'Q';
                            StdFctrsCon.ConCost[17] := Filler;
                          End
                        Else If UpperCase(Copy(FilStr,1,3)) = 'COM' Then
                          Begin
                            Con.Base[B].Category[Fil] := 'M';
                            StdFctrsCon.ConCost[13] := Filler;
                          End
                        Else If UpperCase(Copy(FilStr,1,3)) = 'ENV' Then
                          Begin
                            Con.Base[B].Category[Fil] := 'S';
                            StdFctrsCon.ConCost[19] := Filler;
                          End
                        Else If UpperCase(Copy(FilStr,1,3)) = 'MED' Then
                          Begin
                            Con.Base[B].Category[Fil] := 'R';
                            StdFctrsCon.ConCost[18] := Filler;
                          End
                        Else If UpperCase(Copy(FilStr,1,3)) = 'POL' Then
                          Begin
                            Con.Base[B].Category[Fil] := 'P';
                            StdFctrsCon.ConCost[16] := Filler;
                          End
                        Else If (UpperCase(Copy(FilStr,1,3)) = 'R&D') Or
                                (UpperCase(Copy(FilStr,1,3)) = 'RDT') Then
                          Begin
                            Con.Base[B].Category[Fil] := 'O';
                            StdFctrsCon.ConCost[15] := Filler;
                          End
                        Else If (UpperCase(Copy(FilStr,1,3)) = 'SHI') Or
                                (UpperCase(Copy(FilStr,1,3)) = 'SHP') Then
                          Begin
                            Con.Base[B].Category[Fil] := 'N';
                            StdFctrsCon.ConCost[14] := Filler;
                          End
                        Else Con.Base[B].Category[Fil] := '@';
                      End;
                    Inc (Fil);
                  End;
              End;
        End;
      For P := 1 To 5 Do
        If StdFctrsCon.ConCost[P+19] = 0
          Then StdFctrsCon.Name[P] := '';

      Close (F);
    End;
End;
```

```
Procedure ReadBaseFromDB (Var Fil : Text; WhichBase : Integer);

  Var
    Filler : Real;

  Begin
    Read (Fil, Screen1Data^.Base[WhichBase].Name);
    Screen1Data^.Base[WhichBase].Name :=
      StripSpace (Screen1Data^.Base[WhichBase].Name);
    Read (Fil, Screen1Data^.Base[WhichBase].State);
    Screen1Data^.Base[WhichBase].State :=
      StripSpace (Screen1Data^.Base[WhichBase].State);
    Readln (Fil, IndustFundSwitch.Base[WhichBase], HAPSwitch.Base[WhichBase],
      ActionEnvir^.Base[WhichBase].BaseTotOffEmploy,
      ActionEnvir^.Base[WhichBase].BaseTotEnlEmploy,
      ActionEnvir^.Base[WhichBase].BaseTotStuEmploy,
      ActionEnvir^.Base[WhichBase].PercFamLivOnBase,
      ActionEnvir^.Base[WhichBase].BaseTotCivEmploy,
      ActionEnvir^.Base[WhichBase].VolRIFPerc, Filler,
      ActionEnvir^.Base[WhichBase].OffHousUnitsVacant,
      ActionEnvir^.Base[WhichBase].EnlHousUnitsVacant,
      ActionEnvir^.Base[WhichBase].BaseFacilSqFt,
      ActionEnvir^.Base[WhichBase].AcreOnBase,
      PersCost.Base[WhichBase].OffVHA,
      PersCost.Base[WhichBase].EnlVHA,
      PersCost.Base[WhichBase].PerDiemRate,
      FreightCost.Base[WhichBase],
      Con.Base[WhichBase].AreaCostFactor,
      BudgetPropData^.Base[WhichBase].RPMANonPayBudget,
      BudgetPropData^.Base[WhichBase].RPMAPayBudget,
      BudgetPropData^.Base[WhichBase].CommunBudget,
      BudgetPropData^.Base[WhichBase].BaseOpnsNonPayBudget,
      BudgetPropData^.Base[WhichBase].BaseOpnsPay,
      BudgetPropData^.Base[WhichBase].FamHousBudget,
      Champus.Base[WhichBase].ChampInPatCostVisit,
      Champus.Base[WhichBase].ChampOutPatCostVisit,
      ActionEnvir^.Base[WhichBase].ChampusToMedicare);

      BudgetPropData^.Base[WhichBase].RPMANonPayBudget :=
        BudgetPropData^.Base[WhichBase].RPMANonPayBudget * 1000.0;
      BudgetPropData^.Base[WhichBase].RPMAPayBudget :=
        BudgetPropData^.Base[WhichBase].RPMAPayBudget * 1000.0;
      BudgetPropData^.Base[WhichBase].CommunBudget :=
        BudgetPropData^.Base[WhichBase].CommunBudget * 1000.0;
      BudgetPropData^.Base[WhichBase].BaseOpnsNonPayBudget :=
        BudgetPropData^.Base[WhichBase].BaseOpnsNonPayBudget * 1000.0;
      BudgetPropData^.Base[WhichBase].BaseOpnsPay :=
        BudgetPropData^.Base[WhichBase].BaseOpnsPay * 1000.0;
      BudgetPropData^.Base[WhichBase].FamHousBudget :=
        BudgetPropData^.Base[WhichBase].FamHousBudget * 1000.0;
    If IndustFundSwitch.Base[WhichBase] = 'Y' Then
```

```
      Begin
        If IndustFund.Base[WhichBase] = Nil
          Then New (IndustFund.Base[WhichBase]);
        FillChar (IndustFund.Base[WhichBase]^,
          SizeOf (IndustFund.Base[WhichBase]^), 0);
      End;
  End;

Procedure WriteBaseToDB (Var Fil : Text; WhichBase : Integer);

  Begin
    RightJust (Fil, Screen1Data^.Base[WhichBase].Name, 20);
    RightJust (Fil, Screen1Data^.Base[WhichBase].State, 2);
    Writeln (Fil, IndustFundSwitch.Base[WhichBase], HAPSwitch.Base[WhichBase],
      ActionEnvir^.Base[WhichBase].BaseTotOffEmploy:6:0,
      ActionEnvir^.Base[WhichBase].BaseTotEnlEmploy:6:0,
      ActionEnvir^.Base[WhichBase].BaseTotStuEmploy:6:0,
      ActionEnvir^.Base[WhichBase].PercFamLivOnBase:6:3,
      ActionEnvir^.Base[WhichBase].BaseTotCivEmploy:6:0,
      ActionEnvir^.Base[WhichBase].VolRIFPerc:6:3, ' 0 ',
      ActionEnvir^.Base[WhichBase].OffHousUnitsVacant:6:0,
      ActionEnvir^.Base[WhichBase].EnlHousUnitsVacant:6:0,
      ActionEnvir^.Base[WhichBase].BaseFacilSqFt:10:0,
      ActionEnvir^.Base[WhichBase].AcreOnBase:10:0,
      PersCost.Base[WhichBase].OffVHA:6:0,
      PersCost.Base[WhichBase].EnlVHA:6:0,
      PersCost.Base[WhichBase].PerDiemRate:4:0,
      FreightCost.Base[WhichBase]:7:2,
      Con.Base[WhichBase].AreaCostFactor:5:2,
      (BudgetPropData^.Base[WhichBase].RPMANonPayBudget/1000):10:0,
      (BudgetPropData^.Base[WhichBase].RPMAPayBudget/1000):10:0,
      (BudgetPropData^.Base[WhichBase].CommunBudget/1000):10:0,
      (BudgetPropData^.Base[WhichBase].BaseOpnsNonPayBudget/1000):10:0,
      (BudgetPropData^.Base[WhichBase].BaseOpnsPay/1000):10:0,
      (BudgetPropData^.Base[WhichBase].FamHousBudget/1000):10:0,
      Champus.Base[WhichBase].ChampInPatCostVisit:10:0,
      Champus.Base[WhichBase].ChampOutPatCostVisit:10:0,
      ActionEnvir^.Base[WhichBase].ChampusToMedicare:6:3);
  End;
```

```
{*************************************************************************
 *   Name          : DBNames
 *
 *   Purpose       : Returns a linked list of names in the DataBase
 *
 *   Invoked As    : DBNames(Fname)
 *
 *   Files/Tables
 *     Names                 Access    Purpose
 *     ------------------    -------   ----------------------------------------
 *     fname                 String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 5.0+
 *     Production : MS-DOS 5.0+
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   22Jun92   AML   Creation
 *************************************************************************}

Procedure DBNames(FName : PathStr; Var Names : PDBaseRec; Var Number : Integer);

  Var
    Buffer : IOBuffer;
    F      : Text;
    P      : PDBaseRec;
    S      : String [245];

  Begin
    Names^.BaseName := 'Null Entry';
    Names^.Next := Nil;
    Number := 0;
    P := Names;
    If FileExists (FName) Then
      Begin
        Assign (F, FName);
        SetTextBuf (F, Buffer);
        Reset (F);
        While Not EOF (F) Do
          Begin
            Readln (F, S);
            New (P^.Next);
            P := P^.Next;
            P^.BaseName := StripSpace (Copy (S, 1, 20));
            P^.Next := Nil;
            Inc (Number);
          End;
        Close (F);
      End;
  End;
```

```
{*****************************************************************************
 *   Name           : DBCheck
 *
 *   Purpose        : Searches a DataBase For a particular base
 *
 *   Notes          : Returns True is base is present.
 *
 *   Invoked As     : DBCheck(WhichBase, Fname)
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     fname                String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 5.0+
 *     Production : MS-DOS 5.0+
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   22Jun92  AML   Creation
 *****************************************************************************}

Function DBCheck (WhichBase : Integer; FName : PathStr) : Boolean;

  Var
    Buffer  : IOBuffer;
    Fil     : Text;
    Match   : Boolean;
    TempStr : String [245];

  Begin
    Match := False;
    If FileExists (FName) Then
      Begin
        Assign (Fil, FName);
        SetTextBuf (Fil, Buffer);
        Reset (Fil);
        While Not EOF (Fil) Do
          Begin
            Readln (Fil, TempStr);
            Match := Match Or
                (UpperCase (Screen1Data^.Base[WhichBase].Name)
              = UpperCase (StripSpace (Copy (TempStr, 1, 20))))
          End;
        Close (Fil);
      End;
    DBCheck := Match;
  End;
```

```
{*************************************************************************
 *   Name          : DBLoad
 *
 *   Purpose       : Loads a Base from the DataBase
 *
 *   Invoked As    : DBLoad(WhichBase, Fname)
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     fname                String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 5.0+
 *     Production : MS-DOS 5.0+
 *
 *   Date     Init  Modifications
 *   -------  ----  ---------------------------------------------------------
 *   22Jun92  AML   Creation
 *************************************************************************}

Procedure DBLoad (BName : String20; WhichBase : Integer; FName : PathStr);

  Var
    B      : Integer;
    Buffer : IOBuffer;
    F      : Text;

  Begin
    If FileExists (FName) Then
      Begin
        Assign (F, FName);
        SetTextBuf (F, Buffer);
        Reset (F);
        Repeat
          ReadBaseFromDB (F, WhichBase);
        Until (UpperCase(BName) = UpperCase(Screen1Data^.Base[WhichBase].Name))
           Or EOF (F);
        Close (F);
        ChangeFlag := True;
      End
  End;
```

```
{*****************************************************************************
 *   Name          : DBSave
 *
 *   Purpose       : Saves a Base in the DataBase (alphabetically sorted)
 *
 *   Invoked As    : DBSave(WhichBase, Fname)
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     fname                String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 5.0+
 *     Production : MS-DOS 5.0+
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   22Jun92  AML   Creation
 *****************************************************************************}

Procedure DBSave (WhichBase : Integer; FName : PathStr);

  Var
    FIn, FOut : Text;
    InBuffer,
    OutBuffer : IOBuffer;
    TempStr   : String [245];
    Written   : Boolean;

  Begin
    If FileExists (FName) Then
      Begin
        Written := False;
        Assign (FIn, FName);
        SetTextBuf (FIn, InBuffer);
        Reset (FIn);
        Assign (FOut, 'COBDBASE.TMP');
        SetTextBuf (FOut, OutBuffer);
        ReWrite (FOut);
        While Not EOF (Fin) Do
          Begin
            Readln (FIn, TempStr);
            If (UpperCase (Screen1Data^.Base[WhichBase].Name) >
                UpperCase (StripSpace (Copy (TempStr, 1, 20))))
              Then Writeln (FOut, TempStr)
            Else If (UpperCase (Screen1Data^.Base[WhichBase].Name)
                   = UpperCase (StripSpace (Copy (TempStr, 1, 20)))) Then
              Begin
                WriteBaseToDB (FOut, WhichBase);
                Written := True;
              End
```

```
            Else If Not Written Then
              Begin
                WriteBaseToDB (FOut, WhichBase);
                Writeln (FOut, TempStr);
                Written := True;
              End
            Else Writeln (FOut, TempStr);
          End;
        If Not Written
          Then WriteBaseToDB (FOut, WhichBase);
        Close (FIn);
        Erase (FIn);
        Close (FOut);
        Rename (FOut, FName);
      End
    Else
      Begin
        Assign (FOut, FName);
        ReWrite (FOut);
        WriteBaseToDB (FOut, WhichBase);
        Close (FOut);
      End;
  End;

{*****************************************************************************
 *   Name          : DBDistSave
 *
 *   Purpose       : Saves Base distance in the DataBase
 *
 *   Notes         : If a base pair is already in the database, the old entry
 *                   is removed.  If the memory value is zero, no new value
 *                   is put in (since zero is the default).
 *
 *   Invoked As    : DBDistSave(Fname, Base1, Base2, DistDB, DistMem)
 *
 *   Files/Tables
 *     Names                 Access    Purpose
 *     ------------------    --------  ----------------------------------------
 *     fname                 String    COBRA data file name
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 5.0+
 *     Production : MS-DOS 5.0+
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   06Jul92   AML   Creation
 *****************************************************************************}

Procedure DBDistSave (FName        : PathStr;
                      Base1, Base2 : String20;
                      DistDB, DistMem : Real);
```

```
Var
  FIn, Fout : Text;
  InBuffer,
  OutBuffer : IOBuffer;
  TempName  : String20;
  TempStr   : String60;

Begin
  Base1 := StripSpace(UpperCase(Base1));
  Base2 := StripSpace(UpperCase(Base2));
  If Base2 < Base1 Then
    Begin
      TempName := Base1;
      Base1 := Base2;
      Base2 := TempName;
    End;
  If DistDB = 0 Then
    Begin
      Assign (FOut, FName);
      SetTextBuf (FOut, InBuffer);
      If FileExists (FName)
        Then Append (FOut)
        Else ReWrite (FOut);
      RightJust (FOut, Base1, 20);
      RightJust (FOut, Base2, 20);
      Writeln (FOut, DistMem:7:1);
      Close (FOut);
    End
  Else
    Begin
      Assign (FIn, FName);
      SetTextBuf (FIn, InBuffer);
      Reset (FIn);
      Assign (FOut, 'COBDBASE.TMP');
      SetTextBuf (FOut, OutBuffer);
      ReWrite (FOut);
      While Not EOF (Fin) Do
        Begin
          Readln (FIn, TempStr);
          If (StripSpace(Copy(TempStr,1,20)) <> Base1) Or
             (StripSpace(Copy(TempStr,21,20)) <> Base2)
            Then Writeln (FOut, TempStr);
        End;
      If DistMem <> 0 Then
        Begin
          RightJust (FOut, Base1, 20);
          RightJust (FOut, Base2, 20);
          Writeln (FOut, DistMem:7:1);
        End;
      Close (FIn);
      Erase (FIn);
      Close (FOut);
```

```
        Rename (FOut, FName);
      End
    End;

  End.
```

```
{*************************************************************************
 *  Name          : COBRA
 *
 *  Purpose       : User Interface/main module for COBRA
 *
 *  Notes         :
 *
 *  Invoked As    : "COBRA [/[L¦E¦X]=Filename][/D=directory]" from DOS prompt
 *
 *  Return value  : -none-
 *
 *  Constraints
 *    Development: Turbo Pascal 6.0 w/TurboVision, MS-DOS 3.30+
 *    Production : MS-DOS 3.30+
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------------
 *  11Jan91   ASB   Creation
 *  31Jan91   GJG   Addition of save/load routines, file viewer
 *  10Apr92   AML   Finished removing ALL With..Do commands
 *                  (all variables now listed by full name
 *                  for easier searching and comprehension)
 *  08Jun92   AML   Added command-line parameter handling
 *************************************************************************}

{$F+,O-,S-,X+,D+,L+}
{$UNDEF  DEBUG}    { <-- Define DEBUG for Heap counter                      }
{$UNDEF  OVER1}    { <-- Define OVER1 to overlay Report Execution modules   }
{$DEFINE OVER2}    { <-- Define OVER2 to overlay the other COBRA modules    }
{$DEFINE OVER3}    { <-- Define OVER3 to overlay various systems modules    }
{$UNDEF  PROTO}    { <-- Define PROTO for Prototype label '*P' on screen    }

PROGRAM COBRA;

{$M 49152,16384,655360}       { Memory sizes   :  Stack, HeapMin, HeapMax   }
                              { All along were :  32768,   16384,  655360   }

  USES
    APP,          { The TApplication type                                   }
    CALC,         { TurboVision on-screen Calculator                        }
    CALENDAR,     { TurboVision on-screen Calendar                          }
    DIALOGS,      { Definitions used in building dialog boxes               }
    DOS,          { DOS-specific stuff used in DosShell                     }
    DRIVERS,      { Event manager, screen/keyboard/mouse drivers, etc.      }
    FINPUT,       { The Field input routines from CIS                       }
    GADGETS,      { System Clock & Avail. Heap space counters               }
    HELPFILE,     { Context-sensitive help                                  }
    HISTLIST,     { Definitions for history lists                           }
    MEMORY,       { TVision's memory management routines                    }
    MENUS,        { Definitions for pull-down & pop-up menus                }
    MSGBOX,       { MessageBox Function                                     }
    OBJECTS,      { Basic OOP definitions for TVision, including TObject    }
```

```
    OVERLAY,     { Overlay manager                                          }
    STDDLG,      { File open & change directory dialog boxes                }
    VIEWS,       { Basic definitions of TView & derivatives                 }

    CINIT,       { Initializes all Global Variables                         }
    CIO,         { File I/O routines for saving/loading entered data        }
    CDEC,        { TYPE definitions/global vars used for calculations       }
    CCAL,        { Realignment Summary Report                               }
    CSUP,        { Functions and Procedures used throughout Cobra           }

    CSCR1,       { Data Input Screens                                       }
    CSCR2,
    CSCR3,
    CSCR4,
    CSCR5,
    CSCR6,
    CSCR7,
    CSCR8,
    CSCR9,

    CFAC1,       { Standard factors screens                                 }
    CFAC2,
    CFAC3,
    CFAC4;

{$IFDEF OVER2}
  {$O CIO}
  {$O CDEC}
  {$O CINIT}
  {$O CSCR1}
  {$O CSCR2}
  {$O CSCR3}
  {$O CSCR4}
  {$O CSCR5}
  {$O CSCR6}
  {$O CSCR7}
  {$O CSCR8}
  {$O CSCR9}
  {$O CFAC1}
  {$O CFAC2}
  {$O CFAC3}
  {$O CFAC4}
{$ENDIF}
{$IFDEF OVER3}
  {$O CALC}
  {$O CALENDAR}
  {$O DOS}
  {$O FINPUT}
  {$O HELPFILE}
  {$O HISTLIST}
  {$O MENUS}
  {$O MSGBOX}
```

```
  {$O STDDLG}
  {$O VIEWS}
{$ENDIF}
{$IFDEF OVER1}
  {$O CSUP}
  {$O CCAL}
  {$O CCALCON}
  {$O CCALPER}
  {$O CCALMOV}
  {$O CCALOTH}
  {$O CCALOVR}
  {$O CRPTIN}
  {$O CRPTOUT1}
  {$O CRPTOUT2}
{$ENDIF}

TYPE
  PCobraApp = ^TCobraApp;
  TCobraApp = OBJECT (TApplication)
                Clock : PClockView;
{$IFDEF DEBUG}
                Heap  : PHeapView;
{$ENDIF}

                CONSTRUCTOR Init;
                     { Creates/Initializes system                    }

                Procedure AboutWindow;

                Procedure Calculator;

                Procedure Calendar;

                Procedure CascadeOrTile (Mode : Integer);

                Procedure ChangeDir;

                Procedure ChooseBase (Mode : Integer);

                Procedure DeleteData (Mode : Integer);

                Procedure DistanceDataBase;

                Procedure DosShell;

                Procedure GetEvent(VAR Event: TEvent); virtual;
                     { Checks for context-sensitive help before      }
                     {    handling an event                          }

                Function GetFileForLoad (Ext    : String04;
                                         WTitle : String20;
                                         CLabel : String60) : PathStr;
```

```
Function GetFileForSave (VAR FName : PathStr;
                         Mode : Integer) : WORD;

Function GetPalette : PPalette; virtual;

Procedure HandleEvent(VAR Event: TEvent); virtual;
     { Process any keyboard/mouse/etc. events that     }
     {    the TApplication type doesn't understand     }

Procedure Idle; virtual;
     { Used to keep the clock and heap count updated   }

Procedure InitMenuBar; virtual;
     { Set up the top-line menu                        }

Procedure InitStatusLine; virtual;
     { Set up the bottom line status "menu"            }

Procedure LoadFromDataBase;

Procedure LoadPrevious(Mode:Integer);

Procedure OpenFile (Mode : Integer);

Procedure OutOfMemory; virtual;
     { Out of Memory trap                              }

Procedure PrinterSetup;

Procedure PrintGroup (Mode : Integer);

Procedure PrintReport (Mode : Integer);

Procedure PutTab;

Procedure SaveData (Mode : Integer);

Function ScreenOneNine (Which_Screen:Integer) : WORD;

Function ScreenTwo : WORD;

Function ScreenThree : WORD;

Function ScreenFourFiveSixSeven (Which_Screen:Integer):WORD;

Function ScreenEight : WORD;

Procedure SetBackGroundChar;

Procedure StandardFactors (FactWin : Integer);

Procedure ViewFile (FileName : PathStr);
```

```
                Procedure ViewUsed;

              End;{ TCobraApp }

  { TLineCollection }

  PLineCollection = ^TLineCollection;
  TLineCollection = OBJECT (TCollection)
                      Procedure FreeItem (P : Pointer); virtual;
                    End;

  { TFileViewer }

  PFileViewer = ^TFileViewer;
  TFileViewer = OBJECT (TScroller)
                  FileLines : PCollection;
                  IsValid   : BOOLEAN;

                  CONSTRUCTOR Init (var Bounds       : TRect;
                                        AHScrollBar,
                                        AVScrollBar  : PScrollBar;
                                    var FileName     : PathStr);
                  DESTRUCTOR  Done; virtual;
                  Procedure   Draw; virtual;
                  Function    Valid (Command : Word) : Boolean; virtual;
                End;

  { TFileWindow }

  PFileWindow = ^TFileWindow;
  TFileWindow = OBJECT (TWindow)
                  CONSTRUCTOR Init (var FileName : PathStr);
                End;

  { TFileWindow }

CONSTRUCTOR TFileWindow.Init (var FileName : PathStr);

  CONST
    WinNumber : Integer = 1;

  VAR
    R : TRect;

  Begin
    Desktop^.GetExtent (R);
    TWindow.Init (R, Filename, WinNumber);
    Options := Options or ofTileable;
    Inc (WinNumber);
    GetExtent (R);
    R.Grow (-1, -1);
    Insert (New (PFileViewer,
```

```
      Init (R, StandardScrollBar (sbHorizontal + sbHandleKeyboard),
               StandardScrollBar (sbVertical   + sbHandleKeyboard), Filename)));
  End;

Procedure TLineCollection.FreeItem (P : Pointer);

  Begin
    DisposeStr(P);
  End;

{ TFileViewer }

CONSTRUCTOR TFileViewer.Init (var Bounds       : TRect;
                                  AHScrollBar,
                                  AVScrollBar  : PScrollBar;
                              var FileName     : PathStr);

  VAR
    Buffer     : IOBuffer;
    FileToView : Text;
    Line       : String;
    MaxWidth   : Integer;
    tempword   : Word;

  Begin
    TScroller.Init (Bounds, AHScrollbar, AVScrollBar);
    GrowMode := gfGrowHiX + gfGrowHiY;
    FileLines := New (PLineCollection, Init (5,5));
    IsValid := FileExists (FileName);
    If Not IsValid
      Then tempword := MessageBox
        ('Cannot open file '+Filename+'.', nil, mfError + mfOkButton)
    Else
      Begin
        {$I-}
        Assign (FileToView, FileName);
        SetTextBuf (FileToView, Buffer);
        Reset (FileToView);
        MaxWidth := 0;
        WHILE NOT eof (FileToView) AND NOT LowMemory DO
          Begin
            Readln (FileToView, Line);
            If Length (Line) > MaxWidth THEN
               MaxWidth := Length (Line);
            FileLines^.Insert (NewStr (Line));
          End;
        Close (FileToView);
        {$I+}
      End;
    SetLimit (MaxWidth, FileLines^.Count);
  End;
```

```
DESTRUCTOR TFileViewer.Done;

  Begin
    Dispose (FileLines, Done);
    TScroller.Done;
  End;

Procedure TFileViewer.Draw;

  VAR
    B : TDrawBuffer;
    C : Byte;
    I : Integer;
    S : String;
    P : PString;

  Begin
    C := GetColor (1);
    FOR I := 0 to Size.Y - 1 DO
      Begin
        MoveChar (B, ' ', C, Size.X);
        If Delta.Y + I < FileLines^.Count THEN
          Begin
            P := FileLines^.At (Delta.Y + I);
            If P <> nil THEN
              S := Copy (P^, Delta.X + 1, Size.X)
            ELSE
              S := '';
            MoveStr (B, S, C);
          End;
        WriteLine (0, I, Size.X, 1, B);
      End;
  End;

Function TFileViewer.Valid (Command : Word) : Boolean;

   Begin
      Valid := IsValid;
   End;
```

```
{*--------------------------TCobraApp Methods-------------------------------*}

{*********************************************************************
 *  Name          : TCobraApp.Init
 *
 *  Purpose       : Constructor for the TCobraApp type
 *
 *  Notes         : Sets up the desktop, and places the clock in the
 *                  upper right corner.
 *
 *  Invoked As    :
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  11Jan91   ASB   Creation
 *********************************************************************}

CONSTRUCTOR TCobraApp.Init;

{ CONST
    OvrMaxSize = 80000; }

  Var
    ErrIndex : integer;
    event    : TEvent;
    F        : Text;
    StpVer,
    SetupVer : Real;
    r        : TRect;
    TempWord : Word;
  Var
    OvrName  : string[79];

{$IFNDEF OVER1}
  {$IFNDEF OVER2}
    {$IFNDEF OVER3}
      {$DEFINE OVER0}
    {$ENDIF}
  {$ENDIF}
{$ENDIF}

  Begin
{$IFNDEF OVER0}
    OvrName := 'COBRA.OVR';
    REPEAT
      OvrInit(OvrName);

      If OvrResult = OvrNotFound THEN
        Begin
          WriteLn('Overlay file not found: ', OvrName, '.');
```

```
          Write('Enter correct overlay file name: ');
          ReadLn(OvrName);
        End;
    UNTIL OvrResult <> OvrNotFound;

    If OvrResult <> OvrOk THEN
      Begin
        WriteLn('Overlay manager error.');
        Halt(1);
      End;

    OvrInitEMS;

    If OvrResult = ovrIOError THEN
      Begin
        Write('Overlay file I/O error');
        Halt (1);
      End;

{   OvrSetBuf(OvrMaxSize); }
{$ENDIF}

    TApplication.Init;
    BackGroundChar := #176;
    SetBackGroundChar;

    {*********************************************************************
     * Insert the System Clock in the upper-right corner of the screen
     *********************************************************************}

    GetExtent (r);
    r.A.X := r.B.X - 9;
    r.B.Y := r.A.Y + 1;
    Clock := New (PClockView, Init (r));
    Insert (Clock);

{$IFDEF DEBUG}
    GetExtent(r);
    r.B.X := r.B.X - 10;
    r.A.X := r.B.X - 7;
    r.B.Y := r.A.Y + 1;
    Heap := New(PHeapView, Init(R));
    Insert(Heap);
{$ENDIF}

    ShadowSize.X := 0;
    ShadowSize.Y := 0;

    Val (SetupVersion, SetupVer, ErrIndex);
    Info.DeviceName := 'PRN';
    Info.ReportDir := '';
    Info.HighASCII := 1;
```

```
  If FileExists('Cobra.Stp') Then
    Begin
      Assign (F, 'Cobra.Stp');
      Reset (F);
      Readln (F, StpVer);
      If (StpVer >= 1.0) And (StpVer <= SetupVer) Then
        Begin
          Readln (F, Info.HighASCII, Info.DeviceName);
          Info.DeviceName := StripSpace(Info.DeviceName);
          If Info.DeviceName = ''
            Then Info.DeviceName := 'PRN';
        End;
      If (StpVer >= 1.1) And (StpVer <= SetupVer) Then
        Begin
          Readln (F, Info.ReportDir);
          Info.ReportDir := StripSpace (Info.ReportDir);
        End;
      If StpVer > SetupVer
        Then Tempword := MessageBox
          ('Wrong Setup File Version', nil, mfError + mfOkButton);
      Close (F);
    End;
  If AutoStart.ReportDir <> '' Then
    Begin
      Info.ReportDir := FExpand(StripSpace(AutoStart.ReportDir));
      If Info.ReportDir[Length(Info.ReportDir)] <> '\'
        Then Info.ReportDir := Concat (Info.ReportDir, '\');
    End;

  {*********************************************************************
   * Force the "About" at system start if filename not specified
   *********************************************************************}

  event.What    := evCommand;
  If AutoStart.FileName = ''
    Then event.Command := cmAbout
    Else event.Command := cmAutoLoad;
  PutEvent (event);

  InitPointers;
  InitGlobalVariables;
  InitStandardFactors;
  Screen1Data^.SFFile := '';
  ChangeFlag := False;

  RegisterHelpFile;

  SavedExitProc := ExitProc;
  ExitProc := @RunTimeError;
End; {TCobraApp.Init}
```

```
{*********************************************************************
 *  Name          : TCobraApp.GetPalette
 *
 *  Purpose       : Adds to the standard palette colors for the Help windows
 *
 *  Notes         : Help palettes (CHelpColor, etc.) defined in HELPFILE.TPU
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------------
 *  10Jun92   AML   Creation
 *********************************************************************}

Function TCobraApp.GetPalette: PPalette;
const
  CNewColor = CColor + CHelpColor;
  CNewBlackWhite = CBlackWhite + CHelpBlackWhite;
  CNewMonochrome = CMonochrome + CHelpMonochrome;
  P: array[apColor..apMonochrome] of string[Length(CNewColor)] =
    (CNewColor, CNewBlackWhite, CNewMonochrome);
Begin
  GetPalette := @P[AppPalette];
end;

{*********************************************************************
 *  Name          : TCobraApp.SetBackGroundChar
 *
 *  Purpose       : Sets the character to fill the desktop background ...
 *                  usually Chr(176) [ ]
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------------
 *  19Jun92   AML   Creation
 *********************************************************************}

Procedure TCobraApp.SetBackGroundChar;

  Begin
    DeskTop^.Background^.Pattern := BackGroundChar;
    DeskTop^.Background^.Draw;
  End;

{*********************************************************************
 *  Name          : TCobraApp.GetEvent
 *
 *  Purpose       : Overrides standard GetEvent routine to allow for context-
 *                  sensitive help
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------------
 *  16Jun92   AML   Creation
 *********************************************************************}
```

```
Procedure TCobraApp.GetEvent(var Event: TEvent);

  var
    HelpStrm : PDosStream;
    HFile    : PHelpFile;
    W        : PWindow;

  const
    HelpInUse: Boolean = False;

  Begin
    TApplication.GetEvent(Event);
    case Event.What of
      evCommand:
        if (Event.Command = cmHelp) and not HelpInUse then
          Begin
            HelpInUse := True;
            DisableCommands([cmHelp]);
            HelpStrm := New(PDosStream,
              Init('COBRA.CSH', stOpenRead));
            HFile := New(PHelpFile, Init(HelpStrm));
            if HelpStrm^.Status <> stOk then
              Begin
                MessageBox('Could not open help file.',
                  nil, mfError + mfOkButton);
                Dispose(HFile, Done);
              end
            else
              Begin
                W := New(PHelpWindow,Init(HFile, GetHelpCtx));
                if ValidView(W) <> nil then
                  Begin
                    ExecView(W);
                    Dispose(W, Done);
                  end;
                ClearEvent(Event);
              end;
            HelpInUse := False;
            EnableCommands([cmHelp]);
          end;
      evMouseDown:
        if Event.Buttons <> 1 then Event.What := evNothing;
    end;
  end;
```

```
{*************************************************************************
 *  Name          : TCobraApp.HandleEvent
 *
 *  Purpose       : Handle generated mouse/keyboard events
 *
 *  Notes         :
 *
 *  Invoked As    : HandleEvent(event)
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    Event             IN      Event passed in (an event that TApplication
 *                                does not understand
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  11Jan91   ASB   Creation
 *************************************************************************}

Procedure TCobraApp.HandleEvent (var event : TEvent);

  Procedure Warning;
    Begin
      If ChangeFlag
        Then Messagebox ('You may wish to re-Execute '
          + 'the scenario before viewing or printing Reports.',
          Nil, mfOKButton);
    End;

  VAR
    i             : INTEGER;
    saveflag,
    usetemp       : BOOLEAN;
    tempevent     : TEvent;
    tempstr       : String60;
    tempword      : WORD;

  Begin { TCobraApp.HandleEvent }
    TApplication.HandleEvent (event);
    CASE event.what OF
      evCommand:
        Begin
          usetemp := FALSE;
          CASE event.command OF
            cmAbout          : AboutWindow;
            cmViewHelp       : OpenFile(mdHelp);
            cmPrintHelp      : PrintReport(mdHelp);
            cmViewUsed       : ViewUsed;
            cmCalculator     : Calculator;
```

```
cmCalendar      : Calendar;
cmNewData       : Begin
                    If ChangeFlag
                      Then Tempword := MessageBox ('Are you sure '
                        + 'you want to clear data in memory?',
                        Nil, mfYesNoCancel + mfConfirmation)
                      Else Tempword := cmYes;
                    If TempWord = cmYes Then
                      Begin
                        InitGlobalVariables;
                        UpdateDataFile
                          (StatusLine^.Items^.Next^.Next^.Text^);
                      End;
                  End;
cmViewReport    : Begin
                    Warning;
                    OpenFile(mdReport);
                  End;
cmViewGroup     : Begin
                    Warning;
                    PrintGroup (mdView);
                  End;
cmPrintReport   : Begin
                    Warning;
                    PrintReport (mdReport);
                  End;
cmPrintGroup    : Begin
                    Warning;
                    PrintGroup (mdReport);
                  End;
cmDeleteReport  : DeleteData(mdReport);
cmDeleteGroup   : DeleteData(mdGroup);
cmPrinterSetup  : PrinterSetup;
cmDeleteData    : DeleteData(mdDataFile);
cmSaveData      : SaveData(mdDataFile);
cmLoadPrev      : Begin
                    If ChangeFlag
                      Then Tempword := MessageBox ('Do you want to '
                        + 'save data in memory before loading a '
                        + 'new data file?', Nil, mfYesNoCancel
                        + mfConfirmation)
                      Else Tempword := cmNo;
                    If TempWord = cmYes
                      Then SaveData (mdDataFile);
                    If TempWord <> cmCancel
                      Then LoadPrevious(mdDataFile);
                  End;
cmSaveSF        : SaveData(mdStdFctrs);
cmLoadSF        : Begin
                    If ChangeFlag
                      Then Tempword := MessageBox ('Do you want to '
                        + 'save Standard Factors in memory before '
```

```
                    + 'loading a new file?', Nil, mfYesNoCancel
                    + mfConfirmation)
                  Else Tempword := cmNo;
                If TempWord = cmYes
                  Then SaveData (mdStdFctrs);
                If TempWord <> cmCancel
                  Then LoadPrevious(mdStdFctrs);
              End;
cmAutoLoad    : LoadPrevious(mdAutoLoad);
cmNewSF       : Begin
                If ChangeFlag
                  Then Tempword := MessageBox ('Are you sure '
                    + 'you want to clear Standard Factors in '
                    + 'memory?', Nil, mfYesNoCancel
                    + mfConfirmation)
                  Else Tempword := cmYes;
                If TempWord = cmYes Then
                  Begin
                    InitStandardFactors;
                    Screen1Data^.SFFile := '';
                  End;
              End;
cmLoadOld     : LoadPrevious(mdOldData);
cmExitCobra   : Begin
                Tempword := MessageBox ('Are you sure you want '
                  + 'to exit COBRA?', Nil, mfYesNoCancel
                  + mfConfirmation);
                If TempWord = cmYes Then
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    tempevent.Command := cmQuit;
                  End;
              End;
cmLoadDB      : LoadFromDataBase;
cmSaveDB      : ChooseBase (mdDataBase);
cmDistanceDB  : DistanceDataBase;
cmChangeDir   : ChangeDir;
cmDosShell    : DosShell;
cmCobraFiles  : Begin
                If DirExists (Info.ReportDir) Then
                  Begin
                    OptionSearch;
                    If FileExists
                        (Info.ReportDir + 'COBFILES.RPT')
                      Then ViewFile
                        (Info.ReportDir + 'COBFILES.RPT')
                      Else MessageBox ('No COBRA files found '
                        + 'in current directory.', Nil, mfError
                        + mfOKButton);
                  End
                Else MessageBox ('Report Directory '
```

```
                        + Info.ReportDir + ' not found.', Nil, mfError
                        + mfOKButton);
                    End;
cmDeleteBase      : ChooseBase (mdDelBase);
cmMakeReport      : Begin
                      If ChangeFlag
                        Then Messagebox ('You may wish to re-Execute '
                          + 'the scenario before creating individual '
                          + 'Base reports.', Nil, mfOKButton);
                      ChooseBase (mdReport);
                    End;
cmVideoToggle     : Begin { Toggles video between Mono and Color.
                            VideoToggle was added for demonstrations to
                            clients using overhead projectors.  It can
                            also be used on laptops.  This feature will
                            probably not be documented. }
                      If ScreenMode = smMono
                        Then SetScreenMode (smCO80)
                        Else SetScreenMode (smMono);
                      ShadowSize.X := 0;
                      ShadowSize.Y := 0;
                      Redraw;
                    End;
cmScreenOne       : Begin
                      tempword := ScreenOneNine (1);
                      If (tempword = cmNextScr) THEN
                        Begin
                          usetemp           := TRUE;
                          tempevent.What    := evCommand;
                          If (MaxBases = 0) And
                             (Screen1Data^.Base[0].Name = '')
                            Then tempevent.Command := cmScreenNine
                          Else If (MaxBases = 0)
                            Then tempevent.Command := cmScreenFour
                          Else tempevent.Command := cmScreenTwo;
                        End;
                    End;
cmScreenTwo       : Begin
                      tempword := ScreenTwo;
                      If (tempword = cmNextScr) THEN
                        Begin
                          usetemp           := TRUE;
                          tempevent.What    := evCommand;
                          tempevent.Command := cmScreenThree;
                        End;
                      If (tempword = cmPrevScr) THEN
                        Begin
                          usetemp           := TRUE;
                          tempevent.What    := evCommand;
                          tempevent.Command := cmScreenOne;
                        End;
                    End;
```

```
cmScreenThree  : Begin
                   tempword := ScreenThree;
                   If (tempword = cmNextScr) THEN
                     Begin
                       usetemp            := TRUE;
                       tempevent.What     := evCommand;
                       tempevent.Command  := cmScreenFour;
                     End;
                   If (tempword = cmPrevScr) THEN
                     Begin
                       usetemp            := TRUE;
                       tempevent.What     := evCommand;
                       tempevent.Command  := cmScreenTwo;
                     End;
                 End;
cmScreenFour   : Begin
                   tempword := ScreenFourFiveSixSeven (4);
                   If (tempword = cmNextScr) THEN
                     Begin
                       usetemp            := TRUE;
                       tempevent.What     := evCommand;
                       tempevent.Command  := cmScreenFive;
                     End;
                   If (tempword = cmPrevScr) THEN
                     Begin
                       usetemp            := TRUE;
                       tempevent.What     := evCommand;
                       If MaxBases > 0
                         Then tempevent.Command := cmScreenThree
                         Else tempevent.Command := cmScreenOne;
                     End;
                 End;
cmScreenFive   : Begin
                   tempword := ScreenFourFiveSixSeven (5);
                   If (tempword = cmNextScr) THEN
                     Begin
                       usetemp            := TRUE;
                       tempevent.What     := evCommand;
                       tempevent.Command  := cmScreenSix;
                     End;
                   If (tempword = cmPrevScr) THEN
                     Begin
                       usetemp            := TRUE;
                       tempevent.What     := evCommand;
                       tempevent.Command  := cmScreenFour;
                     End;
                 End;
cmScreenSix    : Begin
                   tempword := ScreenFourFiveSixSeven (6);
                   If (tempword = cmNextScr) THEN
                     Begin
                       usetemp            := TRUE;
```

```
                    tempevent.What    := evCommand;
                    tempevent.Command := cmScreenSeven;
                  End;
                If (tempword = cmPrevScr) THEN
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    tempevent.Command := cmScreenFive;
                  End;
              End;
cmScreenSeven  : Begin
                tempword := ScreenFourFiveSixSeven (7);
                If (tempword = cmNextScr) THEN
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    tempevent.Command := cmScreenEight;
                  End;
                If (tempword = cmPrevScr) THEN
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    tempevent.Command := cmScreenSix;
                  End;
              End;
cmScreenEight  : Begin
                tempword := ScreenEight;
                If (tempword = cmNextScr) THEN
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    tempevent.Command := cmScreenNine;
                  End;
                If (tempword = cmPrevScr) THEN
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    tempevent.Command := cmScreenSeven;
                  End;
              End;
cmScreenNine   : Begin
                tempword := ScreenOneNine (9);
                If (tempword = cmPrevScr) THEN
                  Begin
                    usetemp           := TRUE;
                    tempevent.What    := evCommand;
                    If (MaxBases > 0) Or
                       (Screen1Data^.Base[0].Name <> '')
                      Then tempevent.Command := cmScreenEight
                      Else tempevent.Command := cmScreenOne;
                  End;
              End;
```

```
          cmStdFactors1  : StandardFactors (1);
          cmStdFactors2  : StandardFactors (2);
          cmStdFactors3  : StandardFactors (3);
          cmStdFactors4  : StandardFactors (4);
          cmTile,
          cmCascade      : CascadeOrTile (event.Command);
          cmRedraw       : Redraw;
          cmExecute      : Begin
                             If DirExists (Info.ReportDir) Then
                               Begin
                                 DisableCommands
                                   ([CmClose, cmHelp, cmViewUsed]);
                                 Redraw;
                                 ExecCalc;
                                 ChangeFlag := False;
                                 EnableCommands
                                   ([CmClose, cmHelp, cmViewUsed]);
                                 If AutoStart.ExitCOBRA Then
                                   Begin
                                     usetemp            := TRUE;
                                     tempevent.What     := evCommand;
                                     tempevent.Command := cmQuit;
                                   End
                                 Else
                                   Begin
                                     If FileExists
                                        (Info.ReportDir + 'Error.Rpt')
                                       Then Tempword := MessageBox
                                          ('There were problems with some of '
                                         + 'the scenario data.  See '
                                         + 'ERROR.RPT for more details.',
                                            nil, mfOkButton);
                                     ViewFile (Info.ReportDir + 'COBSUM.RPT');
                                     AutoStart.Execute := False;
                                   End;
                               End
                             Else Messagebox ('Report Directory '
                               + Info.ReportDir + ' not found.', Nil, mfError
                               + mfOKButton);
                           End;
        End; { Case }

        ClearEvent (event);
        If (usetemp = True)
          THEN PutEvent (tempevent);
      End;
  End; { Case }
End; { TCobraApp.HandleEvent }
```

```
{*************************************************************************
 *  Name          : ViewUsed
 *
 *  Purpose       : Display the data files in use
 *
 *  Notes         :
 *
 *  Invoked As    : ViewUsed;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  30Mar92   AML   Creation
 *************************************************************************}

Procedure TCobraApp.ViewUsed;

  VAR
    dialog   : PDialog;
    I        : Integer;
    rect     : TRect;
    temp     : Integer;
    tempword : WORD;

  Begin
    temp := 16;
    If Length (SFFileUsed) > temp
      Then temp := Length (SFFileUsed);
    If Length (DataFileUsed) > temp
      Then temp := Length (DataFileUsed);
    rect.Assign (34 - temp div 2, 4, 43 + temp div 2, 14);
    dialog := New (PDialog, Init (rect, 'Files in Use'));
    rect.Assign (2, 2, 19, 3);
    dialog^.Insert(New(PStaticText,Init(rect,'Current Data File')));
    rect.Assign (2, 4, 23, 5);
    dialog^.Insert(New(PStaticText,Init(rect,'Standard Factors File')));
    rect.Assign (6, 3, temp + 6, 4);
    dialog^.Insert(New(PStaticText,Init(rect,Copy(DataFileUsed,1,
      Length (DataFileUsed)))));
    rect.Assign (6, 5, temp + 6, 6);
    dialog^.Insert(New(PStaticText,Init(rect,Copy(SFFileUsed,1,
      Length (SFFileUsed)))));
    rect.Assign (temp div 2, 7, temp div 2 + 8, 9);
    dialog^.Insert(New(PButton, Init(rect, 'O~K', cmOK, bfDefault)));
    If ValidView(dialog) <> NIL THEN
      Begin
        tempword := Desktop^.ExecView(dialog);
        Dispose(dialog, Done);
      End;
  End; {ViewUsed}
```

```
{*************************************************************************
 *  Name          : AboutWindow
 *
 *  Purpose       : Display the "About" (Title) window
 *
 *  Notes         :
 *
 *  Invoked As    : AboutWindow;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  13Jan91   ASB   Creation
 *************************************************************************}

Procedure TCobraApp.AboutWindow;

  VAR
    dialog   : PDialog;
    rect     : TRect;
    tempword : WORD;

  Begin
    rect.Assign (0,0,63,21);
    dialog := New (PDialog, Init (rect, 'About COBRA'));
    dialog^.HelpCtx := hcDragging;
    dialog^.Options := dialog^.Options OR ofCentered;

    rect.Assign ( 1, 2,62, 6);
    dialog^.Insert(New(PStaticText,Init(rect,
      ^C'[illegible]                                                    '#13 +
      ^C'[illegible]                                                    '#13 +
      ^C'[illegible]                                                    '#13 +
      ^C'[illegible]                                                    '#13)));
    rect.Assign ( 1, 6,62, 9);
    dialog^.Insert(New(PStaticText,Init(rect,
      ^C'[illegible]                                                    '#13 +
      ^C'[illegible]                                                    '#13 +
      ^C'[illegible]                                                    '#13)));
    rect.Assign ( 1, 9,62,12);
    dialog^.Insert(New(PStaticText,Init(rect,
      ^C'[illegible]                                                    '#13 +
      ^C'[illegible]                                                    '#13 +
      ^C'                                                               '#13)));
    rect.Assign ( 1,12,62,17);
    dialog^.Insert(New(PStaticText, Init(rect,#13
      + ^C'Cost Of Base Realignment Action'#13
      + ^C'Version ' + VERSION + #13
      + ^C'R&K Engineering, Inc.'#13
      + ^C'(703) 683-7100')));
    rect.Assign (56,19,60,20);
```

```
    dialog^.Insert(New(PStaticText, Init (rect,
      WithCommas (MemAvail / 1000.0, 0) + 'K')));

    rect.Assign (26,18,36,20);
    dialog^.Insert(New(PButton, Init(rect, 'O~K', cmOK, bfDefault)));

    If ValidView(dialog) <> NIL THEN
      Begin
        tempword := Desktop^.ExecView(dialog);
        Dispose(dialog, Done);
      End;
  End;

{*************************************************************************
 *  Name          : Calculator
 *
 *  Notes         : Rescued from archived Cobra 1.21, uses TurboVision demo
 *
 *  Invoked As    : Calculator
 *
 *  Date      Init  ModIfications
 *  -------   ----  ------------------------------------------------------
 *  13Jan91   AJD   Creation
 *  30Apr92   AML   Reinstated
 *************************************************************************}

Procedure TCobraApp.Calculator;

  VAR
    P : PCalculator;

  Begin
    P := New (PCalculator, Init);
    P^.HelpCtx := hcCalculator;
    If ValidView (P) <> nil
      Then Desktop^.Insert (P);
  End;

{*************************************************************************
 *  Name          : Calendar
 *
 *  Notes         : Like Calculator, uses TurboVision demos
 *
 *  Invoked As    : Calendar
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  05May92   AML   Creation
 *************************************************************************}

Procedure TCobraApp.Calendar;
```

```
  VAR
    P : PCalendarWindow;

  Begin
    P := New (PCalendarWindow, Init);
    P^.HelpCtx := hcCalendar;
    If ValidView (P) <> nil
      Then Desktop^.Insert (P);
  End;

{*********************************************************************
 *  Name          : CascadeOrTile
 *
 *  Notes         : Like Calculator, uses TurboVision demos
 *
 *  Invoked As    : CascadeOrTile (Mode)
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------
 *  05Jun92   AML   Creation
 *********************************************************************}

Procedure TCobraApp.CascadeOrTile (Mode : Integer);

  Var
    R: TRect;
  Begin
    Desktop^.GetExtent(R);
    If Mode = cmCascade
      Then Desktop^.Cascade(R)
      Else Desktop^.Tile(R);
  End;

{*********************************************************************
 *  Name          : ChangeDir
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    : ChangeDir;
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------
 *  13Jan91   ASB   Creation
 *********************************************************************}

Procedure TCobraApp.ChangeDir;

  Var
    d        : PChDirDialog;
    tempword : Word;
```

```
  Begin
    d := New (PChDirDialog, Init (cdNormal, hlChangeDir));
    d^.HelpCtx := hcShellCD;
    If ValidView(d) <> Nil Then
      Begin
        tempword := DeskTop^.ExecView (d);
        Dispose (d, Done);
      End;
  End; {ChangeDir}

{*********************************************************************
 *  Name          : DosShell
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    : DosShell;
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  13Jan91   ASB   Creation
 *********************************************************************}

Procedure TCobraApp.DosShell;

   Begin
     DoneSysError;
     DoneEvents;
     DoneVideo;
     DoneMemory;
     SetMemTop     (HeapPtr);
     PrintStr      ('Type EXIT to return...');
     SwapVectors;
     Exec(GetEnv   ('COMSPEC'), '');
     SwapVectors;
     SetMemTop     (HeapEnd);
     InitMemory;
     InitVideo;
     InitEvents;
     InitSysError;
     Redraw;
   End; {DosShell}
```

```
{*********************************************************************
 *  Name          : LoadFromDataBase
 *
 *  Purpose       : Load Bases w/ Static Info from DataBase
 *
 *  Invoked As    : LoadFromDataBase;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  22Jun92   AML   Creation
 *********************************************************************}

Procedure TCobraApp.LoadFromDataBase;

  Var
    B, B2      : Integer;
    DBase      : DBaseSearchRec;
    Dialog     : PDBaseDlg;
    Event      : TEvent;
    FileName   : PathStr;
    Found      : Boolean;
    Match,
    NumBases,
    OffSet     : Integer;
    P, Q       : PDBaseRec;
    S          : BaseArray;
    TempWord,
    TempWord2  : Word;

  Begin
    FileName := GetFileForLoad ('.DBS', 'Load Base(s)', '~S~elect DataBase...');
    If (FileName <> '') Then
      Begin
        OffSet := 0;
        DBase.SearchName := '';
        New (P);
        DBNames (FileName, P, NumBases);
        If NumBases > 0 Then
          Repeat
            PutTab;
            DBase.Selected := 0;
            DBase.SearchName := UpperCase(DBase.SearchName);
            Q := P;
            For B := 1 To OffSet Do
              Q := Q^.Next;
            For B := 0 To 14 Do
              If Q^.Next = Nil
                Then S[B] := ''
              Else
                Begin
```

```
      Q := Q^.Next;
      S[B] := Q^.BaseName;
    End;
Dialog := New (PDBaseDlg,Init(S));
Dialog^.HelpCtx := hcLoadDB;

If TApplication.ValidView (dialog) <> NIL THEN
  Begin
    dialog^.SetData (DBase);
    tempword := Desktop^.ExecView (dialog);
    If (tempword <> cmCancel) THEN
      Begin
        dialog^.GetData (DBase);
        If DBase.Selected > 0
          Then For B := 0 To 14 Do
            If S[B] <> ''
              Then If Odd(DBase.Selected Div Round(YtoX(2,B))) Then
                Begin
                  Found := False;
                  If (MaxBases > 0) Or
                     (Screen1Data^.Base[0].Name <> '')
                    Then For B2 := 0 To MaxBases Do
                      If UpperCase(Screen1Data^.Base[B2].Name) =
                         UpperCase(S[B]) Then
                        Begin
                          Found := True;
                          Match := B2;
                        End;
                  If Found Then
                    Begin
                      TempWord2 := MessageBox
                        ('Base ' + UpperCase(S[B]) +
                         ' already in memory.  Overwrite?',
                         nil, mfYesNoCancel + mfConfirmation);
                      If TempWord2 = cmYes
                        Then DBLoad (S[B], Match, FileName);
                    End
                  Else If MaxBases < AbsMaxBases Then
                    Begin
                      If Screen1Data^.Base[0].Name = ''
                        Then DBLoad (S[B], 0, FileName)
                      Else
                        Begin
                          MaxBases := MaxBases + 1;
                          DBLoad (S[B], MaxBases, FileName);
                        End
                    End
                  Else MessageBox (
                    'Maximum number of bases loaded.',
                    nil, mfOkButton + mfError);
                End;
        If (TempWord = cmOK) And (NumBases > 15) Then
```

```
                    Begin
                      OffSet := OffSet + 15;
                      If OffSet + 15 > NumBases
                        Then OffSet := NumBases - 15;
                    End;
                  If (TempWord = cmYes) And (NumBases > 15) Then
                    Begin
                      OffSet := OffSet - 15;
                      If OffSet < 0
                        Then OffSet := 0;
                    End;
                  If (TempWord = cmNo) And (DBase.SearchName = '')
                    Then TempWord := cmCancel;
                  If (TempWord = cmNo) And (NumBases > 15) Then
                    Begin
                      Q := P;
                      For B := 1 To OffSet + 1 Do
                        Q := Q^.Next;
                      Found := False;
                      B := 0;
                      While (Q^.Next <> Nil) And Not Found Do
                        Begin
                          Q := Q^.Next;
                          Inc (B);
                          If Pos(UpperCase(DBase.SearchName),
                            UpperCase(StripSpace(Copy(Q^.BaseName,1,20)))) > 0
                              Then Begin
                                Found := True;
                                OffSet := OffSet + B;
                              End;
                        End;
                      If OffSet + 15 > NumBases
                        Then OffSet := NumBases - 15;
                    End;
                End;
              Dispose (dialog,Done);
            End;
        Until (TempWord = cmCancel) or (MaxBases = AbsMaxBases)
      Else Tempword := MessageBox (FileName + ' contains no bases.',
        nil, mfOkButton + mfError);
      Dispose (P);
    End;
End;
```

```
{*************************************************************************
 *  Name          : ChooseBase
 *
 *  Purpose       : Picks a base for deletion, special report, or to be
 *                  saved to the DataBase
 *
 *  Notes         :
 *
 *  Invoked As    : ChooseBase (Mode);
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  31Mar92   AML   Creation
 *  22Jun92   AML   Added DataBase link
 *************************************************************************}

Procedure TCobraApp.ChooseBase (Mode : Integer);

  VAR
    B        : Integer;
    Bases    : Word;
    D        : PFileDialog;
    dialog   : PDialog;
    FileName : PathStr;
    rect     : TRect;
    result   : Boolean;
    S        : Array [0..14] Of String20;
    tempword : Word;

  Begin
    If Mode = mdDataBase Then
      Begin
        D := PFileDialog(ValidView(New(PFileDialog, Init('*.DBS',
          'Save Base(s)', '~S~elect DataBase...', fdOKButton, 100))));
        D^.HelpCtx := hcLoadWindow;
        If D <> nil THEN
          If Desktop^.ExecView (D) <> cmCancel THEN
            Begin
              D^.GetFileName (FileName);
              If FileName = ''
                Then Mode := mdNull
              Else If Pos ('.', FileName) = 0
                Then FileName := Concat (FileName, '.DBS');
              Dispose (D, Done);
            End
          Else Mode := mdNull
        Else Mode := mdNull;
      End;
    If Mode <> mdNull Then
      Begin
```

```
For B := 0 To 14 Do
  If B <= MaxBases
    Then S[B] := Screen1Data^.Base[B].Name
    Else S[B] := '';
Bases := 0;
rect.Assign (24, 1, 56, 22);
Case Mode of
  mdDataBase: Begin
                dialog := New(PDialog, Init(rect, 'Save to DataBase'));
                Dialog^.HelpCtx := hcSaveDB;
              End;
  mdDelBase:  Begin
                dialog := New(PDialog, Init(rect, 'Delete Bases'));
                Dialog^.HelpCtx := hcDeleteBase;
              End;
  mdReport:   Begin
                dialog := New(PDialog, Init(rect, 'Base Report(s)'));
                Dialog^.HelpCtx := hcMakeReport;
              End;
  End;
rect.Assign (3, 1, 28, 2);
If Mode = mdDataBase
  Then Dialog^.Insert(New(PStaticText,Init(rect,
    'Mark Bases for DataBase:')));
If Mode = mdDelBase
  Then Dialog^.Insert(New(PStaticText,Init(rect,
    'Mark Bases for Deletion:')));
If Mode = mdReport
  Then Dialog^.Insert(New(PStaticText,Init(rect,
    'Select Bases for Reports:')));
rect.Assign (4, 18, 14, 20);
If Mode = mdDelBase
  Then Dialog^.Insert(New(PButton, Init(rect,
    '~D~elete', cmOK, bfNormal)));
If (Mode = mdReport) Or (Mode = mdDataBase)
  Then Dialog^.Insert(New(PButton, Init(rect,
    'O~K', cmOK, bfDefault)));
rect.Assign (17, 18, 27, 20);
Dialog^.Insert(New(PButton, Init(rect,
  'Cancel', cmCancel, bfNormal)));
rect.Assign (2, 2, 30, 17);

If Mode = mdReport Then
  Begin
    Inc (rect.B.Y);
    Dialog^.Insert (New (PCheckBoxes, Init (rect,
      NewSItem ('~0~-'+S[0],
      NewSItem ('~1~-'+S[1],
      NewSItem ('~2~-'+S[2],
      NewSItem ('~3~-'+S[3],
      NewSItem ('~4~-'+S[4],
      NewSItem ('~5~-'+S[5],
```

```
        NewSItem ('~6~-'+S[6],
        NewSItem ('~7~-'+S[7],
        NewSItem ('~8~-'+S[8],
        NewSItem ('~9~-'+S[9],
        NewSItem ('~A~-'+S[10],
        NewSItem ('~B~-'+S[11],
        NewSItem ('~C~-'+S[12],
        NewSItem ('~E~-'+S[13],
        NewSItem ('~F~-'+S[14],
        NewSItem ('~T~-(Totals)',
        nil)))))))))))))))))));
    End
  Else Dialog^.Insert (New (PCheckBoxes, Init (rect,
    NewSItem ('~0~-'+S[0],
    NewSItem ('~1~-'+S[1],
    NewSItem ('~2~-'+S[2],
    NewSItem ('~3~-'+S[3],
    NewSItem ('~4~-'+S[4],
    NewSItem ('~5~-'+S[5],
    NewSItem ('~6~-'+S[6],
    NewSItem ('~7~-'+S[7],
    NewSItem ('~8~-'+S[8],
    NewSItem ('~9~-'+S[9],
    NewSItem ('~A~-'+S[10],
    NewSItem ('~B~-'+S[11],
    NewSItem ('~C~-'+S[12],
    NewSItem ('~E~-'+S[13],
    NewSItem ('~F~-'+S[14],
    nil)))))))))))))))))));
  If ValidView (Dialog) <> Nil Then
    Begin
      Dialog^.SetData (Bases);
      TempWord := Desktop^.ExecView(Dialog);
      If (TempWord <> cmCancel)
        Then Dialog^.GetData (Bases);
      Dispose(Dialog, Done);
    End;
  If (TempWord = cmOK)
    Then If (Bases <> 0)
      Then For B := MaxBases DownTo 0 Do
        If Odd(Bases Div Round(YtoX(2,B)))
          Then Case Mode Of
            mdDataBase:
              Begin
                Result := DBCheck (B, FileName);
                If Result
                  Then Tempword := MessageBox (Screen1Data^.Base[B].Name
                    + ' is already in DataBase.  Overwrite?', Nil,
                    mfYesNoCancel + mfConfirmation);
                If (Not Result) Or (TempWord = cmYes)
                  Then DBSave (B, FileName);
              End;
```

```
                    mdDelBase:
                      Begin
                        PutTab;
                        TempWord := Messagebox ('Are you sure you want to delete '
                          + Screen1Data^.Base[B].Name + '?', Nil, mfYesNoCancel
                          + mfConfirmation);
                        If TempWord = cmYes
                          Then KillBase (B);
                      End;
                    mdReport:
                      Begin
                        FileName := Concat('BASE',
                          Chr (48 + B Div 10),
                          Chr (48 + B Mod 10));
                        TempWord := GetFileForSave(FileName, mdReport);
                        If (TempWord <> cmCancel) And (FileName <> '') Then
                          Begin
                            Result := MakeReport(B, FileName);
                            If Not Result
                              Then tempword := MessageBox
                                ('Unable to Create ' + FileName + '.RPT',
                                nil, mfError + mfOkButton);
                          End;
                      End;
                  End;
          If (Bases > 32767) Then
            Begin
              FileName := 'TOTALS';
              TempWord := GetFileForSave(FileName, mdReport);
              If (TempWord <> cmCancel) And (FileName <> '') Then
                Begin
                  Result := MakeReport(MaxBases + 1, FileName);
                  If Not Result
                    Then tempword := MessageBox
                      ('Unable to Create ' + FileName + '.RPT',
                      nil, mfError + mfOkButton);
                End;
            End;
        End;
    End; {ChooseBase}

Procedure TCobraApp.DistanceDataBase;

  VAR
    B, B2, C    : Integer;
    Bases       : TwoWords;
    D           : PFileDialog;
    DBDistance  : ^DistanceRec;
    dialog      : PDialog;
    Event       : TEvent;
    F           : Text;
    FileName    : PathStr;
```

```
  Finished    : Boolean;
  InBuffer    : IOBuffer;
  rect        : TRect;
  S           : Array [0..14] Of String20;
  SLabel,
  TempBase1,
  TempBase2   : String20;
  TempDist    : Real;
  TempStr     : String;
  tempword,
  TransMode   : Word;

Begin
  Bases.First := 0;
  Bases.Second := 0;
  FileName := '';
  Finished := False;
  TransMode := 0;
  D := PFileDialog(ValidView(New(PFileDialog, Init('*.DDB',
    'Distances', '~S~elect DataBase...', fdOKButton, 100))));
  D^.HelpCtx := hcLoadWindow;
  If D <> nil THEN
    If Desktop^.ExecView (D) <> cmCancel THEN
      Begin
        D^.GetFileName (FileName);
        If (FileName <> '') And (Pos ('.', FileName) = 0)
          Then FileName := Concat (FileName, '.DDB');
        Dispose (D, Done);
      End;
  If FileName <> '' Then
    Begin
      For B := 0 To 14 Do
        If B <= MaxBases
          Then S[B] := Screen1Data^.Base[B].Name
          Else S[B] := '';
      New (DBDistance);
      FillChar (DBDistance^, SizeOf (DBDistance^), 0);
      If FileExists (FileName) Then
        Begin
          Assign (F, FileName);
          SetTextBuf (F, InBuffer);
          Reset (F);
          While Not EOF(F) Do
            Begin
              Readln (F, TempBase1, TempBase2, TempDist);
              For B := 0 To MaxBases - 1 Do
                For B2 := B + 1 To MaxBases Do
                  If ((StripSpace(UpperCase(TempBase1)) =
                       StripSpace(UpperCase(Screen1Data^.Base[B].Name))) And
                      (StripSpace(UpperCase(TempBase2)) =
                       StripSpace(UpperCase(Screen1Data^.Base[B2].Name)))) Or
                     ((StripSpace(UpperCase(TempBase1)) =
```

```
                    StripSpace(UpperCase(Screen1Data^.Base[B2].Name))) And
                   (StripSpace(UpperCase(TempBase2)) =
                    StripSpace(UpperCase(Screen1Data^.Base[B].Name)))) Then
                 Begin
                   DBDistance^.Base[B].Base[B2] := TempDist;
                   DBDistance^.Base[B2].Base[B] := TempDist;
                 End;
        End;
      Close (F);
    End;
  Repeat
    PutTab;
    rect.Assign (0, 0, 62, 21);
    dialog := New(PDialog, Init(rect, 'Distance DataBase'));
    Dialog^.Options := Dialog^.Options OR OfCentered;
    Dialog^.HelpCtx := hcDistanceDB;
    rect.Assign (2, 1, 58, 2);
    Dialog^.Insert(New(PStaticText,Init(rect,
      'Select two Bases for Distance DataBase check:')));
    rect.Assign (3, 18, 15, 20);
    Dialog^.Insert(New(PButton, Init(rect, '~O~K', cmOK, bfDefault)));
    rect.Assign (17, 18, 29, 20);
    Dialog^.Insert(New(PButton,
      Init(rect, 'Cancel', cmCancel, bfNormal)));
    rect.Assign (31, 18, 44, 20);
    Dialog^.Insert(New(PButton,
      Init(rect, 'All>~M~em', cmYes, bfNormal)));
    rect.Assign (46, 18, 58, 20);
    Dialog^.Insert(New(PButton, Init(rect, 'All>~D~B', cmNo, bfNormal)));
    For B := 0 To 1 Do
      Begin
        rect.Assign (2 + B * 30, 2, 30 + B * 30, 17);
        If B = 0
          Then SLabel := '0123456789ABCEF'
          Else SLabel := 'GHIJKLNPQRSTUVW';
        Dialog^.Insert (New (PRadioButtons, Init (rect,
          NewSItem ('~'+SLabel[1]+'~-'+S[0],
          NewSItem ('~'+SLabel[2]+'~-'+S[1],
          NewSItem ('~'+SLabel[3]+'~-'+S[2],
          NewSItem ('~'+SLabel[4]+'~-'+S[3],
          NewSItem ('~'+SLabel[5]+'~-'+S[4],
          NewSItem ('~'+SLabel[6]+'~-'+S[5],
          NewSItem ('~'+SLabel[7]+'~-'+S[6],
          NewSItem ('~'+SLabel[8]+'~-'+S[7],
          NewSItem ('~'+SLabel[9]+'~-'+S[8],
          NewSItem ('~'+SLabel[10]+'~-'+S[9],
          NewSItem ('~'+SLabel[11]+'~-'+S[10],
          NewSItem ('~'+SLabel[12]+'~-'+S[11],
          NewSItem ('~'+SLabel[13]+'~-'+S[12],
          NewSItem ('~'+SLabel[14]+'~-'+S[13],
          NewSItem ('~'+SLabel[15]+'~-'+S[14],
          nil)))))))))))))))));
```

```
  End;
TempWord := CmCancel;
If ValidView (Dialog) <> Nil Then
  Begin
    Dialog^.SetData (Bases);
    TempWord := Desktop^.ExecView(Dialog);
    If (TempWord <> cmCancel)
      Then Dialog^.GetData (Bases);
    Dispose(Dialog, Done);
  End;
If TempWord = cmOK Then
  Begin
    If (Bases.First  <= MaxBases) And
       (Bases.Second <= MaxBases) And
       (Bases.First <> Bases.Second) Then
      Begin
        PutTab;
        rect.Assign (0, 0, 30, 12);
        dialog := New(PDialog, Init(rect,
          'DB/Memory Transfers'));
        Dialog^.Options := Dialog^.Options OR OfCentered;
        Dialog^.HelpCtx := hcDistanceDB;
        Rect.Assign (2,1,28,2);
        Dialog^.Insert(New(PStaticText,Init(rect,
          'From: ' + Screen1Data^.Base[Bases.First].Name)));
        Rect.Assign (4,2,28,3);
        Dialog^.Insert(New(PStaticText,Init(rect,
          'To: ' + Screen1Data^.Base[Bases.Second].Name)));
        Str (DBDistance^.Base[Bases.First].Base[Bases.Second]:8:1,
          TempStr);
        Rect.Assign (2,4,28,5);
        Dialog^.Insert(New(PStaticText,Init(rect,
          'DataBase Distance:' + TempStr)));
        Str (Distance^.Base[Bases.First].Base[Bases.Second]:8:1,
          TempStr);
        Rect.Assign (4,5,28,6);
        Dialog^.Insert(New(PStaticText,Init(rect,
          'Memory Distance:' + TempStr)));
        rect.Assign (2,6,28,8);
        Dialog^.Insert (New (PRadioButtons, Init (rect,
          NewSItem ('~M~emory to DataBase',
          NewSItem ('~D~ataBase to Memory',
          nil)))));
        rect.Assign (2, 9, 14, 11);
        Dialog^.Insert(New(PButton,
          Init(rect, '~O~K', cmOK, bfDefault)));
        rect.Assign (16, 9, 28, 11);
        Dialog^.Insert(New(PButton,
          Init(rect, 'Cancel', cmOK, bfNormal)));
        If ValidView (Dialog) <> Nil Then
          Begin
            Dialog^.SetData (TransMode);
```

```
                TempWord := Desktop^.ExecView(Dialog);
                If (TempWord <> cmCancel) Then
                  Dialog^.GetData (TransMode);
                If (TransMode = 0) And (TempWord <> cmCancel) And
                   (DBDistance^.Base[Bases.First].Base[Bases.Second] <>
                    Distance^.Base[Bases.First].Base[Bases.Second]) Then
                  Begin
                    DBDistSave (FileName,
                      Screen1Data^.Base[Bases.First].Name,
                      Screen1Data^.Base[Bases.Second].Name,
                      DBDistance^.Base[Bases.First].Base[Bases.Second],
                      Distance^.Base[Bases.First].Base[Bases.Second]);
                    DBDistance^.Base[Bases.First].Base[Bases.Second] :=
                      Distance^.Base[Bases.First].Base[Bases.Second];
                    DBDistance^.Base[Bases.Second].Base[Bases.First] :=
                      Distance^.Base[Bases.Second].Base[Bases.First];
                  End
                Else If (TransMode = 1) And (TempWord <> cmCancel) Then
                  Begin
                    Distance^.Base[Bases.First].Base[Bases.Second] :=
                      DBDistance^.Base[Bases.First].Base[Bases.Second];
                    Distance^.Base[Bases.Second].Base[Bases.First] :=
                      DBDistance^.Base[Bases.Second].Base[Bases.First];
                    ChangeFlag := True;
                  End;
                Dispose(Dialog, Done);
              End;
          End;
      End
    Else If TempWord = cmYes Then
      Begin
        C := 0;
        For B := 0 To MaxBases - 1 Do
          For B2 := B + 1 To MaxBases Do
            If DBDistance^.Base[B].Base[B2] <> 0 Then
              Begin
                Distance^.Base[B].Base[B2] :=
                  DBDistance^.Base[B].Base[B2];
                Distance^.Base[B2].Base[B] :=
                  DBDistance^.Base[B2].Base[B];
                Inc (C);
              End;
        If C > 0
          Then ChangeFlag := True;
        Str (C, TempStr);
        MessageBox (TempStr + ' matches found in database.', Nil,
          mfOKButton + mfInformation);
        Finished := True;
      End
    Else If TempWord = cmNo Then
      Begin
        C := 0;
```

```
            For B := 0 To MaxBases - 1 Do
              For B2 := B + 1 To MaxBases Do
                If (Distance^.Base[B].Base[B2] <> 0) And
                   (Distance^.Base[B].Base[B2] <>
                    DBDistance^.Base[B].Base[B2]) Then
                  Begin
                    DBDistSave (FileName,
                      Screen1Data^.Base[B].Name,
                      Screen1Data^.Base[B2].Name,
                      DBDistance^.Base[B].Base[B2],
                      Distance^.Base[B].Base[B2]);
                    Inc (C);
                  End;
            Str (C, TempStr);
            MessageBox (TempStr + ' pairs written to database.', Nil,
              mfOKButton + mfInformation);
            Finished := True;
          End
        Else Finished := True;
      Until Finished;
    End;
  End; {DistanceDataBase}

{*********************************************************************
 *  Name          : ViewFile
 *
 *  Purpose       : Opens specified file for viewing
 *
 *  Notes         :
 *
 *  Invoked As    : ViewFile(FileName);
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Jan91   GJG   Creation
 *********************************************************************}

Procedure TCobraApp.ViewFile (FileName : PathStr);

  VAR
    W : PWindow;

  Begin
    W := PWindow (ValidView (New (PFileWindow, Init (FileName))));
    If W <> nil THEN
      Desktop^.Insert (W);
  End;
```

```
{*************************************************************************
 *  Name          : OpenFile
 *
 *  Purpose       : Open report or help files
 *
 *  Notes         :
 *
 *  Invoked As    : OpenFile (Mode);
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Jan91   AJD   Creation
 *  20May92   AML   Combined OpenReport and OpenHelp into OpenFile
 *************************************************************************}

Procedure TCobraApp.OpenFile(Mode:Integer);

  VAR
    FileName : PathStr;

  Begin
    Case Mode Of
      mdReport: FileName := GetFileForLoad
        ('.RPT', 'View Reports', '~S~elect Report...');
      mdHelp:   FileName := GetFileForLoad
        ('.HLP', 'View Help', '~S~elect Help File...');
      End;
    If FileName <> ''
      Then Viewfile (FileName)
  End;

{*************************************************************************
 *  Name          : PrintReport
 *
 *  Purpose       : Print report files
 *
 *  Notes         :
 *
 *  Invoked As    : PrintReport;
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  30Mar92   AML   Creation
 *************************************************************************}

Procedure TCobraApp.PrintReport (Mode : Integer);

  VAR
    FileName  : PathStr;
    Result    : Boolean;
```

```
  Begin
    Case Mode Of
      mdReport : FileName := GetFileForLoad ('.RPT', 'Print Reports',
        '~S~elect Report...');
      mdHelp   : FileName := GetFileForLoad ('.HLP', 'Print Help',
        '~S~elect Help File...');
      End;
    If FileName <> '' Then
      Begin
        Result := HardCopy (FileName);
        Redraw;
        If Not Result
          Then MessageBox ('Unable to print ' + FileName + ' to '
            + Info.DeviceName, nil, mfError + mfOkButton);
      End;
  End;

{*************************************************************************
 *  Name          : PrintGroup
 *
 *  Purpose       : Print group of report files
 *
 *  Notes         :
 *
 *  Invoked As    : PrintGroup;
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  06Apr92   AML   Creation
 *  28Jul92   AML   Added View Group functionality
 *************************************************************************}

Procedure TCobraApp.PrintGroup (Mode : Integer);

  Procedure OutputReport (FileName : PathStr; OMode : Integer);
    Var
      Result : Boolean;
    Begin
      If OMode = mdView
        Then ViewFile (FileName)
      Else If OMode = mdReport Then
        Begin
          Result := HardCopy (FileName);
          If Not Result
            Then MessageBox ('Unable to print ' + FileName + ' to '
              + Info.DeviceName, nil, mfError + mfOkButton);
        End
    End;
```

```
Const
  RptName : Array [0..20] Of String[12] =
    ('1TIMCOST.RPT', 'APPDET.RPT',    'APPDETFX.RPT', 'APPDETSS.RPT',
     'APPSUM.RPT',   'COBSUM.RPT',    'DELTAS.RPT',   'ERROR.RPT',
     'FINANCE.RPT',  'INPUTDAT.RPT',  'MILCONAS.RPT', 'MISSION.RPT',
     'MOVING.RPT',   'NPV.RPT',       'OTHER.RPT',    'OVERHEAD.RPT',
     'PERSMOVE.RPT', 'PERSONEL.RPT',  'PERSONNE.RPT', 'PERSPERC.RPT',
     'RPMABOS.RPT');
  RptOrder : Array [0..2,0..20] of Integer =
    ((1, 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1),
     (32, 4, 1, 1024, 32, 64, 256, 16, 2, 8, 4, 1, 128, 16, 2, 8, 64, 512,
      256, 128, 512), (5, 13, 0, 10, 16, 17, 19, 4, 1, 3, 2, 11, 18, 15, 12,
      14, 6, 20, 8, 7, 9));

VAR
  BaseRpt  : SearchRec;
  C        : Integer;
  Dialog   : PDialog;
  Dialog2  : PDialog;
  F        : Text;
  FileName : PathStr;
  GroupCol : TwoWords;
  GrpVer   : String;
  GrpWord  : Word;
  Match    : Boolean;
  rect     : TRect;
  Result   : Boolean;
  TempInt  : Integer;
  TempStr  : String;
  TempWord : Word;

Begin
        GroupCol.First := 0;
        GroupCol.Second := 0;

  If Mode = mdReport Then
    FileName := GetFileForLoad ('.GRP', 'Print Group', '~S~elect Group...')
  Else If Mode = mdView Then
    FileName := GetFileForLoad ('.GRP', 'View Group', '~S~elect Group...');
  If (FileName <> '') And (FileName <> '<NONE>') Then
    Begin
      Assign (F, FileName);
      Reset (F);
      Readln (F, GrpVer);
      Readln (F, GroupCol.First, GroupCol.Second);
      Close (F);
      FileName := StripFile (FileName);
      If GrpVer > GroupVersion Then
        Begin
          tempword := MessageBox
            ('Cannot load Group files from newer COBRA versions.',
            nil, mfError + mfOkButton);
```

```
        GroupCol.First  := 0;
        GroupCol.Second := 0;
      End;
    If GrpVer = '1.00' Then
      Begin
        GroupCol.First  := (GroupCol.First  Mod 128)
                         + (GroupCol.First  Div 128) * 256;
        GroupCol.Second := (GroupCol.Second Mod 256)
                         + (GroupCol.Second Div 256) * 512;
        GrpVer := '1.10';
      End;
    If GrpVer = '1.10' Then
      Begin
        GroupCol.First  := (GroupCol.First Mod 128)
                         + (GroupCol.First Div 128) * 256;
        GrpVer := '1.20';
      End;
  End;

If FileName <> '' Then
  Begin
    If FileName = '<NONE>'
      Then FileName := '';
    rect.Assign (17, 2, 63, 20);
    dialog := New (PDialog, Init (rect, 'Reports in Group'));
    rect.Assign(7,2,30,3);
    Dialog^.Insert(New(PStaticText,Init(rect,'Select Desired Reports:')));
    rect.Assign(3,15,15,17);
    Dialog^.Insert(New(PButton, Init(rect, '~S~ave', cmOK, bfNormal)));
    rect.Assign(17,15,28,17);
    If Mode = mdReport
      Then Dialog^.Insert(New(PButton,Init(rect,'~P~rint',cmNO,bfNormal)))
    Else If Mode = mdView
      Then Dialog^.Insert(New(PButton,Init(rect,'~V~iew',cmNO,bfNormal)));
    rect.Assign(30,15,42,17);
    Dialog^.Insert(New(PButton, Init(rect, 'Cancel', cmCancel, bfNormal)));
    rect.Assign (2, 3, 22, 14);
    Dialog^.Insert (New (PCheckBoxes, Init (rect,
      NewSItem ('~A~-' + RptName[0],
      NewSItem ('~B~-' + RptName[1],
      NewSItem ('~C~-' + RptName[2],
      NewSItem ('~D~-' + RptName[3],
      NewSItem ('~E~-' + RptName[4],
      NewSItem ('~F~-' + RptName[5],
      NewSItem ('~G~-' + RptName[6],
      NewSItem ('~H~-' + RptName[7],
      NewSItem ('~I~-' + RptName[8],
      NewSItem ('~J~-' + RptName[9],
      NewSItem ('~K~-' + RptName[10],
      nil)))))))))))))));
    rect.Assign (24, 3, 44, 14);
```

```
Dialog^.Insert (New (PCheckBoxes, Init (rect,
  NewSItem ('~L~-' + RptName[11],
  NewSItem ('~M~-' + RptName[12],
  NewSItem ('~N~-' + RptName[13],
  NewSItem ('~O~-' + RptName[14],
  NewSItem ('~Q~-' + RptName[15],
  NewSItem ('~R~-' + RptName[16],
  NewSItem ('~T~-' + RptName[17],
  NewSItem ('~U~-' + RptName[18],
  NewSItem ('~W~-' + RptName[19],
  NewSItem ('~X~-' + RptName[20],
  NewSItem ('~Y~-(other RPTs)',
  nil)))))))))))))));
TempWord := 0;
If Mode = mdReport
  Then Dialog^.HelpCtx := hcPrintGroup
Else If Mode = mdView
  Then Dialog^.HelpCtx := hcViewGroup;
If TApplication.ValidView(dialog) <> Nil Then
  While (TempWord <> cmCancel) Do Begin
    dialog^.SetData(GroupCol);
    tempword := Desktop^.ExecView(dialog);
    dialog^.GetData(GroupCol);
    If (TempWord = cmNO) Then
      Begin
        For C := 0 To 20 Do
          If ((RptOrder [0,C] = 1) And
              (Odd (GroupCol.First Div RptOrder [1,C]))) Or
             ((RptOrder [0,C] = 2) And
              (Odd (GroupCol.Second Div RptOrder [1,C])))
            Then OutputReport
              (Info.ReportDir + RptName[RptOrder [2,C]], Mode);
        If Odd (GroupCol.Second Div 1024) Then
          Begin
            FindFirst (Info.ReportDir + '*.RPT', 0, BaseRpt);
            While DosError = 0 Do
              Begin
                Match := False;
                For C := 0 To 20 Do
                  Match := Match Or (BaseRpt.Name = RptName[C]);
                If Not Match
                  Then OutputReport (Info.ReportDir+BaseRpt.Name,Mode);
                FindNext (BaseRpt);
              End;
          End;
        Redraw;
      End;
    If (TempWord = cmOK) Then
      Begin
        rect.Assign(20,7,55,18);
        dialog2 := New (PDialog, Init (rect,
          'Save Report Group List'));
```

```
                rect.Assign(20,5,30,7);
                Dialog2^.Insert(New(PButton, Init(rect,
                  '~S~ave', cmOK, bfNormal)));
                rect.Assign(20,7,30,9);
                Dialog2^.Insert(New(PButton, Init(rect,
                  'Cancel', cmCancel, bfNormal)));
                rect.Assign(4,2,24,3);
                Dialog2^.Insert(New(PStaticText,Init(rect,
                  'Enter File Name')));
                rect.Assign(4,3,14,4);
                Dialog2^.Insert(New(PFInputLine,Init
                  (rect,8,DFileNameSet,DString,0, 0.0, 0.0)));
                rect.Assign(14,3,18,4);
                Dialog2^.Insert(New(PStaticText,Init(rect,'.GRP')));
                Dialog2^.HelpCtx := hcSaveWindow;
                If TApplication.ValidView(dialog2) <> Nil Then
                  Begin
                    dialog2^.SetData(FileName);
                    tempword := Desktop^.ExecView(dialog2);
                    If (tempword <> cmCancel) Then
                      Begin
                        dialog2^.GetData(FileName);
                        If (FileName <> '') Then
                          Begin
                            FileName := FExpand (ConCat (FileName, '.Grp'));
                            Assign (F, FileName);
                            ReWrite (F);
                            Writeln (F, GroupVersion);
                            Writeln (F, GroupCol.First, ' ', GroupCol.Second);
                            Close (F);
                            FileName := StripFile (FileName);
                          End;
                      End
                    Else TempWord := cmYES;
                    Dispose(dialog2,Done);
                  End;
              End;
          End;
        Dispose (dialog, Done);
      End;
  End;
```

```
{*************************************************************************
 *  Name          : PrinterSetup
 *
 *  Purpose       : Changes output (printer) device
 *
 *  Notes         : Default value is PRN
 *
 *  Invoked As    : PrinterSetup;
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  01Apr92   AML   Creation
 *  20May92   AML   Added Migrate Window button and Setup file
 *  12Jul92   AML   Added Report Directory
 *************************************************************************}

Procedure TCobraApp.PrinterSetup;

  Var
    dialog          : PDialog;
    event           : TEvent;
    F               : Text;
    rect            : TRect;
    P               : PView;
    tempword        : Word;

  Begin
    rect.Assign (0, 0, 30, 14);
    dialog := New (PDialog, Init (rect, 'Printer Setup'));
    dialog^.Options := dialog^.Options OR ofCentered;
    PutTab;
    rect.Assign (2, 3, 28, 4);
    P := New(PFInputLine,Init(rect,79,DPathSet,DString,0,0.0,0.0));
    Dialog^.Insert (P);
    rect.Assign (1, 2, 29, 3);
    Dialog^.Insert(New(PLabel,Init(rect, 'Enter ~P~rinter Device Name:', P)));
    rect.Assign (2, 5, 28, 6);
    P := New(PFInputLine,Init(rect,79,DPathSet,DString,0,0.0,0.0));
    Dialog^.Insert (P);
    rect.Assign (1, 4, 26, 5);
    Dialog^.Insert(New(PLabel,Init(rect, 'Enter ~R~eport Directory:', P)));
    rect.Assign (2, 6, 29, 7);
    Dialog^.Insert(New(PStaticText,Init(rect, 'PERSMOVE Migrate Diagrams:')));
    rect.Assign (2, 7, 28, 10);
    Dialog^.Insert (New (PRadioButtons, Init (rect,
      NewSItem ('~A~SCII Char Borders',
      NewSItem ('~S~ingle Line Borders',
      NewSItem ('~D~ouble Line Borders',
      nil))))));
    rect.Assign (3, 11, 13, 13);
    Dialog^.Insert(New(PButton, Init(rect, 'O~K', cmOK, bfDefault)));
    rect.Assign (16, 11, 26, 13);
```

```
    Dialog^.Insert(New(PButton, Init(rect, '~C~ancel', cmCancel, bfNormal)));
    Dialog^.HelpCtx := hcPrinterSetup;
    If ValidView (Dialog) <> Nil Then
      Begin
        Dialog^.SetData (Info);
        TempWord := Desktop^.ExecView(Dialog);
        If (TempWord <> cmCancel) Then
          Begin
            Dialog^.GetData (Info);
            If Info.DeviceName = ''
              Then Info.DeviceName := 'PRN'
              Else Info.DeviceName := StripSpace (Info.DeviceName);
            Info.ReportDir := StripSpace (Info.ReportDir);
            If Info.ReportDir <> '' Then
              Begin
                Info.ReportDir := FExpand (Info.ReportDir);
                If Info.ReportDir[Length(Info.ReportDir)] <> '\'
                  Then Info.ReportDir := Concat (Info.ReportDir, '\');
              End;
            Assign (F, 'Cobra.Stp');
            ReWrite (F);
            Writeln (F, SetupVersion);
            Writeln (F, Info.HighASCII, ' ', Info.DeviceName);
            RightJust (F, Info.ReportDir, 79);
            Writeln (F);
            Close (F);
          End;
        Dispose(Dialog, Done);
      End;
  End; {PrinterSetup}

{*************************************************************************
 *  Name          : LoadPrevious
 *
 *  Purpose       : Load previous data set
 *
 *  Notes         :
 *
 *  Invoked As    : LoadPrevious(Mode);
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  27Jan91   GJG   Creation
 *  27Apr92   AML   Used Mode variable to distinguish between CBR and SFF loads
 *************************************************************************}

Procedure TCobraApp.LoadPrevious(Mode:Integer);

  VAR
    Event    : TEvent;
    FileName : PathStr;
```

```
Begin
  If Mode = mdAutoLoad Then
    Begin
      event.What     := evCommand;
      If AutoStart.Execute
        Then event.Command := cmExecute
        Else event.Command := cmAbout;
      PutEvent (event);
    End;
  Case Mode Of
    mdDataFile: FileName := GetFileForLoad ('.CBR', 'Load Data File',
      '~S~elect Data File...');
    mdStdFctrs: FileName := GetFileForLoad ('.SFF', 'Load Standard Factors',
      '~S~elect SF File...');
    mdOldData:  FileName := GetFileForLoad ('.COB', 'Load Old Data File',
      '~S~elect Old Data File...');
    mdAutoLoad: FileName := FExpand (AutoStart.FileName);
    mdNewSF:    Begin
                  If Screen1Data^.SFFile = ''
                    Then Mode := mdNull
                  Else
                    Begin
                      FileName := FExpand (Screen1Data^.SFFile);
                      If (Pos ('.', FileName) = 0)
                        Then FileName := ConCat (FileName, '.SFF');
                      If Not FileExists (FileName)
                        Then Mode := mdNull
                        Else Mode := mdStdFctrs;
                    End;
                End;
    End;
  If FileName = ''
    Then Mode := mdNull;
  If (Mode = mdAutoLoad) Then
    Begin
      If (Pos ('.COB', FileName) > 0)
        Then Mode := mdOldData
        Else Mode := mdDataFile;
    End;
  If (Mode = mdDataFile) Or (Mode = mdOldData)
    Then InitGlobalVariables;
  If (Mode = mdStdFctrs) Or (Mode = mdOldData) Then
    Begin
      InitStandardFactors;
      Screen1Data^.SFFile := '';
    End;
  If (Mode = mdDataFile) Then
    Begin
      If Not InputData (FileName) Then
        Begin
          If AutoStart.ExitCobra Then
            Begin
```

```
            event.What    := evCommand;
            event.Command := cmQuit;
            PutEvent (event);
          End
        Else MessageBox ('Load data failed', nil, mfOKButton + mfError);
      End
    Else DataFileUsed := FileName;
    UpdateDataFile (StatusLine^.Items^.Next^.Next^.Text^);
    If Screen1Data^.SFFile <> '' Then
      Begin
        FileName := Screen1Data^.SFFile;
        UpdateDataFile (StatusLine^.Items^.Next^.Next^.Text^);
        If Pos ('.', FileName) = 0
          Then FileName := Concat (FileName, '.SFF');
        InitStandardFactors;
        If InputSF (FileName) <> True
          Then MessageBox
            ('Load Standard Factors failed', nil, mfOKButton + mfError)
          Else SFFileUsed := FExpand (FileName);
        UpdateDataFile (StatusLine^.Items^.Next^.Next^.Text^);
      End;
  End;
If Mode = mdStdFctrs Then
  Begin
    If Not InputSF (FileName)
      Then MessageBox
        ('Load Standard Factors failed', nil, mfOKButton + mfError)
    Else
      Begin
        SFFileUsed := FileName;
        Screen1Data^.SFFile := FileName;
        UpdateDataFile (StatusLine^.Items^.Next^.Next^.Text^);
      End;
  End;
If Mode = mdOldData Then
  Begin
    If Not InputOld (FileName) Then
      Begin
        If AutoStart.ExitCobra Then
          Begin
            event.What    := evCommand;
            event.Command := cmQuit;
            PutEvent (event);
          End
        Else MessageBox ('Load Old Data File failed', nil,
          mfOKButton + mfError);
      End
    Else
      Begin
        If Not AutoStart.Execute
          Then MessageBox
           ('Don''t forget to save data AND standard factors '
```

```
            + 'in the new format!', nil, mfOKButton);
          DataFileUsed := FileName;
          SFFileUsed := FileName;
          UpdateDataFile (StatusLine^.Items^.Next^.Next^.Text^);
        End;
      End;
    AutoStart.FileName := '';
  End;

{*************************************************************************
 *  Name          : DeleteData
 *
 *  Purpose       : Delete Datafile, Group, or Report
 *
 *  Notes         :
 *
 *  Invoked As    : DeleteData;
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Jan91   GJG   Creation
 *  20May92   AML   Added Group deletion
 *************************************************************************}

Procedure TCobraApp.DeleteData (Mode:Integer);

  VAR
    FileName : PathStr;
    F        : Text;
    TempWord : Word;

  Begin
    TempWord := cmCancel;
    Case Mode Of
      mdDataFile: FileName := GetFileForLoad ('.CBR', 'Delete Data File',
        '~D~elete Data File...');
      mdGroup:    FileName := GetFileForLoad ('.GRP', 'Delete Report Group',
        '~D~elete Group File...');
      mdReport:   FileName := GetFileForLoad ('.RPT', 'Delete Report',
        '~D~elete Report File...');
      End;
    If FileName <> '' Then
      Begin
        PutTab;
        TempWord := Messagebox ('Are you sure you want to delete '
          + Filename + '?', Nil, mfYesNoCancel + mfConfirmation);
      End;
    If TempWord <> cmYes
      Then FileName := '';
    If FileName <> '' Then
      Begin
        Assign(F,FileName);
```

```
        Erase(F);
      End
    End;

{*************************************************************************
 *  Name          : GetFileForLoad
 *
 *  Purpose       : Prompts user for file name to load data or open file
 *
 *  Notes         :
 *
 *  Invoked As    : GetFileForLoad (Ext, WTitle, CLabel)
 *
 *  Return value  : FileName
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  25Jun92   AML   Creation
 *************************************************************************}

Function TCobraApp.GetFileForLoad
  (Ext : String04; WTitle : String20; CLabel : String60) : PathStr;

  Var
    Buttons,
    TempWord : Word;
    D        : PFileDialog;
    FileName,
    TempStr  : PathStr;

  Begin
    FileName := '';
    TempStr := '';
    If (Ext = '.RPT') Then
      Begin
        If DirExists (Info.ReportDir) Then
          Begin
            GetDir (0, TempStr);
            ChDir (Copy (Info.ReportDir, 1, Length (Info.ReportDir) - 1));
          End
        Else
          Begin
            MessageBox ('Report Directory ' + Info.ReportDir + ' not found.',
              Nil, mfError + mfOKButton);
            TempStr := '<ERROR>';
          End;
      End;
    If TempStr <> '<ERROR>' Then
      Begin
        Buttons := fdOKButton;
        If Ext = '.GRP'
          Then Buttons := Buttons + fdClearButton;
```

```
      D := PFileDialog(ValidView(New(PFileDialog, Init('*' + Ext, WTitle,
        CLabel, Buttons, 100))));
      D^.HelpCtx := hcLoadWindow;
    End
  Else
    Begin
      D := Nil;
      TempStr := '';
    End;
  If D <> nil Then
    Begin
      TempWord := Desktop^.ExecView (D);
      If (TempWord <> cmCancel) And (TempWord <> cmFileClear) Then
        Begin
          D^.GetFileName (FileName);
          If FileName <> '' Then
            Begin
              If Pos ('.', FileName) = 0
                Then FileName := Concat (FileName, Ext);
              If Not FileExists (FileName) Then
                Begin
                  Tempword := MessageBox ('File Not Found:  ' + FileName,
                    nil, mfError + mfOkButton);
                  FileName := '';
                End;
            End;
        End
      Else If TempWord = cmFileClear
        Then FileName := '<NONE>';
      Dispose (D, Done);
    End;
  GetFileForLoad := FileName;
  If TempStr <> '' Then
    Begin
      If (TempStr[Length(TempStr)] = '\') And (Length (TempStr) > 3)
        Then TempStr := Copy (TempStr, 1, Length(TempStr) -1);
      ChDir (TempStr);
    End;
End;
```

```
{*********************************************************************
 *  Name          : GetFileForSave
 *
 *  Purpose       : Prompts user to enter file name to save data or stdfctrs,
 *                  or to give a name for a custom base report
 *
 *  Notes         :
 *
 *  Invoked As    : GetFileForSave(FName,Mode)
 *
 *  Return value :
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ------------------------------------------
 *    FName              VAR     initial/entered values for the screen
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  29Jan91   GJG   Creation
 *********************************************************************}

Function TCobraApp.GetFileForSave (VAR FName : PathStr; Mode:Integer) : WORD;

  VAR
    event    : TEvent;
    rect     : TRect;
    dialog   : PDialog;
    tempword : WORD;

  Begin
    Case Mode Of
      mdDataFile: If (DataFileUsed = '(none)')
                    Then FName := ''
                    Else FName := StripFile (DataFileUsed);
      mdStdFctrs: If SFFileUsed = '(none)'
                    Then FName := ''
                    Else FName := StripFile (SFFileUsed);
      End;

    {*********************************************************************
     * Create a dialog box to display data choices in
     *********************************************************************}

    rect.Assign(20,7,55,15);
    If Mode = mdDataFile
      Then dialog := New(PDialog, Init(rect, 'Save Data File'));
    If Mode = mdStdFctrs
      Then dialog := New(PDialog, Init(rect, 'Save Standard Factors'));
    If Mode = mdReport
      Then dialog := New(PDialog, Init(rect, 'Save Base Report'));
```

```
{*****************************************************************
 * Data fields
 *****************************************************************}

rect.Assign(4,2,24,3);
dialog^.Insert(New(PStaticText,Init(rect,'Enter File Name')));
rect.Assign(4,3,14,4);
dialog^.Insert(New(PFInputLine,Init
  (rect,8,DFileNameSet,DString,0, 0.0, 0.0)));
rect.Assign(14,3,18,4);
If Mode = mdDataFile
  Then dialog^.Insert(New(PStaticText,Init(rect,'.CBR')));
If Mode = mdStdFctrs
  Then dialog^.Insert(New(PStaticText,Init(rect,'.SFF')));
If Mode = mdReport
  Then dialog^.Insert(New(PStaticText,Init(rect,'.RPT')));
rect.Assign(20,2,30,4);
dialog^.Insert(New(PButton, Init(rect, '~S~ave', cmOK, bfNormal)));
rect.Assign(20,5,30,7);
dialog^.Insert(New(PButton, Init(rect, 'Cancel', cmCancel, bfNormal)));
dialog^.HelpCtx := hcSaveWindow;
If (Mode = mdReport) Then
  Begin
    rect.Assign(4,5,19,6);
    If FName = 'TOTALS'
      Then Dialog^.Insert(New(PStaticText,Init(rect,'(Totals)')))
      Else Dialog^.Insert(New(PStaticText,Init(rect,Copy(Screen1Data^.Base
        [(Ord(FName[5])-48)*10 + Ord(FName[6])-48].Name,1,15))))
  End;
PutTab;

{*********************************************************************
 * Load the default values, execute the screen, and save the selected
 *   values if user did not cancel the window
 *********************************************************************}

If TApplication.ValidView(dialog) <> NIL THEN
   Begin
      dialog^.SetData(FName);
      tempword := Desktop^.ExecView(dialog);
      If (tempword <> cmCancel) THEN
        Begin
          dialog^.GetData(FName);
        End;
      Dispose(dialog,Done);
   End;

If (tempword = cmOK) THEN
  Case Mode Of
    mdDataFile: GetFileForSave := cmSaveData;
    mdStdFctrs: GetFileForSave := cmSaveSF;
```

```
        mdReport:   Begin
                      If DirExists (Info.ReportDir) Then
                        Begin
                          FName := Info.ReportDir + FName;
                          GetFileForSave := cmMakeReport;
                        End
                      Else
                        Begin
                          Messagebox ('Report Directory ' + Info.ReportDir
                            + ' not found.', Nil, mfError + mfOKButton);
                          GetFileForSave := cmCancel;
                        End
                    End;
        End
    ELSE
      GetFileForSave := tempword;
  End; {GetFileForSave}

{*********************************************************************
 *  Name          : SaveData
 *
 *  Purpose       : Saves the current data set
 *
 *  Notes         :
 *
 *  Invoked As    : SaveData
 *
 *  Return value : status of exit (a cmXXXX value)
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    FName              VAR     initial/entered values for the screen
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  29Jan91   GJG   Creation
 *********************************************************************}

Procedure TCobraApp.SaveData (Mode : Integer);

  VAR
    savetest : Boolean;
    tempword : Word;
    F        : TEXT;
    filename : PathStr;

  Begin
    Tempword := GetFileForSave(FileName, Mode);
    If (FileName <> '') And (tempword <> cmCancel) Then
```

```
      Begin
        If Mode = mdDataFile
          Then FileName := FExpand(Concat(FileName, '.CBR'));
        If Mode = mdStdFctrs
          Then FileName := FExpand(Concat(FileName, '.SFF'));
        TempWord := cmYes;
        If FileExists (FileName)
          Then Tempword := MessageBox ('File ' + FileName
            + ' already exists.  Overwrite?', nil, mfYesNoCancel
            + mfConfirmation);
        If tempword = cmYes Then
          Begin
            If Mode = mdDataFile
              Then SaveTest := OutputData(filename)
            Else If Mode = mdStdFctrs
              Then SaveTest := OutputSF(filename);
            If SaveTest Then
              Begin
                If Mode = mdDataFile Then
                  Begin
                    DataFileUsed := FileName;
                  End
                Else If Mode = mdStdFctrs Then
                  Begin
                    SFFileUsed := FileName;
                    Screen1Data^.SFFile := FileName;
                  End;
                UpdateDataFile (StatusLine^.Items^.Next^.Next^.Text^);
              End
            ELSE tempword := MessageBox
                ('Save data failed', nil, mfOKButton + mfError);
          End;
      End;
  End;

{*************************************************************************
 *  Name          : ScreenOneNine
 *
 *  Purpose       : Handle the first and eighth data entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : ScreenOneNine (Which_Screen)
 *
 *  Return value  : status of exit (a cmXXXX value)
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  15Jan91   ASB   Creation
 *  11Feb92   GJG   Added check to find MaxBases
 *  18May92   AML   Created Screen Eight
 *************************************************************************}
```

```
Function TCobraApp.ScreenOneNine (Which_Screen : Integer) : WORD;

  VAR
    B        : INTEGER;
    event    : TEvent;
    dialog1  : PScr1Dlg;
    dialog9  : PScr9Dlg;
    rect     : TRect;
    tempStr  : Pathstr;
    tempword : WORD;

  Begin
    If Which_Screen = 1
      Then Dialog1 := New (PScr1Dlg,Init)
      Else Dialog9 := New (PScr9Dlg,Init);
    PutTab;
    If Which_Screen = 1
      Then If TApplication.ValidView (dialog1) <> NIL THEN
        Begin
          dialog1^.HelpCtx := hcScreenOne;
          Screen1Data^.SetDateFlag := 0;
          TempStr := Screen1Data^.SFFile;
          dialog1^.SetData (Screen1Data^);
          tempword := Desktop^.ExecView (dialog1);
          If (tempword <> cmCancel) THEN
            Begin
              dialog1^.GetData (Screen1Data^);
              ChangeFlag := True;
            End;
          If Screen1Data^.SetDateFlag = 1
            Then Screen1Data^.DateLine := AsOfDateTime;
          Screen1Data^.ModelDoShutDn := UpCase (Screen1Data^.ModelDoShutDn);
          LastYear := 0;
          For B := 1 To AbsMaxBases Do
            If Screen1Data^.Base[B].Name <> ''
              Then MaxBases := B;
          For B := 0 To MaxBases Do
            Begin
              Screen1Data^.Base[B].Deactivate :=
                UpCase (Screen1Data^.Base[B].Deactivate);
              Screen1Data^.Base[B].State :=
                UpperCase (Screen1Data^.Base[B].State);
              If Screen1Data^.Base[B].CloseYear > LastYear
                Then LastYear := Screen1Data^.Base[B].CloseYear;
              CapCheck (Screen1Data^.Base[B].Name);
            End;
          If LastYear = 0
            Then LastYear := MaxYears;
          Dispose (dialog1,Done);
          If StripSpace(UpperCase(TempStr)) <>
             StripSpace(UpperCase(Screen1Data^.SFFile)) Then
```

```
            Begin
              LoadPrevious(mdNewSF);
              If FExpand(Screen1Data^.SFFile) <> FExpand (SFFileUsed)
                Then Screen1Data^.SFFile := TempStr
                Else MessageBox ('New Standard Factor File '
                  + SFFileUsed + ' has been loaded.', Nil, mfOKButton
                  + mfInformation);
            End;
        End;
    If Which_Screen = 9
      Then If TApplication.ValidView (dialog9) <> NIL THEN
        Begin
          dialog9^.HelpCtx := hcScreenNine;
          dialog9^.SetData (Screen9Data);
          tempword := Desktop^.ExecView (dialog9);
          If (tempword <> cmCancel) THEN
            Begin
              dialog9^.GetData (Screen9Data);
              ChangeFlag := True;
            End;
          Dispose (dialog9,Done);
        End;
    If (tempword = cmOK)
      Then ScreenOneNine := cmNextScr
    ELSE If (tempword = cmYes)
      Then ScreenOneNine := cmPrevScr
    Else ScreenOneNine := tempword;
  End; {ScreenOneNine}

{*************************************************************************
 *  Name          : ScreenTwo
 *
 *  Purpose       : Handle the second data entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : ScreenTwo
 *
 *  Return value  : status of exit (a cmXXXX value)
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Apr92    AML  Creation
 *************************************************************************}

Function TCobraApp.ScreenTwo : WORD;
```

```
VAR
  DataPage : Integer;
  dialog   : PScr2Dlg;
  event    : TEvent;
  recycle  : BOOLEAN;
  tempword : WORD;

Begin
  DataPage := 0;
  Recycle := True;
  While Recycle Do
    Begin
      Dialog := New (PScr2Dlg, Init (DataPage));
      Dialog^.InitScr2Info (DataPage);
      Dialog^.HelpCtx := hcScreenTwo;
      PutTab;
      If TApplication.ValidView (dialog) <> NIL THEN
        Begin
          Dialog^.SetData (Dialog^.Scr2Data);
          Tempword := Desktop^.ExecView (dialog);
          If (tempword <> cmCancel) THEN
            Begin
              dialog^.GetData (dialog^.Scr2Data);
              dialog^.CopyScr2Info (DataPage);
            End
          Else Recycle := False;
          Dispose (dialog,Done);
        End;
      If (tempword = cmOK) Then
        Begin
          TempWord := cmNextScr;
          DataPage := DataPage + 15;
          If DataPage >= ((MaxBases * (MaxBases + 1)) Div 2)
            Then Recycle := False;
        End;
      If (tempword = cmYes) Then
        Begin
          TempWord := cmPrevScr;
          DataPage := DataPage - 15;
          If DataPage < 0
            Then Recycle := False;
        End;
      If (tempword = cmNo)
        Then Recycle := False;
    End;
  ScreenTwo := tempword;
End; {ScreenTwo}
```

```
{*************************************************************************
 *  Name          : ScreenThree
 *
 *  Purpose       : Handle the third data entry screen
 *
 *  Notes         :
 *
 *  Invoked As    : ScreenThree
 *
 *  Return value  : status of exit (a cmXXXX value)
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Jan91   ASB   Creation
 *************************************************************************}

Function TCobraApp.ScreenThree : WORD;

  Procedure Advance(Var Data1, Data2:Integer);
  Var Finished : Boolean;
    Begin
      Finished := False;
      Repeat
        Inc (Data2);
        If Data2 > MaxBases Then
          Begin
            Inc (Data1);
            Data2 := Succ(Data1);
          End;
        If Data1 >= MaxBases Then
          Begin
            Data1 := -1;
            Finished := True;
          End
        Else If Distance^.Base[Data1].Base[Data2] > 0
          Then Finished := True;
      Until Finished;
    End;

  Procedure Reverse(Var Data1, Data2:Integer);
  Var Finished : Boolean;
    Begin
      Finished := False;
      Repeat
        Dec (Data2);
        If Data2 <= Data1 Then
          Begin
            Dec (Data1);
            Data2 := MaxBases;
          End;
        If Data1 < 0
          Then Finished := True
```

```
      Else If Distance^.Base[Data1].Base[Data2] > 0
        Then Finished := True;
    Until Finished;
  End;

VAR
  event    : TEvent;
  rect     : TRect;
  dialog   : PScr3Dlg;
  dialog2  : PDialog;
  tempword : WORD;
  recycle  : BOOLEAN;
  Data1,
  Data2    : Integer;

Begin
  Data1 := 0;
  Data2 := 0;
  Advance(Data1,Data2);
  Recycle := (Data1 > -1);
  If Not Recycle Then
    Begin
      rect.Assign(10,1,61,10);
      dialog2 := New(PDialog, Init(rect, 'Screen Three - Transfer Table'));
      dialog2^.HelpCtx := hcNullDataThree;
      rect.Assign(6,2,49,3);
      dialog2^.Insert(New(PStaticText,Init(rect,
        'No transfer data can be entered until there')));
      rect.Assign(2,3,49,4);
      dialog2^.Insert(New(PStaticText,Init(rect,
        'are bases defined with a distance between them.')));
      rect.Assign (4,5,16,7);
      dialog2^.Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));
      rect.Assign (19,5,31,7);
      dialog2^.Insert(New(PButton, Init(rect,
        '~P~revious', cmYes, bfNormal)));
      rect.Assign (34,5,46,7);
      dialog2^.Insert(New(PButton, Init(rect, '~D~one', cmCancel, bfNormal)));
      PutTab;
      If TApplication.ValidView(Dialog2) <> Nil Then
        Begin
          Tempword := Desktop^.ExecView(Dialog2);
          Dispose(Dialog2,Done);
        End;
      If (tempword = cmOK)
        Then TempWord := cmNextScr;
      If (tempword = cmYes)
        Then TempWord := cmPrevScr;
    End;
```

```
  While Recycle Do
    Begin
      Dialog := New (PScr3Dlg, Init (Data1 * 16 + Data2));
      Dialog^.InitScr3Info (Data1 * 16 + Data2);
      Dialog^.HelpCtx := hcScreenThree;
      PutTab;

      {*****************************************************************
       * Load the default values, execute the screen, and save the
       *   selected values if user did not cancel the window
       *****************************************************************}

      If TApplication.ValidView (dialog) <> NIL THEN
        Begin
          Dialog^.SetData (Dialog^.Scr3Data);
          Tempword := Desktop^.ExecView (dialog);
          If (tempword <> cmCancel) THEN
            Begin
              dialog^.GetData (dialog^.Scr3Data);
              dialog^.CopyScr3Info (Data1 * 16 + Data2);
            End
          Else Recycle := False;
          Dispose (dialog,Done);
        End;

      If (tempword = cmOK) Then
        Begin
          TempWord := cmNextScr;
          Advance(Data1, Data2);
          If Data1 < 0
            Then Recycle := False;
        End;
      If (tempword = cmYes) Then
        Begin
          TempWord := cmPrevScr;
          Reverse(Data1, Data2);
          If Data1 < 0
            Then Recycle := False;
        End;
      If (tempword = cmNo)
        Then Recycle := False;
    End;
  ScreenThree := tempword;
End; {ScreenThree}
```

```
{*********************************************************************
 *  Name          : ScreenFourFiveSixSeven
 *
 *  Purpose       : Handles the fourth, fifth, sixth, seventh data entry screens
 *
 *  Notes         : Were screens six, eight, and seven previous to multibasing
 *                : MilCon screen was originally separated by requirements,
 *                  capacity, and rehab rather than by base -- and in a
 *                  radically different format.
 *
 *  Invoked As    : ScreenFourFiveSixSeven (Which_Screen)
 *
 *  Return value : status of exit (a cmXXXX value)
 *
 *  Date      Init  Modifications
 *  -------   ----  ----------------------------------------------------
 *  29Jan91   ASB   Creation
 *  24Apr92   AML   Redid screens with multiBase info, one screen per base
 *  13May92   AML   Combined screen handlers for four, five, and six
 *********************************************************************}

Function TCobraApp.ScreenFourFiveSixSeven (Which_Screen: Integer) : WORD;

  VAR
    event    : TEvent;
    dialog4  : PScr4Dlg;
    dialog5  : PScr5Dlg;
    dialog6  : PScr6Dlg;
    dialog7  : PScr7Dlg;
    tempword : WORD;
    recycle  : BOOLEAN;
    DataPage : Integer;

  Begin
    DataPage := 0;
    Recycle := True;
    Repeat
      Case Which_Screen Of
        4 : Begin
              Dialog4 := New (PScr4Dlg, Init (DataPage));
              Dialog4^.InitScr4Info (DataPage);
              Dialog4^.HelpCtx := hcScreenFour;
            End;
        5 : Begin
              Dialog5 := New (PScr5Dlg, Init (DataPage));
              Dialog5^.InitScr5Info (DataPage);
              Dialog5^.HelpCtx := hcScreenFive;
            End;
        6 : Begin
              Dialog6 := New (PScr6Dlg, Init (DataPage));
              Dialog6^.InitScr6Info (DataPage);
              Dialog6^.HelpCtx := hcScreenSix;
```

```
      End;
  7 : Begin
        Dialog7 := New (PScr7Dlg, Init (DataPage));
        Dialog7^.InitScr7Info (DataPage);
        Dialog7^.HelpCtx := hcScreenSeven;
      End;
  End;
PutTab;

{*****************************************************************
 * Load the default values, execute the screen, and save the
 *   selected values If user did not cancel the window
 *****************************************************************}

Case Which_Screen Of
  4: If TApplication.ValidView (dialog4) <> NIL THEN
       Begin
         Dialog4^.SetData (Dialog4^.Scr4Data);
         Tempword := Desktop^.ExecView (dialog4);
         If (tempword <> cmCancel) THEN
           Begin
             dialog4^.GetData (dialog4^.Scr4Data);
             dialog4^.CopyScr4Info (DataPage);
           End
         Else Recycle := False;
         Dispose (dialog4,Done);
       End;
  5: If TApplication.ValidView (dialog5) <> NIL THEN
       Begin
         Dialog5^.SetData (Dialog5^.Scr5Data);
         Tempword := Desktop^.ExecView (dialog5);
         If (tempword <> cmCancel) THEN
           Begin
             dialog5^.GetData (dialog5^.Scr5Data);
             dialog5^.CopyScr5Info (DataPage);
           End
         Else Recycle := False;
         Dispose (dialog5,Done);
       End;
  6: If TApplication.ValidView (dialog6) <> NIL THEN
       Begin
         Dialog6^.SetData (Dialog6^.Scr6Data);
         Tempword := Desktop^.ExecView (dialog6);
         If (tempword <> cmCancel) THEN
           Begin
             dialog6^.GetData (dialog6^.Scr6Data);
             dialog6^.CopyScr6Info (DataPage);
           End
         Else Recycle := False;
         Dispose (dialog6,Done);
       End;
```

```
        7: If TApplication.ValidView (dialog7) <> NIL THEN
             Begin
               Dialog7^.SetData (Dialog7^.Scr7Data);
               Tempword := Desktop^.ExecView (dialog7);
               If (tempword <> cmCancel) THEN
                 Begin
                   dialog7^.GetData (dialog7^.Scr7Data);
                   dialog7^.CopyScr7Info (DataPage);
                 End
               Else Recycle := False;
               Dispose (dialog7,Done);
             End;
        End;

      If (tempword = cmOK) Then
        Begin
          TempWord := cmNextScr;
          Inc (DataPage);
          If DataPage > MaxBases
            Then Recycle := False;
        End;
      If (tempword = cmYes) Then
        Begin
          TempWord := cmPrevScr;
          Dec (DataPage);
          If DataPage < 0
            Then Recycle := False;
        End;
      If (tempword = cmNo)
        Then Recycle := False;
    Until Not Recycle;
    ScreenFourFiveSixSeven := tempword;
  End; {ScreenFourFiveSixSeven}

{*************************************************************************
 *  Name          : ScreenEight
 *
 *  Purpose       : Handle the eighth data entry screen
 *
 *  Notes         : Was screen nine previous to multibasing
 *
 *  Invoked As    : ScreenEight
 *
 *  Return value  : status of exit (a cmXXXX value)
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  28Jan91   ASB   Creation
 *************************************************************************}
```

```
Function TCobraApp.ScreenEight : WORD;

  VAR
    B, DataPage : Integer;
    event       : TEvent;
    rect        : TRect;
    dialog      : PScr8Dlg;
    dialog2     : PDialog;
    tempword    : WORD;
    recycle     : BOOLEAN;

  Begin
    DataPage := 0;
    Recycle := False;
    For B := MaxBases DownTo 0 Do
      If IndustFundSwitch.Base[B] = 'Y' Then
        Begin
          Recycle := True;
          DataPage := B;
        End;
    If Not Recycle Then
      Begin
        rect.Assign(10,1,61,10);
        dialog2 := New(PDialog, Init(rect,
          'Screen Eight - Base Info (Unique Activity)'));
        dialog2^.HelpCtx := hcNullDataEight;
        rect.Assign(7,2,49,3);
        dialog2^.Insert(New(PStaticText,Init(rect,
          'There are no bases which are designated as')));
        rect.Assign(2,3,49,4);
        dialog2^.Insert(New(PStaticText,Init(rect,
          'having Unique Activities to describe here.')));
        rect.Assign(4,5,16,7);
        dialog2^.Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));
        rect.Assign (19,5,31,7);
        dialog2^.Insert(New(PButton, Init(rect,
          '~P~revious', cmYes, bfNormal)));
        rect.Assign (34,5,46,7);
        dialog2^.Insert(New(PButton, Init(rect, '~D~one', cmCancel, bfNormal)));
        PutTab;
        If TApplication.ValidView(Dialog2) <> Nil Then
          Begin
            Tempword := Desktop^.ExecView(Dialog2);
            Dispose(Dialog2,Done);
          End;
        If (tempword = cmOK)
          Then TempWord := cmNextScr;
        If (tempword = cmYes)
          Then TempWord := cmPrevScr;
      End
```

```
Else Repeat
  Dialog := New (PScr8Dlg, Init (DataPage));
  Dialog^.InitScr8Info (DataPage);
  Dialog^.HelpCtx := hcScreenEight;
  PutTab;

  {*****************************************************************
   * Load the default values, execute the screen, and save the
   *   selected values If user did not cancel the window
   *****************************************************************}

  If TApplication.ValidView (dialog) <> NIL THEN
    Begin
      Dialog^.SetData (Dialog^.Scr8Data);
      Tempword := Desktop^.ExecView (dialog);
      If (tempword <> cmCancel) THEN
        Begin
          dialog^.GetData (dialog^.Scr8Data);
          dialog^.CopyScr8Info (DataPage);
        End
      Else Recycle := False;
      Dispose (dialog,Done);
    End;

  If (tempword = cmOK) Then
    Begin
      TempWord := cmNextScr;
      Inc (DataPage);
      While (IndustFundSwitch.Base[DataPage] <> 'Y')
        And (DataPage < MaxBases) Do
          Inc (DataPage);
      If IndustFundSwitch.Base[DataPage] <> 'Y'
        Then Inc (DataPage);
      If DataPage > MaxBases
        Then Recycle := False;
    End;
  If (tempword = cmYes) Then
    Begin
      TempWord := cmPrevScr;
      Dec (DataPage);
      While (IndustFundSwitch.Base[DataPage] <> 'Y')
        And (DataPage > 0) Do
          Dec (DataPage);
      If IndustFundSwitch.Base[DataPage] <> 'Y'
        Then Dec (DataPage);
      If DataPage < 0
        Then Recycle := False;
    End;
  If (tempword = cmNo)
    Then Recycle := False;
Until Not Recycle;
ScreenEight := tempword;
```

```
  End; {ScreenEight}

{*********************************************************************
 *  Name          : StandardFactors
 *
 *  Purpose       : Display the 4 standard factors screens
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ----------------------------------------------------
 *  25Jan90   GJG   Creation
 *  03Apr92   AML   Modified (percentages displayed properly)
 *********************************************************************}

Procedure TCobraApp.StandardFactors (factwin  : INTEGER);

  VAR
    fact1dlg : PFact1Dlg;
    fact2dlg : PFact2Dlg;
    fact3dlg : PFact3Dlg;
    fact4dlg : PFact4Dlg;
    factdone : BOOLEAN;
    event    : TEvent;
    P, Y     : Integer;
    tempword : WORD;

  Begin
    factdone := FALSE;
    tempword := cmNextScr;
    Dec (FactWin);
    While factdone <> True Do
      Begin
        If tempword = cmNextScr
          Then Inc (FactWin)
        Else If tempword = cmPrevScr
          Then Dec (FactWin)
        Else If (TempWord = cmCancel) Or (TempWord = cmNo)
          Then factdone := True;
        If Not factdone Then
          Begin
            CASE factwin OF
              1: Begin
                   fact1dlg := New (PFact1Dlg,Init);
                   fact1dlg^.HelpCtx := hcStdFctrsOne;
                   PutTab;
```

```
If TApplication.ValidView (fact1dlg) <> NIL THEN
  Begin
    StdFctrsPers.PercOffMarried :=
      StdFctrsPers.PercOffMarried * 100.0;
    StdFctrsPers.PercEnlMarried :=
      StdFctrsPers.PercEnlMarried * 100.0;
    StdFctrsPers.PercEnlHous :=
      StdFctrsPers.PercEnlHous * 100.0;
    StdFctrsPers.CivTurnover :=
      StdFctrsPers.CivTurnover * 100.0;
    StdFctrsPers.CivEarlyRetire :=
      StdFctrsPers.CivEarlyRetire * 100.0;
    StdFctrsPers.PriPlacementPerc :=
      StdFctrsPers.PriPlacementPerc * 100.0;
    StdFctrsPers.PPSInvolvePCS :=
      StdFctrsPers.PPSInvolvePCS * 100.0;
    StdFctrsPers.CivRIfPayFctr :=
      StdFctrsPers.CivRIfPayFctr * 100.0;
    StdFctrsPers.CivRetirePayFctr :=
      StdFctrsPers.CivRetirePayFctr * 100.0;
    StdFctrsPers.PercCivsQuitting :=
      StdFctrsPers.PercCivsQuitting * 100.0;
    StdFctrsPers.HomeSaleReimburseRate :=
      StdFctrsPers.HomeSaleReimburseRate * 100.0;
    StdFctrsPers.HomePurchReimburseRate :=
      StdFctrsPers.HomePurchReimburseRate * 100.0;
    StdFctrsPers.CivHomeOwnRate :=
      StdFctrsPers.CivHomeOwnRate * 100.0;
    StdFctrsPers.HAPFactor :=
      StdFctrsPers.HAPFactor * 100.0;
    StdFctrsPers.DARSEFactor :=
      StdFctrsPers.DARSEFactor * 100.0;
     StdFctrsPers.HAPReceive :=
      StdFctrsPers.HAPReceive * 100.0;
    StdFctrsPers.DARSEReceive :=
      StdFctrsPers.DARSEReceive * 100.0;
    StdFctrsPers.NatlAvgHomePrice :=
      StdFctrsPers.NatlAvgHomePrice / 1000.0;
    fact1dlg^.SetData (StdFctrsPers);
    tempword := Desktop^.ExecView (fact1dlg);
    IF (tempword <> cmCancel) THEN
      Begin
        fact1dlg^.GetData (StdFctrsPers);
        ChangeFlag := True;
      End;
    StdFctrsPers.NatlAvgHomePrice :=
      StdFctrsPers.NatlAvgHomePrice * 1000.0;
    StdFctrsPers.PercOffMarried :=
      StdFctrsPers.PercOffMarried / 100.0;
    StdFctrsPers.PercEnlMarried :=
      StdFctrsPers.PercEnlMarried / 100.0;
```

```
            StdFctrsPers.PercEnlHous :=
              StdFctrsPers.PercEnlHous / 100.0;
            StdFctrsPers.CivTurnover :=
              StdFctrsPers.CivTurnover / 100.0;
            StdFctrsPers.CivEarlyRetire :=
              StdFctrsPers.CivEarlyRetire / 100.0;
            StdFctrsPers.PriPlacementPerc :=
              StdFctrsPers.PriPlacementPerc / 100.0;
            StdFctrsPers.PPSInvolvePCS :=
              StdFctrsPers.PPSInvolvePCS / 100.0;
            StdFctrsPers.CivRifPayFctr :=
              StdFctrsPers.CivRifPayFctr / 100.0;
            StdFctrsPers.CivRetirePayFctr :=
              StdFctrsPers.CivRetirePayFctr / 100.0;
            StdFctrsPers.PercCivsQuitting :=
              StdFctrsPers.PercCivsQuitting / 100.0;
            StdFctrsPers.HomeSaleReimburseRate :=
              StdFctrsPers.HomeSaleReimburseRate / 100.0;
            StdFctrsPers.HomePurchReimburseRate :=
              StdFctrsPers.HomePurchReimburseRate / 100.0;
            StdFctrsPers.CivHomeOwnRate :=
              StdFctrsPers.CivHomeOwnRate / 100.0;
            StdFctrsPers.HAPFactor
              := StdFctrsPers.HAPFactor / 100.0;
            StdFctrsPers.DARSEFactor
              := StdFctrsPers.DARSEFactor / 100.0;
            StdFctrsPers.HAPReceive
              := StdFctrsPers.HAPReceive / 100.0;
            StdFctrsPers.DARSEReceive
              := StdFctrsPers.DARSEReceive / 100.0;
            Dispose (fact1dlg,Done);
          End;
        IF (tempword = cmOK)
          Then tempword := cmNextScr
      End;
    2: Begin
        fact2dlg      := New (PFact2Dlg,Init);
        fact2dlg^.HelpCtx := hcStdFctrsTwo;
        PutTab;
        IF TApplication.ValidView (fact2dlg) <> NIL THEN
          Begin
            For Y := 1 To 6 Do
              StdFctrsFac.InflationRate.Year[Y] :=
                StdFctrsFac.InflationRate.Year[Y] * 100.0;
            StdFctrsFac.InflationNPV :=
              StdFctrsFac.InflationNPV * 100.0;
            StdFctrsFac.DiscountNPV :=
              StdFctrsFac.DiscountNPV * 100.0;
            StdFctrsFac.SuppToMoveFctr :=
              StdFctrsFac.SuppToMoveFctr * 100.0;
            StdFctrsFac.CaretakerPercOrigRPMACost :=
              StdFctrsFac.CaretakerPercOrigRPMACost * 100.0;
```

```
            StdFctrsFac.RehabCostVsNewConstr :=
              StdFctrsFac.RehabCostVsNewConstr * 100.0;
            StdFctrsFac.IMAPerc := StdFctrsFac.IMAPerc * 100.0;
            StdFctrsFac.Design := StdFctrsFac.Design * 100.0;
            StdFctrsFac.SIOH := StdFctrsFac.SIOH * 100.0;
            StdFctrsFac.Contingency :=
              StdFctrsFac.Contingency * 100.0;
            StdFctrsFac.SitePrep := StdFctrsFac.SitePrep * 100.0;
            fact2dlg^.SetData (StdFctrsFac);
            tempword := Desktop^.ExecView (fact2dlg);
            If (tempword <> cmCancel) Then
              Begin
                fact2dlg^.GetData (StdFctrsFac);
                ChangeFlag := True;
              End;
            For Y := 1 To 6 Do
              StdFctrsFac.InflationRate.Year[Y] :=
                StdFctrsFac.InflationRate.Year[Y] / 100.0;
            StdFctrsFac.InflationNPV :=
              StdFctrsFac.InflationNPV / 100.0;
            StdFctrsFac.DiscountNPV :=
              StdFctrsFac.DiscountNPV / 100.0;
            StdFctrsFac.SuppToMoveFctr :=
              StdFctrsFac.SuppToMoveFctr / 100.0;
            StdFctrsFac.CaretakerPercOrigRPMACost :=
              StdFctrsFac.CaretakerPercOrigRPMACost / 100.0;
            StdFctrsFac.RehabCostVsNewConstr :=
              StdFctrsFac.RehabCostVsNewConstr / 100.0;
            StdFctrsFac.IMAPerc := StdFctrsFac.IMAPerc / 100.0;
            StdFctrsFac.Design := StdFctrsFac.Design / 100.0;
            StdFctrsFac.SIOH := StdFctrsFac.SIOH / 100.0;
            StdFctrsFac.Contingency :=
              StdFctrsFac.Contingency / 100.0;
            StdFctrsFac.SitePrep := StdFctrsFac.SitePrep / 100.0;
            Dispose (fact2dlg,Done);
          End;
        IF (tempword = cmOK)
          Then tempword := cmNextScr
        Else IF (tempword = cmYes)
          Then tempword := cmPrevScr
      End;
  3: Begin
        fact3dlg := New (PFact3Dlg,Init);
        fact3dlg^.HelpCtx := hcStdFctrsThree;
        PutTab;
        IF TApplication.ValidView (fact3dlg) <> NIL THEN
          Begin
            StdFctrsTrans.ShippingLossRate :=
              StdFctrsTrans.ShippingLossRate * 100.0;
            StdFctrsTrans.HHGCost := StdFctrsTrans.HHGCost * 100.0;
            fact3dlg^.SetData (StdFctrsTrans);
            tempword := Desktop^.ExecView (fact3dlg);
```

```
                    IF (tempword <> cmCancel) Then
                      Begin
                        fact3dlg^.GetData (StdFctrsTrans);
                        ChangeFlag := True;
                      End;
                    StdFctrsTrans.ShippingLossRate :=
                      StdFctrsTrans.ShippingLossRate / 100.0;
                    StdFctrsTrans.HHGCost := StdFctrsTrans.HHGCost / 100.0;
                    Dispose (fact3dlg,Done);
                  End;
                IF (tempword = cmOK)
                  Then tempword := cmNextScr
                ELSE IF (tempword = cmYes)
                  Then tempword := cmPrevScr
              End;
            4: Begin
                fact4dlg := New (PFact4Dlg,Init);
                fact4dlg^.HelpCtx := hcStdFctrsFour;
                PutTab;
                IF TApplication.ValidView (fact4dlg) <> NIL THEN
                  Begin
                    fact4dlg^.SetData (StdFctrsCon);
                    tempword := Desktop^.ExecView (fact4dlg);
                    IF (tempword <> cmCancel) Then
                      Begin
                        Fact4dlg^.GetData (StdFctrsCon);
                        ChangeFlag := True;
                      End;
                    For P := 0 To 5 Do
                      StdFctrsCon.Units[P] :=
                        UpperCase (StdFctrsCon.Units[P]);
                    Dispose (fact4dlg, Done);
                  End;
                IF (tempword = cmYes)
                  Then tempword := cmPrevScr
              End;
            End; { Case }
        End;
    End; { While }
End;  { StandardFactors }
```

```
{*************************************************************************
 *  Name          : TCobraApp.InitMenuBar
 *
 *  Purpose       : Initialize the top-line menu bar
 *
 *  Notes         :
 *
 *  Invoked As    : InitMenuBar;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------------
 *  11Jan91   ASB   Creation
 *************************************************************************}

Procedure TCobraApp.InitMenuBar;

  VAR
    R : TRect;

  Begin
    GetExtent (R);
    R.B.Y := R.A.Y + 1;
    MenuBar := New (PMenuBar, Init (R,
      NewMenu (
        NewSubMenu ('~H~elp', hcHelpMenu,
          NewMenu (
            NewItem ('~A~bout COBRA', '', kbNoKey, cmAbout, hcAboutCobra,
            NewLine (
            NewItem ('~V~iew Help', 'ALT-F1', kbAltF1, cmViewHelp, hcViewHelp,
            NewItem ('~P~rint Help', '', kbNoKey, cmPrintHelp, hcPrintHelp,
            NewItem ('~F~iles Used', '', kbNoKey, cmViewUsed, hcFilesUsed,
            NewLine (
            NewItem ('~C~alculator', 'ALT-C', kbAltC, cmCalculator,
              hcCalculator,
            NewItem ('Calen~D~ar', '', kbNoKey, cmCalendar, hcCalendar,
            nil)))))))),
        NewSubMenu ('~F~ile', hcFileMenu,
          NewMenu (
            NewItem ('~L~oad Data File', 'ALT-L', kbAltL, cmLoadPrev,
              hcLoadPrevious,
            NewItem ('~S~ave Data File', 'ALT-S', kbAltS, cmSaveData,
              hcSaveCurrent,
            NewItem ('~C~lear Data', '', kbNoKey, cmNewData,
              hcCreateNew,
            NewItem ('~D~elete Data File', '', kbNoKey, cmDeleteData,
              hcDeleteData,
            NewLine (
            NewItem ('L~O~ad Standard Factors', '', kbNoKey, cmLoadSF, hcLoadSF,
            NewItem ('Sa~V~e Standard Factors', '', kbNoKey, cmSaveSF, hcSaveSF,
            NewItem ('Clea~R~ Standard Factors', '', kbNoKey, cmNewSF, hcNewSF,
```

```
      NewLine (
      NewItem ('~F~ile Directory', 'F2', kbF2, cmCobraFiles,
        hcCobraFiles,
      NewLine (
      NewItem ('E~X~it COBRA', 'ALT-X', kbAltX, cmExitCobra, hcExitCobra,
      nil))))))))))))),
  NewSubMenu ('~D~ataBase', hcDBaseMenu,
    NewMenu (
      NewItem ('~L~oad Base(s)', '', kbNoKey, cmLoadDB, hcLoadDB,
      NewItem ('~S~ave Base(s)', '', kbNoKey, cmSaveDB, hcSaveDB,
      NewItem ('~D~istances', '', kbNoKey, cmDistanceDB, hcDistanceDB,
      nil)))),
  NewSubMenu ('~I~nput Data', hcEnterData,
    NewMenu (
      NewItem ('~1~ - General Scenario', '', kbAlt1, cmScreenOne,
        hcEnterData,
      NewItem ('~2~ - Distance Table', '', kbAlt2, cmScreenTwo,
        hcEnterData,
      NewItem ('~3~ - Movement Table', '', kbAlt3, cmScreenThree,
        hcEnterData,
      NewItem ('~4~ - Base Info (Static)', '', kbAlt4, cmScreenFour,
        hcEnterData,
      NewItem ('~5~ - Base Info (Dynamic)', '', kbAlt5, cmScreenFive,
        hcEnterData,
      NewItem ('~6~ - Base Info (Personnel)', '', kbAlt6, cmScreenSix,
        hcEnterData,
      NewItem ('~7~ - Base Info (MilCon)', '', kbAlt7, cmScreenSeven,
        hcEnterData,
      NewItem ('~8~ - Base Info (Unique)', '', kbAlt8, cmScreenEight,
        hcEnterData,
      NewItem ('~9~ - Explanatory Notes', '', kbAlt9, cmScreenNine,
        hcEnterData,
      NewLine (
      NewItem ('~P~ersonnel Factors', '', kbAlt0, cmStdFactors1,
        hcEnterData,
      NewItem ('~F~acility Factors', '', kbNoKey, cmStdFactors2,
        hcEnterData,
      NewItem ('~T~ransportation Factors', '', kbNoKey, cmStdFactors3,
        hcEnterData,
      NewItem ('~C~onstruction Factors', '', kbNoKey, cmStdFactors4,
        hcEnterData,
      NewLine (
      NewItem ('~D~elete Base(s)', '', kbNoKey, cmDeleteBase,
        hcDeleteBase,
      nil)))))))))))))))),
  NewSubMenu ('~R~eports', hcReportMenu,
    NewMenu (
      NewItem ('~E~xecute', 'ALT-E', kbAltE, cmExecute, hcExecute,
      NewItem ('~B~ase Report(s)', '', kbNoKey,cmMakeReport, hcMakeReport,
      NewLine (
      NewItem ('~V~iew Report', 'ALT-V', kbAltV, cmViewReport,
        hcViewReport,
```

```
            NewItem ('~P~rint Report', 'ALT-P', kbAltP, cmPrintReport,
              hcPrintReport,
            NewItem ('Delete ~R~eport', '', kbNoKey, cmDeleteReport,
              hcDeleteReport,
            NewLine (
            NewItem ('Vie~W~ Group', '', kbNoKey, cmViewGroup, hcViewGroup,
            NewItem ('Print ~G~roup', 'ALT-G', kbAltG, cmPrintGroup,
              hcPrintGroup,
            NewItem ('~D~elete Group', '', kbNoKey, cmDeleteGroup,
              hcDeleteGroup,
            nil)))))))))))),
        NewSubMenu ('~U~tilities', hcUtilityMenu,
          NewMenu (
            NewItem ('~D~OS Shell', '', kbNoKey, cmDosShell, hcShellCD,
            NewItem ('~C~hange Dir', '', kbNoKey, cmChangeDir, hcShellCD,
            NewItem ('~P~rinter Setup', '', kbNoKey, cmPrinterSetup,
              hcPrinterSetUp,
            NewItem ('~L~oad Old Data', '', kbNoKey, cmLoadOld, hcLoadOld,
            nil))))),
        NewSubMenu ('~W~indows', hcWindowMenu,
          NewMenu (
            NewItem ('~S~ize/Move', '^F5', kbNoKey, cmResize, hcSizeMove,
            NewItem ('~Z~oom', 'F5', kbNoKey, cmZoom, hcZoom,
            NewItem ('~N~ext', 'F6', kbNoKey, cmNext, hcNext,
            NewLine (
            NewItem ('~T~ile', '^F7', kbNoKey, cmTile, hcCascadeTile,
            NewItem ('~C~ascade', 'F7', kbNoKey, cmCascade, hcCascadeTile,
            nil)))))))),
      nil)))))))))))
  End; {*InitMenuBar*}

{*************************************************************************
 *  Name          : TCobraApp.InitStatusLine
 *
 *  Purpose       : Initialize the bottom line "menu" on the screen
 *
 *  Notes         :
 *
 *  Invoked As    : InitStatusLine;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  11Jan91   ASB   Creation
 *************************************************************************}

Procedure TCobraApp.InitStatusLine;

  VAR
    R   : TRect;
    Ver : String60;
```

```
  Begin
{$IFDEF PROTO}
    Ver := 'COBRA ' + VERSION + 'P, R&K Engineering, Inc., 1992';
{$ELSE}
    Ver := 'COBRA v' + VERSION + ', R&K Engineering, Inc., 1992';
{$ENDIF}
    GetExtent(R);
    R.A.Y := R.B.Y - 1;
    StatusLine := New (PStatusLine, Init (R, NewStatusDef (0, $FFFF,
      NewStatusKey ('~ESC~-Close window', kbEsc,       cmClose,
      NewStatusKey ('~F1~-Help',          kbF1,        cmHelp,
      NewStatusKey (' (none)  ',          kbAltEqual,  cmViewUsed,
      NewStatusKey (Ver,                  kbAltSpace,  cmAbout,
      NewStatusKey ('',                   kbF3,        cmRedraw,
      NewStatusKey ('',                   kbF5,        cmZoom,
      NewStatusKey ('',                   kbCtrlF5,    cmResize,
      NewStatusKey ('',                   kbF6,        cmNext,
      NewStatusKey ('',                   kbShiftF6,   cmPrev,
      NewStatusKey ('',                   kbF7,        cmCascade,
      NewStatusKey ('',                   kbCtrlF7,    cmTile,
      NewStatusKey ('',                   kbF9,        cmVideoToggle,
      NewStatusKey ('',                   kbF10,       cmMenu,
      nil)))))))))))),
      nil)));
  End;

{*************************************************************************
 *  Name          : TCobraApp.Idle
 *
 *  Purpose       : Update the system clock
 *
 *  Notes         :
 *
 *  Invoked As    : Idle;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  11Jan91   ASB   Creation
 *************************************************************************}

Procedure TCobraApp.Idle;

Function IsTileable(P: PView): Boolean; far;
Begin
  IsTileable := P^.Options and ofTileable <> 0;
end;

  Var
    Event : Tevent;
```

```
  Begin
    TApplication.Idle;
    Clock^.Update;
{$IFDEF DEBUG}
    Heap^.Update;
{$ENDIF}
    if Desktop^.FirstThat(@IsTileable) <> nil
      Then EnableCommands([cmTile, cmCascade])
      Else DisableCommands([cmTile, cmCascade]);
    if StatusLine^.Items^.Next^.Next^.Text^ <> ' (none)  '
      Then EnableCommands([cmViewUsed])
      Else DisableCommands([cmViewUsed]);
    If Screen1Data^.Base[0].Name <> ''
      Then EnableCommands([cmScreenFour, cmScreenFive, cmScreenSix,
        cmScreenSeven, cmScreenEight, cmExecute, cmDeleteBase, cmMakeReport,
        cmSaveData, cmSaveDB])
      Else DisableCommands([cmScreenFour, cmScreenFive, cmScreenSix,
        cmScreenSeven, cmScreenEight, cmExecute, cmDeleteBase, cmMakeReport,
        cmSaveData, cmSaveDB]);
    If MaxBases > 0
      Then EnableCommands ([cmScreenTwo, cmScreenThree, cmDistanceDB])
      Else DisableCommands ([cmScreenTwo, cmScreenThree, cmDistanceDB]);
    If MaxBases < AbsMaxBases
      Then EnableCommands([cmLoadDB])
      Else DisableCommands([cmLoadDB]);
  End; (*TCobraApp.Idle*)

{*************************************************************************
 *  Name          : TCobraApp.OutOfMemory
 *
 *  Purpose       : Display out of memory message
 *
 *  Notes         :
 *
 *  Invoked As    : OutOfMemory;
 *
 *  Return value  : -none-
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  11Jan91   ASB   Creation
 *************************************************************************}

Procedure TCobraApp.OutOfMemory;

  VAR
    C : Word;

  Begin
    C := MessageBox ('Not enough memory available to complete operation.',
      nil, mfError + mfOkButton);
  End;
```

```
{*************************************************************************
 *  Name          : TCobraApp.PutTab
 *
 *  Purpose       : Causes TurboVision to act as though <TAB> had been pressed.
 *
 *  Notes         : Used to change where the focus is when a window opens
 *
 *  Invoked As    : PutTab;
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------------
 *  01Jul92   AML   Creation
 *************************************************************************}

Procedure TCobraApp.PutTab;

  Var
    Event : TEvent;

  Begin
    Event.What    := evKeyDown;
    Event.KeyCode := kbTab;
    PutEvent (Event);
  End;

{*************************************************************************
 * MAIN PROGRAM
 *************************************************************************}

VAR
  AutoError   : Boolean;
  Counter     : Integer;
  ThisProgram : TCobraApp;

Begin
  AutoError := False;
  AutoStart.Execute := False;
  AutoStart.ExitCOBRA := False;
  AutoStart.FileName := '';
  AutoStart.ReportDir := '';
  For Counter := 1 To 2 Do
    If (Counter <= ParamCount) And
       (Length (Paramstr(Counter)) > 3) Then
      Begin
        If ((Copy(ParamStr(Counter),1,3) = '/L=') Or
            (Copy(ParamStr(Counter),1,3) = '/l=') Or
            (Copy(ParamStr(Counter),1,3) = '/E=') Or
            (Copy(ParamStr(Counter),1,3) = '/e=') Or
            (Copy(ParamStr(Counter),1,3) = '/X=') Or
            (Copy(ParamStr(Counter),1,3) = '/x=')) And
            (AutoStart.FileName = '')
```

```
          Then AutoStart.FileName :=
            Copy(ParamStr(Counter),4,Length(ParamStr(Counter))-3)
        Else If ((Copy(ParamStr(Counter),1,3) = '/D=') Or
            (Copy(ParamStr(Counter),1,3) = '/d=')) And
            (AutoStart.ReportDir = '')
          Then AutoStart.ReportDir :=
            Copy(ParamStr(Counter),4,Length(ParamStr(Counter))-3)
        Else AutoError := True;
        If (Copy(ParamStr(Counter),1,3) = '/E=') Or
           (Copy(ParamStr(Counter),1,3) = '/e=') Or
           (Copy(ParamStr(Counter),1,3) = '/X=') Or
           (Copy(ParamStr(Counter),1,3) = '/x=')
          Then AutoStart.Execute := True;
        If (Copy(ParamStr(Counter),1,3) = '/X=') Or
           (Copy(ParamStr(Counter),1,3) = '/x=')
          Then AutoStart.ExitCobra := True;
      End
  Else If (Counter <= ParamCount) And
          (Length (Paramstr(Counter)) < 4)
    Then AutoError := True;
  If ParamCount > 2
    Then AutoError := True;
  If AutoError Then
    Begin
      Writeln;
      Writeln ('COBRA Version ', Version, ' - R&K Engineering, Inc., 1992');
      Writeln;
      Writeln ('Enter ''COBRA'' by itself to run COBRA.');
      Writeln
        ('Adding ''/L=filename'' will automatically load a data file, or');
      Writeln
        ('adding ''/E=filename'' will automatically execute a data file, or');
      Writeln
        ('adding ''/X=filename'' will execute a data file and exit to DOS.');
      Writeln
        ('Also, ''/D=directory'' will specify a directory for all reports.');
    End
  Else
    Begin
      If (AutoStart.FileName <> '') And (Pos('.', AutoStart.FileName) = 0)
        Then AutoStart.FileName := Concat (AutoStart.FileName, '.CBR');
      ThisProgram.Init;
      ThisProgram.Run;
      ThisProgram.Done;
    End
End.
```

```
UNIT CRPTIN;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE EnteredDataReport;

IMPLEMENTATION

{*****************************************************************************
 *   Name           : EnteredDataReport
 *
 *   Purpose        : Writes a report of the entered data to a file
 *
 *   Notes          :
 *
 *   Invoked As     : WriteEnteredDataReport
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     INPUTDAT.RPT           Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *****************************************************************************}

PROCEDURE EnteredDataReport;

  VAR
    Buffer : IOBuffer;
    F      : Text;
    One,
    OneK   : Year1ToMaxRealRec;
    Page,
    Y      : Integer;

  PROCEDURE DataInputScreen1;
```

```
VAR
  B, B2, C, P      : INTEGER;
  UserEntryError   : Boolean;
  UserEntryError2  : Word;

Begin
  If (Screen1Data^.Service = '')
    Then ErrorOut ('No Service was specified for this scenario.')
  Else If
      (Pos ('ARM', UpperCase(Screen1Data^.Service)) = 0) And

      (Pos ('NAV', UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('USN', UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('S.N', UpperCase(Screen1Data^.Service)) = 0) And

      (Pos ('AIR', UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('AF',  UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('A.F', UpperCase(Screen1Data^.Service)) = 0) And

      (Pos ('MAR', UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('MC',  UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('M.C', UpperCase(Screen1Data^.Service)) = 0) And

      (Pos ('DLA', UpperCase(Screen1Data^.Service)) = 0) And
      (Pos ('LOG', UpperCase(Screen1Data^.Service)) = 0)
    Then ErrorOut ('"' + Screen1Data^.Service
                 + '" is not a recognized Service.');
  If (Screen1Data^.OptionPkgName = '')
    Then ErrorOut ('No Option Package Name was given for this scenario.');
  If FileExists (Info.ReportDir + 'ERROR.RPT')
    Then ErrorOut ('');
  Title (F,'INPUT SCREEN ONE - GENERAL SCENARIO', -1);
  writeln (F, Blank:10, 'Model Year One : FY',  Screen1Data^.FY1:5);
  writeln (F);
  write (F, Blank:10, 'Model does Time-Phasing of Construction/Shutdown:');
  If Screen1Data^.ModelDoShutDn = 'Y'
    Then Writeln (F, '  Yes')
    Else Writeln (F, '  No');
  writeln (F);
  writeln (F, Blank:10, 'Base Name', 'Strategy:':26);
  writeln (F, Blank:10, '---------', '---------':26);
  UserEntryError2 := 0;
  FOR B := 0 to MaxBases DO
    Begin
      write (F, Blank:10, BaseNState(B),
        Blank:(26 - Length (BaseNState(B))));
      If Screen1Data^.Base[B].CloseYear = 0
        Then Writeln (F, 'Realignment')
      Else If Screen1Data^.Base[B].Deactivate = 'Y'
        Then Writeln (F, 'Deactivates in ',
          FiscalYear (Screen1Data^.Base[B].CloseYear))
```

```
        Else Writeln (F, 'Closes in ',
          FiscalYear (Screen1Data^.Base[B].CloseYear));
      If (Screen1Data^.Base[B].Deactivate = 'Y') And
         (Screen1Data^.Base[B].CloseYear = 0)
        Then UserEntryError2 := UserEntryError2 + Round(YtoX(2,B));
    END;
  C := 0;
  For P := 1 To 6 Do
    C := C + Length (Screen1Data^.Summary[P]);
  If C > 0 Then
    Begin
      writeln (F);
      Writeln (F, Blank:10, 'Summary:');
      For P := 1 To 6 Do
        Writeln (F, Blank:10, Screen1Data^.Summary[P]);
      End;
  UserEntryError := False;
  For B := 0 To MaxBases Do
    If Screen1Data^.Base[B].Name = ''
      Then UserEntryError := True;
  If UserEntryError Then
    Begin
      UserEntryError := False;
      ErrorOut ('One or more bases has no name.');
      ErrorOut ('');
    End;
  If MaxBases > 0
    Then For B := 0 To MaxBases - 1 Do
      For B2 := B + 1 To MaxBases Do
        If UpperCase(StripSpace(BaseNState(B))) =
           UpperCase(StripSpace(BaseNState(B2))) Then
          Begin
            UserEntryError := True;
            ErrorOut ('There are two bases with the same name:  ');
            ErrorOut ('     '
              + StripSpace(BaseNState(B)) + ' and '
              + StripSpace(BaseNState(B2)));
          End;
  If UserEntryError
    Then ErrorOut ('');
  If UserEntryError2 > 0 Then
    Begin
      ErrorOut ('The following base(s) deactivate with no year specified:');
      ErrorList (UserEntryError2);
    End;
  If Odd (Screen9Data.Pages) Then
    Begin
      Writeln (F);
      Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
    End;
END;  { DataInputScreen1 }
```

```
PROCEDURE DataInputScreen2;

  VAR
    B, B2,
    Line    : INTEGER;

  Begin { DataInputScreen2 }
    Title (F, 'INPUT SCREEN TWO - DISTANCE TABLE', 2);
    Writeln (F);
    Writeln (F, Blank:10, 'From Base:',
                Blank:15, 'To Base:',
                Blank:18, 'Distance:');
    Writeln (F, Rept('-',10):20, Rept('-',8):23, Rept('-',9):27);
    Line := 0;
    For B := 0 To MaxBases - 1 Do
      For B2 := B + 1 To MaxBases Do
        Begin
          Inc (Line);
          If Line = 48 Then
            Begin
              Line := 1;
              If Odd (Screen9Data.Pages Div 2) Then
                Begin
                  Writeln (F);
                  Writeln (F, Blank:10,
                    '(See final page for Explanatory Notes)');
                End;
              Title (F, 'INPUT SCREEN TWO - DISTANCE TABLE', Page);
              Inc (Page);
              Writeln (F);
              Writeln (F, Blank:10, 'From Base:',
                          Blank:15, 'To Base:',
                          Blank:18, 'Distance:');
              Writeln (F, Rept('-',10):20, Rept('-',8):23, Rept('-',9):27);
            End;
          Write (F, Blank:10, BaseNState(B),
            Blank:(25 - Length(BaseNState(B))),
            BaseNState(B2),
            Blank:(25 - Length(BaseNState(B2))));
          If Distance^.Base[B].Base[B2] > 0
            Then Writeln (F,
              WithCommas (Distance^.Base[B].Base[B2],1):7, ' mi')
            Else Writeln (F, ' not given');
        End;
    If Odd (Screen9Data.Pages Div 2) Then
      Begin
        Writeln (F);
        Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
      End;
  END; { DataInputScreen2 }
```

```
PROCEDURE DataInputScreen3;

  VAR
    BaseFrom, BaseTo, B, B2, L, Y : INTEGER;

  Begin { DataInputScreen3 }
    For B := 0 To MaxBases - 1 Do
      For B2 := B + 1 To MaxBases Do
        If Distance^.Base[B].Base[B2] > 0 Then
          Begin
            Title (F, 'INPUT SCREEN THREE - MOVEMENT TABLE', Page);
            Inc (Page);
            For L := 0 To 1 Do
              Begin
                Case L Of
                  0: Begin
                       BaseTo := B2;
                       BaseFrom := B;
                     End;
                  1: Begin
                       BaseTo := B;
                       BaseFrom := B2;
                     End;
                  End;
                writeln (F);
                Writeln (F, Blank:10, 'Transfers from ',
                  BaseNState(BaseFrom), ' to ', BaseNState(BaseTo));
                Writeln (F);
                FiscOut (F, 28, 7, 0);
                Write (F, Blank:28);
                For Y := 0 To MaxYears - 1 Do
                  Write(F, '   ----');
                Writeln(F);
                Write (F, Blank:10, 'Officers:          ');
                For Y := 1 To MaxYears Do
                  Write (F, WithCommas
                    (PosTrans.Base[BaseFrom].Base[BaseTo]^.Off.Year[Y],0):7);
                Writeln (F);
                Write (F, Blank:10, 'Enlisted:          ');
                For Y := 1 To MaxYears Do
                  Write (F, WithCommas
                    (PosTrans.Base[BaseFrom].Base[BaseTo]^.Enl.Year[Y],0):7);
                Writeln (F);
                Write (F, Blank:10, 'Civilians:         ');
                For Y := 1 To MaxYears Do
                  Write (F, WithCommas
                    (PosTrans.Base[BaseFrom].Base[BaseTo]^.Civ.Year[Y],0):7);
                Writeln (F);
                Write (F, Blank:10, 'Students:          ');
                For Y := 1 To MaxYears Do
                  Write (F, WithCommas
                    (PosTrans.Base[BaseFrom].Base[BaseTo]^.Stu.Year[Y],0):7);
```

```
                    Writeln (F);
                    YearOut (F, 'Missn Eqpt (tons):',
                      ActionEnvir^.Base[BaseFrom].Base[BaseTo]^.MoveMissionEqpmt,
                      One, 7, 0);
                    YearOut (F, 'Suppt Eqpt (tons):',
                      ActionEnvir^.Base[BaseFrom].Base[BaseTo]^.MoveSupportEqpmt,
                      One, 7, 0);
                    If (Pos ('ARM', UpperCase(Screen1Data^.Service)) = 0) Then
                      Begin
                        YearOut (F, 'Mil Light Vehic:  ',
                          ActionEnvir^.Base[BaseFrom].Base[BaseTo]^.MilLightVehic,
                          One, 7, 0);
                        YearOut (F, 'Heavy/Spec Vehic: ',

ActionEnvir^.Base[BaseFrom].Base[BaseTo]^.HeavyOrSpecVehic,
                          One, 7, 0);
                      End
                    Else
                      Begin
                        YearOut (F, 'Mil Lt Vehi(tons):',
                          ActionEnvir^.Base[BaseFrom].Base[BaseTo]^.MilLightVehic,
                          One, 7, 0);
                        YearOut (F, 'Hvy/Spc Veh(tons):',

ActionEnvir^.Base[BaseFrom].Base[BaseTo]^.HeavyOrSpecVehic,
                          One, 7, 0);
                      End
                  End;
                If Odd (Screen9Data.Pages Div 4) Then
                  Begin
                    Writeln (F);
                    Writeln (F, Blank:10,
                      '(See final page for Explanatory Notes)');
                  End;
              End;
    END;  { DataInputScreen3 }

  PROCEDURE DataInputScreen4;

    VAR
      B : Integer;

    Begin { DataInputScreen4 }
      For B := 0 To MaxBases Do
        Begin
          Title (F,'INPUT SCREEN FOUR - STATIC BASE INFO', Page);
          Inc (Page);
          Writeln (F);
          Writeln (F, Blank:10, 'Name:  ', BaseNState(B));
          Writeln (F);
          Write (F, Blank:10, 'Homeowner Assistance Program:  ');
```

```
If HapSwitch.Base[B] = 'Y'
  Then Writeln (F, 'Yes')
  Else Writeln (F, 'No');
Write (F, Blank:10, 'Unique Activity Information:   ');
If IndustFundSwitch.Base[B] = 'Y'
  Then Writeln (F, 'Yes')
  Else Writeln (F, 'No');
Writeln (F);
Writeln (F,Blank:10,'Total Officer Employees:',
  WithCommas(ActionEnvir^.Base[B].BaseTotOffEmploy,0):35);
Writeln (F,Blank:10,'Total Enlisted Employees:',
  WithCommas(ActionEnvir^.Base[B].BaseTotEnlEmploy,0):34);
Writeln (F,Blank:10,'Total Student Employees:',
  WithCommas(ActionEnvir^.Base[B].BaseTotStuEmploy,0):35);
Writeln (F,Blank:10,'Percent of Military Families Living On Base:',
  (ActionEnvir^.Base[B].PercFamLivOnBase*100.0):15:1, '%');
Writeln (F,Blank:10,'Total Civilian Employees:',
  WithCommas(ActionEnvir^.Base[B].BaseTotCivEmploy,0):34);
Writeln (F,Blank:10,'Percent of Civilians Not Willing To Move:',
  (ActionEnvir^.Base[B].VolRIFPerc*100.0):18:1, '%');
Writeln (F,Blank:10,'Officer Housing Units Available:',
  WithCommas(ActionEnvir^.Base[B].OffHousUnitsVacant,0):27);
Writeln (F,Blank:10,'Enlisted Housing Units Available:',
  WithCommas(ActionEnvir^.Base[B].EnlHousUnitsVacant,0):26);
Writeln (F,Blank:10,'Total Base Facilities (Square Feet):',
  WithCommas(ActionEnvir^.Base[B].BaseFacilSqFt,0):23);
Writeln (F,Blank:10,'Total Acreage on Base (Acres):',
  WithCommas(ActionEnvir^.Base[B].AcreOnBase,0):29);
Writeln (F,Blank:10,'Officer Variable Housing Allowance ($/Month):',
  WithCommas(PersCost.Base[B].OffVHA,0):14);
Writeln (F,Blank:10,'Enlisted Variable Housing Allowance ($/Month):',
  WithCommas(PersCost.Base[B].EnlVHA,0):13);
Writeln (F,Blank:10,'Per Diem Rate ($/Day):',
  WithCommas(PersCost.Base[B].PerDiemRate,0):37);
Writeln (F,Blank:10,'Freight Cost ($/Ton/Mile):',
  WithCommas(FreightCost.Base[B],2):33);
writeln (F,Blank:10,'Area Cost Factor:',
  Con.Base[B].AreaCostFactor:42:2);
Writeln (F);
Writeln (F,Blank:10,'RPMA Non-Payroll Costs ($K/Year):',
  WithCommas(BudgetPropData^.Base[B].RPMANonPayBudget/1000.0,0):26);
Writeln (F,Blank:10,'RPMA Payroll Costs ($K/Year):',
  WithCommas(BudgetPropData^.Base[B].RPMAPayBudget/1000.0,0):30);
Writeln (F,Blank:10,'Communications Costs ($K/Year):',
  WithCommas(BudgetPropData^.Base[B].CommunBudget/1000.0,0):28);
Writeln (F,Blank:10,'Base Ops Non-Payroll Costs ($K/Year):',
  WithCommas(BudgetPropData^.Base[B].BaseOpnsNonPayBudget
    /1000.0,0):22);
Writeln (F,Blank:10,'Base Ops Payroll Costs ($K/Year):',
  WithCommas(BudgetPropData^.Base[B].BaseOpnsPay/1000.0,0):26);
Writeln (F,Blank:10,'Family Housing Costs ($K/Year):',
  WithCommas(BudgetPropData^.Base[B].FamHousBudget/1000.0,0):28);
```

```
        Writeln (F);
        Writeln (F,Blank:10,'CHAMPUS On-Base In-Patient Cost/Visit ($):',
          WithCommas(Champus.Base[B].ChampInPatCostVisit,0):17);
        Writeln (F,Blank:10,'CHAMPUS On-Base Out-Patient Cost/Visit ($):',
          WithCommas(Champus.Base[B].ChampOutPatCostVisit,0):16);
        Writeln (F,Blank:10,'CHAMPUS Shift To Medicare',
          (ActionEnvir^.Base[B].ChampusToMedicare * 100.0):34:1, '%');
        If Odd (Screen9Data.Pages Div 8) Then
          Begin
            Writeln (F);
            Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
          End;
      End;
  END;  { DataInputScreen4 }

PROCEDURE DataInputScreen5;

  VAR
    B, Y,
    TimePhasTemp    : INTEGER;
    UserEntryError : Word;

  Begin { DataInputScreen5 }
    UserEntryError := 0;
    For B := 0 To MaxBases Do
      Begin
        If Not Odd(B) Then
          Begin
            Title (F,'INPUT SCREEN FIVE - DYNAMIC BASE INFO', Page);
            Inc (Page);
          End;
        Writeln (F);
        Writeln (F, Blank:10, 'Name:  ', BaseNState(B));
        FiscOut (F, 28, 7, 0);
        Write (F, Blank:28);
        For Y := 0 To MaxYears - 1 Do
          Write (F,'   ----');
        Writeln (F);
        YearOut (F, '1-Time Unique($K):',
          ActionEnvir^.Base[B].OneTimeUniqueCosts, OneK, 7, 0);
        YearOut (F, '1-Time Moving($K):',
          ActionEnvir^.Base[B].OneTimeMovingCosts, OneK, 7, 0);
        YearOut (F, 'Env Mitig Req($K):',
          ActionEnvir^.Base[B].EnvirMitigReqd, OneK, 7, 0);
        YearOut (F, 'Act Misn Cost($K):', ActivMisCost^.Base[B], OneK, 7, 0);
        YearOut (F, 'Misc Rec Cost($K):',
          BudgetPropData^.Base[B].MiscRecurCost, OneK, 7, 0);
        Writeln (F);
        YearOut (F, 'Property (Acres): ',
          RealPropProc.RealPropPur.Base[B].Acre, One, 7, 0);
        YearOut (F, 'Property ($K):    ',
          RealPropProc.RealPropPur.Base[B].Cost, OneK, 7, 0);
```

```
      Center (F, '(Positive indicates buys, negative indicates sales)');
      Writeln (F);
      TimePhasTemp := 0;
      Write (F, Blank:10, 'Construc Sched(%): ');
      For Y := 1 TO MaxYears Do
        Begin
          Write (F,(TimePhasShutDn.Base[B].Constr.Year[Y]*100.0):6:0, '%');
          TimePhasTemp := TimePhasTemp
            + Round(TimePhasShutDn.Base[B].Constr.Year[Y]*100);
        End;
      If (TimePhasTemp <> 100) And
          Not Odd(UserEntryError Div Round(YtoX(2,B)))
          And (Screen1Data^.ModelDoShutDn <> 'Y')
        Then UserEntryError := UserEntryError + Round(YtoX(2,B));
      Writeln (F);
      TimePhasTemp := 0;
      Write (F, Blank:10, 'Shutdown Sched(%): ');
      For Y := 1 TO MaxYears Do
        Begin
          Write (F,(TimePhasShutDn.Base[B].ShutDn.Year[Y]*100.0):6:0, '%');
          TimePhasTemp := TimePhasTemp
            + Round(TimePhasShutDn.Base[B].ShutDn.Year[Y]*100.0);
        End;
      If (TimePhasTemp <> 100) And
          Not Odd(UserEntryError Div Round(YtoX(2,B)))
          And (Screen1Data^.ModelDoShutDn <> 'Y')
        Then UserEntryError := UserEntryError + Round(YtoX(2,B));
      Writeln (F);
      Writeln (F);
      YearOut (F, 'Constr Avoid ($K):', CostAvoid.Base[B].Constr,
        OneK, 7, 0);
      YearOut (F, 'FamHousAvoid ($K):', CostAvoid.Base[B].FamQtr,
        OneK, 7, 0);
      YearOut (F, 'Procur Avoid ($K):', CostAvoid.Base[B].Procure,
        OneK, 7, 0);
      Writeln (F);
      Writeln (F, Blank:10, 'Facility Shut Down (SqFt):',
        WithCommas(RealPropProc.Base[B].FacilSqFtShutDown,0):20);
      Writeln (F, Blank:10, 'Percent of Family Housing ShutDown:',
        (ActionEnvir^.Base[B].FamHousShutDn*100.0):11:1, '%');
      Writeln (F);
      If Odd(Screen9Data.Pages Div 16) And Odd(B)
        Then Writeln (F, Blank:10,
          '(See final page for Explanatory Notes)');
    End;
  If UserEntryError > 0 Then
    Begin
      ErrorOut
        ('The Time-Phasing percentages for the following base(s)');
      ErrorOut
        ('do not add up to 100% :');
      ErrorList (UserEntryError);
```

```
      End;
  END;

PROCEDURE DataInputScreen6;

  VAR
    B, Y,
    CareTakeTemp     : INTEGER;
    UserEntryError,
    UserEntryError2 : Word;

  Begin { DataInputScreen6 }
    UserEntryError := 0;
    UserEntryError2 := 0;
    For B := 0 To MaxBases Do
      Begin
        If (B Div 4) * 4 = B Then
          Begin
            Title (F,'INPUT SCREEN SIX - BASE PERSONNEL INFO', Page);
            Inc (Page);
          End;
        Writeln (F);
        Writeln (F, Blank:10, 'Name:  ', BaseNState(B));
        FiscOut (F, 28, 7, 0);
        Write (F, Blank:28);
        For Y := 0 To MaxYears - 1 Do
          Write(F,'   ----');
        Writeln(F);
        Write (F, Blank:10, 'Officer FS Chg:   ');
        For Y := 1 TO MaxYears Do
          Write (F, WithCommas(-PosReduc.Base[B].Off.Year[Y],0):7);
        Writeln (F);
        Write (F, Blank:10, 'Enlisted FS Chg:  ');
        For Y := 1 TO MaxYears Do
          Write (F, WithCommas(-PosReduc.Base[B].Enl.Year[Y],0):7);
        Writeln (F);
        Write (F, Blank:10, 'Civilian FS Chg:  ');
        For Y := 1 TO MaxYears Do
          Write (F, WithCommas(-PosReduc.Base[B].Civ.Year[Y],0):7);
        Writeln (F);
        Write (F, Blank:10, 'Officers Elim:    ');
        For Y := 1 TO MaxYears Do
          Write (F, WithCommas(PosElim.Base[B].Off.Year[Y],0):7);
        Writeln (F);
        Write (F, Blank:10, 'Enlisted Elim:    ');
        For Y := 1 TO MaxYears Do
          Write (F, WithCommas(PosElim.Base[B].Enl.Year[Y],0):7);
        Writeln (F);
        Write (F, Blank:10, 'Civilians Elim:   ');
        For Y := 1 TO MaxYears Do
          Write (F, WithCommas(PosElim.Base[B].Civ.Year[Y],0):7);
        Writeln (F);
```

```
      CareTakeTemp := 0;
      Write (F, Blank:10, 'Caretakers - Mil: ');
      For Y := 1 TO MaxYears Do
        Begin
          Write (F, WithCommas(PosTrans.Base[B].CaretakeMil.Year[Y],0):7);
          CareTakeTemp := CareTakeTemp
            + PosTrans.Base[B].CaretakeMil.Year[Y];
          If (CareTakeTemp < 0) And
              Not Odd(UserEntryError Div Round(YtoX(2,B)))
            Then UserEntryError := UserEntryError + Round(YtoX(2,B));
          If (CaretakeTemp > 0) And (Screen1Data^.Base[B].Deactivate <> 'Y')
              And Not Odd(UserEntryError2 Div Round(YtoX(2,B)))
            Then UserEntryError2 := UserEntryError2 + Round(YtoX(2,B));
        End;
      Writeln (F);
      CareTakeTemp := 0;
      Write (F, Blank:10, 'Caretakers - Civ: ');
      For Y := 1 TO MaxYears Do
        Begin
          Write (F, WithCommas(PosTrans.Base[B].CaretakeCiv.Year[Y],0):7);
          CareTakeTemp := CareTakeTemp
            + PosTrans.Base[B].CaretakeCiv.Year[Y];
          If (CareTakeTemp < 0) And
              Not Odd(UserEntryError Div Round(YtoX(2,B)))
            Then UserEntryError := UserEntryError + Round(YtoX(2,B));
          If (CaretakeTemp > 0) And (Screen1Data^.Base[B].Deactivate <> 'Y')
              And Not Odd(UserEntryError2 Div Round(YtoX(2,B)))
            Then UserEntryError2 := UserEntryError2 + Round(YtoX(2,B));
        End;
      Writeln (F);
      Write (F, Blank:10, 'CHAMPUS InPat/Yr: ');
      For Y := 1 TO MaxYears Do
        Write (F,
          WithCommas(Champus.Base[B].OnBaseInPatVisits.Year[Y],0):7);
      Writeln (F);
      Write (F, Blank:10, 'CHAMPUS OutPat/Yr:');
      For Y := 1 TO MaxYears Do
        Write (F,
          WithCommas(Champus.Base[B].OnBaseOutPatVisits.Year[Y],0):7);
      Writeln (F);
      Writeln (F);
      If Odd (Screen9Data.Pages Div 32)
        And (((B + 1) Div 4) * 4 - 1 = B) Then
          Begin
            Writeln (F);
            Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
          End;
    End;
  If UserEntryError > 0 Then
    Begin
      ErrorOut
        ('The following base(s), at some point, have a negative number');
```

```
        ErrorOut
          ('of Civilian and/or Military caretakers:');
        ErrorList (UserEntryError);
      End;
    If UserEntryError2 > 0 Then
      Begin
        ErrorOut
          ('The following base(s) have Civilian and/or Military caretakers');
        ErrorOut
          ('assigned even though they are not being deactivated.');
        ErrorList (UserEntryError2);
      End;
  END;

PROCEDURE DataInputScreen7;

  VAR
    B, P : INTEGER;

  Begin
    For B := 0 To MaxBases Do
      Begin
        Title (F,'INPUT SCREEN SEVEN - MILCON BASE INFO', Page);
        Inc (Page);
        Writeln (F);
        Writeln (F, Blank:10, 'Name:  ', BaseNState(B));
        Writeln (F);
        Writeln (F, Blank:10, 'Description',
                    Blank:11, 'Category   New Con     Rehab  Cost($K)');
        Writeln (F, Rept('-',12):22,
                    Blank:10, '--------   -------     -----  --------');
        For P := 1 To 16 Do
          Begin
            If Con.Base[B].ProjectName[P] <> Nil
              Then Write (F, Blank:10, Con.Base[B].ProjectName[P]^,
                   Blank:(22 - Length (Con.Base[B].ProjectName[P]^)))
              Else Write (F, Blank:32);
            Case Con.Base[B].Category[P] Of
              '@': Write (F, ' (Other)');
              'A': Write (F, 'Horizont');
              'B': Write (F, 'Watrfrnt');
              'C': Write (F, 'Air Opns');
              'D': Write (F, 'Operatnl');
              'E': Write (F, 'Administ');
              'F': Write (F, 'SchlBldg');
              'G': Write (F, 'MaintShp');
              'H': Write (F, 'BachQtrs');
              'I': Write (F, 'FamlQtrs');
              'J': Write (F, 'StoFacil');
              'K': Write (F, 'DinFacil');
              'L': Write (F, 'RecFacil');
              'M': Write (F, 'ComFacil');
```

```
                'N': Write (F, 'Shipyard');
                'O': Write (F, 'RDT&EFac');
                'P': Write (F, 'POLStore');
                'Q': Write (F, 'AmmoStor');
                'R': Write (F, 'MedFacil');
                'S': Write (F, 'Environm');
                'T', 'U', 'V', 'W', 'X',
                'Y', 'Z', '[', '\', ']',
                '^', '_', '`', 'a', 'b', 'c':
                  If StdFctrsCon.Name[Ord(Con.Base[B].Category[P])-83] = ''
                                  Then   Write   (F,   'OptCat-',
Chr(Ord(Con.Base[B].Category[P])-19))
                    Else Write (F, Copy(
                      StdFctrsCon.Name[Ord(Con.Base[B].Category[P])-83],1,8):8)
                End;
              Writeln (F, WithCommas(Con.Base[B].New[P],0):10,
                WithCommas(Con.Base[B].Rehab[P],0):10,
                WithCommas(Con.Base[B].Cost[P],0):10);
              If Con.Base[B].Comment[P] <> Nil
                Then Center (F, Concat ('(', Con.Base[B].Comment[P]^, ')'));
            End;
          If Odd (Screen9Data.Pages Div 64) Then
            Begin
              Writeln (F);
              Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
            End;
        End;
    End;

  PROCEDURE DataInputScreen8;

    VAR
      B, Y : INTEGER;

    Begin { DataInputScreen8 }
      For B := 0 To MaxBases Do
        If IndustFundSwitch.Base[B] = 'Y' Then
          Begin
            Title (F,'INPUT SCREEN EIGHT - UNIQUE ACTIVITY INFO', Page);
            Inc (Page);
            Writeln (F);
            Writeln (F, Blank:10, 'Base:  ', BaseNState(B));
            FiscOut (F, 22, 8, 0);
            Write (F, BLANK:22);
            For Y := 1 TO MaxYears DO
              write (F, '    ----');
            Writeln (F);
            Writeln (F, Blank:10, 'OVHD COST ($K):');
            YearOut (F, '  Admin/Plan', IndustFund.Base[B]^.OvhdCost.AdminPlan,
              OneK, 8, 0);
            YearOut (F, '  Operating*', IndustFund.Base[B]^.OvhdCost.Operating,
              OneK, 8, 0);
```

```
          YearOut (F, '  Mothball  ', IndustFund.Base[B]^.OvhdCost.Mothball,
            OneK, 8, 0);
          YearOut (F, '  Caretaker*', IndustFund.Base[B]^.OvhdCost.Caretaker,
            OneK, 8, 0);
          YearOut (F, '  Maint Spc*', IndustFund.Base[B]^.OvhdCost.MaintSpace,
            OneK, 8, 0);
          YearOut (F, '  Other    *', IndustFund.Base[B]^.OvhdCost.Other,
            OneK, 8, 0);
          Writeln (F);
          Writeln (F, Blank:10, 'OVHD SAVE ($K):');
          YearOut (F, '  Operating*', IndustFund.Base[B]^.OvhdSave.Operating,
            OneK, 8, 0);
          YearOut (F, '  Other    *', IndustFund.Base[B]^.OvhdSave.Other,
            OneK, 8, 0);
          Writeln (F);
          Writeln (F, Blank:10, 'MOVE COST ($K):');
          YearOut (F, '  Pack/Unpac', IndustFund.Base[B]^.MoveCost.PackUnpack,
            OneK, 8, 0);
          YearOut (F, '  Freight   ', IndustFund.Base[B]^.MoveCost.Freight,
            OneK, 8, 0);
          YearOut (F, '  Vehic Move', IndustFund.Base[B]^.MoveCost.Vehic,
            OneK, 8, 0);
          YearOut (F, '  Driving   ', IndustFund.Base[B]^.MoveCost.Driv,
            OneK, 8, 0);
          YearOut (F, '  Loss Rate ', IndustFund.Base[B]^.MoveCost.LossRate,
            OneK, 8, 0);
          Writeln (F);
          Writeln (F, Blank:10,
            '* = Values used in Beyond Year calculations');
          If Odd (Screen9Data.Pages Div 128) Then
            Begin
              Writeln (F);
              Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
            End;
        End;
  End; {DataScreens}

  {Standard Factors Screen 1}

PROCEDURE StandardFactors;

  Var
    P, Y : Integer;

  Begin { StandardFactors }
    Title (F,'STANDARD PERSONNEL FACTORS', Page);
    Inc (Page);
    writeln (F);
    writeln (F, Blank:10, 'Percentage of Officers Married',
      (StdFctrsPers.PercOffMarried * 100.0):29:2, '%');
    writeln (F, Blank:10, 'Percentage of Enlisted Married',
      (StdFctrsPers.PercEnlMarried * 100.0):29:2, '%');
```

```
writeln (F, Blank:10, 'Enlisted Housing Military Construction',
  (StdFctrsPers.PercEnlHous * 100.0):21:2, '%');
writeln (F);
writeln (F, Blank:10, 'Officer Salary ($/Year)',
  WithCommas (StdFctrsPers.OffSal,2):36);
writeln (F, Blank:10, 'Officer BAQ with Dependents',
  WithCommas (StdFctrsPers.OffBAQDepends,2):32);
writeln (F, Blank:10, 'Enlisted Salary ($/Year)',
  WithCommas (StdFctrsPers.EnlSal,2):35);
writeln (F, Blank:10, 'Enlisted BAQ with Dependents',
  WithCommas (StdFctrsPers.EnlBAQDepends,2):31);
writeln (F, Blank:10, 'Average Unemployment Cost ($/Week)',
  WithCommas (StdFctrsPers.AvgUnemployCostPerWeek,2):25);
writeln (F, Blank:10, 'Unemployment Eligibility Period (Weeks)',
  WithCommas (StdFctrsPers.UnemployEligible,0):17);
writeln (F, Blank:10, 'Civilian Salary ($)',
  WithCommas (StdFctrsPers.CivSal,2):40);
writeln (F);
writeln (F, Blank:10, 'Civilian Turnover Rate',
  (StdFctrsPers.CivTurnover * 100.0):37:2, '%');
writeln (F, Blank:10, 'Civilian Early Retirement Rate',
  (StdFctrsPers.CivEarlyRetire * 100.0):29:2, '%');
writeln (F, Blank:10, 'Civilian Quitting Rate',
  (StdFctrsPers.PercCivsQuitting * 100.0):37:2, '%');
writeln (F, Blank:10, 'Civilian RIF Pay Factor',
  (StdFctrsPers.CivRifPayFctr * 100.0):36:2, '%');
writeln (F, Blank:10, 'Civilian Retirement Pay Factor',
  (StdFctrsPers.CivRetirePayFctr * 100.0):29:2, '%');
writeln (F, Blank:10, 'Priority Placement Service',
  (StdFctrsPers.PriPlacementPerc * 100.0):33:2, '%');
Writeln (F, Blank:10, 'PPS Actions Involving PCS',
  (StdFctrsPers.PPSInvolvePCS * 100.0):34:2, '%');
writeln (F, Blank:10, 'Civilian PCS Costs ($)',
  WithCommas (StdFctrsPers.CivPCSCosts,2):37);
writeln (F, Blank:10, 'New Hire Cost ($)',
  WithCommas (StdFctrsPers.NewHireCost,2):42);
writeln (F);
writeln (F, Blank:10, 'National Median Home Price ($)',
  WithCommas (StdFctrsPers.NatlAvgHomePrice,2):29);
writeln (F, Blank:10, 'Home Sale Reimbursement',
  (StdFctrsPers.HomeSaleReimburseRate * 100.0):36:2, '%');
writeln (F, Blank:10, 'Maximum Home Sale Reimbursement ($)',
  WithCommas (StdFctrsPers.HomeSaleReimburseMax,2):24);
writeln (F, Blank:10, 'Home Purch Reimbursement',
  (StdFctrsPers.HomePurchReimburseRate * 100.0):35:2, '%');
writeln (F, Blank:10, 'Maximum Home Purch Reimbursement ($)',
  WithCommas (StdFctrsPers.HomePurchReimburseMax,2):23);
writeln (F, Blank:10, 'Civilian Homeowning Rate',
  (StdFctrsPers.CivHomeOwnRate * 100.0):35:2, '%');
writeln (F, Blank:10, 'HAP Home Value Reimbursement Rate',
  (StdFctrsPers.HAPFactor * 100.0):26:2, '%');
```

```
writeln (F, Blank:10, 'HAP Homeowner Receiving Rate',
  (StdFctrsPers.HAPReceive * 100.0):31:2, '%');
writeln (F, Blank:10, 'RSE Home Value Reimbursement Rate',
  (StdFctrsPers.DARSEFactor * 100.0):26:2, '%');
writeln (F, Blank:10, 'RSE Homeowner Receiving Rate',
  (StdFctrsPers.DARSEReceive * 100.0):31:2, '%');
writeln (F);
writeln (F, Blank:10, 'Standard Factors File Description:',
  StripSpace(StdFctrsPers.FileDescription):25);
If Odd (Screen9Data.Pages Div 256) Then
  Begin
    Writeln (F);
    Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
  End;

Title (F, 'STANDARD FACILITY FACTORS', Page);
Inc (Page);
writeln (F);
         writeln (F, Blank:10, 'RPMA Building SF Cost Index',
  WithCommas (StdFctrsFac.RPMACostBldgPower,2):32);
writeln (F, Blank:10, 'BOS Index (RPMA vs population)',
  WithCommas (StdFctrsFac.BOSIndex,2):29);
writeln (F, Blank:10, '(Indices are used as exponents)');
writeln (F);
writeln (F, Blank:10, 'Support for Move Factor',
  (StdFctrsFac.SuppToMoveFctr * 100.0):36:2, '%');
writeln (F);
writeln (F, Blank:10, 'Caretaker Costs:');
writeln (F, Blank:10, '--------- ------');
writeln (F, Blank:10, 'Administrative Space Needs (SF/Caretaker)',
  WithCommas (StdFctrsFac.CaretakerAdminSpace,2):18);
writeln (F, Blank:10, 'Percentage of Original RPMA Cost',
  (StdFctrsFac.CaretakerPercOrigRPMACost * 100.0):27:2, '%');
writeln (F, Blank:10, 'Mothball Cost ($/SqFt)',
  WithCommas (StdFctrsFac.MothballCostPerSqrFt,2):37);
Writeln (F);
writeln (F, Blank:10, 'Discount Rate for NPV.RPT/ROI:',
  (StdFctrsFac.DiscountNPV * 100.0):8:1, '%');
writeln (F, Blank:10, 'Inflation Rate for NPV.RPT/ROI:',
  (StdFctrsFac.InflationNPV * 100.0):7:1, '%');
writeln (F);
Write (F, Blank:10, 'Inflation Rate');
FiscOut (F, 3, 7, 0);
write (F, Blank:10, 'for FINANCE.RPT:  ');
For Y := 1 To MaxYears Do
  Write (F, (StdFctrsFac.InflationRate.Year[Y] * 100.0):6:1, '%');
Writeln (F);
writeln (F);
writeln (F, Blank:10, 'Average Bachelor Quarters Size (SF):',
  WithCommas (StdFctrsFac.AvgBachQtrs,2):23);
writeln (F, Blank:10, 'Average Family Quarters Size   (SF):',
  WithCommas (StdFctrsFac.AvgFamQtrs,2):23);
```

```
writeln (F);
writeln (F, Blank:10, 'Rehabilitation Cost vs. New Construction Cost',
  (StdFctrsFac.RehabCostVsNewConstr * 100.0):14:2, '%');
writeln (F, Blank:10, 'Information Management Account',
  (StdFctrsFac.IMAPerc * 100.0):29:2, '%');
writeln (F);
writeln (F, Blank:10, 'Design Rate',
  (StdFctrsFac.Design * 100.0):48:2, '%');
writeln (F, Blank:10, 'Supervision, Inspection, OverHead Rate',
  (StdFctrsFac.SIOH * 100.0):21:2, '%');
writeln (F, Blank:10, 'Contingency Planning Rate',
  (StdFctrsFac.Contingency * 100.0):34:2, '%');
writeln (F, Blank:10, 'Site Preparation Rate',
  (StdFctrsFac.SitePrep * 100.0):38:2, '%');
If Odd (Screen9Data.Pages Div 512) Then
  Begin
    Writeln (F);
    Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
  End;

Title (F,'STANDARD TRANSPORTATION FACTORS', Page);
Inc (Page);
writeln (F);
writeln (F, Blank:10, 'Material per Assigned Person (Lbs)',
  WithCommas (StdFctrsTrans.MaterialPerPerson,0):25);
writeln (F, Blank:10, 'HHG Weight Per Officer Family (Lb)',
  WithCommas (StdFctrsTrans.HHGWtPerOffFam,2):25);
writeln (F, Blank:10, 'HHG Weight Per Enlisted Family (Lb)',
  WithCommas (StdFctrsTrans.HHGWtPerEnlFam,2):24);
writeln (F, Blank:10, 'HHG Weight Per Military Single (Lb)',
  WithCommas (StdFctrsTrans.HHGWtPerMilSing,2):24);
writeln (F, Blank:10, 'HHG Weight Per Civilian (Lb)',
  WithCommas (StdFctrsTrans.HHGWtPerCiv,2):31);
writeln (F);
writeln (F, Blank:10, 'Household Goods Cost ($/100Lb)',
  WithCommas (StdFctrsTrans.HHGCost * 100.0,2):29);
writeln (F, Blank:12,
  '(Includes Packing, Unpacking, Storage, and Misc. Costs)');
writeln (F);
writeln (F, Blank:10, 'Shipping Loss Rate',
  (StdFctrsTrans.ShippingLossRate * 100.0):41:1, '%');
writeln (F);
writeln (F, Blank:10, 'Equipment Packing & Crating Cost ($/Ton)',
  WithCommas (StdFctrsTrans.PackNCrate,2):19);
writeln (F, Blank:10, 'Military Light Vehicle Cost ($/Mile)',
  WithCommas (StdFctrsTrans.MilLightVehicCostPerMile,2):23);
writeln (F, Blank:10, 'Heavy or Special Vehicle Cost ($/Mile)',
  WithCommas (StdFctrsTrans.MilVehicCostPerMile,2):21);
writeln (F, Blank:10, 'Pers Owned Vehic Reimburse ($/Mile)',
  WithCommas (StdFctrsTrans.POVReimbursePerMile,2):24);
writeln (F, Blank:10, 'Air Transport Per Passenger Mile ($)',
  WithCommas (StdFctrsTrans.AirTransPerPassMile,2):23);
```

```
writeln (F, Blank:10, 'Misc Expenses Per Direct Employee ($)',
  WithCommas (StdFctrsTrans.MiscExpPerDirectEmp,2):22);
writeln (F);
writeln (F, Blank:10, 'Avg Military Service Tour Length (Years)',
  WithCommas (StdFctrsTrans.AverageTourLength,2):19);
writeln (F, Blank:10, 'Routine PCS Costs/Person/Tour ($)',
  WithCommas (StdFctrsTrans.RoutinePCSCostsPerPerson,2):26);
writeln (F, Blank:10, 'One-Time Officer PCS Cost ($)',
  WithCommas (StdFctrsTrans.OneTimeOffPCSCosts,2):30);
writeln (F, Blank:10, 'One-Time Enlisted PCS Cost ($)',
  WithCommas (StdFctrsTrans.OneTimeEnlPCSCosts,2):29);
If Odd (Screen9Data.Pages Div 1024) Then
  Begin
    Writeln (F);
    Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
  End;

Title (F,'STANDARD CONSTRUCTION FACTORS',Page);
Inc(Page);
writeln (F);
writeln (F, BLANK:10, 'Category:', 'Units:':27, 'Cost/UM($):':24);
writeln (F);
writeln (F, Blank:10, 'Horizontal', '(SY)':25,
  WithCommas (StdFctrsCon.ConCost[1],0):24);
writeln (F, Blank:10, 'Waterfront', '(LF)':25,
  WithCommas (StdFctrsCon.ConCost[2],0):24);
writeln (F, Blank:10, 'Air Operations', '(SF)':21,
  WithCommas (StdFctrsCon.ConCost[3],0):24);
writeln (F, Blank:10, 'Operational', '(SF)':24,
  WithCommas (StdFctrsCon.ConCost[4],0):24);
writeln (F, Blank:10, 'Administrative', '(SF)':21,
  WithCommas (StdFctrsCon.ConCost[5],0):24);
writeln (F, Blank:10, 'School Buildings', '(SF)':19,
  WithCommas (StdFctrsCon.ConCost[6],0):24);
writeln (F, Blank:10, 'Maintenance Shops', '(SF)':18,
  WithCommas (StdFctrsCon.ConCost[7],0):24);
Writeln (F, Blank:10, 'Bachelor Quarters', '(':15,
  StripSpace(StdFctrsCon.Units[-2]):2, ')',
  WithCommas (StdFctrsCon.ConCost[8],0):24);
Writeln (F, Blank:10, 'Family Quarters', '(':17,
  StripSpace(StdFctrsCon.Units[-1]):2, ')',
  WithCommas (StdFctrsCon.ConCost[9],0):24);
writeln (F, Blank:10, 'Covered Storage', '(SF)':20,
  WithCommas (StdFctrsCon.ConCost[10],0):24);
writeln (F, Blank:10, 'Dining Facilities', '(SF)':18,
  WithCommas (StdFctrsCon.ConCost[11],0):24);
writeln (F, Blank:10, 'Recreation Facilities', '(SF)':14,
  WithCommas (StdFctrsCon.ConCost[12],0):24);
writeln (F, Blank:10, 'Communications Facilities', '(SF)':10,
  WithCommas (StdFctrsCon.ConCost[13],0):24);
writeln (F, Blank:10, 'Shipyard Maintenance', '(SF)':15,
  WithCommas (StdFctrsCon.ConCost[14],0):24);
```

```
      writeln (F, Blank:10, 'RDT & E Facilities', '(SF)':17,
        WithCommas (StdFctrsCon.ConCost[15],0):24);
      writeln (F, Blank:10, 'POL Storage', '(BL)':24,
        WithCommas (StdFctrsCon.ConCost[16],0):24);
      writeln (F, Blank:10, 'Ammunition Storage', '(SF)':17,
        WithCommas (StdFctrsCon.ConCost[17],0):24);
      writeln (F, Blank:10, 'Medical Facilities', '(SF)':17,
        WithCommas (StdFctrsCon.ConCost[18],0):24);
      Writeln (F, Blank:10, 'Environmental', '(':19,
        StripSpace(StdFctrsCon.Units[0]):2, ')',
        WithCommas (StdFctrsCon.ConCost[19],0):24);
      For P := 1 To 16 Do
        Begin
          If StdFctrsCon.Name[P] = ''
            Then write (F, Blank:10, 'Optional Category ', Chr(64+P), Blank:12)
            Else write (F, Blank:10, StdFctrsCon.Name[P],
                           Blank:(31 - Length (StdFctrsCon.Name[P])));
          writeln (F, '(', StripSpace(StdFctrsCon.Units[P]):2, ')',
            WithCommas (StdFctrsCon.ConCost[P+19],0):24);
        End;
      If Odd (Screen9Data.Pages Div 2048) Then
        Begin
          Writeln (F);
          Writeln (F, Blank:10, '(See final page for Explanatory Notes)');
        End;
    End;

  PROCEDURE Footnotes;

    Var
      I, Ch : Integer;

    Begin
      Ch := 0;
      For I := 1 To 15 Do
        Ch := Ch + Length (Screen9Data.Line[I]);
      If Ch > 0 Then
        Begin
          Title(F, 'EXPLANATORY NOTES',Page);
          For I := 1 To 15 Do
            Begin
              Writeln (F);
              If Screen9Data.Line[I] <> ''
                Then Writeln (F, Blank:10, Screen9Data.Line[I]);
            End;
        End;
    End;

  Begin
    Page := 3; { First two pages are constant }
    For Y := 1 TO AbsMaxYears Do
```

```
      Begin
        One.Year[Y] := 1.0;
        OneK.Year[Y] := 0.001;
      End;

  { One?.Year[Y] are used to guarantee that YearOut and BeyondOut do NOT
    inflate the values they're given to print. }

    assign  (F, Info.ReportDir + 'INPUTDAT.RPT');
    SetTextBuf (F, Buffer);
    rewrite (F);

    DataInputScreen1;
    DataInputScreen2;
    DataInputScreen3;
    DataInputScreen4;
    DataInputScreen5;
    DataInputScreen6;
    DataInputScreen7;
    DataInputScreen8;
    StandardFactors;
    Footnotes;

    Close (F)
  END;  { EnteredDataReport }

END. { UNIT }
```

```
UNIT CRPTOUT1;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE ApproDetailReport              (ApproReport : ApproReportRec;
                                          Adjust1000  : Year1ToMaxRealRec);

PROCEDURE ApproDetailFixedReport         (ApproReport : ApproReportRec;
                                          Adjust1000  : Year1ToMaxRealRec);

PROCEDURE ApproDetailSteadyStateReport (ApproReport : ApproReportRec;
                                          Adjust1000  : Year1ToMaxRealRec);

IMPLEMENTATION

{*************************************************************************
 *   Name          : AppropriationsDetailReport
 *   Purpose       : Writes appropriations detail report
 *   Invoked As    : AppropriationsDetailReport (ApproReport)
 *   Files/Tables
 *     Names                 Access    Purpose
 *     ------------------    --------  ----------------------------------------
 *     APPDET.RPT            Write
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   27Mar92  AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   10Apr92  AML   Streamlining
 *************************************************************************}

PROCEDURE ApproDetailReport (ApproReport : ApproReportRec;
                             Adjust1000  : Year1ToMaxRealRec);

  VAR
    Buffer : IOBuffer;
    F      : TEXT;
    Y      : INTEGER;

  Begin { AppropriationsDetailReport }
    assign (F, Info.ReportDir + 'APPDET.RPT');
    SetTextBuf (F, Buffer);
    reWrite (F);
```

```
Title (F, 'APPROPRIATIONS DETAIL', -1);
Write (F, Blank:10, 'COSTS($K)  ');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F, '  Beyond');
Write (F, Blank:10, '---------  ');
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F, '  ------');
BeyondOut (F, 'MilCon     ', ApproReport^.Costs.MilCon, Adjust1000, 8, 1);
Writeln (F, Blank:10, 'FAM HOUSING');
BeyondOut (F, ' Construct ', ApproReport^.Costs.FamHous.Construction,
  Adjust1000, 8, 1);
BeyondOut (F, ' Operations', ApproReport^.Costs.FamHous.Operations,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'O&M');
BeyondOut (F, ' RPMA      ', ApproReport^.Costs.OM.RPMA, Adjust1000, 8, 1);
BeyondOut (F, ' BOS       ', ApproReport^.Costs.OM.BOS, Adjust1000, 8, 1);
BeyondOut (F, ' UniqOperat', ApproReport^.Costs.OM.OperatingIDF,
  Adjust1000, 8, 1);
Writeln (F, Blank:11, 'CIV SALARY');
BeyondOut (F, '  Civ RIF  ', ApproReport^.Costs.OM.CivSalary.CivRIF,
  Adjust1000, 8, 1);
BeyondOut (F, '  Civ Retir', ApproReport^.Costs.OM.CivSalary.CivRetire,
  Adjust1000, 8, 1);
Writeln (F, Blank:11, 'CIV MOVING');
BeyondOut (F, '  Per Diem ', ApproReport^.Costs.OM.CivMoving.PerDiem,
  Adjust1000, 8, 1);
BeyondOut (F, '  POV Miles', ApproReport^.Costs.OM.CivMoving.POVMiles,
  Adjust1000, 8, 1);
BeyondOut (F, '  Home Purc', ApproReport^.Costs.OM.CivMoving.HomePurchSale,
  Adjust1000, 8, 1);
BeyondOut (F, '  HHG      ', ApproReport^.Costs.OM.CivMoving.HHG,
  Adjust1000, 8, 1);
BeyondOut (F, '  Misc     ', ApproReport^.Costs.OM.CivMoving.Misc,
  Adjust1000, 8, 1);
BeyondOut (F, '  Hous Hunt', ApproReport^.Costs.OM.CivMoving.HousHunt,
  Adjust1000, 8, 1);
BeyondOut (F, '  PPS      ', ApproReport^.Costs.OM.CivMoving.PPS,
  Adjust1000, 8, 1);
BeyondOut (F, '  RITA     ', ApproReport^.Costs.OM.CivMoving.RITA,
  Adjust1000, 8, 1);
Writeln (F, Blank:11, 'FREIGHT');
BeyondOut (F, '  Packing  ', ApproReport^.Costs.OM.Freight.Pack,
  Adjust1000, 8, 1);
BeyondOut (F, '  Freight  ', ApproReport^.Costs.OM.Freight.Freight,
  Adjust1000, 8, 1);
BeyondOut (F, '  Vehicles ', ApproReport^.Costs.OM.Freight.Vehicles,
  Adjust1000, 8, 1);
BeyondOut (F, '  Driving  ', ApproReport^.Costs.OM.Freight.Driving,
  Adjust1000, 8, 1);
```

```
BeyondOut (F, '  Loss Rate', ApproReport^.Costs.OM.Freight.LossRate,
  Adjust1000, 8, 1);
BeyondOut (F, ' CHAMPUS   ', ApproReport^.Costs.OM.Champus,
  Adjust1000, 8, 1);
BeyondOut (F, ' Unemploymt', ApproReport^.Costs.OM.Unemployment,
  Adjust1000, 8, 1);
Writeln (F, Blank:11, 'OTHER');
BeyondOut (F, '  Caretaker', ApproReport^.Costs.OM.Other.Caretaker,
  Adjust1000, 8, 1);
BeyondOut (F, '  AdminPlan', ApproReport^.Costs.OM.Other.AdminPlan,
  Adjust1000, 8, 1);
BeyondOut (F, '  Shutdown ', ApproReport^.Costs.OM.Other.Shutdown,
  Adjust1000, 8, 1);
BeyondOut (F, '  Maintain ', ApproReport^.Costs.OM.Other.Maintain,
  Adjust1000, 8, 1);
BeyondOut (F, '  New Hire ', ApproReport^.Costs.OM.Other.NewHire,
  Adjust1000, 8, 1);
BeyondOut (F, '  1TimeMove', ApproReport^.Costs.OM.Other.OneTimeMove,
  Adjust1000, 8, 1);
BeyondOut (F, '  Unique   ', ApproReport^.Costs.OM.Other.OtherIDF,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'MIL PERSONNEL');
Writeln (F, Blank:11, 'MIL MOVING');
BeyondOut (F, '  Elim PCS ',
  ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS, Adjust1000, 8, 1);
BeyondOut (F, '  Per Diem ', ApproReport^.Costs.MilPers.MilMoving.PerDiem,
  Adjust1000, 8, 1);
BeyondOut (F, '  POV Miles', ApproReport^.Costs.MilPers.MilMoving.POVMiles,
  Adjust1000, 8, 1);
BeyondOut (F, '  HHG      ', ApproReport^.Costs.MilPers.MilMoving.HHG,
  Adjust1000, 8, 1);
BeyondOut (F, '  Misc     ', ApproReport^.Costs.MilPers.MilMoving.Misc,
  Adjust1000, 8, 1);
Title (F, 'APPROPRIATIONS DETAIL', 2);
Writeln (F);
BeyondOut (F, ' Hous Allow', ApproReport^.Costs.MilPers.Other.HousAllow,
  Adjust1000, 8, 1);
BeyondOut (F, 'Procurement', ApproReport^.Costs.Procure, Adjust1000, 8, 1);
BeyondOut (F, 'HAP / RSE  ', ApproReport^.Costs.HomeownAsst,
  Adjust1000, 8, 1);
BeyondOut (F, 'Envir Mitig', ApproReport^.Costs.EnvirMitig,
  Adjust1000, 8, 1);
BeyondOut (F, 'Info Manage', ApproReport^.Costs.InfoMgt, Adjust1000, 8, 1);
Writeln (F, Blank:10, 'OTHER');
BeyondOut (F, ' Mission   ', ApproReport^.Costs.Other.Mission,
  Adjust1000, 8, 1);
BeyondOut (F, ' One-Time  ', ApproReport^.Costs.Other.OneTime,
  Adjust1000, 8, 1);
BeyondOut (F, 'Land Purch ',
  ApproReport^.Costs.LandPurch, Adjust1000, 8, 1);
BeyondOut (F, 'Misc Recur ',
  ApproReport^.Costs.MiscRecur, Adjust1000, 8, 1);
```

```
Writeln (F);
BeyondOut (F, 'TOT COSTS  ', ApproReport^.Costs.Total, Adjust1000, 8, 1);

Writeln (F);
Writeln (F);
Write (F, Blank:10, 'SAVINGS($K)');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F, '  Beyond');
Write (F, Rept('-',11):21);
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F, '  ------');
Writeln (F, Blank:10, 'MILCON');
BeyondOut (F, ' Cost Avoid', ApproReport^.Savings.MilCon, Adjust1000, 8, 1);
Writeln (F, Blank:10, 'FAM HOUSING');
BeyondOut (F, ' Construct ', ApproReport^.Savings.FamHous.Construction,
  Adjust1000, 8, 1);
BeyondOut (F, ' Operations', ApproReport^.Savings.FamHous.Operations,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'O&M');
BeyondOut (F, ' RPMA      ',
  ApproReport^.Savings.OM.RPMA, Adjust1000, 8, 1);
BeyondOut (F, ' BOS       ', ApproReport^.Savings.OM.BOS, Adjust1000, 8, 1);
BeyondOut (F, ' UniqOperat', ApproReport^.Savings.OM.OperatingIDF,
  Adjust1000, 8, 1);
BeyondOut (F, ' Civ Salary', ApproReport^.Savings.OM.CivSalary,
  Adjust1000, 8, 1);
BeyondOut (F, ' Civ Moving', ApproReport^.Savings.OM.CivMoving,
  Adjust1000, 8, 1);
BeyondOut (F, ' Freight   ', ApproReport^.Savings.OM.Freight,
  Adjust1000, 8, 1);
BeyondOut (F, ' CHAMPUS   ', ApproReport^.Savings.OM.Champus,
  Adjust1000, 8, 1);
BeyondOut (F, ' UniqOther ', ApproReport^.Savings.OM.OtherIDF,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'MIL PERSONNEL');
Writeln (F, Blank:11, 'SALARY');
BeyondOut (F, '  Officer  ', ApproReport^.Savings.MilPers.MilSalary.Off,
  Adjust1000, 8, 1);
BeyondOut (F, '  Enlisted ', ApproReport^.Savings.MilPers.MilSalary.Enl,
  Adjust1000, 8, 1);
BeyondOut (F, ' Mil Moving', ApproReport^.Savings.MilPers.MilMoving,
  Adjust1000, 8, 1);
BeyondOut (F, ' Hous Allow', ApproReport^.Savings.MilPers.Other.HousAllow,
  Adjust1000, 8, 1);
BeyondOut (F, 'Procurement',
  ApproReport^.Savings.Procure, Adjust1000, 8, 1);
BeyondOut (F, 'HAP / RSE  ', ApproReport^.Savings.HomeownAsst,
  Adjust1000, 8, 1);
BeyondOut (F, 'Envir Mitig', ApproReport^.Savings.EnvirMitig,
  Adjust1000, 8, 1);
```

```
BeyondOut (F, 'Info Manage',
  ApproReport^.Savings.InfoMgt, Adjust1000, 8, 1);
Writeln (F, Blank:10, 'OTHER');
BeyondOut (F, ' Mission   ', ApproReport^.Savings.Other, Adjust1000, 8, 1);
BeyondOut (F, 'LandRevenue', ApproReport^.Savings.LandRevenue,
  Adjust1000, 8, 1);
BeyondOut (F, 'Misc Recur ', ApproReport^.Savings.MiscRecur,
  Adjust1000, 8, 1);
Writeln (F);
BeyondOut (F, 'TOT SAVINGS', ApproReport^.Savings.Total, Adjust1000, 8, 1);

Title (F, 'APPROPRIATIONS DETAIL', 3);
Write (F, Blank:10, 'NETCOST($K)');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F, '  Beyond');
Write (F, Rept('-',11):21);
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F, '  ------');
BeyondOut (F, 'MilCon     ', ApproReport^.Net.MilCon, Adjust1000, 8, 1);
Writeln (F, Blank:10, 'FAM HOUSING');
BeyondOut (F, ' Construct ', ApproReport^.Net.FamHous.Construction,
  Adjust1000, 8, 1);
BeyondOut (F, ' Operations', ApproReport^.Net.FamHous.Operations,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'O&M');
BeyondOut (F, ' RPMA      ', ApproReport^.Net.OM.RPMA, Adjust1000, 8, 1);
BeyondOut (F, ' BOS       ', ApproReport^.Net.OM.BOS, Adjust1000, 8, 1);
BeyondOut (F, ' UniqOperat', ApproReport^.Net.OM.OperatingIDF,
  Adjust1000, 8, 1);
BeyondOut (F, ' Civ Salary', ApproReport^.Net.OM.CivSalary,
  Adjust1000, 8, 1);
BeyondOut (F, ' Civ Moving', ApproReport^.Net.OM.CivMoving,
  Adjust1000, 8, 1);
BeyondOut (F, ' Other     ', ApproReport^.Net.OM.Other, Adjust1000, 8, 1);
Writeln (F, Blank:10, 'MIL PERSONNEL');
BeyondOut (F, ' Mil Salary', ApproReport^.Net.MilPers.MilSalary,
  Adjust1000, 8, 1);
BeyondOut (F, ' Mil Moving', ApproReport^.Net.MilPers.MilMoving,
  Adjust1000, 8, 1);
BeyondOut (F, ' Other     ', ApproReport^.Net.MilPers.Other,
  Adjust1000, 8, 1);
BeyondOut (F, 'Procurement', ApproReport^.Net.Procure, Adjust1000, 8, 1);
BeyondOut (F, 'HAP / RSE  ',
  ApproReport^.Net.HomeownAsst, Adjust1000, 8, 1);
BeyondOut (F, 'Envir Mitig', ApproReport^.Net.EnvirMitig, Adjust1000, 8, 1);
BeyondOut (F, 'Info Manage', ApproReport^.Net.InfoMgt, Adjust1000, 8, 1);
BeyondOut (F, 'Other      ', ApproReport^.Net.Other, Adjust1000, 8, 1);
BeyondOut (F, 'Land       ', ApproReport^.Net.Land, Adjust1000, 8, 1);
BeyondOut (F, 'CHAMPUS    ', ApproReport^.Net.Champus, Adjust1000, 8, 1);
BeyondOut (F, 'Misc Recur ', ApproReport^.Net.MiscRecur, Adjust1000, 8, 1);
```

```
    Writeln (F);
    BeyondOut (F, 'TOT NETCOST', ApproReport^.Net.Total, Adjust1000, 8, 1);
    close (F)
  END;  { AppropriationsDetailReport }

{*************************************************************************
 *   Name           : ApproDetailFixedReport
 *
 *   Purpose        : Writes appropriations detail fixed costs report
 *
 *   Notes          :
 *
 *   Invoked As     : AppropriationsDetailFixedReport
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     APPDETFX.RPT           Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas, titles, etc.)
 *   10Apr92   AML   Streamlining
 *************************************************************************}

PROCEDURE ApproDetailFixedReport (ApproReport : ApproReportRec;
                                  Adjust1000  : Year1ToMaxRealRec);

  VAR
    Buffer : IOBuffer;
    F      : TEXT;
    Y      : INTEGER;

  Begin { ApproDetailFixedReport }
    assign (F, Info.ReportDir + 'APPDETFX.RPT');
    SetTextBuf (F, Buffer);
    reWrite (F);

    Title (F, 'APPROPRIATIONS DETAIL, FIXED COSTS', -1);
    Write (F, Blank:10, 'COSTS($K)   ');
    For Y := 1 To MaxYears Do
      write (F, FiscalYear(Y):8);
    writeln (F);
    Write (F, Blank:10, '---------  ');
```

```
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F);
BeyondOut (F, 'MilCon     ', ApproReport^.Costs.MilCon, Adjust1000, 8, 0);
Writeln (F, Blank:10, 'FAM HOUSING');
BeyondOut (F, ' Construct ', ApproReport^.Costs.FamHous.Construction,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, 'O&M');
Writeln (F, Blank:10, ' CIV SALARY');
BeyondOut (F, '  RIF      ', ApproReport^.Costs.OM.CivSalary.CivRIF,
  Adjust1000, 8, 0);
BeyondOut (F, '  Retire   ', ApproReport^.Costs.OM.CivSalary.CivRetire,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, ' CIV MOVING');
BeyondOut (F, '  Per Diem ', ApproReport^.Costs.OM.CivMoving.PerDiem,
  Adjust1000, 8, 0);
BeyondOut (F, '  POV Miles', ApproReport^.Costs.OM.CivMoving.POVMiles,
  Adjust1000, 8, 0);
BeyondOut (F, '  Home Purc', ApproReport^.Costs.OM.CivMoving.HomePurchSale,
  Adjust1000, 8, 0);
BeyondOut (F, '  HHG      ', ApproReport^.Costs.OM.CivMoving.HHG,
  Adjust1000, 8, 0);
BeyondOut (F, '  Misc     ', ApproReport^.Costs.OM.CivMoving.Misc,
  Adjust1000, 8, 0);
BeyondOut (F, '  Hous Hunt', ApproReport^.Costs.OM.CivMoving.HousHunt,
  Adjust1000, 8, 0);
BeyondOut (F, '  PPS      ', ApproReport^.Costs.OM.CivMoving.PPS,
  Adjust1000, 8, 0);
BeyondOut (F, '  RITA     ', ApproReport^.Costs.OM.CivMoving.RITA,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, ' FREIGHT');
BeyondOut (F, '  Packing  ', ApproReport^.Costs.OM.Freight.Pack,
  Adjust1000, 8, 0);
BeyondOut (F, '  Freight  ', ApproReport^.Costs.OM.Freight.Freight,
  Adjust1000, 8, 0);
BeyondOut (F, '  Vehicles ', ApproReport^.Costs.OM.Freight.Vehicles,
  Adjust1000, 8, 0);
BeyondOut (F, '  Driving  ', ApproReport^.Costs.OM.Freight.Driving,
  Adjust1000, 8, 0);
BeyondOut (F, '  Loss Rate', ApproReport^.Costs.OM.Freight.LossRate,
  Adjust1000, 8, 0);
BeyondOut (F, ' Unemploymt', ApproReport^.Costs.OM.Unemployment,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, ' OTHER');
BeyondOut (F, '  AdminPlan', ApproReport^.Costs.OM.Other.AdminPlan,
  Adjust1000, 8, 0);
BeyondOut (F, '  Shutdown ', ApproReport^.Costs.OM.Other.Shutdown,
  Adjust1000, 8, 0);
BeyondOut (F, '  Maintain ', ApproReport^.Costs.OM.Other.Maintain,
  Adjust1000, 8, 0);
BeyondOut (F, '  New Hire ', ApproReport^.Costs.OM.Other.NewHire,
  Adjust1000, 8, 0);
```

```
BeyondOut (F, '  1TimeMove', ApproReport^.Costs.OM.Other.OneTimeMove,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, 'MIL PERSONNEL');
Writeln (F, Blank:10, ' MOVING');
BeyondOut (F, '  Elim PCS ',
  ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS, Adjust1000, 8, 0);
BeyondOut (F, '  Per Diem ', ApproReport^.Costs.MilPers.MilMoving.PerDiem,
  Adjust1000, 8, 0);
BeyondOut (F, '  POV Miles', ApproReport^.Costs.MilPers.MilMoving.POVMiles,
  Adjust1000, 8, 0);
BeyondOut (F, '  HHG      ', ApproReport^.Costs.MilPers.MilMoving.HHG,
  Adjust1000, 8, 0);
BeyondOut (F, '  Misc     ', ApproReport^.Costs.MilPers.MilMoving.Misc,
  Adjust1000, 8, 0);
BeyondOut (F, ' Hous Allow', ApproReport^.Costs.MilPers.Other.HousAllow,
  Adjust1000, 8, 0);
BeyondOut (F, 'Procuremnt*', ApproReport^.Costs.Procure, Adjust1000, 8, 0);
BeyondOut (F, 'HAP / RSE  ', ApproReport^.Costs.HomeownAsst,
  Adjust1000, 8, 0);
BeyondOut (F, 'Envir Mitig', ApproReport^.Costs.EnvirMitig,
  Adjust1000, 8, 0);
BeyondOut (F, 'Info Manage', ApproReport^.Costs.InfoMgt, Adjust1000, 8, 0);
Writeln (F, Blank:10, 'OTHER');
BeyondOut (F, ' One-Time  ', ApproReport^.Costs.Other.OneTime,
  Adjust1000, 8, 0);
BeyondOut (F, 'Land Purch ', ApproReport^.Costs.LandPurch,
  Adjust1000, 8, 0);
Writeln (F);
Write (F, Blank:10, 'TOT COSTS  ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.MilCon.Year[Y]
    + ApproReport^.Costs.FamHous.Construction.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
    + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
    + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
    + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
    + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
    + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
    + ApproReport^.Costs.OM.Unemployment.Year[Y]
    + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
    + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
    + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
    + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
```

```
    + ApproReport^.Costs.OM.Other.OneTimeMove.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
    + ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
    + ApproReport^.Costs.HomeownAsst.Year[Y]
    + ApproReport^.Costs.EnvirMitig.Year[Y]
    + ApproReport^.Costs.InfoMgt.Year[Y]
    + ApproReport^.Costs.Other.OneTime.Year[Y]
    + ApproReport^.Costs.LandPurch.Year[Y]
    + ApproReport^.Costs.Procure.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F);

Title (F, 'APPRO. DETAIL FIXED COSTS', 2);
Write (F, Blank:10, 'SAVINGS($K)');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F);
Write (F, Rept('-',11):21);
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F);
Writeln (F, Blank:10, 'MILCON');
BeyondOut (F, ' Cost Avoid', ApproReport^.Savings.MilCon, Adjust1000, 8, 0);
Writeln (F, Blank:10, 'FAM HOUSING');
BeyondOut (F, ' Construct ', ApproReport^.Savings.FamHous.Construction,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, 'O&M');
BeyondOut (F, ' Civ Moving', ApproReport^.Savings.OM.CivMoving,
  Adjust1000, 8, 0);
BeyondOut (F, ' Freight   ', ApproReport^.Savings.OM.Freight,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, 'MIL PERSONNEL');
BeyondOut (F, ' Moving    ', ApproReport^.Savings.MilPers.MilMoving,
  Adjust1000, 8, 0);
BeyondOut (F, 'Procuremnt*', ApproReport^.Savings.Procure,
  Adjust1000, 8, 0);
BeyondOut (F, 'HAP / RSE  ', ApproReport^.Savings.HomeownAsst,
  Adjust1000, 8, 0);
BeyondOut (F, 'Envir Mitig', ApproReport^.Savings.EnvirMitig,
  Adjust1000, 8, 0);
BeyondOut (F, 'Info Manage', ApproReport^.Savings.InfoMgt,
  Adjust1000, 8, 0);
BeyondOut (F, 'LandRevenue', ApproReport^.Savings.LandRevenue,
  Adjust1000, 8, 0);
Writeln (F);
Write (F, Blank:10, 'TOT SAVINGS');
```

```
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Savings.MilCon.Year[Y]
    + ApproReport^.Savings.MilPers.MilMoving.Year[Y]
    + ApproReport^.Savings.Procure.Year[Y]
    + ApproReport^.Savings.LandRevenue.Year[Y]) * Adjust1000.Year[Y],0):8);
writeln (F);
Writeln (F);
Writeln (F);

Write (F, Blank:10, 'NETCOST($K)');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F);
Write (F, Rept('-',11):21);
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F);
BeyondOut (F, 'MilCon     ', ApproReport^.Net.MilCon, Adjust1000, 8, 0);
Writeln (F, Blank:10, 'FAM HOUSING');
BeyondOut (F, ' Constructn', ApproReport^.Costs.FamHous.Construction,
  Adjust1000, 8, 0);
Writeln (F, Blank:10, 'O&M');
Write (F, Blank:10, ' Civ Salary');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F);
Write (F, Blank:10, ' Civ Moving');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F);
Write (F, Blank:10, ' Freight   ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.Freight.Pack.Year[Y]
    + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
    + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
    + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
    + ApproReport^.Costs.OM.Freight.LossRate.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F);
BeyondOut (F, ' Unemploymt', ApproReport^.Costs.OM.Unemployment,
  Adjust1000, 8, 0);
Write (F, Blank:10, ' Other     ');
```

```
For Y := 1 to MaxYears DO
  write (F, WithCommas ((
      ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
    + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
    + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
    + ApproReport^.Costs.OM.Other.NewHire.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F);
Writeln (F, Blank:10, 'MIL PERSONNEL');
Write (F, Blank:10, ' Moving    ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((
      ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
    - ApproReport^.Savings.MilPers.MilMoving.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F);
BeyondOut (F, ' Other     ', ApproReport^.Costs.MilPers.Other.HousAllow,
  Adjust1000, 8, 0);
Writeln (F);
Write (F, Blank:10, 'Procuremnt*');
For Y := 1 to MaxYears DO
  write (F, WithCommas((ApproReport^.Costs.Procure.Year[Y]
                      - ApproReport^.Savings.Procure.Year[Y])
                      * Adjust1000.Year[Y],0):8);
Writeln (F);
BeyondOut (F, 'HAP / RSE  ', ApproReport^.Costs.HomeownAsst,
  Adjust1000, 8, 0);
BeyondOut (F, 'Envir Mitig', ApproReport^.Costs.EnvirMitig,
  Adjust1000, 8, 0);
BeyondOut (F, 'Info Manage', ApproReport^.Costs.InfoMgt, Adjust1000, 8, 0);
Writeln (F, Blank:10, 'OTHER');
BeyondOut (F, ' One-Time  ', ApproReport^.Costs.Other.OneTime,
  Adjust1000, 8, 0);
Write (F, Blank:10, 'Land       ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.LandPurch.Year[Y]
    - ApproReport^.Savings.LandRevenue.Year[Y])* Adjust1000.Year[Y],0):8);
Writeln (F);
Writeln (F);
Write (F, Blank:10, 'TOT NETCOST');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((
      ApproReport^.Costs.MilCon.Year[Y]
    - ApproReport^.Savings.MilCon.Year[Y]
    + ApproReport^.Costs.FamHous.Construction.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
```

```
      + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
      + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
      + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
      + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
      + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
      + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
      + ApproReport^.Costs.OM.Unemployment.Year[Y]
      + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
      + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
      + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
      + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
      + ApproReport^.Costs.OM.Other.OneTimeMove.Year[Y]
      + ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
      + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
      + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
      + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
      + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
      - ApproReport^.Savings.MilPers.MilMoving.Year[Y]
      + ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
      + ApproReport^.Costs.Procure.Year[Y]
      - ApproReport^.Savings.Procure.Year[Y]
      + ApproReport^.Costs.HomeownAsst.Year[Y]
      + ApproReport^.Costs.EnvirMitig.Year[Y]
      + ApproReport^.Costs.InfoMgt.Year[Y]
      + ApproReport^.Costs.Other.OneTime.Year[Y]
      + ApproReport^.Costs.LandPurch.Year[Y]
      - ApproReport^.Savings.LandRevenue.Year[Y])
      * Adjust1000.Year[Y],0):8);
  Writeln (F);
  Writeln (F);
  Writeln (F, Blank:10, '* Procurement values continue into Beyond years.');
  close (F)
END;  { ApproDetailFixedReport }
```

```
{*****************************************************************************
 *   Name           : ApproDetailSteadyStateReport
 *
 *   Purpose        : Writes appropriations detail steady-state costs report
 *
 *   Notes          :
 *
 *   Invoked As     : AppropriationsDetailSteadyStateReport
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     APPDETSS.RPT           Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (titles)
 *   10Apr92   AML   Streamlining
 *****************************************************************************}

PROCEDURE ApproDetailSteadyStateReport (ApproReport : ApproReportRec;
                                        Adjust1000  : Year1ToMaxRealRec);

  VAR
    Buffer : IOBuffer;
    F      : TEXT;
    Y      : INTEGER;

  Begin { ApproDetailSteadyStateReport }
    assign (F, Info.ReportDir + 'APPDETSS.RPT');
    SetTextBuf (F, Buffer);
    reWrite (F);

    Title (F, 'APP. DETAIL, STEADY-STATE COSTS', -1);
    Write (F, Blank:10, 'COSTS($K)   ');
    For Y := 1 To MaxYears Do
      write (F, FiscalYear(Y):8);
    writeln (F, '  Beyond');
    Write (F, Blank:10, '---------  ');
    For Y := 0 To MaxYears - 1 Do
      write (F, '    ----');
    writeln (F, '  ------');
    Writeln (F, Blank:10, 'FAM HOUSING');
    BeyondOut (F, ' Operations', ApproReport^.Costs.FamHous.Operations,
      Adjust1000, 8, 1);
```

```
Writeln (F, Blank:10, 'O&M');
BeyondOut (F, ' RPMA       ', ApproReport^.Costs.OM.RPMA, Adjust1000, 8, 1);
BeyondOut (F, ' BOS        ', ApproReport^.Costs.OM.BOS, Adjust1000, 8, 1);
BeyondOut (F, ' UniqOperat', ApproReport^.Costs.OM.OperatingIDF,
  Adjust1000, 8, 1);
BeyondOut (F, ' CHAMPUS    ', ApproReport^.Costs.OM.Champus,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, ' OTHER');
BeyondOut (F, '  Caretaker', ApproReport^.Costs.OM.Other.Caretaker,
  Adjust1000, 8, 1);
BeyondOut (F, '  Unique    ', ApproReport^.Costs.OM.Other.OtherIDF,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'OTHER');
BeyondOut (F, ' Mission    ', ApproReport^.Costs.Other.Mission,
  Adjust1000, 8, 1);
BeyondOut (F, 'Misc Recur ', ApproReport^.Costs.MiscRecur,
  Adjust1000, 8, 1);
Writeln (F);
Write (F, Blank:10, 'TOT COSTS  ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.FamHous.Operations.Year[Y]
    + ApproReport^.Costs.OM.RPMA.Year[Y]
    + ApproReport^.Costs.OM.BOS.Year[Y]
    + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
    + ApproReport^.Costs.OM.Champus.Year[Y]
    + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
    + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
    + ApproReport^.Costs.MiscRecur.Year[Y]
    + ApproReport^.Costs.Other.Mission.Year[Y])* Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.FamHous.Operations.Beyond
    + ApproReport^.Costs.OM.RPMA.Beyond
    + ApproReport^.Costs.OM.BOS.Beyond
    + ApproReport^.Costs.OM.OperatingIDF.Beyond
    + ApproReport^.Costs.OM.Champus.Beyond
    + ApproReport^.Costs.OM.Other.Caretaker.Beyond
    + ApproReport^.Costs.OM.Other.OtherIDF.Beyond
    + ApproReport^.Costs.MiscRecur.Beyond
    + ApproReport^.Costs.Other.Mission.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
Writeln (F);
Writeln (F);

Write (F, Blank:10, 'SAVINGS($K)');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F, '  Beyond');
Write (F, Rept('-',11):21);
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F, '  ------');
Writeln (F, Blank:10, 'FAM HOUSING');
```

```
BeyondOut (F, ' Operations', ApproReport^.Savings.FamHous.Operations,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'O&M');
BeyondOut (F, ' RPMA      ', ApproReport^.Savings.OM.RPMA,
  Adjust1000, 8, 1);
BeyondOut (F, ' BOS       ', ApproReport^.Savings.OM.BOS, Adjust1000, 8, 1);
BeyondOut (F, ' UniqOperat', ApproReport^.Savings.OM.OperatingIDF,
  Adjust1000, 8, 1);
BeyondOut (F, ' Civ Salary', ApproReport^.Savings.OM.CivSalary,
  Adjust1000, 8, 1);
BeyondOut (F, ' CHAMPUS   ', ApproReport^.Savings.OM.Champus,
  Adjust1000, 8, 1);
BeyondOut (F, ' UniqOther ', ApproReport^.Savings.OM.OtherIDF,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'MIL PERSONNEL');
Writeln (F, Blank:10, ' SALARY');
BeyondOut (F, '  Officer  ', ApproReport^.Savings.MilPers.MilSalary.Off,
  Adjust1000, 8, 1);
BeyondOut (F, '  Enlisted ', ApproReport^.Savings.MilPers.MilSalary.Enl,
  Adjust1000, 8, 1);
BeyondOut (F, ' Hous Allow', ApproReport^.Savings.MilPers.Other.HousAllow,
  Adjust1000, 8, 1);
Writeln (F, Blank:10, 'OTHER');
BeyondOut (F, ' Mission   ', ApproReport^.Savings.Other, Adjust1000, 8, 1);
BeyondOut (F, 'Misc Recur ', ApproReport^.Savings.MiscRecur,
  Adjust1000, 8, 1);
Writeln (F);
Write (F, Blank:10, 'TOT SAVINGS');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Savings.FamHous.Operations.Year[Y]
    + ApproReport^.Savings.OM.RPMA.Year[Y]
    + ApproReport^.Savings.OM.BOS.Year[Y]
    + ApproReport^.Savings.OM.OperatingIDF.Year[Y]
    + ApproReport^.Savings.OM.CivSalary.Year[Y]
    + ApproReport^.Savings.OM.Champus.Year[Y]
    + ApproReport^.Savings.OM.OtherIDF.Year[Y]
    + ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
    + ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
    + ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y]
    + ApproReport^.Savings.MiscRecur.Year[Y]
    + ApproReport^.Savings.Other.Year[Y])* Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Savings.FamHous.Operations.Beyond
    + ApproReport^.Savings.OM.RPMA.Beyond
    + ApproReport^.Savings.OM.BOS.Beyond
    + ApproReport^.Savings.OM.OperatingIDF.Beyond
    + ApproReport^.Savings.OM.CivSalary.Beyond
    + ApproReport^.Savings.OM.Champus.Beyond
    + ApproReport^.Savings.OM.OtherIDF.Beyond
    + ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
    + ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond
    + ApproReport^.Savings.MilPers.Other.HousAllow.Beyond
    + ApproReport^.Savings.MiscRecur.Beyond
```

```
    + ApproReport^.Savings.Other.Beyond)* Adjust1000.Year[MaxYears],0):8);

Title (F, 'APP DET, STEADY-STATE COSTS',2);
Write (F, Blank:10, 'NETCOST($K)');
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):8);
writeln (F, '  Beyond');
Write (F, Rept('-',11):21);
For Y := 0 To MaxYears - 1 Do
  write (F, '    ----');
writeln (F, '  ------');
Writeln (F, Blank:10, 'FAM HOUSING');
Write (F, Blank:10, ' Operations');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.FamHous.Operations.Year[Y]
    - ApproReport^.Savings.FamHous.Operations.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.FamHous.Operations.Beyond
    - ApproReport^.Savings.FamHous.Operations.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
Writeln (F, Blank:10, 'O&M');
Write (F, Blank:10, ' RPMA      ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.RPMA.Year[Y]
    - ApproReport^.Savings.OM.RPMA.Year[Y]) * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.OM.RPMA.Beyond
    - ApproReport^.Savings.OM.RPMA.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
Write (F, Blank:10, ' BOS       ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.BOS.Year[Y]
    - ApproReport^.Savings.OM.BOS.Year[Y]) * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.OM.BOS.Beyond
    - ApproReport^.Savings.OM.BOS.Beyond) * Adjust1000.Year[MaxYears],0):8);
Write (F, Blank:10, ' UniqOperat');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.OperatingIDF.Year[Y]
    - ApproReport^.Savings.OM.OperatingIDF.Year[Y])
    * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.OM.OperatingIDF.Beyond
    - ApproReport^.Savings.OM.OperatingIDF.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
Write (F, Blank:10, ' Civ Salary');
For Y := 1 to MaxYears DO
  write (F, WithCommas
    (-ApproReport^.Savings.OM.CivSalary.Year[Y] * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas
    (-ApproReport^.Savings.OM.CivSalary.Beyond
    * Adjust1000.Year[MaxYears],0):8);
Write (F, Blank:10, ' CHAMPUS   ');
```

```
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.Champus.Year[Y]
    - ApproReport^.Savings.OM.Champus.Year[Y]) * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.OM.Champus.Beyond
    - ApproReport^.Savings.OM.Champus.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
Writeln (F, Blank:10, ' OTHER');
BeyondOut (F, '  Caretaker', ApproReport^.Costs.OM.Other.Caretaker,
  Adjust1000, 8, 1);
Write (F, Blank:10, '  Unique   ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
    - ApproReport^.Savings.OM.OtherIDF.Year[Y]) * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.OM.Other.OtherIDF.Beyond
    - ApproReport^.Savings.OM.OtherIDF.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
Writeln (F, Blank:10, 'MIL PERSONNEL');
Writeln (F, Blank:10, ' SALARY');
Write (F, Blank:10, '  Officer  ');
For Y := 1 to MaxYears DO
  write (F, WithCommas (-ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
    * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas (-ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
    * Adjust1000.Year[MaxYears],0):8);
Write (F, Blank:10, '  Enlisted ');
For Y := 1 to MaxYears DO
  write (F, WithCommas (-ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
    * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas (-ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond
    * Adjust1000.Year[MaxYears],0):8);
Write (F, Blank:10, ' Hous Allow');
For Y := 1 to MaxYears DO
  write (F, WithCommas (
    - ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y]
    * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas (-ApproReport^.Savings.MilPers.Other.HousAllow.Beyond
    * Adjust1000.Year[MaxYears],0):8);
Writeln (F, Blank:10, 'OTHER');
Write (F, Blank:10, ' Mission   ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.Other.Mission.Year[Y]
    - ApproReport^.Savings.Other.Year[Y]) * Adjust1000.Year[Y],0):8);
Writeln (F, WithCommas ((ApproReport^.Costs.Other.Mission.Beyond
    - ApproReport^.Savings.Other.Beyond) * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, 'Misc Recur ', ApproReport^.Net.MiscRecur, Adjust1000, 8, 1);
Writeln (F);
Write (F, Blank:10, 'TOT NETCOST');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.FamHous.Operations.Year[Y]
    + ApproReport^.Costs.OM.RPMA.Year[Y]
    + ApproReport^.Costs.OM.BOS.Year[Y]
    + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
```

```
        + ApproReport^.Costs.OM.Champus.Year[Y]
        + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
        + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
        + ApproReport^.Costs.Other.Mission.Year[Y]
        + ApproReport^.Net.MiscRecur.Year[Y]
        - ApproReport^.Savings.FamHous.Operations.Year[Y]
        - ApproReport^.Savings.OM.RPMA.Year[Y]
        - ApproReport^.Savings.OM.BOS.Year[Y]
        - ApproReport^.Savings.OM.OperatingIDF.Year[Y]
        - ApproReport^.Savings.OM.CivSalary.Year[Y]
        - ApproReport^.Savings.OM.Champus.Year[Y]
        - ApproReport^.Savings.OM.OtherIDF.Year[Y]
        - ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
        - ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
        - ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y]
        - ApproReport^.Savings.Other.Year[Y]) * Adjust1000.Year[Y],0):8);
    Writeln (F, WithCommas ((ApproReport^.Costs.FamHous.Operations.Beyond
        + ApproReport^.Costs.OM.RPMA.Beyond
        + ApproReport^.Costs.OM.BOS.Beyond
        + ApproReport^.Costs.OM.OperatingIDF.Beyond
        + ApproReport^.Costs.OM.Champus.Beyond
        + ApproReport^.Costs.OM.Other.Caretaker.Beyond
        + ApproReport^.Costs.OM.Other.OtherIDF.Beyond
        + ApproReport^.Costs.Other.Mission.Beyond
        + ApproReport^.Net.MiscRecur.Beyond
        - ApproReport^.Savings.FamHous.Operations.Beyond
        - ApproReport^.Savings.OM.RPMA.Beyond
        - ApproReport^.Savings.OM.BOS.Beyond
        - ApproReport^.Savings.OM.OperatingIDF.Beyond
        - ApproReport^.Savings.OM.CivSalary.Beyond
        - ApproReport^.Savings.OM.Champus.Beyond
        - ApproReport^.Savings.OM.OtherIDF.Beyond
        - ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
        - ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond
        - ApproReport^.Savings.MilPers.Other.HousAllow.Beyond
        - ApproReport^.Savings.Other.Beyond) * Adjust1000.Year[MaxYears],0):8);
    close (F)
  END;  { ApproDetailSteadyStateReport }

END. { UNIT }
```

```
UNIT CRPTOUT2;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  CDEC,
  CSUP;

PROCEDURE DeltaChartReport   (DeltaChart        : DeltaChartRec);

PROCEDURE SummaryReport      (MissionReport     : CostSaveNetByYearRec;
                              PersFinal         : CostSaveNetByYearRec;
                              MilConFinal       : MilConFinalRec;
                              MovCostFinal      : YearAndBeyondRealRec;
                              OtherCostFinal    : CostSaveNetByYearRec;
                              OvhdCostFinal     : CostSaveNetByYearRec;
                              CalcFctrsPerPos   : CalcPersPosRec;
                              TotalOneTimeCost  : Real;
                              PersMoves         : CalcMovesRec;
                              MilMov            : Year1ToMaxRealRec;
                              Adjust1000        : Year1ToMaxRealRec);

PROCEDURE ApproSummaryReport (ApproReport       : ApproReportRec;
                              Adjust1000        : Year1ToMaxRealRec);

Procedure ApproFinanceReport (ApproReport       : ApproReportRec;
                              CalcFctrsPerPos   : CalcPersPosRec);

IMPLEMENTATION
{*************************************************************************
 *   Name           : DeltaChartItem
 *
 *   Purpose        : Writes an item for the Delta Chart Report
 *
 *   Notes          : Since the code for each item on the Delta Chart was so
 *                    similar, it made sense to have one procedure for
 *                    all of them.
 *
 *   Invoked As     : DeltaChartItem (Fil, DeltaLabel, DeltaItem, DeltaPers);
 *
 *   Return value   :
 *
 *   Constraints
 *     Development  : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production   : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ---------------------------------------------------------
 *   02Jun92  AML   Creation
 *************************************************************************}
```

```
Procedure DeltaChartItem (Var Fil        : Text;
                              DeltaLabel : String10;
                              DeltaItem,
                              DeltaPers  : DeltaChartSubRec1);

  Var
    B : Integer;

  Begin
    writeln (Fil);
    writeln (Fil, Blank:(48 - ((Length(DeltaLabel) + 1) Div 2)), DeltaLabel);
    Write (Fil, Blank:10, 'Base', Blank:26, 'Change    %Chng');
    If Pos ('($)', DeltaLabel) > 0
      Then DeltaLabel := Copy (DeltaLabel, 1, Length(DeltaLabel) - 3);
    If DeltaLabel = 'Personnel'
      Then Writeln (Fil)
      Else Writeln (Fil, DeltaLabel:10, '/Pers');
    Write (Fil, Blank:10, '----', Blank:26, '------    -----');
    If DeltaLabel = 'Personnel'
      Then Writeln (Fil)
      Else Writeln (Fil, Rept ('-', Length(DeltaLabel) + 5):15);
    For B := 0 To MaxBases Do
      Begin
        Write (Fil, Blank:10, Screen1Data^.Base[B].Name,
          Blank:(21 - Length (Screen1Data^.Base[B].Name)),
          WithCommas(DeltaItem.Base[B].Delta,0):15,
          (DeltaItem.Base[B].PercChange*100.0):8:2, '%');
        If DeltaLabel = 'Personnel'
          Then Writeln (Fil)
          Else Writeln (Fil, WithCommas(ZeroCheck(
            DeltaItem.Base[B].Delta,
            DeltaPers.Base[B].Delta),2):15);
      End;
   writeln (Fil);
  End;
```

```
{*************************************************************************
 *   Name          : DeltaChartReport
 *
 *   Purpose       : Writes a report of the deltas and percentage changes in
 *                   BOS, personnel, land, SF, and RPMA for each base to a file
 *
 *   Notes         :
 *
 *   Invoked As    : DeltaChartReport (DeltaChart)
 *
 *   Return value  :
 *
 *   Files/Tables
 *     Names                Access    Purpose
 *     ------------------   --------  ----------------------------------------
 *     DELTAS.RPT           Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *************************************************************************}

PROCEDURE DeltaChartReport (DeltaChart : DeltaChartRec);

  VAR
    B      : INTEGER;
    Buffer : IOBuffer;
    F      : TEXT;

  Begin { DeltaChartReport }
    assign (F, Info.ReportDir + 'DELTAS.RPT');
    SetTextBuf (F, Buffer);
    reWrite (F);

    Title (F, 'BOS, PERSONNEL, LAND, SF, AND RPMA DELTAS', -1);
    DeltaChartItem (F, 'Personnel', DeltaChart^.Pers, DeltaChart^.Pers);
    DeltaChartItem (F, 'BOS($)', DeltaChart^.BOS, DeltaChart^.Pers);
    Title (F, 'DELTA CHART REPORT', 2);
    DeltaChartItem (F, 'RPMA($)', DeltaChart^.RPMA, DeltaChart^.Pers);
    DeltaChartItem (F, 'RPMABOS($)', DeltaChart^.RPMABOS, DeltaChart^.Pers);
    Title (F, 'DELTA CHART REPORT', 3);
    DeltaChartItem (F, 'Land', DeltaChart^.Land, DeltaChart^.Pers);
    DeltaChartItem (F, 'SF', DeltaChart^.SF, DeltaChart^.Pers);
    Close (F)
  END;  { DeltaChartReport }
```

```
{*****************************************************************************
 *   Name           : CalcNPV
 *
 *   Purpose        : Calculates net present values, payback year, and breakeven
 *                    year
 *
 *   Notes          :
 *
 *   Invoked As     : CalcNPV (Cost, BreakEvenYear, PaybackYear, OptionNPV)
 *
 *   Return values :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, Titles, etc.)
 *   08Apr92   AML   Streamlining
 *****************************************************************************}

PROCEDURE CalcNPV (     NetForSummary : YearAndBeyondRealRec;
                    VAR BreakEvenYear : INTEGER;
                    VAR PaybackYear   : INTEGER;
                    VAR OptionNPV     : REAL);

  VAR
    Buffer        : IOBuffer;
    Finished, Sw  : BOOLEAN;
    Y, Y2         : INTEGER;
    NPVTable      : NPVTableRec;
    CostThisYear,
    InflThisYear,
    NPVLastYear,
    NPVThisYear,
    Temp          : REAL;
    F             : TEXT;

  Begin { CalcNPV }
    BreakEvenYear := 0;
    OptionNPV     := 0.0;
    PaybackYear   := 0;

    assign  (F, Info.ReportDir + 'NPV.RPT');
    SetTextBuf (F, Buffer);
    rewrite (F);
```

```
Title (F, 'NET PRESENT VALUES REPORT', 1);
writeln (F, BLANK:10, 'Year',
            BLANK:11, 'Cost($)',
            BLANK:06, 'Inflated Cost($)',
            BLANK:10, 'NPV($)');
writeln (F);

FOR Y := 1 to NUMYEARS DO
  Begin
    IF (Y < 7)
      Then NPVTable.Cost.Year[Y] := NetForSummary.Year[Y]
      ELSE NPVTable.Cost.Year[Y] := NetForSummary.Beyond;
    NPVTable.Inflated.Year[Y] :=
        NPVTable.Cost.Year[Y]
      * YtoX (1.0 + StdFctrsFac.InflationNPV, Y - 0.5);
    NPVTable.NPV.Year[Y] :=
        NPVTable.Inflated.Year[Y]
      / YtoX (1.0 + StdFctrsFac.DiscountNPV, Y - 0.5);
    If Y > 1
      Then NPVTable.NPV.Year[Y] :=
           NPVTable.NPV.Year[Y]
         + NPVTable.NPV.Year[Y-1];
    writeln (F, FiscalYear(Y):14,
      WithCommas (NPVTable.Cost.Year[Y], 0):18,
      WithCommas (NPVTable.Inflated.Year[Y], 0):19,
      WithCommas (NPVTable.NPV.Year[Y], 0):19)
  END;

{ Set OptionNPV = NPV at year NUMYEARS }

OptionNPV := NPVTable.NPV.Year[NUMYEARS];

{ Initialize switches to continue calculating }

Finished  := False;
Sw        := False;

IF (NPVTable.NPV.Year[NUMYEARS] > 0.0) and
   (NPVTable.Cost.Year[MaxYears+1] > 0.0)
  Then Begin
    BreakEvenYear := 999;                    { Will never pay back }
    PaybackYear   := 999;
    Finished      := True
  END
ELSE
  IF NPVTable.NPV.Year[NUMYEARS] < 0.0 THEN
    Begin
      Y := NUMYEARS;                          { Has already paid back }
      WHILE NPVTable.NPV.Year[Y] < 0.0 DO
        Begin
          Dec (Y);
          Sw := True
```

```
        END
      END;

  IF not Finished THEN
    IF Sw THEN
      Begin
        BreakEvenYear := Y;
        PaybackYear   := BreakEvenYear - LastYear
      END
    ELSE
      Begin                                          { Has not paid back, }
        Y := NumYears + 1;                           { but will, so check }
        NPVLastYear  := NPVTable.NPV.Year[NUMYEARS]; { for up to 100 yrs  }
        WHILE not Finished DO
          Begin
            CostThisYear := NetForSummary.Beyond;
            InflThisYear := CostThisYear
              * YtoX (1.0 + StdFctrsFac.InflationNPV, Y - 0.5);
            NPVThisYear  := NPVLastYear
              + (InflThisYear
              / YtoX (1.0 + StdFctrsFac.DiscountNPV, Y - 0.5));
            writeln (F, FiscalYear(Y):14,
              WithCommas (CostThisYear,0):18,
              WithCommas (InflThisYear,0):19,
              WithCommas (NPVThisYear,0):19);
            If Y = 49
              Then Title (F, 'NET PRESENT VALUES REPORT', 2);
            IF Y = 101 THEN
              Begin
                BreakEvenYear := 100;
                PaybackYear   := 100;
                Finished      := True
              END
            ELSE
              IF NPVThisYear < 0.0 THEN
                Begin
                  BreakEvenYear := Y - 1;
                  PaybackYear   := BreakEvenYear - LastYear;
                  Finished      := True
                END
              ELSE
                Begin
                  NPVLastYear   := NPVThisYear;
                  Inc (Y);
                END
          END
      END;

  { Do not allow BreakEvenYear or PaybackYear to be negative }

  BreakEvenYear := Round (AtLeastZero (BreakEvenYear));
  PayBackYear   := Round (AtLeastZero (PayBackYear));
```

```
  Close (F)
END;  { CalcNPV }

{*****************************************************************************
 *   Name            : SummaryTable
 *
 *   Purpose         : Writes summary table
 *
 *   Notes           : After Savings and Costs were added to COBSUM.RPT, this one
 *                     procedure was created to generate Cost, Savings, and Net
 *                     Cost tables.
 *
 *   Invoked As      : SummaryTable (BreakEvenYear, PaybackYear, OptionNPV,
 *                                   MissionCost, PersFinal, MilConFinal,
 *                                   MovCostFinal, OtherCostFinal,
 *                                   OvhdCostFinal, NetForSummary,
 *                                   CalcFctrsPerPos, TotalOneTimeCost)
 *
 *   Return value    :
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   01Jun92   AML   Creation
 *****************************************************************************}

Procedure SummaryTable (Var Fil        : Text;
                            Header     : String10;
                            Misn,
                            Pers,
                            Ovhd       : YearAndBeyondRealRec;
                            Cons       : Year1ToMaxRealRec;
                            Movg,
                            Othr       : YearAndBeyondRealRec;
                            Adjust1000 : Year1ToMaxRealRec);
 Var
   Y : Integer;

 Begin
   Writeln (Fil);
   Writeln (Fil, Blank:10, Header, ' ($K) Constant Dollars');
   FiscOut (Fil, 14, 8, 1);
   writeln (Fil, BLANK:14, Rept ('   -----', 6), '  ------');
   BeyondOut (Fil, 'Misn', Misn, Adjust1000, 8, 1);
   BeyondOut (Fil, 'Pers', Pers, Adjust1000, 8, 1);
   BeyondOut (Fil, 'Ovhd', Ovhd, Adjust1000, 8, 1);
   YearOut   (Fil, 'Cons', Cons, Adjust1000, 8, 1);
   BeyondOut (Fil, 'Movg', Movg, Adjust1000, 8, 1);
   BeyondOut (Fil, 'Othr', Othr, Adjust1000, 8, 1);
```

```
    Writeln (Fil);
    write (Fil, Blank:10, 'TOT ');
    FOR Y := 1 to MaxYears DO
      write (Fil, WithCommas (Within(-9999999.0,(
          Misn.Year[Y]
        + Pers.Year[Y]
        + Ovhd.Year[Y]
        + Cons.Year[Y]
        + Movg.Year[Y]
        + Othr.Year[Y])
        * Adjust1000.Year[Y],9999999.0),0):8);
    writeln (Fil, WithCommas (Within(-9999999.0,(
          Misn.Beyond
        + Pers.Beyond
        + Ovhd.Beyond
        + Othr.Beyond)
        * Adjust1000.Year[MaxYears],9999999.0),0):8);
    Writeln (Fil);
  End;

{*****************************************************************************
 *   Name          : SummaryReport
 *
 *   Purpose       : Writes summary report to a file
 *
 *   Notes         :
 *
 *   Invoked As    : SummaryReport (BreakEvenYear, PaybackYear, OptionNPV,
 *                                  MissionCost, PersFinal, MilConFinal,
 *                                  MovCostFinal, OtherCostFinal,
 *                                  OvhdCostFinal, NetForSummary,
 *                                  CalcFctrsPerPos, TotalOneTimeCost)
 *
 *   Return value  :
 *
 *   Files/Tables
 *     Names                 Access    Purpose
 *     ------------------    --------  ----------------------------------------
 *     COBSUM.RPT            Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *   01Jun92   AML   Added Savings and Costs
 *****************************************************************************}
```

```
PROCEDURE SummaryReport (     MissionReport     : CostSaveNetByYearRec;
                              PersFinal         : CostSaveNetByYearRec;
                              MilConFinal       : MilConFinalRec;
                              MovCostFinal      : YearAndBeyondRealRec;
                              OtherCostFinal    : CostSaveNetByYearRec;
                              OvhdCostFinal     : CostSaveNetByYearRec;
                              CalcFctrsPerPos   : CalcPersPosRec;
                              TotalOneTimeCost  : REAL;
                              PersMoves         : CalcMovesRec;
                              MilMov            : Year1ToMaxRealRec;
                              Adjust1000        : Year1ToMaxRealRec);

  VAR
    B, C, Y,
    BreakEvenYear,
    PayBackYear     : INTEGER;
    Buffer          : IOBuffer;
    OffSum, EnlSum,
    CivSum, StuSum,
    OptionNPV       : Real;
    F               : TEXT;
    Civ, Enl,
    Off, Stu        : Year1ToMaxIntegerRec;
    NetForSummary,
    TempMovCost,
    TempMovSave     : YearAndBeyondRealRec;
    TempMilConCost  : Year1ToMaxRealRec;

  Begin { SummaryReport }
    FillChar (NetForSummary, SizeOf (NetForSummary), 0);
    For Y := 1 To MaxYears Do
      NetForSummary.Year[Y] :=
          MissionReport.Net.Year[Y]
        + PersFinal.Net.Year[Y]
        + MilConFinal.Net.Year[Y]
        + MovCostFinal.Year[Y]
        + OtherCostFinal.Net.Year[Y]
        + OvhdCostFinal.Net.Year[Y];
    NetForSummary.Beyond :=
        MissionReport.Net.Beyond
      + PersFinal.Net.Beyond
      + OtherCostFinal.Net.Beyond
      + OvhdCostFinal.Net.Beyond;
    CalcNPV (NetForSummary, BreakEvenYear, PaybackYear, OptionNPV);

    assign (F, Info.ReportDir + 'COBSUM.RPT');
    SetTextBuf (F, Buffer);
    rewrite (F);
    Title (F, 'COBRA REALIGNMENT SUMMARY', -1);
    OptionNPV :=
      Within (-9999999.0, OptionNPV / 1000.0, 9999999.0);
```

```
TotalOneTimeCost :=
  Within (-9999999.0, TotalOneTimeCost / 1000.0, 9999999.0);
writeln (F, Blank:10, 'Starting Year  : ', Screen1Data^.FY1:4);
write (F, Blank:10, 'Break Even Year: ');
If BreakEvenYear = 0
  Then writeln (F, 'Immediate')
Else If BreakEvenYear = 999
  Then writeln (F, 'Never')
Else
  Begin
    Write (F, (Screen1Data^.FY1 + BreakEvenYear):4);
    If BreakEvenYear = 100
      Then Write (F, '+ (Year ')
      Else Write (F, '  (Year ');
    Write (F, BreakEvenYear + 1);
    If BreakEvenYear = 100
      Then Writeln (F, '+)')
      Else Writeln (F, ')');
  End;
write (F, Blank:10, 'ROI Year       : ');
If PayBackYear = 0
  Then writeln (F, 'Immediate')
Else If PayBackYear = 999
  Then writeln (F, 'Never')
Else
  Begin
    Write (F, (Screen1Data^.FY1 + PayBackYear + LastYear):4);
    If PayBackYear = 100
      Then Write (F, '+ (')
      Else Write (F, '  (');
    Write (F, PayBackYear);
    If PayBackYear = 100
      Then Writeln (F, '+ Years)')
      Else Writeln (F, ' Years)');
  End;
writeln (F);
writeln (F, Blank:10, 'Option NPV in', FiscalYear (NumYears):5,
  '  ($K) :', WithCommas(OptionNPV,0):8);
writeln (F, Blank:10, 'Total One-Time Cost ($K) :',
  WithCommas(TotalOneTimeCost,0):8);

SummaryTable (F, 'Net Costs', MissionReport.Net, PersFinal.Net,
  OvhdCostFinal.Net, MilConFinal.Net, MovCostFinal, OtherCostFinal.Net,
  Adjust1000);

OffSum := 0;
EnlSum := 0;
CivSum := 0;
StuSum := 0;
```

```
writeln (F);
write (F, BLANK:20);
For Y := 1 To MaxYears Do
  write (F, FiscalYear(Y):7);
writeln (F, '   TOTAL');
writeln (F, BLANK:22,'-----  -----  -----  -----  -----  -----   -----');
writeln (F, Blank:10, 'FORCE STRUCTURE REDUCTIONS');
write (F, Blank:12, 'Officers');
FOR Y := 1 to MaxYears DO
  Begin
    write (F, WithCommas(CalcFctrsPerPos.OffPosReduc.Year[Y],0):7);
    OffSum := OffSum + CalcFctrsPerPos.OffPosReduc.Year[Y];
  End;
writeln (F, WithCommas(OffSum,0):8);
write (F, Blank:12, 'Enlisted');
FOR Y := 1 to MaxYears DO
  Begin
    write (F, WithCommas(CalcFctrsPerPos.EnlPosReduc.Year[Y],0):7);
    EnlSum := EnlSum + CalcFctrsPerPos.EnlPosReduc.Year[Y];
  End;
writeln (F, WithCommas(EnlSum,0):8);
write (F, Blank:12, 'Civilian');
FOR Y := 1 to MaxYears DO
  Begin
    write (F, WithCommas(CalcFctrsPerPos.CivPosReduc.Year[Y],0):7);
    CivSum := CivSum + CalcFctrsPerPos.CivPosReduc.Year[Y];
  End;
writeln (F, WithCommas(CivSum,0):8);

OffSum := 0;
EnlSum := 0;
CivSum := 0;
StuSum := 0;
writeln (F, Blank:10, 'POSITIONS ELIMINATED');
write (F, Blank:12, 'Officers');
FOR Y := 1 to MaxYears DO
  Begin
    write (F, WithCommas(CalcFctrsPerPos.OffPosElim.Year[Y],0):7);
    OffSum := OffSum + CalcFctrsPerPos.OffPosElim.Year[Y];
  End;
writeln (F, WithCommas(OffSum,0):8);
write (F, Blank:12, 'Enlisted');
FOR Y := 1 to MaxYears DO
  Begin
    write (F, WithCommas(CalcFctrsPerPos.EnlPosElim.Year[Y],0):7);
    EnlSum := EnlSum + CalcFctrsPerPos.EnlPosElim.Year[Y];
  End;
writeln (F, WithCommas(EnlSum,0):8);
write (F, Blank:12, 'Civilian');
FOR Y := 1 to MaxYears DO
```

```
    Begin
      write (F, WithCommas(CalcFctrsPerPos.CivPosElim.Year[Y],0):7);
      CivSum := CivSum + CalcFctrsPerPos.CivPosElim.Year[Y];
    End;
  writeln (F, WithCommas(CivSum,0):8);

  OffSum := 0;
  EnlSum := 0;
  CivSum := 0;
  StuSum := 0;
  FOR Y := 1 to ABSMAXYEARS DO
    Begin
      Civ.Year[Y] := 0;
      Enl.Year[Y] := 0;
      Off.Year[Y] := 0;
      Stu.Year[Y] := 0;
    END;
  FOR Y := 1 to MaxYears DO
    FOR B := 0 to MaxBases DO
      Begin
        Civ.Year[Y] := Civ.Year[Y]
          + PersMoves.Civ.Into.Base[B].Year[Y];
        Enl.Year[Y] := Enl.Year[Y]
          + PersMoves.Enl.Into.Base[B].Year[Y];
        Off.Year[Y] := Off.Year[Y]
          + PersMoves.Off.Into.Base[B].Year[Y];
        Stu.Year[Y] := Stu.Year[Y]
          + PersMoves.Stu.Into.Base[B].Year[Y];
      END;

  writeln (F, Blank:10, 'PERSONNEL REALIGNMENTS');
  write (F, Blank:12, 'Officers');
  FOR Y := 1 to MaxYears DO
    Begin
      write (F, WithCommas(Off.Year[Y],0):7);
      OffSum := OffSum + Off.Year[Y];
    End;
  writeln (F, WithCommas(OffSum,0):8);
  write (F, Blank:12, 'Enlisted');
  FOR Y := 1 to MaxYears DO
    Begin
      write (F, WithCommas(Enl.Year[Y],0):7);
      EnlSum := EnlSum + Enl.Year[Y];
    End;
  writeln (F, WithCommas(EnlSum,0):8);
  write (F, Blank:12, 'Students');
  FOR Y := 1 to MaxYears DO
    Begin
      write (F, WithCommas(Stu.Year[Y],0):7);
      StuSum := StuSum + Stu.Year[Y];
    End;
  writeln (F, WithCommas(StuSum,0):8);
```

```
    write (F, Blank:12, 'TOT MIL ');
    FOR Y := 1 to MaxYears DO
      write (F, WithCommas(Off.Year[Y] + Enl.Year[Y] + Stu.Year[Y],0):7);
    writeln (F, WithCommas(OffSum + EnlSum + StuSum,0):8);
    write (F, Blank:12, 'Civilian');
    FOR Y := 1 to MaxYears DO
      Begin
        write (F, WithCommas(Civ.Year[Y],0):7);
        CivSum := CivSum + Civ.Year[Y];
      End;
    writeln (F, WithCommas(CivSum,0):8);
    write (F, Blank:12, 'TOTAL   ');
    FOR Y := 1 to MaxYears DO
      write (F, WithCommas(Off.Year[Y] + Enl.Year[Y]
                          + Stu.Year[Y] + Civ.Year[Y],0):7);
    writeln (F, WithCommas(OffSum + EnlSum + StuSum + CivSum,0):8);
    writeln (F);
    If  Length(Screen1Data^.Summary[1]) + Length(Screen1Data^.Summary[2])
      + Length(Screen1Data^.Summary[3]) + Length(Screen1Data^.Summary[4])
      + Length(Screen1Data^.Summary[5]) + Length(Screen1Data^.Summary[6]) > 0
      Then
        Begin
          Writeln (F, Blank:10, 'Summary:');
          Writeln (F, Blank:10, '--------');
          For C := 1 To 6 Do
            writeln (F, Blank:10, Screen1Data^.Summary[C]);
        End;
    Title (F, 'COBRA REALIGNMENT SUMMARY', 2);
    For Y := 1 To MaxYears Do
      Begin
        TempMovSave.Year[Y] := -MilMov.Year[Y];
        TempMovCost.Year[Y] := MovCostFinal.Year[Y] - MilMov.Year[Y];
        TempMilConCost.Year[Y] := MilConFinal.Net.Year[Y]
                                + MilConFinal.CostAvoidConstr.Year[Y];
      End;
    TempMovCost.Beyond := 0.0;
    TempMovSave.Beyond := 0.0;

    SummaryTable (F, 'Costs', MissionReport.Cost, PersFinal.Cost,
      OvhdCostFinal.Cost, TempMilConCost, TempMovCost, OtherCostFinal.Cost,
      Adjust1000);

    SummaryTable (F, 'Savings', MissionReport.Save, PersFinal.Save,
      OvhdCostFinal.Save, MilConFinal.CostAvoidConstr, TempMovSave,
      OtherCostFinal.Save, Adjust1000);

    Close (F);
  END;  { SummaryReport }
```

```
{*****************************************************************************
 *   Name           : AppropriationsSummaryReport
 *
 *   Purpose        : Writes appropriations summary report
 *
 *   Notes          :
 *
 *   Invoked As     : AppropriationsSummaryReport (ApproReport)
 *
 *   Return value   :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     APPSUM.RPT             Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   27Mar92   AML   Adjusted Output (Commas in numbers, titles, etc.)
 *****************************************************************************}

PROCEDURE ApproSummaryReport (ApproReport : ApproReportRec;
                              Adjust1000  : Year1ToMaxRealRec);

  VAR
    Buffer : IOBuffer;
    F      : TEXT;
    Y      : INTEGER;

  Begin { AppropriationsSummaryReport }
    assign (F, Info.ReportDir + 'APPSUM.RPT');
    SetTextBuf (F, Buffer);
    reWrite (F);

    Title (F, 'APPROPRIATIONS SUMMARY', -1);
    FiscOut (F, 21, 8, 1);
    Write (F, Blank:21);
    For Y := 0 To MaxYears - 1 Do
      write (F, '    ----');
    writeln (F, '  ------');
    writeln (F, BLANK:10, 'COSTS ($K)');
    BeyondOut (F, ' MilCon    ', ApproReport^.Costs.MilCon, Adjust1000, 8, 1);
    writeln (F, Blank:11, 'FAM HOUSING');
    BeyondOut (F, '  Construct', ApproReport^.Costs.FamHous.Construction,
      Adjust1000, 8, 1);
    BeyondOut (F, '  Operation', ApproReport^.Costs.FamHous.Operations,
      Adjust1000, 8, 1);
```

```
write (F, Blank:11, 'O&M        ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.OM.RPMA.Year[Y]
    + ApproReport^.Costs.OM.BOS.Year[Y]
    + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
    + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
    + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
    + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
    + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
    + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
    + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
    + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
    + ApproReport^.Costs.OM.Champus.Year[Y]
    + ApproReport^.Costs.OM.Unemployment.Year[Y]
    + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
    + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
    + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
    + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
    + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
    + ApproReport^.Costs.OM.Other.OneTimeMove.Year[Y]
    + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Costs.OM.RPMA.Beyond
    + ApproReport^.Costs.OM.BOS.Beyond
    + ApproReport^.Costs.OM.OperatingIDF.Beyond
    + ApproReport^.Costs.OM.CivSalary.CivRIF.Beyond
    + ApproReport^.Costs.OM.CivSalary.CivRetire.Beyond
    + ApproReport^.Costs.OM.CivMoving.PerDiem.Beyond
    + ApproReport^.Costs.OM.CivMoving.POVMiles.Beyond
    + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Beyond
    + ApproReport^.Costs.OM.CivMoving.HHG.Beyond
    + ApproReport^.Costs.OM.CivMoving.Misc.Beyond
    + ApproReport^.Costs.OM.CivMoving.HousHunt.Beyond
    + ApproReport^.Costs.OM.CivMoving.PPS.Beyond
    + ApproReport^.Costs.OM.CivMoving.RITA.Beyond
    + ApproReport^.Costs.OM.Freight.Pack.Beyond
    + ApproReport^.Costs.OM.Freight.Freight.Beyond
    + ApproReport^.Costs.OM.Freight.Vehicles.Beyond
    + ApproReport^.Costs.OM.Freight.Driving.Beyond
    + ApproReport^.Costs.OM.Freight.LossRate.Beyond
    + ApproReport^.Costs.OM.Champus.Beyond
    + ApproReport^.Costs.OM.Unemployment.Beyond
    + ApproReport^.Costs.OM.Other.Caretaker.Beyond
    + ApproReport^.Costs.OM.Other.AdminPlan.Beyond
```

```
    + ApproReport^.Costs.OM.Other.Shutdown.Beyond
    + ApproReport^.Costs.OM.Other.Maintain.Beyond
    + ApproReport^.Costs.OM.Other.NewHire.Beyond
    + ApproReport^.Costs.Om.Other.OtherIDF.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
write (F, Blank:11, 'Mil Pers  ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((
    + ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
    + ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((
    + ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Beyond
    + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Beyond
    + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Beyond
    + ApproReport^.Costs.MilPers.MilMoving.HHG.Beyond
    + ApproReport^.Costs.MilPers.MilMoving.Misc.Beyond
    + ApproReport^.Costs.MilPers.Other.HousAllow.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, ' Envir Mit ', ApproReport^.Costs.EnvirMitig,
  Adjust1000, 8, 1);
BeyondOut (F, ' HAP / RSE ', ApproReport^.Costs.HomeownAsst,
  Adjust1000, 8, 1);
BeyondOut (F, ' Land Purch', ApproReport^.Costs.LandPurch,
  Adjust1000, 8, 1);
BeyondOut (F, ' Procuremts', ApproReport^.Costs.Procure, Adjust1000, 8, 1);
write (F, Blank:11, 'Other     ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Costs.InfoMgt.Year[Y]
    + ApproReport^.Costs.Other.Mission.Year[Y]
    + ApproReport^.Costs.Other.OneTime.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Costs.InfoMgt.Beyond
    + ApproReport^.Costs.Other.Mission.Beyond
    + ApproReport^.Costs.Other.OneTime.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, ' Misc Recur', ApproReport^.Costs.MiscRecur,
  Adjust1000, 8, 1);
writeln (F);
BeyondOut (F, ' TOTAL     ', ApproReport^.Costs.Total, Adjust1000, 8, 1);
writeln (F);
writeln (F, BLANK:10, 'SAVINGS ($K)');
BeyondOut (F, ' MilCon    ', ApproReport^.Savings.MilCon, Adjust1000, 8, 1);
writeln (F, Blank:11, 'FAM HOUSING ');
BeyondOut (F, '  Construct', ApproReport^.Savings.FamHous.Construction,
  Adjust1000, 8, 1);
BeyondOut (F, '  Operation', ApproReport^.Savings.FamHous.Operations,
  Adjust1000, 8, 1);
```

```
write (F, Blank:11, 'O&M       ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Savings.OM.RPMA.Year[Y]
    + ApproReport^.Savings.OM.BOS.Year[Y]
    + ApproReport^.Savings.OM.OperatingIDF.Year[Y]
    + ApproReport^.Savings.OM.CivSalary.Year[Y]
    + ApproReport^.Savings.OM.CivMoving.Year[Y]
    + ApproReport^.Savings.OM.Freight.Year[Y]
    + ApproReport^.Savings.OM.Champus.Year[Y]
    + ApproReport^.Savings.OM.OtherIDF.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Savings.OM.RPMA.Beyond
    + ApproReport^.Savings.OM.BOS.Beyond
    + ApproReport^.Savings.OM.OperatingIDF.Beyond
    + ApproReport^.Savings.OM.CivSalary.Beyond
    + ApproReport^.Savings.OM.CivMoving.Beyond
    + ApproReport^.Savings.OM.Freight.Beyond
    + ApproReport^.Savings.OM.Champus.Beyond
    + ApproReport^.Savings.OM.OtherIDF.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
write (F, Blank:11, 'Mil Pers  ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
    + ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
    + ApproReport^.Savings.MilPers.MilMoving.Year[Y]
    + ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Savings.MilPers.MilSalary.Off.Beyond
    + ApproReport^.Savings.MilPers.MilSalary.Enl.Beyond
    + ApproReport^.Savings.MilPers.MilMoving.Beyond
    + ApproReport^.Savings.MilPers.Other.HousAllow.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, ' Envir Mit ', ApproReport^.Savings.EnvirMitig,
  Adjust1000, 8, 1);
BeyondOut (F, ' HAP / RSE ', ApproReport^.Savings.HomeownAsst,
  Adjust1000, 8, 1);
BeyondOut (F, ' Land Reven', ApproReport^.Savings.LandRevenue,
  Adjust1000, 8, 1);
BeyondOut (F, ' Procuremts', ApproReport^.Savings.Procure,
  Adjust1000, 8, 1);
write (F, Blank:11, 'Other     ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Savings.Other.Year[Y]
    + ApproReport^.Savings.InfoMgt.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Savings.Other.Beyond
    + ApproReport^.Savings.InfoMgt.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, ' Misc Recur', ApproReport^.Savings.MiscRecur,
  Adjust1000, 8, 1);
writeln (F);
BeyondOut (F, ' TOTAL     ', ApproReport^.Savings.Total, Adjust1000, 8, 1);
```

```
writeln (F);
Writeln (F, Blank:10, 'NET COSTS ($K)');
BeyondOut (F, ' MilCon    ', ApproReport^.Net.MilCon, Adjust1000, 8, 1);
writeln (F, Blank:11, 'FAM HOUSING');
BeyondOut (F, '  Construct', ApproReport^.Net.FamHous.Construction,
  Adjust1000, 8, 1);
BeyondOut (F, '  Operation', ApproReport^.Net.FamHous.Operations,
  Adjust1000, 8, 1);
write (F, Blank:11, 'O&M       ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Net.OM.RPMA.Year[Y]
    + ApproReport^.Net.OM.BOS.Year[Y]
    + ApproReport^.Net.OM.OperatingIDF.Year[Y]
    + ApproReport^.Net.OM.CivSalary.Year[Y]
    + ApproReport^.Net.OM.CivMoving.Year[Y]
    + ApproReport^.Net.OM.Other.Year[Y]
    + ApproReport^.Net.Champus.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Net.OM.RPMA.Beyond
    + ApproReport^.Net.OM.BOS.Beyond
    + ApproReport^.Net.OM.OperatingIDF.Beyond
    + ApproReport^.Net.OM.CivSalary.Beyond
    + ApproReport^.Net.OM.CivMoving.Beyond
    + ApproReport^.Net.OM.Other.Beyond
    + ApproReport^.Net.Champus.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
write (F, Blank:11, 'Mil Pers  ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Net.MilPers.MilSalary.Year[Y]
    + ApproReport^.Net.MilPers.MilMoving.Year[Y]
    + ApproReport^.Net.MilPers.Other.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Net.MilPers.MilSalary.Beyond
    + ApproReport^.Net.MilPers.MilMoving.Beyond
    + ApproReport^.Net.MilPers.Other.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, ' Envir Mit ', ApproReport^.Net.EnvirMitig, Adjust1000, 8, 1);
BeyondOut (F, ' HAP / RSE ', ApproReport^.Net.HomeownAsst,
  Adjust1000, 8, 1);
BeyondOut (F, ' Land      ', ApproReport^.Net.Land, Adjust1000, 8, 1);
BeyondOut (F, ' Procuremts', ApproReport^.Net.Procure, Adjust1000, 8, 1);
write (F, Blank:11, 'Other     ');
For Y := 1 to MaxYears DO
  write (F, WithCommas ((ApproReport^.Net.Other.Year[Y]
    + ApproReport^.Net.InfoMgt.Year[Y])
    * Adjust1000.Year[Y],0):8);
writeln (F, WithCommas ((ApproReport^.Net.Other.Beyond
    + ApproReport^.Net.InfoMgt.Beyond)
    * Adjust1000.Year[MaxYears],0):8);
BeyondOut (F, ' Misc Recur', ApproReport^.Net.MiscRecur, Adjust1000, 8, 1);
writeln (F);
BeyondOut (F, ' TOTAL     ', ApproReport^.Net.Total, Adjust1000, 8, 1);
```

```
  Close (F)
END;  { AppropriationsSummaryReport }

{*****************************************************************************
 *   Name            : AppropriationsFinanceReport
 *
 *   Purpose         : Writes appropriations finance summary report
 *
 *   Notes           : FinAdjust.Year[] contains the $ to $K conversion factor
 *                     (0.001), and the cumulative inflation rates.
 *                   : Chr(12) is the FormFeed character.
 *
 *   Invoked As      : ApproFinanceReport (ApproReport)
 *
 *   Return value    :
 *
 *   Files/Tables
 *     Names                  Access    Purpose
 *     ------------------     --------  ----------------------------------------
 *     FINANCE.RPT             Write
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   22May92   AML   Creation
 *****************************************************************************}

Procedure ApproFinanceReport (ApproReport      : ApproReportRec;
                              CalcFctrsPerPos  : CalcPersPosRec);

  Procedure FinOut (Var Fil      : Text;
                        OutLabel : String;
                        Value    : YearAndBeyondRealRec;
                        Adjst    : Year1ToMaxRealRec;
                        StartYr,
                        Size     : Integer);

  Var
    Y : Integer;

  Begin
    Write (Fil, Blank:10, OutLabel);
    For Y := StartYr * 3 + 1 to StartYr * 3 + 3 DO
      Write (Fil, WithCommas (Within ( -(YtoX(10,Size-2)-1),
        Value.Year[Y] * Adjst.Year[Y],  (YtoX(10,Size-2)-1) ),0):Size);
    Writeln (Fil)
  End;
```

```
VAR
  Buffer    : IOBuffer;
  C, Y, Y2  : INTEGER;
  F         : TEXT;
  FinAdjust : Year1ToMaxRealRec;
  NavyFacil : Boolean;
  TmpStr    : String;

Begin { AppropriationsSummaryReport }
  FinAdjust.Year[1] := (1 + StdFctrsFac.InflationRate.Year[1]) / 1000.0;
  For Y := 2 To MaxYears Do
    FinAdjust.Year[Y] := FinAdjust.Year[Y-1]
      * (1 + StdFctrsFac.InflationRate.Year[Y]);
  NavyFacil :=
    (Pos ('NAV', UpperCase(Screen1Data^.Service)) > 0) Or
    (Pos ('USN', UpperCase(Screen1Data^.Service)) > 0) Or
    (Pos ('S.N', UpperCase(Screen1Data^.Service)) > 0) Or
    (Pos ('MAR', UpperCase(Screen1Data^.Service)) > 0) Or
    (Pos ('MC',  UpperCase(Screen1Data^.Service)) > 0) Or
    (Pos ('M.C', UpperCase(Screen1Data^.Service)) > 0);
  assign (F, Info.ReportDir + 'FINANCE.RPT');
  SetTextBuf (F, Buffer);
  reWrite (F);
  For Y2 := 0 To 1 Do
    Begin
      For C := 1 To 6 Do
        Writeln (F);
      If Not NavyFacil
        Then Center (F, 'BASE REALIGNMENT AND CLOSURE')
      Else
        Begin
          Center (F, 'DEPARTMENT OF THE NAVY');
          Center (F, 'BASE REALIGNMENT, CLOSURE, OR CONSOLIDATION');
        End;
      Center (F, ConCat('FINANCIAL SUMMARY (COBRA v', Version, ')'));
      If Not NavyFacil
        Then Center (F, UpperCase(Concat
          ('(',Screen1Data^.Service,'/',Screen1Data^.OptionPkgName,')')));
      Center (F, Concat ('Data As Of ', Screen1Data^.DateLine));
      Center (F, Concat ('Report Created ', AsOfDateTime));
      If Not NavyFacil
        Then Center (F, '(DOLLARS IN THOUSANDS)')
      Else
        Begin
          Center (F, '(In Thousands of Dollars)');
          Writeln (F);
          Writeln (F, Blank:10, 'Closure/Realignment Summary:  ',
            Screen1Data^.OptionPkgName);
        End;
      Writeln (F);
      Write (F, Blank:40);
      For Y := 0 To 2 Do
```

```
  Write (F, '   FY', (Screen1Data^.FY1 + Y + Y2*3):5);
Writeln (F);
Write (F, Blank:10, 'ONE-TIME IMPLEMENTATION COSTS:',
  Rept ('   -------', 3));
Writeln (F);
Writeln (F, Blank:10, Rept ('-', 30));
FinOut (F, '  Military Construction       ', ApproReport^.Costs.MilCon,
  FinAdjust, Y2, 10);
If Not NavyFacil Then
  Begin
    Writeln (F, Blank:12, 'Family Housing');
    FinOut (F, '    Construction                ',
      ApproReport^.Costs.FamHous.Construction, FinAdjust, Y2, 10);
    FinOut (F, '    Operations                  ',
      ApproReport^.Costs.FamHous.Operations, FinAdjust, Y2, 10);
    FinOut (F, '  Environment                   ',
      ApproReport^.Costs.EnvirMitig, FinAdjust, Y2, 10);
  End
Else
  Begin
    FinOut (F, '  Family Housing: Construction',
      ApproReport^.Costs.FamHous.Construction, FinAdjust, Y2, 10);
    Write (F, Blank:28, 'Operations  ');
    For Y := 1 To 3 Do
      Write (F, 0:10);
    Writeln (F);
  End;
Write (F, Blank:12, 'Operation and Maintenance   ');
If NavyFacil
  Then For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
    Write (F, WithCommas (
        (ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
      + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
      + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
      + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
      + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
      + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
      + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
      + ApproReport^.Costs.OM.Unemployment.Year[Y]
      + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
      + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
      + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
      + ApproReport^.Costs.OM.Other.NewHire.Year[Y])
      * FinAdjust.Year[Y], 0):10)
```

```
  Else For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
    Write (F, WithCommas (
       (ApproReport^.Costs.OM.RPMA.Year[Y]
      + ApproReport^.Costs.OM.BOS.Year[Y]
      + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
      + ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
      + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
      + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
      + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
      + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
      + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
      + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
      + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
      + ApproReport^.Costs.OM.Champus.Year[Y]
      + ApproReport^.Costs.OM.Unemployment.Year[Y]
      + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
      + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
      + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
      + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
      + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
      + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y])
      * FinAdjust.Year[Y], 0):10);
Writeln (F);
If NavyFacil
  Then Write (F, Blank:12, 'Military Personnel (PCS)    ')
  Else Write (F, Blank:12, 'Military Personnel - PCS    ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas ((
      ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
    + ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y])
    * FinAdjust.Year[Y], 0):10);
Writeln (F);
If NavyFacil
  Then FinOut (F, '  Homeowner Assistance Program',
    ApproReport^.Costs.HomeOwnAsst, FinAdjust, Y2, 10)
Else
  Begin
    Write (F, Blank:12, 'Other                        ');
    For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
```

```
      Write (F, WithCommas (
         (ApproReport^.Costs.InfoMgt.Year[Y]
        + ApproReport^.Costs.Other.Mission.Year[Y]
        + ApproReport^.Costs.Other.OneTime.Year[Y]
        + ApproReport^.Costs.MiscRecur.Year[Y]
        + ApproReport^.Costs.Procure.Year[Y]
        + ApproReport^.Costs.HomeOwnAsst.Year[Y])
        * FinAdjust.Year[Y], 0):10);
    Writeln (F);
  End;
If NavyFacil
  Then FinOut (F, '  Revenues from Land Sales    ',
    ApproReport^.Net.Land, FinAdjust, Y2, 10)
  Else FinOut (F, '  Revenues from Land Sales (-)',
    ApproReport^.Net.Land, FinAdjust, Y2, 10);
If NavyFacil Then
  Begin
    FinOut (F, '  Environmental: Planning &   ',
      ApproReport^.Costs.EnvirMitig, FinAdjust, Y2, 10);
    Writeln (F, Blank:14, 'Cleanup/Compliance');
  End
Else Writeln (F);
If Not NavyFacil
  Then FinOut (F, '  TOTAL COSTS                 ',
    ApproReport^.Costs.Total, FinAdjust, Y2, 10)
Else
  Begin
    Write (F, Blank:10, 'TOTAL COSTS                  ');
    For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
      Write (F, WithCommas (
         (ApproReport^.Costs.MilCon.Year[Y]
        + ApproReport^.Costs.FamHous.Construction.Year[Y]
        + ApproReport^.Costs.OM.CivSalary.CivRIF.Year[Y]
        + ApproReport^.Costs.OM.CivSalary.CivRetire.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.PerDiem.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.POVMiles.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.HomePurchSale.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.HHG.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.Misc.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.HousHunt.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.PPS.Year[Y]
        + ApproReport^.Costs.OM.CivMoving.RITA.Year[Y]
        + ApproReport^.Costs.OM.Freight.Pack.Year[Y]
        + ApproReport^.Costs.OM.Freight.Freight.Year[Y]
        + ApproReport^.Costs.OM.Freight.Vehicles.Year[Y]
        + ApproReport^.Costs.OM.Freight.Driving.Year[Y]
        + ApproReport^.Costs.OM.Freight.LossRate.Year[Y]
        + ApproReport^.Costs.OM.Unemployment.Year[Y]
        + ApproReport^.Costs.OM.Other.AdminPlan.Year[Y]
        + ApproReport^.Costs.OM.Other.Shutdown.Year[Y]
        + ApproReport^.Costs.OM.Other.Maintain.Year[Y]
        + ApproReport^.Costs.OM.Other.NewHire.Year[Y]
```

```
    + ApproReport^.Costs.MilPers.MilMoving.ElimMilPCS.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.PerDiem.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.POVMiles.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.HHG.Year[Y]
    + ApproReport^.Costs.MilPers.MilMoving.Misc.Year[Y]
    + ApproReport^.Costs.MilPers.Other.HousAllow.Year[Y]
    + ApproReport^.Costs.HomeOwnAsst.Year[Y]
    + ApproReport^.Net.Land.Year[Y]
    + ApproReport^.Costs.EnvirMitig.Year[Y])
    * FinAdjust.Year[Y], 0):10);
Writeln (F);
Writeln (F, Blank:10, '(BASE CLOSURE ', FiscalYear(0), ' ACCOUNT)');
Writeln (F);
Writeln (F, Blank:10, 'RECURRING COSTS:');
Writeln (F, Blank:10, Rept ('-',16));
FinOut (F, '  Family Housing: Operations  ',
  ApproReport^.Costs.FamHous.Operations, FinAdjust, Y2, 10);
Write (F, Blank:12, 'Operation and Maintenance   ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas (
     (ApproReport^.Costs.OM.RPMA.Year[Y]
    + ApproReport^.Costs.OM.BOS.Year[Y]
    + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
    + ApproReport^.Costs.OM.Champus.Year[Y]
    + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
    + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y])
    * FinAdjust.Year[Y], 0):10);
Writeln (F);
Write (F, Blank:12, 'Other: APN                  ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas (
     (ApproReport^.Costs.InfoMgt.Year[Y]
    + ApproReport^.Costs.Other.Mission.Year[Y]
    + ApproReport^.Costs.Other.OneTime.Year[Y]
    + ApproReport^.Costs.MiscRecur.Year[Y]
    + ApproReport^.Costs.Procure.Year[Y]
    + ApproReport^.Costs.HomeOwnAsst.Year[Y])
    * FinAdjust.Year[Y], 0):10);
Writeln (F);
Write (F, Blank:10, 'TOTAL COSTS                    ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas (
     (ApproReport^.Costs.FamHous.Operations.Year[Y]
    + ApproReport^.Costs.OM.RPMA.Year[Y]
    + ApproReport^.Costs.OM.BOS.Year[Y]
    + ApproReport^.Costs.OM.OperatingIDF.Year[Y]
    + ApproReport^.Costs.OM.Champus.Year[Y]
    + ApproReport^.Costs.OM.Other.Caretaker.Year[Y]
    + ApproReport^.Costs.OM.Other.OtherIDF.Year[Y]
    + ApproReport^.Costs.InfoMgt.Year[Y]
    + ApproReport^.Costs.Other.Mission.Year[Y]
    + ApproReport^.Costs.Other.OneTime.Year[Y]
```

```
          + ApproReport^.Costs.MiscRecur.Year[Y]
          + ApproReport^.Costs.Procure.Year[Y]
          + ApproReport^.Costs.HomeOwnAsst.Year[Y])
          * FinAdjust.Year[Y], 0):10);
      Writeln (F);
    End;
  Writeln (F);
  Writeln (F, Blank:10, 'SAVINGS:');
  Writeln (F, Blank:10, '--------');
  If Not NavyFacil Then
    Begin
      FinOut (F, '  Military Construction       ',
        ApproReport^.Savings.MilCon, FinAdjust, Y2, 10);
      Writeln (F, Blank:12, 'Family Housing');
      FinOut (F, '    Construction              ',
        ApproReport^.Savings.FamHous.Construction, FinAdjust, Y2, 10);
      FinOut (F, '    Operations                ',
        ApproReport^.Savings.FamHous.Operations, FinAdjust, Y2, 10);
    End
  Else
    Begin
      FinOut (F, '  Military Construction       ',
        ApproReport^.Savings.MilCon, FinAdjust, Y2, 10);
      FinOut (F, '  Family Housing: Construction',
        ApproReport^.Savings.FamHous.Construction, FinAdjust, Y2, 10);
      FinOut (F, '                   Operations  ',
        ApproReport^.Savings.FamHous.Operations, FinAdjust, Y2, 10);
    End;
  Write (F, Blank:12, 'Operation and Maintenance   ');
  For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
    Write (F, WithCommas (
       (ApproReport^.Savings.OM.RPMA.Year[Y]
      + ApproReport^.Savings.OM.BOS.Year[Y]
      + ApproReport^.Savings.OM.OperatingIDF.Year[Y]
      + ApproReport^.Savings.OM.CivSalary.Year[Y]
      + ApproReport^.Savings.OM.CivMoving.Year[Y]
      + ApproReport^.Savings.OM.Freight.Year[Y]
      + ApproReport^.Savings.OM.Champus.Year[Y]
      + ApproReport^.Savings.OM.OtherIDF.Year[Y])
      * FinAdjust.Year[Y], 0):10);
  Writeln (F);
  If NavyFacil
    Then Write (F, Blank:12, 'Military Personnel (PCS)    ')
    Else Write (F, Blank:12, 'Military Personnel          ');
  For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
    Write (F, WithCommas ((
        ApproReport^.Savings.MilPers.MilSalary.Off.Year[Y]
      + ApproReport^.Savings.MilPers.MilSalary.Enl.Year[Y]
      + ApproReport^.Savings.MilPers.MilMoving.Year[Y]
      + ApproReport^.Savings.MilPers.Other.HousAllow.Year[Y])
      * FinAdjust.Year[Y], 0):10);
```

```
Writeln (F);
If NavyFacil
  Then Write (F, Blank:12, 'Other: APN                    ')
  Else Write (F, Blank:12, 'Other                         ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas ((
      ApproReport^.Savings.Other.Year[Y]
    + ApproReport^.Savings.InfoMgt.Year[Y]
    + ApproReport^.Savings.MiscRecur.Year[Y]
    + ApproReport^.Savings.Procure.Year[Y])
    * FinAdjust.Year[Y], 0):10);
Writeln (F);
Write (F, Blank:10, 'Civilian ES                      ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas (CalcFctrsPerPos.CivPosElim.Year[Y],-2):10);
Writeln (F);
Write (F, Blank:10, 'Military ES                      ');
For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
  Write (F, WithCommas (CalcFctrsPerPos.OffPosElim.Year[Y]
                      + CalcFctrsPerPos.EnlPosElim.Year[Y],-2):10);
Writeln (F);
If NavyFacil Then
  Begin
    FinOut (F, 'TOTAL SAVINGS                    ',
      ApproReport^.Savings.Total, FinAdjust, Y2, 10);
    Writeln (F);
  End
Else
  Begin
    Writeln (F);
    FinOut (F, '  TOTAL SAVINGS                  ',
      ApproReport^.Savings.Total, FinAdjust, Y2, 10);
  End;
Writeln (F);
If NavyFacil
  Then FinOut (F, 'GRAND TOTAL (BASE CLOSURE NET)',
    ApproReport^.Net.Total, FinAdjust, Y2, 10)
Else
  Begin
    Writeln (F, Blank:10, 'NET IMPLEMENTATION COSTS:');
    Writeln (F, Blank:10, Rept ('-',25));
    FinOut (F, '  Military Construction          ',
      ApproReport^.Net.MilCon, FinAdjust, Y2, 10);
    Writeln (F, Blank:12, 'Family Housing');
    FinOut (F, '    Construction                 ',
      ApproReport^.Net.FamHous.Construction, FinAdjust, Y2, 10);
    FinOut (F, '    Operations                   ',
      ApproReport^.Net.FamHous.Operations, FinAdjust, Y2, 10);
    FinOut (F, '  Environment                    ',
      ApproReport^.Net.EnvirMitig, FinAdjust, Y2, 10);
    Write (F, Blank:12, 'Operation and Maintenance   ');
    For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
```

```
            Write (F, WithCommas ((ApproReport^.Net.OM.RPMA.Year[Y]
              + ApproReport^.Net.OM.BOS.Year[Y]
              + ApproReport^.Net.OM.OperatingIDF.Year[Y]
              + ApproReport^.Net.OM.CivSalary.Year[Y]
              + ApproReport^.Net.OM.CivMoving.Year[Y]
              + ApproReport^.Net.OM.Other.Year[Y]
              + ApproReport^.Net.Champus.Year[Y]) * FinAdjust.Year[Y], 0):10);
          Writeln (F);
          Write (F, Blank:12, 'Military Personnel          ');
          For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
            Write (F, WithCommas ((
                ApproReport^.Net.MilPers.MilSalary.Year[Y]
              + ApproReport^.Net.MilPers.MilMoving.Year[Y]
              + ApproReport^.Net.MilPers.Other.Year[Y])
              * FinAdjust.Year[Y], 0):10);
          Writeln (F);
          Write (F, Blank:12, 'Other', Blank:23);
          For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
            Write (F, WithCommas ((
                ApproReport^.Net.Other.Year[Y]
              + ApproReport^.Net.InfoMgt.Year[Y]
              + ApproReport^.Net.MiscRecur.Year[Y]
              + ApproReport^.Net.Procure.Year[Y]
              + ApproReport^.Net.HomeOwnAsst.Year[Y])
              * FinAdjust.Year[Y], 0):10);
          Writeln (F);
          FinOut (F, '  Revenues from Land Sales    ', ApproReport^.Net.Land,
            FinAdjust, Y2, 10);
          Write (F, Blank:10, 'Civilian ES                    ');
          For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
            Write (F, WithCommas (CalcFctrsPerPos.CivPosElim.Year[Y],-2):10);
          Writeln (F);
          Write (F, Blank:10, 'Military ES                    ');
          For Y := Y2 * 3 + 1 To Y2 * 3 + 3 Do
            Write (F, WithCommas (CalcFctrsPerPos.OffPosElim.Year[Y]
                                  + CalcFctrsPerPos.EnlPosElim.Year[Y],-2):10);
          Writeln (F);
          Writeln (F);
          FinOut (F, '  NET IMPLEMENTATION COSTS    ', ApproReport^.Net.Total,
            FinAdjust, Y2, 10);
        End;
      If Y2 = 0
        Then Write(F, Chr(12));
    End;
  Close (F);
End;

END. { UNIT }
```

```
UNIT CSCR1;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT,
  CDEC;

TYPE
  PScr1Dlg = ^TScr1Dlg;
  TScr1Dlg = OBJECT (TDialog)
               CONSTRUCTOR Init;
             END; {Object}

  PDBaseDlg = ^TDBaseDlg;
  TDBaseDlg = OBJECT (TDialog)
               CONSTRUCTOR Init (S : BaseArray);
             END; {Object}

IMPLEMENTATION

{*************************************************************************
 *  Name          : TScr1Dlg.Init
 *
 *  Purpose       : Set up screen one
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value  :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
 *************************************************************************}

CONSTRUCTOR TScr1Dlg.Init;

  VAR
    B1,
    B2,
    C    : INTEGER;
    P    : PView;
    rect : TRect;
```

```
BEGIN
  rect.Assign (0,0,78,22);
  TDialog.Init (rect,'Screen One - General Scenario');
  Options := Options OR ofCentered;

  rect.Assign (16,1,38,2);
  P := New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0));
  Insert (P);
  rect.Assign (1,1,16,2);
  Insert(New(PLabel,Init(rect,'~O~ptn Pkg Name:', P)));

  rect.Assign (16,2,38,3);
  Insert(New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0)));
  rect.Assign (8,2,16,3);
  Insert(New(PStaticText,Init(rect,'Service:')));

  rect.Assign (16,3,38,4);
  Insert(New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0)));
  rect.Assign (10,3,16,4);
  Insert(New(PStaticText,Init(rect,'Group:')));

  rect.Assign (40,1,61,2);
  Insert(New(PStaticText,Init(rect,'Standard Factor File:')));
  rect.Assign (61,1,76,2);
  Insert(New(PFInputLine,Init(rect,79,DPathSet,DString,0,0.0,0.0)));

  rect.Assign (70,2,76,3);
  Insert(New(PFInputLine,Init(rect,4,DUnSignedSet,DInteger,0,1990.0,2100.0)));
  rect.Assign (40,2,70,3);
  Insert(New(PStaticText,Init(rect,'Model Year One is Fiscal Year:')));

  rect.Assign (73,3,76,4);
  Insert(New(PFInputLine,Init(rect,1,DResponseSet,DChar,0,0.0,0.0)));
  rect.Assign (40,3,73,4);
  Insert(New(PStaticText,Init(rect,'Let Model do Time-Phasing? (Y/N):')));

  For B2 := 0 To 1 Do
    Begin
    rect.Assign (24 + B2 * 38, 4, 38 + B2 * 38, 5);
    Insert(New(PStaticText,Init(rect,
      'State  CY* BD*')));
    For B1 := 0 To 7 Do
      If (B1 < 7) Or (B2 = 1) Then
        Begin
          rect.Assign (2 + 38 * B2,5 + B1,24 + 38 * B2,6 + B1);
          P := New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0));
          Insert (P);
          If B1 = 0 Then
            Begin
              rect.Assign (1 + B2 * 38, 4, 11 + B2 * 38, 5);
```

```
            If B2 = 0
              Then Insert(New(PLabel,Init(rect,'~B~ase Name',P)))
              Else Insert(New(PLabel,Init(rect,'B~a~se Name',P)));
          End;
        rect.Assign (25 + 38 * B2,5 + B1,29 + 38 * B2,6 + B1);
        Insert(New(PFInputLine,Init(rect,2,DCharSet,DString,0,0.0,6.0)));
        rect.Assign (31 + 38 * B2,5 + B1,34 + 38 * B2,6 + B1);
        Insert(New(PFInputLine,
          Init(rect,1,DStrategySet,DInteger,0,0.0,6.0)));
        rect.Assign (35 + 38 * B2,5 + B1,38 + 38 * B2,6 + B1);
        Insert(New(PFInputLine,Init(rect,1,DResponseSet,DChar,0,0.0,0.0)));
      End;
  End;

For C := 1 To 6 Do
  Begin
    rect.Assign (2,12 + C,64,13 + C);
    P := New(PFInputLine,Init(rect,60,DCharSet,DString,0,0.0,0.0));
    Insert (P);
    If C = 1 Then
      Begin
        rect.Assign (1,12,22,13);
        Insert(New(PLabel,Init(rect,'~S~ummary/Description:',P)));
      End
  End;

rect.Assign (65,13,76,16);
Insert(New(PStaticText,Init(rect,'*CY = Close /Deact Year (0=Realign)')));
rect.Assign (65,17,76,19);
Insert(New(PStaticText,Init(rect,'*BD = Base Deact (Y/N)')));

rect.Assign (2,20,24,21);
P := New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0));
Insert (P);
rect.Assign (1,19,21,20);
Insert(New(PLabel,Init(rect,'~T~ime/Date of Data:', P)));
rect.Assign (24, 20, 29, 21);
Insert (New (PCheckBoxes, Init (rect, NewSItem (' ', nil))));
rect.Assign (24,19,28,20);
Insert(New(PStaticText,Init(rect,'Set:')));

{*****************************************************************
 * "OK" and "Cancel" buttons
 *****************************************************************}

rect.Assign (48,19,60,21);
Insert(New(PButton,Init(rect,'~N~ext',cmOK,bfNormal)));

rect.Assign (63,19,75,21);
Insert(New(PButton,Init(rect,'~D~one',cmNo,bfNormal)));
END; {TScrlDlg.Init}
```

```
{*************************************************************************
 *  Name          : TDBaseDlg.Init
 *
 *  Purpose       : Set up screen to load from DataBase
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  22Jun92   AML   Creation
 *************************************************************************}

CONSTRUCTOR TDBaseDlg.Init (S : BaseArray);

  Var
    B1, B2   : Integer;
    P        : PView;
    rect     : TRect;
    TempStr  : String17;

  Begin
    rect.Assign (0,0,47,18);
    TDialog.Init (rect,'Load Base(s) From DataBase');
    Options := Options OR ofCentered;

    rect.Assign (2, 2, 30, 17);
    P := New (PCheckBoxes, Init (rect,
      NewSItem ('~0~-'+S[0],
      NewSItem ('~1~-'+S[1],
      NewSItem ('~2~-'+S[2],
      NewSItem ('~3~-'+S[3],
      NewSItem ('~4~-'+S[4],
      NewSItem ('~5~-'+S[5],
      NewSItem ('~6~-'+S[6],
      NewSItem ('~7~-'+S[7],
      NewSItem ('~8~-'+S[8],
      NewSItem ('~9~-'+S[9],
      NewSItem ('~A~-'+S[10],
      NewSItem ('~B~-'+S[11],
      NewSItem ('~C~-'+S[12],
      NewSItem ('~D~-'+S[13],
      NewSItem ('~E~-'+S[14],
      nil)))))))))))))))));
    Insert (P);
    rect.Assign (1, 1, 30, 2);
    Insert (New (PLabel, Init (rect, 'Select Base(s) to ~L~oad:', P)));

    rect.Assign (32,2,45,3);
    P := New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0));
    Insert (P);
    rect.Assign (31,1,45,2);
    Insert(New(PLabel,Init(rect,'~S~earch for:',P)));
```

```
    B1 := MaxBases;
    If (B1 > 1) Or (Screen1Data^.Base[0].Name <> '')
      Then Inc (B1);
    B2 := AbsMaxBases + 1;
    TempStr := Concat ('(', Chr(B1 Div 10 + 48), Chr(B1 Mod 10 + 48),
      '/', Chr(B2 Div 10 + 48), Chr(B2 Mod 10 + 48), ' bases)');
    rect.Assign (32,4,45,5);
    Insert(New(PStaticText,Init(rect,TempStr)));

    rect.Assign (31,6,45,8);
    Insert(New(PButton,Init(rect,'~O~pen',cmNo,bfDefault)));
    rect.Assign (31,9,45,11);
    Insert(New(PButton,Init(rect,'~N~ext',cmOK,bfNormal)));
    rect.Assign (31,12,45,14);
    Insert(New(PButton,Init(rect,'~P~revious',cmYes,bfNormal)));
    rect.Assign (31,15,45,17);
    Insert(New(PButton,Init(rect,'Cancel',cmCancel,bfNormal)));
  End;

END. {Unit CScr1}
```

```
UNIT CSCR2;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  FINPUT;

TYPE
  PScr2Dlg = ^TScr2Dlg;
  TScr2Dlg = OBJECT (TDialog)
               Scr2Data : Array [1..15] of Real;
               CONSTRUCTOR Init (datapage : INTEGER);
               PROCEDURE Advance (Var Base1, Base2 : INTEGER);
               PROCEDURE InitScr2Info (datapage : INTEGER);
               PROCEDURE CopyScr2Info (datapage : INTEGER);
             END; {Object}

IMPLEMENTATION

{*********************************************************************
 *  Name         : InitScr2Info
 *
 *  Purpose      : Reads base distances into Screen2Data
 *
 *  Invoked As   : InitScr2Info(info)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name             Access  Purpose
 *    ---------------  ------  ----------------------------------------
 *    datapage         IN      number of base to get distances from/to
 *
 *  Date     Init  Modifications
 *  -------  ----  ------------------------------------------------------
 *  27Apr92  AML   Creation
 *********************************************************************}

PROCEDURE TScr2Dlg.InitScr2Info (datapage : INTEGER);

  VAR
    B,
    B2,
    L  : INTEGER;
```

```
  BEGIN
    B  := 0;
    B2 := 1;
    If DataPage > 0
      Then For L := 1 To DataPage Do
        Advance (B, B2);
    For L := 1 To 15 Do
      Begin
        If B > -1 Then
          Scr2Data[L] := Distance^.Base[B].Base[B2];
        Advance (B, B2);
      End;
  END; {InitScr2Info}

{*************************************************************************
 *  Name          : CopyScr2Info
 *
 *  Purpose       : Copies Screen2Data into Distance; does error checking;
 *
 *  Invoked As    : CopyScr2Info(scr2info)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     structure to copy data from
 *    datapage          IN      array index in ActionEnvir.Base[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  27Apr92   AML   Creation
 *************************************************************************}

PROCEDURE TScr2Dlg.CopyScr2Info (datapage : INTEGER);

  VAR
    B,
    B2,
    L  : INTEGER;

  BEGIN
    B  := 0;
    B2 := 1;
    If DataPage > 0
      Then For L := 1 To DataPage Do
        Advance (B, B2);
    For L := 1 To 15 Do
      Begin
        If B > -1 Then
```

```
          Begin
            Distance^.Base[B].Base[B2] := Scr2Data[L];
            Distance^.Base[B2].Base[B] := Scr2Data[L];
            If (Scr2Data[L] < 0.1) And (PosTrans.Base[B].Base[B2] <> Nil) Then
              Begin
                Dispose (PosTrans.Base[B].Base[B2]);
                Dispose (PosTrans.Base[B2].Base[B]);
                PosTrans.Base[B].Base[B2] := Nil;
                PosTrans.Base[B2].Base[B] := Nil;
                Dispose (ActionEnvir^.Base[B].Base[B2]);
                Dispose (ActionEnvir^.Base[B2].Base[B]);
                ActionEnvir^.Base[B].Base[B2] := Nil;
                ActionEnvir^.Base[B2].Base[B] := Nil;
              End
            Else If (Scr2Data[L] > 0.0) And (PosTrans.Base[B].Base[B2] = Nil) The:
              Begin
                New (PosTrans.Base[B].Base[B2]);
                New (PosTrans.Base[B2].Base[B]);
                FillChar (PosTrans.Base[B].Base[B2]^,
                  SizeOf (PosTrans.Base[B].Base[B2]^), 0);
                FillChar (PosTrans.Base[B2].Base[B]^,
                  SizeOf (PosTrans.Base[B2].Base[B]^), 0);
                New (ActionEnvir^.Base[B].Base[B2]);
                New (ActionEnvir^.Base[B2].Base[B]);
                FillChar (ActionEnvir^.Base[B].Base[B2]^,
                  SizeOf (ActionEnvir^.Base[B].Base[B2]^), 0);
                FillChar (ActionEnvir^.Base[B2].Base[B]^,
                  SizeOf (ActionEnvir^.Base[B2].Base[B]^), 0);
              End;
          End;
        Advance (B, B2);
      End;
    ChangeFlag := True;
  END; {CopyScr2Info}


{*************************************************************************
 *  Name          : TScr2Dlg.Init
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Apr91   AML   Creation
 *************************************************************************}

CONSTRUCTOR TScr2Dlg.Init(datapage : INTEGER);

  VAR
    B,
    B2,
    L    : Integer;
    Rect : TRect;
```

```
BEGIN
  rect.Assign  ( 0, 0,78,22);
  TDialog.Init (rect, 'Screen Two - Distance Table');
  Options := Options OR ofCentered;

  rect.Assign(21,1,54,2);
  Insert(New(PStaticText,Init(rect,'Distance between Bases (in Miles)')));

  B  := 0;
  B2 := 1;
  If DataPage > 0
    Then For L := 1 To DataPage Do
      Advance (B, B2);
  For L := 1 To 15 Do
    Begin
      If B > -1 Then
        Begin
          rect.Assign (7,L+2,37,L+3);
          Insert(New(PStaticText,Init(rect,'From                        To')))
          rect.Assign (12,L+2,32,L+3);
          Insert(New(PStaticText,Init(rect,Screen1Data^.Base[B].Name)));
          rect.Assign (37,L+2,57,L+3);
          Insert(New(PStaticText,Init(rect,Screen1Data^.Base[B2].Name)));
          rect.Assign (59,L+2,71,L+3);
          Insert(New(PFInputLine,
            Init(rect,10,DPosRealSet,DReal,1,0.0,15000.0)));
        End;
      Advance (B, B2);
    End;

  {*************************************************************
   * "Next", "Previous" and "Cancel" buttons
   *************************************************************}

  rect.Assign  (33,19,45,21);
  Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

  rect.Assign  (48,19,60,21);
  Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

  rect.Assign  (63,19,75,21);
  Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
END; {TScr2Dlg.Init}
```

```
{*************************************************************************
 *  Name          : TScr2Dlg.Advance
 *
 *  Purpose       : Copies Screen2Data into Distance; does error checking;
 *
 *  Invoked As    : CopyScr2Info(scr2info)
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     structure to copy data from
 *    datapage          IN      array index in ActionEnvir.Base[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  27Apr92   AML   Creation
 *************************************************************************}

Procedure TScr2Dlg.Advance (Var Base1, Base2 : Integer);

  Begin
    If Base1 > -1 Then
      Begin
        Inc (Base2);
        If Base2 > MaxBases Then
          Begin
            Inc (Base1);
            Base2 := Base1 + 1;
          End;
        If Base1 = MaxBases
          Then Base1 := -1;
      End;
  End;

END. {Unit CScr2}
```

```
UNIT CSCR3;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  CSUP,
  FINPUT;

TYPE

  Screen3BaseRec = RECORD
    Off,
    Enl,
    Civ,
    Stu  : Year1ToMaxIntegerRec;
    Mis,
    Sup,
    Lit,
    Hev  : Year1ToMaxRealRec;
  END;

  Screen3Data = RECORD
    Srce,
    Dest : Screen3BaseRec;
  END;

  PScr3Dlg = ^TScr3Dlg;
  TScr3Dlg = OBJECT (TDialog)
               Scr3Data : Screen3Data;
               CONSTRUCTOR Init (DataPage : INTEGER);
               PROCEDURE InitScr3Info (DataPage : INTEGER);
               PROCEDURE CopyScr3Info (DataPage : INTEGER);
             END; {Object}


IMPLEMENTATION
```

```
{*************************************************************************
 *  Name          : InitScr3Info
 *
 *  Purpose       : Initialize the scr3info data structure with data from
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Accesses the following global variable:
 *                     PosTrans
 *
 *  Invoked As    : InitScr3Info(info)
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ------------------------------------------
 *    DataPage          IN      the two bases exchanging personnel/eqpmt
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  27Jan91   ASB   Creation
 *  11feb92   GJG   Added Screen3Data record to Method
 *  27Apr92   AML   Recreated in MultiBase format
 *************************************************************************}

PROCEDURE TScr3Dlg.InitScr3Info (DataPage : INTEGER);

  VAR
    B, B2, Y : INTEGER;

  BEGIN
    B := DataPage Div 16;
    B2 := DataPage Mod 16;
    FOR Y := 1 TO AbsMaxYears DO
      BEGIN
        Scr3Data.Srce.Off.Year[Y] := PosTrans.Base[B].Base[B2]^.Off.Year[Y];
        Scr3Data.Srce.Enl.Year[Y] := PosTrans.Base[B].Base[B2]^.Enl.Year[Y];
        Scr3Data.Srce.Civ.Year[Y] := PosTrans.Base[B].Base[B2]^.Civ.Year[Y];
        Scr3Data.Srce.Stu.Year[Y] := PosTrans.Base[B].Base[B2]^.Stu.Year[Y];
        Scr3Data.Dest.Off.Year[Y] := PosTrans.Base[B2].Base[B]^.Off.Year[Y];
        Scr3Data.Dest.Enl.Year[Y] := PosTrans.Base[B2].Base[B]^.Enl.Year[Y];
        Scr3Data.Dest.Civ.Year[Y] := PosTrans.Base[B2].Base[B]^.Civ.Year[Y];
        Scr3Data.Dest.Stu.Year[Y] := PosTrans.Base[B2].Base[B]^.Stu.Year[Y];
        Scr3Data.Srce.Mis.Year[Y] :=
          ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y];
        Scr3Data.Srce.Sup.Year[Y] :=
          ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y];
        Scr3Data.Srce.Lit.Year[Y] :=
          ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y];
        Scr3Data.Srce.Hev.Year[Y] :=
          ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y];
        Scr3Data.Dest.Mis.Year[Y] :=
```

```
          ActionEnvir^.Base[B2].Base[B]^.MoveMissionEqpmt.Year[Y];
        Scr3Data.Dest.Sup.Year[Y] :=
          ActionEnvir^.Base[B2].Base[B]^.MoveSupportEqpmt.Year[Y];
        Scr3Data.Dest.Lit.Year[Y] :=
          ActionEnvir^.Base[B2].Base[B]^.MilLightVehic.Year[Y];
        Scr3Data.Dest.Hev.Year[Y] :=
          ActionEnvir^.Base[B2].Base[B]^.HeavyOrSpecVehic.Year[Y];
      END;
  END; {InitScr3Info}

{*************************************************************************
 *  Name         : CopyScr3Info
 *
 *  Purpose      : Copy the data from the scr3info data structure into
 *                 the global variables used by the calculation routine.
 *
 *  Notes        : Alters the following global variable:
 *                    PosTrans
 *
 *  Invoked As   : CopyScr3Info(info)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    Datapage          IN      bases doing transfers
 *
 *  Date     Init  Modifications
 *  -------  ----  ------------------------------------------------------
 *  27Jan91  ASB   Creation
 *  11feb92  GJG   Added Screen3Data record to Method
 *  27Apr92  AML   Redid in multibase format
 *************************************************************************}

PROCEDURE TScr3Dlg.CopyScr3Info (DataPage : INTEGER);

  VAR
    B, B2, Y : INTEGER;

  BEGIN
    B := DataPage Div 16;
    B2 := DataPage Mod 16;
    FOR Y := 1 TO AbsMaxYears DO
      BEGIN
        PosTrans.Base[B].Base[B2]^.Off.Year[Y] := Scr3Data.Srce.Off.Year[Y];
        PosTrans.Base[B].Base[B2]^.Enl.Year[Y] := Scr3Data.Srce.Enl.Year[Y];
        PosTrans.Base[B].Base[B2]^.Civ.Year[Y] := Scr3Data.Srce.Civ.Year[Y];
        PosTrans.Base[B].Base[B2]^.Stu.Year[Y] := Scr3Data.Srce.Stu.Year[Y];
        PosTrans.Base[B2].Base[B]^.Off.Year[Y] := Scr3Data.Dest.Off.Year[Y];
        PosTrans.Base[B2].Base[B]^.Enl.Year[Y] := Scr3Data.Dest.Enl.Year[Y];
        PosTrans.Base[B2].Base[B]^.Civ.Year[Y] := Scr3Data.Dest.Civ.Year[Y];
```

```
        PosTrans.Base[B2].Base[B]^.Stu.Year[Y] := Scr3Data.Dest.Stu.Year[Y];
        ActionEnvir^.Base[B].Base[B2]^.MoveMissionEqpmt.Year[Y] :=
          Scr3Data.Srce.Mis.Year[Y];
        ActionEnvir^.Base[B].Base[B2]^.MoveSupportEqpmt.Year[Y] :=
          Scr3Data.Srce.Sup.Year[Y];
        ActionEnvir^.Base[B].Base[B2]^.MilLightVehic.Year[Y] :=
          Scr3Data.Srce.Lit.Year[Y];
        ActionEnvir^.Base[B].Base[B2]^.HeavyOrSpecVehic.Year[Y] :=
          Scr3Data.Srce.Hev.Year[Y];
        ActionEnvir^.Base[B2].Base[B]^.MoveMissionEqpmt.Year[Y] :=
          Scr3Data.Dest.Mis.Year[Y];
        ActionEnvir^.Base[B2].Base[B]^.MoveSupportEqpmt.Year[Y] :=
          Scr3Data.Dest.Sup.Year[Y];
        ActionEnvir^.Base[B2].Base[B]^.MilLightVehic.Year[Y] :=
          Scr3Data.Dest.Lit.Year[Y];
        ActionEnvir^.Base[B2].Base[B]^.HeavyOrSpecVehic.Year[Y] :=
          Scr3Data.Dest.Hev.Year[Y];
      END;
    ChangeFlag := True;
  END; {CopyScr3Info}

{*********************************************************************
 *  Name          : TScr3Dlg.Init
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value  :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
 *********************************************************************}

CONSTRUCTOR TScr3Dlg.Init(DataPage : INTEGER);

  VAR
    B,
    B2,
    L,
    L2,
    Y    : INTEGER;
    P    : PView;
    rect : TRect;

  BEGIN
    B := DataPage Div 16;
    B2 := DataPage Mod 16;
    rect.Assign(0,0,78,22);
```

```
TDialog.Init(rect, 'Screen Three - Movement Table');
Options := Options OR ofCentered;

For L2 := 0 To 1 Do
  Begin
    rect.Assign(2,1 + L2 * 9,20,2 + L2 * 9);
    Insert(New(PStaticText,init(rect,
      'FROM              TO')));
    For Y := 1 To MaxYears Do
      Begin
        rect.Assign (24 + Y * 8,1 + L2 * 9,28 + Y * 8,2 + L2 * 9);
        Insert(New(PStaticText,init(rect,FiscalYear(Y))));
      End;

    rect.Assign(21,2 + L2 * 9,30,3 + L2 * 9);
    Insert(New(PStaticText,init(rect,'Officers:')));

    rect.Assign(21,3 + L2 * 9,30,4 + L2 * 9);
    Insert(New(PStaticText,init(rect,'Enlisted:')));

    rect.Assign(20,4 + L2 * 9,30,5 + L2 * 9);
    Insert(New(PStaticText,init(rect,'Civilians:')));

    rect.Assign(21,5 + L2 * 9,30,6 + L2 * 9);
    Insert(New(PStaticText,init(rect,'Students:')));

    For L := 0 To 3 Do
      For Y := 0 To 5 Do
        Begin
          rect.Assign(30+Y*8,2+L + L2 * 9,37+Y*8,3+L + L2 * 9);
          P := New(PFInputLine,
            init(rect,5,DUnsignedSet,DInteger,0,0.0,30000.0));
          Insert (P);
          If (L + Y = 0) Then
            Begin
              rect.Assign (1 + L2 * 16,1 + L2 * 9,6 + L2 * 14,2 + L2 * 9);
              If L2 = 0
                Then Insert(New(PLabel,init(rect,'~F~ROM',P)))
                Else Insert(New(PLabel,init(rect,'~T~O',P)));
            End;
        End;

    rect.Assign(9,6 + L2 * 9,30,7 + L2 * 9);
    Insert(New(PStaticText,init(rect,'Mission Equip (tons):')));

    rect.Assign(9,7 + L2 * 9,30,8 + L2 * 9);
    Insert(New(PStaticText,init(rect,'Support Equip (tons):')));

    If (Pos ('ARM', UpperCase(Screen1Data^.Service)) = 0) Then
      Begin
        rect.Assign(6,8 + L2 * 9,30,9 + L2 * 9);
        Insert(New(PStaticText,init(rect,'Military Light Vehicles:')));
```

```
            rect.Assign(7,9 + L2 * 9,30,10 + L2 * 9);
            Insert(New(PStaticText,init(rect,'Heavy/Special Vehicles:')));
          End
        Else
          Begin
            rect.Assign(7,8 + L2 * 9,30,9 + L2 * 9);
            Insert(New(PStaticText,init(rect,'Mil Light Vehic (tons):')));
            rect.Assign(6,9 + L2 * 9,30,10 + L2 * 9);
            Insert(New(PStaticText,init(rect,'Heavy/Spec Vehic (tons):')));
          End;

        For L := 0 To 3 Do
          For Y := 0 To 5 Do
            Begin
              rect.Assign(30+Y*8,6+L + L2 * 9,37+Y*8,7+L + L2 * 9);
              Insert(New(PFInputLine,
                init(rect,5,DPosRealSet,DReal,0,0.0,99999.0)));
            End;
      End;

    rect.Assign(7,1,17,2);
    Insert(New(PStaticText,init(rect,Copy(Screen1Data^.Base[B].Name,1,10))));
    rect.Assign(7,10,17,11);
    Insert(New(PStaticText,init(rect,Copy(Screen1Data^.Base[B2].Name,1,10))));
    rect.Assign(21,10,31,11);
    Insert(New(PStaticText,init(rect,Copy(Screen1Data^.Base[B].Name,1,10))));
    rect.Assign(21,1,31,2);
    Insert(New(PStaticText,init(rect,Copy(Screen1Data^.Base[B2].Name,1,10))));

    rect.Assign(2,20,26,21);
    Insert(New(PStaticText,init(rect,'List Moves in Year ONLY!')));

    {*************************************************************
     * "Next", "Previous" and "Cancel" buttons
     *************************************************************}

    rect.Assign(33,19,45,21);
    Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

    rect.Assign(48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign(63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; {TScr3Dlg.Init}

END. {Unit CScr3}
```

```
UNIT CSCR4;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  FINPUT;

TYPE

  Screen4Data = RECORD
    TotOffEmp,
    TotEnlEmp,
    TotStuEmp,
    PercLivBase,
    TotCivEmp,
    PercNotMove,
    OffVacant,
    EnlVacant,
    TotFacSqFt,
    TotFacAcre,
    OffVHA,
    EnlVHA,
    PerDiem,
    FreightCost,
    RPMANonPayBudget,
    RPMAPayBudget,
    CommunBudget,
    BaseOpnsNonPayBudget,
    BaseOpnsPay,
    FamHousBudget,
    AreaCostFactor,
    ChampusIn,
    ChampusOut,
    ChampusShift : Real;
    BaseFlags    : Word;
  END;

  PScr4Dlg = ^TScr4Dlg;
  TScr4Dlg = OBJECT (TDialog)
               Scr4Data : Screen4Data;
               CONSTRUCTOR Init (DataPage : INTEGER);
               PROCEDURE InitScr4Info (DataPage : INTEGER);
               PROCEDURE CopyScr4Info (DataPage : INTEGER);
             END; {Object}

IMPLEMENTATION
```

```
{*************************************************************************
 *  Name          : InitScr4Info
 *
 *  Purpose       : Initialize the Scr4data data structure with data from
 *                  the global variables used by the calculation routine.
 *
 *  Invoked As    : InitScr4Info(DataPage)
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    DataPage          IN      array index in ActionEnvir^.GainBase[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  29Jan91   ASB   Creation
 *  11feb92   GJG   Added Screen4Data record to Method
 *  24Apr92   AML   Changed to Multibase format
 *************************************************************************}

PROCEDURE TScr4Dlg.InitScr4Info (DataPage : INTEGER);

  BEGIN
    Scr4Data.TotOffEmp := ActionEnvir^.Base[DataPage].BaseTotOffEmploy;
    Scr4Data.TotEnlEmp := ActionEnvir^.Base[DataPage].BaseTotEnlEmploy;
    Scr4Data.TotStuEmp := ActionEnvir^.Base[DataPage].BaseTotStuEmploy;
    Scr4Data.PercLivBase :=
      ActionEnvir^.Base[DataPage].PercFamLivOnBase * 100.0;
    Scr4Data.TotCivEmp := ActionEnvir^.Base[DataPage].BaseTotCivEmploy;
    Scr4Data.PercNotMove := ActionEnvir^.Base[DataPage].VolRIFPerc * 100.0;
    Scr4Data.OffVacant := ActionEnvir^.Base[DataPage].OffHousUnitsVacant;
    Scr4Data.EnlVacant := ActionEnvir^.Base[DataPage].EnlHousUnitsVacant;
    Scr4Data.TotFacSqFt := ActionEnvir^.Base[DataPage].BaseFacilSqFt;
    Scr4Data.TotFacAcre := ActionEnvir^.Base[DataPage].AcreOnBase;
    Scr4Data.OffVHA := PersCost.Base[DataPage].OffVHA;
    Scr4Data.EnlVHA := PersCost.Base[DataPage].EnlVHA;
    Scr4Data.PerDiem := PersCost.Base[DataPage].PerDiemRate;
    Scr4Data.FreightCost := FreightCost.Base[DataPage];
    Scr4Data.AreaCostFactor := Con.Base[DataPage].AreaCostFactor;
    Scr4Data.RPMANonPayBudget :=
      BudgetPropData^.Base[DataPage].RPMANonPayBudget / 1000.0;
    Scr4Data.RPMAPayBudget :=
      BudgetPropData^.Base[DataPage].RPMAPayBudget / 1000.0;
    Scr4Data.CommunBudget :=
      BudgetPropData^.Base[DataPage].CommunBudget / 1000.0;
    Scr4Data.BaseOpnsNonPayBudget :=
      BudgetPropData^.Base[DataPage].BaseOpnsNonPayBudget / 1000.0;
    Scr4Data.BaseOpnsPay :=
      BudgetPropData^.Base[DataPage].BaseOpnsPay / 1000.0;
```

```
    Scr4Data.FamHousBudget :=
      BudgetPropData^.Base[DataPage].FamHousBudget / 1000.0;
    Scr4Data.ChampusIn := Champus.Base[DataPage].ChampInPatCostVisit;
    Scr4Data.ChampusOut := Champus.Base[DataPage].ChampOutPatCostVisit;
    Scr4Data.ChampusShift :=
      ActionEnvir^.Base[DataPage].ChampusToMedicare * 100.0;
    If IndustFundSwitch.Base[DataPage] = 'Y'
      Then Scr4Data.BaseFlags := 2
      Else Scr4Data.BaseFlags := 0;
    If HapSwitch.Base[DataPage] = 'Y'
      Then Inc (Scr4Data.BaseFlags);
  END; {InitScr4Info}

{*************************************************************************
 *  Name          : CopyScr4Info
 *
 *  Purpose       : Copy the data from the Scr4data data structure into
 *                  the global variables used by the calculation routine.
 *
 *  Invoked As    : CopyScr4Info(DataPage)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    DataPage          IN      array index in ActionEnvir^.GainBase[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  29Jan91   ASB   Creation
 *  11feb92   GJG   Added Screen4Data record to Method
 *  24Apr92   AML   Changed to Multibase format
 *************************************************************************}

PROCEDURE TScr4Dlg.CopyScr4Info (DataPage : INTEGER);

  VAR
    Y : Integer;

  BEGIN
    If Odd (Scr4Data.BaseFlags Div 2) Then
      Begin
        IndustFundSwitch.Base[DataPage] := 'Y';
        If IndustFund.Base[DataPage] = Nil Then
          Begin
            New (IndustFund.Base[DataPage]);
            FillChar (IndustFund.Base[DataPage]^,
              SizeOf (IndustFund.Base[DataPage]^), 0);
          End;
      End
```

```
    Else
      Begin
        IndustFundSwitch.Base[DataPage] := 'N';
        If IndustFund.Base[DataPage] <> Nil Then
          Begin
            Dispose (IndustFund.Base[DataPage]);
            IndustFund.Base[DataPage] := Nil;
          End;
      End;
    If Odd (Scr4Data.BaseFlags)
      Then HapSwitch.Base[DataPage] := 'Y'
      Else HapSwitch.Base[DataPage] := 'N';
    ActionEnvir^.Base[DataPage].BaseTotOffEmploy := Scr4Data.TotOffEmp;
    ActionEnvir^.Base[DataPage].BaseTotEnlEmploy := Scr4Data.TotEnlEmp;
    ActionEnvir^.Base[DataPage].BaseTotStuEmploy := Scr4Data.TotStuEmp;
    ActionEnvir^.Base[DataPage].PercFamLivOnBase :=
      Scr4Data.PercLivBase / 100.0;
    ActionEnvir^.Base[DataPage].BaseTotCivEmploy := Scr4Data.TotCivEmp;
    ActionEnvir^.Base[DataPage].VolRIFPerc := Scr4Data.PercNotMove / 100.0;
    ActionEnvir^.Base[DataPage].OffHousUnitsVacant := Scr4Data.OffVacant;
    ActionEnvir^.Base[DataPage].EnlHousUnitsVacant := Scr4Data.EnlVacant;
    ActionEnvir^.Base[DataPage].BaseFacilSqFt := Scr4Data.TotFacSqFt;
    ActionEnvir^.Base[DataPage].AcreOnBase := Scr4Data.TotFacAcre;
    PersCost.Base[DataPage].OffVHA := Scr4Data.OffVHA;
    PersCost.Base[DataPage].EnlVHA := Scr4Data.EnlVHA;
    PersCost.Base[DataPage].PerDiemRate := Scr4Data.PerDiem;
    FreightCost.Base[DataPage] := Scr4Data.FreightCost;
    Con.Base[DataPage].AreaCostFactor := Scr4Data.AreaCostFactor;
    BudgetPropData^.Base[DataPage].RPMANonPayBudget :=
      Scr4Data.RPMANonPayBudget * 1000.0;
    BudgetPropData^.Base[DataPage].RPMAPayBudget :=
      Scr4Data.RPMAPayBudget * 1000.0;
    BudgetPropData^.Base[DataPage].CommunBudget :=
      Scr4Data.CommunBudget * 1000.0;
    BudgetPropData^.Base[DataPage].BaseOpnsNonPayBudget :=
      Scr4Data.BaseOpnsNonPayBudget * 1000.0;
    BudgetPropData^.Base[DataPage].BaseOpnsPay :=
      Scr4Data.BaseOpnsPay * 1000.0;
    BudgetPropData^.Base[DataPage].FamHousBudget :=
      Scr4Data.FamHousBudget * 1000.0;
    Champus.Base[DataPage].ChampInPatCostVisit := Scr4Data.ChampusIn;
    Champus.Base[DataPage].ChampOutPatCostVisit := Scr4Data.ChampusOut;
    ActionEnvir^.Base[DataPage].ChampusToMedicare :=
      Scr4Data.ChampusShift / 100.0;
    ChangeFlag := True;
  END; {CopyScr4Info}
```

```
{*************************************************************************
 *  Name          : TScr4Dlg.Init
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value  :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
 *************************************************************************}

CONSTRUCTOR TScr4Dlg.Init(DataPage : INTEGER);

  VAR
    P    : PView;
    rect : TRect;

  BEGIN
    rect.Assign(0,0,78,22);
    TDialog.Init(rect, 'Screen Four - Base Information (Static)');
    Options := Options OR ofCentered;

    rect.Assign(2,1,7,2);
    Insert(New(PStaticText,Init(rect,'Base:')));
    rect.Assign(9,1,29,2);
    Insert(New(PStaticText,Init(rect,Screen1Data^.Base[DataPage].Name)));

    rect.Assign(27, 2,37, 3);
    P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,50000.0));
    Insert (P);
    rect.Assign(11, 2,27, 3);
    Insert(New(PLabel,Init(rect,'~T~otal Officers:', P)));

    rect.Assign(12, 3,27, 4);
    Insert(New(PStaticText,Init(rect,'Total Enlisted:')));
    rect.Assign(27, 3,37, 4);
    Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,50000.0)));

    rect.Assign(12, 4,27, 5);
    Insert(New(PStaticText,Init(rect,'Total Students:')));
    rect.Assign(27, 4,37, 5);
    Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,50000.0)));

    rect.Assign( 4, 5,38, 6);
    Insert(New(PStaticText,Init(rect, '% Mil Families On Base:          %')));
    rect.Assign(27, 5,37, 6);
    Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));
```

```
rect.Assign( 2, 6,27, 7);
Insert(New(PStaticText,Init(rect,'Total Civilian Employees:')));
rect.Assign(27, 6,37, 7);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,50000.0)));

rect.Assign( 3, 7,38, 8);
Insert(New(PStaticText,Init(rect, '% Civs Not Will to Move:          %')));
rect.Assign(27, 7,37, 8);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));

rect.Assign( 2, 8,27, 9);
Insert(New(PStaticText,Init(rect,'Off Housing Units Vacant:')));
rect.Assign(27, 8,37, 9);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,9000.0)));

rect.Assign( 2, 9,27,10);
Insert(New(PStaticText,Init(rect,'Enl Housing Units Vacant:')));
rect.Assign(27, 9,37,10);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,9000.0)));

rect.Assign( 5,10,27,11);
Insert(New(PStaticText,Init(rect,   'Total Facilities (SF):')));
rect.Assign(27,10,37,11);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999.0)));

rect.Assign( 6,11,27,12);
Insert(New(PStaticText,Init(rect,    'Acreage on Base (AC):')));
rect.Assign(27,11,37,12);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,1E07)));

rect.Assign( 5,12,27,13);
Insert(New(PStaticText,Init(rect,   'Officer VHA ($/Month):')));
rect.Assign(27,12,37,13);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,20000.0)));

rect.Assign( 4,13,27,14);
Insert(New(PStaticText,Init(rect,  'Enlisted VHA ($/Month):')));
rect.Assign(27,13,37,14);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,20000.0)));

rect.Assign( 5,14,27,15);
Insert(New(PStaticText,Init(rect,   'Per Diem Rate ($/Day):')));
rect.Assign(27,14,37,15);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,400.0)));

rect.Assign( 3,15,27,16);
Insert(New(PStaticText,Init(rect, 'Freight Cost ($/Ton/Mi):')));
rect.Assign(27,15,37,16);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,100.0)));
```

```
rect.Assign(66, 2,76, 3);
P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999));
Insert (P);
rect.Assign(40, 2,66, 3);
Insert(New(PLabel,Init(rect, '~R~PMA Non-Payroll ($K/Yr):', P)));

rect.Assign(45, 3,66, 4);
Insert(New(PStaticText,Init(rect, 'RPMA Payroll ($K/Yr):')));
rect.Assign(66, 3,76, 4);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999)));

rect.Assign(40, 4,66, 5);
Insert(New(PStaticText,Init(rect,'Communicatn Costs ($K/Yr):')));
rect.Assign(66, 4,76, 5);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999)));

rect.Assign(41, 5,66, 6);
Insert(New(PStaticText,Init(rect, 'Base Ops Non-Pay ($K/Yr):')));
rect.Assign(66, 5,76, 6);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999)));

rect.Assign(41, 6,66, 7);
Insert(New(PStaticText,Init(rect, 'Base Ops Payroll ($K/Yr):')));
rect.Assign(66, 6,76, 7);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999)));

rect.Assign(40, 7,66, 8);
Insert(New(PStaticText,Init(rect,'Family Hous Costs ($K/Yr):')));
rect.Assign(66, 7,76, 8);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999)));

rect.Assign(49, 8,66, 9);
Insert(New(PStaticText,Init(rect, 'Area Cost Factor:')));
rect.Assign(66, 8,76, 9);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,2,0.0,5.0)));

rect.Assign(40,10,66,11);
Insert(New(PStaticText,Init(rect,'CHAMPUS In-Patient($/Vis):')));
rect.Assign(66,10,76,11);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0, 0.0, 99999999)));

rect.Assign(39,11,66,12);
Insert(New(PStaticText,Init(rect,  'CHAMPUS Out-Patient($/Vis):')));
rect.Assign(66,11,76,12);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0, 0.0, 99999999)));

rect.Assign(40,12,77,13);
Insert(New(PStaticText,Init(rect,'CHAMPUS Shift to Medicare:          %')));
rect.Assign(66,12,76,13);
Insert(New(PFInputLine,Init(rect,8,DPosRealSet,DReal,1,0.0,100.0)));
```

```
    rect.Assign (40, 14, 76, 16);
    Insert (New (PCheckBoxes, Init (rect,
      NewSItem ('~H~omeowner Assistance Program',
      NewSItem ('~U~nique Activity Information',
      nil)))));

    {*************************************************************
     * "Next", "Previous" and "Cancel" buttons
     *************************************************************}

    rect.Assign(33,19,45,21);
    Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

    rect.Assign(48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign(63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; {TScr4Dlg.Init}

END. {Unit CScr4}
```

```
UNIT CSCR5;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  CSUP,
  FINPUT;

TYPE
  Screen5Data = RECORD
                  OneTimUnq,
                  OneTimMov,
                  EnvirMit,
                  ActivMis,
                  MiscRecur,
                  PropAcre,
                  PropCost,
                  ConstrPct,
                  ShutDnPct,
                  ConAvoid,
                  FamAvoid,
                  ProAvoid  : Year1ToMaxRealRec;
                  ShutDnSF,
                  FamHousDn : Real;
                END;

  PScr5Dlg = ^TScr5Dlg;
  TScr5Dlg = OBJECT (TDialog)
               Scr5Data : Screen5Data;
               CONSTRUCTOR Init (datapage : INTEGER);
               PROCEDURE InitScr5Info (datapage : INTEGER);
               PROCEDURE CopyScr5Info (datapage : INTEGER);
             END; {Object}

IMPLEMENTATION
```

```
{*************************************************************************
 *  Name          : InitScr5Info
 *
 *  Purpose       : Initialize the scr5data data structure with data from
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Accesses the following global variable:
 *                     ActionEnvir^
 *
 *  Invoked As    : InitScr5Info (datapage)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    datapage          IN      array index in ActionEnvir^.GainBase[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  30Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen5Data record to Method
 *************************************************************************}

PROCEDURE TScr5Dlg.InitScr5Info (datapage : INTEGER);

  VAR
    Y : INTEGER;

  BEGIN
    Scr5Data.ShutDnSF := RealPropProc.Base[DataPage].FacilSqFtShutDown;
    Scr5Data.FamHousDn := ActionEnvir^.Base[DataPage].FamHousShutDn * 100.0;
    FOR Y := 1 TO ABSMAXYEARS DO
      BEGIN
        Scr5Data.EnvirMit.Year[Y] :=
          ActionEnvir^.Base[DataPage].EnvirMitigReqd.Year[Y] / 1000.0;
        Scr5Data.OneTimUnq.Year[Y] :=
          ActionEnvir^.Base[DataPage].OneTimeUniqueCosts.Year[Y] / 1000.0;
        Scr5Data.OneTimMov.Year[Y] :=
          ActionEnvir^.Base[DataPage].OneTimeMovingCosts.Year[Y] / 1000.0;
        Scr5Data.ActivMis.Year[Y] :=
          ActivMisCost^.Base[DataPage].Year[Y] / 1000.0;
        Scr5Data.MiscRecur.Year[Y] :=
          BudgetPropData^.Base[DataPage].MiscRecurCost.Year[Y] / 1000.0;
        Scr5Data.PropAcre.Year[Y] :=
          RealPropProc.RealPropPur.Base[DataPage].Acre.Year[Y];
        Scr5Data.PropCost.Year[Y] :=
          RealPropProc.RealPropPur.Base[DataPage].Cost.Year[Y] / 1000.0;
        Scr5Data.ConstrPct.Year[Y] :=
          TimePhasShutDn.Base[DataPage].Constr.Year[Y] * 100.0;
        Scr5Data.ShutDnPct.Year[Y] :=
```

```
          TimePhasShutDn.Base[DataPage].ShutDn.Year[Y] * 100.0;
        Scr5Data.ConAvoid.Year[Y] :=
          CostAvoid.Base[DataPage].Constr.Year[Y] / 1000.0;
        Scr5Data.FamAvoid.Year[Y] :=
          CostAvoid.Base[DataPage].FamQtr.Year[Y] / 1000.0;
        Scr5Data.ProAvoid.Year[Y] :=
          CostAvoid.Base[DataPage].Procure.Year[Y] / 1000.0;
      END;

  END; {InitScr5Info}

{*************************************************************************
 *  Name          : CopyScr5Info
 *
 *  Purpose       : Copy the data from the scr5data data structure into
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Alters the following global variable:
 *                     ActionEnvir^
 *
 *  Invoked As    : CopyScr5Info (datapage)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     structure to copy data from
 *    datapage          IN      array index in ActionEnvir^.GainBase[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  30Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen5Data record to Method
 *************************************************************************}

PROCEDURE TScr5Dlg.CopyScr5Info (datapage : INTEGER);

  VAR
    Y : INTEGER;

  BEGIN
    RealPropProc.Base[DataPage].FacilSqFtShutDown := Scr5Data.ShutDnSF;
    ActionEnvir^.Base[DataPage].FamHousShutDn := Scr5Data.FamHousDn / 100.0;
    FOR Y := 1 TO ABSMAXYEARS DO
      BEGIN
        ActionEnvir^.Base[DataPage].EnvirMitigReqd.Year[Y] :=
          Scr5Data.EnvirMit.Year[Y] * 1000.0;
        ActionEnvir^.Base[DataPage].OneTimeUniqueCosts.Year[Y] :=
          Scr5Data.OneTimUnq.Year[Y] * 1000.0;
```

```
        ActionEnvir^.Base[DataPage].OneTimeMovingCosts.Year[Y] :=
          Scr5Data.OneTimMov.Year[Y] * 1000.0;
        ActivMisCost^.Base[DataPage].Year[Y] :=
          Scr5Data.ActivMis.Year[Y] * 1000.0;
        BudgetPropData^.Base[DataPage].MiscRecurCost.Year[Y] :=
          Scr5Data.MiscRecur.Year[Y] * 1000.0;
        RealPropProc.RealPropPur.Base[DataPage].Acre.Year[Y] :=
          Scr5Data.PropAcre.Year[Y];
        RealPropProc.RealPropPur.Base[DataPage].Cost.Year[Y] :=
          Scr5Data.PropCost.Year[Y] * 1000.0;
        TimePhasShutDn.Base[DataPage].Constr.Year[Y] :=
          Scr5Data.ConstrPct.Year[Y] / 100.0;
        TimePhasShutDn.Base[DataPage].ShutDn.Year[Y] :=
          Scr5Data.ShutDnPct.Year[Y] / 100.0;
        CostAvoid.Base[DataPage].Constr.Year[Y] :=
          Scr5Data.ConAvoid.Year[Y] * 1000.0;
        CostAvoid.Base[DataPage].FamQtr.Year[Y] :=
          Scr5Data.FamAvoid.Year[Y] * 1000.0;
        CostAvoid.Base[DataPage].Procure.Year[Y] :=
          Scr5Data.ProAvoid.Year[Y] * 1000.0;
      END;
    ChangeFlag := True;
  END; {CopyScr5Info}

{*************************************************************************
 *  Name          : TScr5Dlg.Init
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value  :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
 *************************************************************************}

CONSTRUCTOR TScr5Dlg.Init (datapage : INTEGER);

  VAR
    P    : PView;
    rect : TRect;
    Y    : INTEGER;

  BEGIN
    {*********************************************************************
     * Create a dialog box to display data choices in
     *********************************************************************}
```

```
rect.Assign  (0,0,78,22);
TDialog.Init (rect, 'Screen Five - Base Information (Dynamic)');
Options := Options OR ofCentered;

{*************************************************************
 * Titles
 *************************************************************}

rect.Assign(2,1,7,2);
Insert(New(PStaticText,Init(rect,'Base:')));
rect.Assign(9,1,29,2);
Insert(New(PStaticText,Init(rect,Screen1Data^.Base[DataPage].Name)));

For Y := 1 To MaxYears Do
  Begin
    rect.Assign (16 + Y * 9, 2, 20 + Y * 9, 3);
    Insert(New(PStaticText,Init(rect,FiscalYear(Y))));
  End;

For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,3, 31+9*Y,4);
    P:=New(PFInputLine,Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0))
    Insert (P);
    If Y = 0 Then
      Begin
        rect.Assign(1,3,23,4);
        Insert(New(PLabel,Init(rect,'~1~-Time Uniq Cost($K):', P)));
      End;
  End;

rect.Assign(2,4,23,5);
Insert(New(PStaticText,Init(rect,'1-Time Move Cost($K):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,4, 31+9*Y,5);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0)));
  End;

rect.Assign(1,5,23,6);
Insert(New(PStaticText,Init(rect,'Env Non-Const Req($K):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 5, 31+9*Y, 6);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0)));
  End;

rect.Assign(2,6,23,7);
Insert(New(PStaticText,Init(rect,'Activ Misn Cost*($K):')));
```

```
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,6, 31+9*Y, 7);
    Insert (New(PFInputLine,
      Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0)));
  End;

For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,7, 31+9*Y,8);
    P:=New(PFInputLine,Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0));
    Insert (P);
    If Y = 0 Then
      Begin
        rect.Assign(1,7,23,8);
        Insert(New(PLabel,Init(rect,'~M~isc Recur Cost*($K):', P)));
      End;
  End;

rect.Assign(1,8,23,9);
Insert(New(PStaticText,Init(rect,'Prop +Buy / -Sale(AC):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,8, 31+9*Y, 9);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0)));
  End;

rect.Assign(1,9,23,10);
Insert(New(PStaticText,Init(rect,'Prop +Buy / -Sale($K):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 9, 31+9*Y,10);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DReal,0,-999999.0,999999.0)));
  End;

rect.Assign(1,10,77,11);
Insert(New(PStaticText,Init(rect,
  'Construction Schedule:' +
  '        %        %        %        %        %        %')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 10, 31+9*Y, 11);
    Insert(New(PFInputLine,Init(rect,6,DPosRealSet,DReal,0,0.0,100.0)));
  End;

rect.Assign(31,11,77,12);
Insert(New(PStaticText,Init(rect,
      '%        %        %        %        %        %')));
```

```
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 11, 31+9*Y, 12);
    P := New(PFInputLine,Init(rect,6,DPosRealSet,DReal,0,0.0,100.0));
    Insert (P);
    If Y = 0 Then
      Begin
        rect.Assign(4,11,23,12);
        Insert(New(PLabel,Init(rect, '~S~hutdown Schedule:', P)));
      End;
  End;

rect.Assign(2,12,23,13);
Insert(New(PStaticText,Init(rect, 'Construct Avoid ($K):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,12, 31+9*Y,13);
    Insert(New(PFInputLine,Init(rect,6,DPosRealSet,DReal,0,0.0,999999.0)));
  End;

rect.Assign(1,13,23,14);
Insert(New(PStaticText,Init(rect, 'FamHousCon Avoid ($K):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,13, 31+9*Y,14);
    Insert(New(PFInputLine,Init(rect,6,DPosRealSet,DReal,0,0.0,999999.0)));
  End;

rect.Assign(4,14,23,15);
Insert(New(PStaticText,Init(rect,   'Procure Avoid*($K):')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,14, 31+9*Y,15);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DReal,0,-99999.0,999999.0)));
  End;

Rect.Assign(22, 16, 32, 17);
P := New(PFInputLine,Init(rect,8,DPosRealSet,DReal,0,0.0,99999999));
Insert (P);
rect.Assign(1,16,22,17);
Insert(New(PLabel,Init(rect,'~F~acil ShutDown (SF):',P)));

rect.Assign(44,16,77,17);
Insert(New(PStaticText,Init(rect,'Family Housing ShutDown:        %')));
rect.assign(68,16,76,17);
Insert(New(PFInputLine,Init(rect,6,DPosRealSet,DReal,1,0.0,100.0)));

rect.Assign(1,20,33,21);
Insert(New(PStaticText,Init(rect,'*' + FiscalYear(MaxYears) +
  ' value used in Beyond years')));
```

```
    {*************************************************************
     * "Next", "Previous" and "Cancel" buttons.  Note that the
     *   "Next" button can only appear if we are on the first
     *   datapage & a second datapage is required.
     *************************************************************}

    rect.Assign  (33,19,45,21);
    Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

    rect.Assign  (48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign  (63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));

  END; {TScr5Dlg.Init}

END. {Unit CScr5}
```

```
UNIT CSCR6;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  CSUP,
  FINPUT;

TYPE
  Screen6Data = RECORD
                  OffReduc,
                  EnlReduc,
                  CivReduc,
                  OffElim,
                  EnlElim,
                  CivElim,
                  MilCare,
                  CivCare    : Year1ToMaxIntegerRec;
                  OnBaseIn,
                  OnBaseOut  : Year1ToMaxRealRec;
                END;

  PScr6Dlg = ^TScr6Dlg;
  TScr6Dlg = OBJECT (TDialog)
               Scr6Data : Screen6Data;
               CONSTRUCTOR Init (datapage : INTEGER);
               PROCEDURE InitScr6Info (datapage : INTEGER);
               PROCEDURE CopyScr6Info (datapage : INTEGER);
             END; {Object}

IMPLEMENTATION
```

```
{*************************************************************************
 *  Name          : InitScr6Info
 *
 *  Purpose       : Initialize the scr6data data structure with data from
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Accesses the following global variables:
 *                     PosElim, PosReduc, PosTrans
 *
 *  Invoked As    : InitScr6Info (datapage)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    datapage          IN      array index in ActionEnvir^.GainBase[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  05Jun92   AML   Creation
 *************************************************************************}

PROCEDURE TScr6Dlg.InitScr6Info (datapage : INTEGER);

  VAR
    Y : INTEGER;

  BEGIN
    FOR Y := 1 TO ABSMAXYEARS DO
      BEGIN
        Scr6Data.OffReduc.Year[Y]  := -PosReduc.Base[DataPage].Off.Year[Y];
        Scr6Data.EnlReduc.Year[Y]  := -PosReduc.Base[DataPage].Enl.Year[Y];
        Scr6Data.CivReduc.Year[Y]  := -PosReduc.Base[DataPage].Civ.Year[Y];
        Scr6Data.OffElim.Year[Y]   := PosElim.Base[DataPage].Off.Year[Y];
        Scr6Data.EnlElim.Year[Y]   := PosElim.Base[DataPage].Enl.Year[Y];
        Scr6Data.CivElim.Year[Y]   := PosElim.Base[DataPage].Civ.Year[Y];
        Scr6Data.MilCare.Year[Y]   :=
          PosTrans.Base[DataPage].CareTakeMil.Year[Y];
        Scr6Data.CivCare.Year[Y]   :=
          PosTrans.Base[DataPage].CareTakeCiv.Year[Y];
        Scr6Data.OnBaseIn.Year[Y]  :=
          Champus.Base[DataPage].OnBaseInPatVisits.Year[Y];
        Scr6Data.OnBaseOut.Year[Y] :=
          Champus.Base[DataPage].OnBaseOutPatVisits.Year[Y];
      END;

  END; {InitScr6Info}
```

```
{*************************************************************************
 *  Name          : CopyScr6Info
 *
 *  Purpose       : Copy the data from the scr6data data structure into
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Alters the following global variable:
 *                     ActionEnvir^
 *
 *  Invoked As    : CopyScr6Info (datapage)
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              VAR     structure to copy data from
 *    datapage          IN      array index in ActionEnvir^.GainBase[] to
 *                                begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  30Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen6Data record to Method
 *************************************************************************}

PROCEDURE TScr6Dlg.CopyScr6Info (datapage : INTEGER);

  VAR
    Y : INTEGER;

  BEGIN
    FOR Y := 1 TO ABSMAXYEARS DO
      BEGIN
        PosReduc.Base[DataPage].Off.Year[Y] := -Scr6Data.OffReduc.Year[Y];
        PosReduc.Base[DataPage].Enl.Year[Y] := -Scr6Data.EnlReduc.Year[Y];
        PosReduc.Base[DataPage].Civ.Year[Y] := -Scr6Data.CivReduc.Year[Y];
        PosElim.Base[DataPage].Off.Year[Y] := Scr6Data.OffElim.Year[Y];
        PosElim.Base[DataPage].Enl.Year[Y] := Scr6Data.EnlElim.Year[Y];
        PosElim.Base[DataPage].Civ.Year[Y] := Scr6Data.CivElim.Year[Y];
        PosTrans.Base[DataPage].CareTakeMil.Year[Y] :=
          Scr6Data.MilCare.Year[Y];
        PosTrans.Base[DataPage].CareTakeCiv.Year[Y] :=
          Scr6Data.CivCare.Year[Y];
        Champus.Base[DataPage].OnBaseInPatVisits.Year[Y] :=
          Scr6Data.OnBaseIn.Year[Y];
        Champus.Base[DataPage].OnBaseOutPatVisits.Year[Y] :=
          Scr6Data.OnBaseOut.Year[Y];
      END;
    ChangeFlag := True;
  END; {CopyScr6Info}
```

```
{*************************************************************************
 *  Name          : TScr6Dlg.Init
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
 *************************************************************************}

CONSTRUCTOR TScr6Dlg.Init (datapage : INTEGER);

  VAR
    P    : PView;
    rect : TRect;
    Y    : INTEGER;

  BEGIN
    {*****************************************************************
     * Create a dialog box to display data choices in
     *****************************************************************}

    rect.Assign  (0,0,78,22);
    TDialog.Init (rect, 'Screen Six - Base Information (Personnel)');
    Options := Options OR ofCentered;

    {*************************************************************
     * Titles
     *************************************************************}

    rect.Assign(2,1,7,2);
    Insert(New(PStaticText,Init(rect,'Base:')));
    rect.Assign(9,1,29,2);
    Insert(New(PStaticText,Init(rect,Screen1Data^.Base[DataPage].Name)));

    For Y := 1 To MaxYears Do
      Begin
        rect.Assign (16 + Y * 9, 2, 20 + Y * 9, 3);
        Insert(New(PStaticText,Init(rect,FiscalYear(Y))));
      End;

    rect.Assign(7,4,24,5);
    Insert(New(PStaticText,Init(rect,'Officer Changes:')));
```

```
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 4, 31+9*Y, 5);
    P := New(PFInputLine,
      Init(rect,6,DSignedSet,DInteger,0,-30000.0,30000.0));
    Insert (P);
    If Y = 0 Then
      Begin
        rect.Assign(1,3,47,4);
        Insert(New(PLabel,Init(rect,
          '~F~orce Structure Changes in Year (+Inc / -Dec)', P)));
      End;
  End;

rect.Assign(6,5,24,6);
Insert(New(PStaticText,Init(rect,'Enlisted Changes:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 5, 31+9*Y, 6);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DInteger,0,-30000.0,30000.0)));
  End;

rect.Assign(6,6,24,7);
Insert(New(PStaticText,Init(rect,'Civilian Changes:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 6, 31+9*Y, 7);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DInteger,0,-30000.0,30000.0)));
  End;

rect.Assign(3,8,24,9);
Insert(New(PStaticText,Init(rect,'Officers Eliminated:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y, 8, 31+9*Y, 9);
    P := New(PFInputLine,Init(rect,6,DUnsignedSet,DInteger,0,0.0,30000.0));
    Insert (P);
    If Y = 0 Then
      Begin
        rect.Assign(1,7,30,8);
        Insert(New(PLabel,Init(rect,'~E~liminated Positions by Year',P)));
      End;
  End;

rect.Assign(3,9,24,10);
Insert(New(PStaticText,Init(rect,'Enlisted Eliminated:')));
```

```
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,9, 31+9*Y,10);
    Insert(New(PFInputLine,
      Init(rect,6,DUnsignedSet,DInteger,0,0.0,30000.0)));
  End;

rect.Assign(2,10,24,11);
Insert(New(PStaticText,Init(rect,'Civilians Eliminated:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,10, 31+9*Y,11);
    Insert(New(PFInputLine,
      Init(rect,6,DUnsignedSet,DInteger,0,0.0,30000.0)));
  End;

rect.Assign(3,12,24,13);
Insert(New(PStaticText,Init(rect,'Military Caretakers:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,12, 31+9*Y,13);
    P := New(PFInputLine,
      Init(rect,6,DSignedSet,DInteger,0,-30000.0,30000.0));
    Insert (P);
    If Y = 0 Then
      Begin
        rect.Assign(1,11,38,12);
        Insert(New(PLabel,Init(rect,
          '~C~aretakers +Added / -Removed by Year', P)));
      End;
  End;

rect.Assign(3,13,24,14);
Insert(New(PStaticText,Init(rect,'Civilian Caretakers:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,13, 31+9*Y,14);
    Insert(New(PFInputLine,
      Init(rect,6,DSignedSet,DInteger,0,-30000.0,30000.0)));
  End;

rect.Assign(4,15,24,16);
Insert(New(PStaticText,Init(rect,'In-Pat Visits/Year:')));
For Y := 0 To 5 Do
  Begin
    Rect.Assign(23+9*Y,15, 31+9*Y,16);
    P := New(PFInputLine,Init(rect,6,DSignedSet,DReal,0,-99999,999999));
    Insert (P);
```

```
      If Y = 0 Then
        Begin
          rect.Assign(1,14,40,15);
          Insert(New(PLabel,Init(rect,
            '~R~etiree On-Base Medical (+Inc / -Dec)', P)));
        End;
    End;

  rect.Assign(3,16,24,17);
  Insert(New(PStaticText,Init(rect,'Out-Pat Visits/Year:')));
  For Y := 0 To 5 Do
    Begin
      Rect.Assign(23+9*Y,16, 31+9*Y,17);
      Insert(New(PFInputLine,Init(rect,6,DSignedSet,DReal,0,-99999,999999)));
    End;

  {*************************************************************
   * "Next", "Previous" and "Cancel" buttons.  Note that the
   *   "Next" button can only appear if we are on the first
   *   datapage & a second datapage is required.
   *************************************************************}

  rect.Assign  (33,19,45,21);
  Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

  rect.Assign  (48,19,60,21);
  Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

  rect.Assign  (63,19,75,21);
  Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));

END; {TScr6Dlg.Init}

END. {Unit CScr6}
```

```
UNIT CSCR7;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  FINPUT,
  CSUP;

TYPE
  Screen7Data = Record
                  Pro  : String20;
                  Cat  : String05;
                  New,
                  Rehab,
                  Cost : Real;
                  Com  : String60;
                End;

  PScr7Dlg = ^TScr7Dlg;
  TScr7Dlg = OBJECT (TDialog)
               Scr7Data : Array [1..16] of Screen7Data;
               CONSTRUCTOR Init (WhichBase : integer);
               PROCEDURE InitScr7Info (WhichBase : INTEGER);
               PROCEDURE CopyScr7Info (WhichBase : INTEGER);
             END; {Object}

IMPLEMENTATION
```

```
{*************************************************************************
 *  Name          : InitScr7Info
 *
 *  Purpose       : Initialize the scr7data data structure with data from
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Accesses the following global variables:
 *                  ConRqmts, ConCap, ConRehab
 *                : Formerly, there were three screen Sixes - one each for
 *                  requirements, capacity, and rehab.  Now, each base has
 *                  its own screen...
 *
 *  Invoked As    : InitScr7Info (WhichBase)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    datapage          IN      array index to begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  30Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen7Data record to Method
 *  31Mar92   AML   Switched to "Each base has a screen" format
 *************************************************************************}

PROCEDURE TScr7Dlg.InitScr7Info (WhichBase : INTEGER);

  VAR
    P : Integer;

  BEGIN
    For P := 1 To 16 Do
      Begin
        If Con.Base[WhichBase].ProjectName[P] <> Nil
          Then Scr7Data[P].Pro := Con.Base[WhichBase].ProjectName[P]^
          Else Scr7Data[P].Pro := '';
        Scr7Data[P].New        := Con.Base[WhichBase].New[P];
        Scr7Data[P].Rehab      := Con.Base[WhichBase].Rehab[P];
        Scr7Data[P].Cost       := Con.Base[WhichBase].Cost[P];
        If Con.Base[WhichBase].Comment[P] <> Nil
          Then Scr7Data[P].Com := Con.Base[WhichBase].Comment[P]^
          Else Scr7Data[P].Com := '';
        Case Con.Base[WhichBase].Category[P] of
          '@': Scr7Data[P].Cat := 'OTHER';
          'A': Scr7Data[P].Cat := 'HORIZ';
          'B': Scr7Data[P].Cat := 'WATER';
          'C': Scr7Data[P].Cat := 'AIROP';
          'D': Scr7Data[P].Cat := 'OPERA';
          'E': Scr7Data[P].Cat := 'ADMIN';
```

```
'F': Scr7Data[P].Cat := 'SCHLB';
'G': Scr7Data[P].Cat := 'MAINT';
'H': Scr7Data[P].Cat := 'BACHQ';
'I': Scr7Data[P].Cat := 'FAMLQ';
'J': Scr7Data[P].Cat := 'STORA';
'K': Scr7Data[P].Cat := 'DINFC';
'L': Scr7Data[P].Cat := 'RECFC';
'M': Scr7Data[P].Cat := 'COMFC';
'N': Scr7Data[P].Cat := 'SHIPY';
'O': Scr7Data[P].Cat := 'RDT&E';
'P': Scr7Data[P].Cat := 'POLST';
'Q': Scr7Data[P].Cat := 'AMMOS';
'R': Scr7Data[P].Cat := 'MEDFC';
'S': Scr7Data[P].Cat := 'ENVIR';
'T': If StdFctrsCon.Name[1] = ''
       Then Scr7Data[P].Cat := 'OPT-A'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[1],1,5));
'U': If StdFctrsCon.Name[2] = ''
       Then Scr7Data[P].Cat := 'OPT-B'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[2],1,5));
'V': If StdFctrsCon.Name[3] = ''
       Then Scr7Data[P].Cat := 'OPT-C'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[3],1,5));
'W': If StdFctrsCon.Name[4] = ''
       Then Scr7Data[P].Cat := 'OPT-D'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[4],1,5));
'X': If StdFctrsCon.Name[5] = ''
       Then Scr7Data[P].Cat := 'OPT-E'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[5],1,5));
'Y': If StdFctrsCon.Name[6] = ''
       Then Scr7Data[P].Cat := 'OPT-F'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[6],1,5));
'Z': If StdFctrsCon.Name[7] = ''
       Then Scr7Data[P].Cat := 'OPT-G'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[7],1,5));
'[': If StdFctrsCon.Name[8] = ''
       Then Scr7Data[P].Cat := 'OPT-H'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[8],1,5));
'\': If StdFctrsCon.Name[9] = ''
       Then Scr7Data[P].Cat := 'OPT-I'
       Else Scr7Data[P].Cat :=
         UpperCase(Copy(StdFctrsCon.Name[9],1,5));
```

```
        ']': If StdFctrsCon.Name[10] = ''
               Then Scr7Data[P].Cat := 'OPT-J'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[10],1,5));
        '^': If StdFctrsCon.Name[11] = ''
               Then Scr7Data[P].Cat := 'OPT-K'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[11],1,5));
        '_': If StdFctrsCon.Name[12] = ''
               Then Scr7Data[P].Cat := 'OPT-L'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[12],1,5));
        '''': If StdFctrsCon.Name[13] = ''
               Then Scr7Data[P].Cat := 'OPT-M'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[13],1,5));
        'a': If StdFctrsCon.Name[14] = ''
               Then Scr7Data[P].Cat := 'OPT-N'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[14],1,5));
        'b': If StdFctrsCon.Name[15] = ''
               Then Scr7Data[P].Cat := 'OPT-O'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[15],1,5));
        'c': If StdFctrsCon.Name[16] = ''
               Then Scr7Data[P].Cat := 'OPT-P'
               Else Scr7Data[P].Cat :=
                 UpperCase(Copy(StdFctrsCon.Name[16],1,5));
      End;
    End;
END; {InitScr7Info}
```

```
{*************************************************************************
 *  Name          : CopyScr7Info
 *
 *  Purpose       : Copy the data from the scr7data data structure into
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Alters the following global variables:
 *                  ConRqmts, ConCap, ConRehab
 *                : Formerly, there were three screen Sixes - one each for
 *                  requirements, capacity, and rehab.  Now, each base has
 *                  its own screen...
 *
 *  Invoked As    : CopyScr7Info (WhichBase)
 *
 *  Return value : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    datapage          IN      array index to begin copying from
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  30Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen7Data record to Method
 *  31Mar92   AML   Switched to "Each base has a screen" format
 *************************************************************************}

PROCEDURE TScr7Dlg.CopyScr7Info (WhichBase : INTEGER);

  Var
    P : Integer;
    S : String;

  Begin
    For P := 1 To 16 Do
      Begin
        If Scr7Data[P].Pro <> '' Then
          Begin
            If Con.Base[WhichBase].ProjectName[P] = Nil
              Then New (Con.Base[WhichBase].ProjectName[P]);
            Con.Base[WhichBase].ProjectName[P]^    := Scr7Data[P].Pro;
          End
        Else If Con.Base[WhichBase].ProjectName[P] <> Nil Then
          Begin
            Dispose (Con.Base[WhichBase].ProjectName[P]);
            Con.Base[WhichBase].ProjectName[P] := Nil;
          End;
        Con.Base[WhichBase].New[P]                 := Scr7Data[P].New;
        Con.Base[WhichBase].Rehab[P]               := Scr7Data[P].Rehab;
        Con.Base[WhichBase].Cost[P]                := Scr7Data[P].Cost;
        If Scr7Data[P].Com <> '' Then
```

```
  Begin
    If Con.Base[WhichBase].Comment[P] = Nil
      Then New (Con.Base[WhichBase].Comment[P]);
    Con.Base[WhichBase].Comment[P]^        := Scr7Data[P].Com;
  End
Else If Con.Base[WhichBase].Comment[P] <> Nil Then
  Begin
    Dispose (Con.Base[WhichBase].Comment[P]);
    Con.Base[WhichBase].Comment[P] := Nil;
  End;
S := UpperCase (Copy (Scr7Data[P].Cat, 1, 3));
If      (S = 'HOR')
  Then Con.Base[WhichBase].Category[P] := 'A'
Else If (S = 'WAT')
  Then Con.Base[WhichBase].Category[P] := 'B'
Else If (S = 'AIR')
  Then Con.Base[WhichBase].Category[P] := 'C'
Else If (S = 'OPE')
  Then Con.Base[WhichBase].Category[P] := 'D'
Else If (S = 'ADM')
  Then Con.Base[WhichBase].Category[P] := 'E'
Else If (S = 'SCH')
  Then Con.Base[WhichBase].Category[P] := 'F'
Else If (S = 'MAI')
  Then Con.Base[WhichBase].Category[P] := 'G'
Else If (S = 'BAC')
  Then Con.Base[WhichBase].Category[P] := 'H'
Else If (S = 'FAM')
  Then Con.Base[WhichBase].Category[P] := 'I'
Else If (S = 'STO') Or (S = 'COV')
  Then Con.Base[WhichBase].Category[P] := 'J'
Else If (S = 'DIN')
  Then Con.Base[WhichBase].Category[P] := 'K'
Else If (S = 'REC')
  Then Con.Base[WhichBase].Category[P] := 'L'
Else If (S = 'COM')
  Then Con.Base[WhichBase].Category[P] := 'M'
Else If (S = 'SHI')
  Then Con.Base[WhichBase].Category[P] := 'N'
Else If (S = 'RDT')
  Then Con.Base[WhichBase].Category[P] := 'O'
Else If (S = 'POL')
  Then Con.Base[WhichBase].Category[P] := 'P'
Else If (S = 'AMM')
  Then Con.Base[WhichBase].Category[P] := 'Q'
Else If (S = 'MED')
  Then Con.Base[WhichBase].Category[P] := 'R'
Else If (S = 'ENV')
  Then Con.Base[WhichBase].Category[P] := 'S'
Else If (S='OC1') Or (UpperCase(Scr7Data[P].Cat)='OPCT1') Or
        (S='OCA') Or (UpperCase(Scr7Data[P].Cat)='OPT-A') Or
        ((S=UpperCase(Copy(StdFctrsCon.Name[1],1,3)))And(S<>''))
```

```
    Then Con.Base[WhichBase].Category[P] := 'T'
  Else If (S='OC2') Or (UpperCase(Scr7Data[P].Cat)='OPCT2') Or
          (S='OCB') Or (UpperCase(Scr7Data[P].Cat)='OPT-B') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[2],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'U'
  Else If (S='OC3') Or (UpperCase(Scr7Data[P].Cat)='OPCT3') Or
          (S='OCC') Or (UpperCase(Scr7Data[P].Cat)='OPT-C') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[3],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'V'
  Else If (S='OC4') Or (UpperCase(Scr7Data[P].Cat)='OPCT4') Or
          (S='OCD') Or (UpperCase(Scr7Data[P].Cat)='OPT-D') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[4],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'W'
  Else If (S='OC5') Or (UpperCase(Scr7Data[P].Cat)='OPCT5') Or
          (S='OCE') Or (UpperCase(Scr7Data[P].Cat)='OPT-E') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[5],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'X'
  Else If (S='OCF') Or (UpperCase(Scr7Data[P].Cat)='OPT-F') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[6],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'Y'
  Else If (S='OCG') Or (UpperCase(Scr7Data[P].Cat)='OPT-G') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[7],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'Z'
  Else If (S='OCH') Or (UpperCase(Scr7Data[P].Cat)='OPT-H') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[8],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := '['
  Else If (S='OCI') Or (UpperCase(Scr7Data[P].Cat)='OPT-I') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[9],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := '\'
  Else If (S='OCJ') Or (UpperCase(Scr7Data[P].Cat)='OPT-J') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[10],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := ']'
  Else If (S='OCK') Or (UpperCase(Scr7Data[P].Cat)='OPT-K') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[11],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := '^'
  Else If (S='OCL') Or (UpperCase(Scr7Data[P].Cat)='OPT-L') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[12],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := '_'
  Else If (S='OCM') Or (UpperCase(Scr7Data[P].Cat)='OPT-M') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[13],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := '`'
  Else If (S='OCN') Or (UpperCase(Scr7Data[P].Cat)='OPT-N') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[14],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'a'
  Else If (S='OCO') Or (UpperCase(Scr7Data[P].Cat)='OPT-O') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[15],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'b'
  Else If (S='OCP') Or (UpperCase(Scr7Data[P].Cat)='OPT-P') Or
          ((S=UpperCase(Copy(StdFctrsCon.Name[16],1,3)))And(S<>''))
    Then Con.Base[WhichBase].Category[P] := 'c'
  Else   Con.Base[WhichBase].Category[P] := '@'
End;
```

```
    ChangeFlag := True;
  End; {CopyScr7Info}

{*********************************************************************
 *  Name          : TScr7Dlg.Info
 *
 *  Purpose       :
 *
 *  Notes         : Formerly, there were three screen Sixes - one each for
 *                  requirements, capacity, and rehab.  Now, each base has
 *                  its own screen...
 *
 *  Invoked As    :
 *
 *  Return value  :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
 *  31Mar92   AML   Switched to "Each base has a screen" format
 *********************************************************************}

CONSTRUCTOR TScr7Dlg.Init (WhichBase : Integer);

  VAR
    P    : INTEGER;
    PV   : PView;
    rect : TRect;

  BEGIN

    {*****************************************************************
     * Create a dialog box to display data choices in
     *****************************************************************}

    rect.Assign  ( 0, 0,78,22);
    TDialog.Init (rect,
      'Screen Seven - Base Information (Military Construction)');
    Options := Options OR ofCentered;

    {*************************************************************
     * Titles
     *************************************************************}

    rect.Assign  ( 2, 1, 59, 2);
    Insert(New(PStaticText,Init(rect,
      'Base:                                NEW                TOTAL')));
    rect.Assign  ( 2, 2, 74, 3);
    Insert(New(PStaticText,Init(rect,
      'DESCRIPTION:            CATEG:  CONSTRUC: REHAB:    COST($K): ' +
      'COMMENTS:')));
    rect.Assign  ( 9, 1, 29, 2);
```

```
    Insert(New(PStaticText,Init(rect,Screen1Data^.Base[WhichBase].Name)));

    {*************************************************************
     * Data fields for all other values
     *************************************************************}

    For P := 3 To 18 Do
      Begin
        rect.Assign(2, P, 24, P + 1);
        Insert(New(PFInputLine,Init(rect,20,DCharSet,DString,0,0.0,0.0)));
        rect.Assign(25, P, 32, P + 1);
        PV := New(PFInputLine,Init(rect,5,DCharSet,DString,0,0.0,0.0));
        PV^.HelpCtx := hcConCategories;
        Insert (PV);
        rect.Assign(33, P, 42, P + 1);
        Insert(New(PFInputLine,
          Init(rect,7,DPosRealSet,DReal,0, 0.0, 9999999.0)));
        rect.Assign(43, P, 52, P + 1);
        Insert(New(PFInputLine,
          Init(rect,7,DPosRealSet,DReal,0, 0.0, 9999999.0)));
        rect.Assign(53, P, 62, P + 1);
        Insert(New(PFInputLine,
          Init(rect,7,DPosRealSet,DReal,0, 0.0, 9999999.0)));
        rect.Assign (63, P, 76, P + 1);
        Insert(New(PFInputLine,Init(rect,60,DCharSet,DString,0,0.0,0.0)));
      End;

    {*************************************************************
     * "Next", "Previous" and "Cancel" buttons
     *************************************************************}

    rect.Assign(33,19,45,21);
    Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

    rect.Assign(48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign(63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; {TScr7Dlg.Init}

END. {Unit Cscr7}
```

```
UNIT CSCR8;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  CDEC,
  CSUP,
  FINPUT;

TYPE

  Screen8Rec1 = RECORD
                  AdminPlan,
                  Operating,
                  Mothball,
                  Caretaker,
                  MaintSpace,
                  Other        : Year1ToMaxReal
                END;

  Screen8Rec2 = RECORD
                  Operating,
                  Other        : Year1ToMaxReal
                END;

  Screen8Rec3 = RECORD
                  PackUnpack,
                  Freight,
                  Vehic,
                  Driv,
                  LossRate     : Year1ToMaxReal
                END;

  Screen8Data = RECORD
                  OvhdCost     : Screen8Rec1;
                  OvhdSave     : Screen8Rec2;
                  MoveCost     : Screen8Rec3;
                END;

  PScr8Dlg = ^TScr8Dlg;
  TScr8Dlg = OBJECT (TDialog)
               Scr8Data : Screen8Data;
               CONSTRUCTOR Init (DataPage : Integer);
               PROCEDURE InitScr8Info (DataPage : Integer);
               PROCEDURE CopyScr8Info (DataPage : Integer);
             END; {Object}
```

```
IMPLEMENTATION

{*********************************************************************
 *  Name          : InitScr8Info
 *
 *  Purpose       : Initialize the scr8data data structure with data from
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Accesses the following global variable:
 *                     IndustFund
 *
 *  Invoked As    : InitScr4Info
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *    info              OUT     structure to initialize
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  28Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen8Data record to Method
 *********************************************************************}

PROCEDURE TScr8Dlg.InitScr8Info (DataPage : Integer);

  VAR
    Y : Integer;

  BEGIN
    FOR Y := 1 to ABSMAXYEARS DO
      BEGIN
        Scr8Data.OvhdCost.AdminPlan[Y]  :=
          IndustFund.Base[DataPage]^.OvhdCost.AdminPlan.Year[Y]  / 1000.0;
        Scr8Data.OvhdCost.Operating[Y]  :=
          IndustFund.Base[DataPage]^.OvhdCost.Operating.Year[Y]  / 1000.0;
        Scr8Data.OvhdCost.Mothball[Y]   :=
          IndustFund.Base[DataPage]^.OvhdCost.Mothball.Year[Y]   / 1000.0;
        Scr8Data.OvhdCost.Caretaker[Y]  :=
          IndustFund.Base[DataPage]^.OvhdCost.Caretaker.Year[Y]  / 1000.0;
        Scr8Data.OvhdCost.MaintSpace[Y] :=
          IndustFund.Base[DataPage]^.OvhdCost.MaintSpace.Year[Y] / 1000.0;
        Scr8Data.OvhdCost.Other[Y]      :=
          IndustFund.Base[DataPage]^.OvhdCost.Other.Year[Y]      / 1000.0;
        Scr8Data.OvhdSave.Operating[Y]  :=
          IndustFund.Base[DataPage]^.OvhdSave.Operating.Year[Y]  / 1000.0;
        Scr8Data.OvhdSave.Other[Y]      :=
          IndustFund.Base[DataPage]^.OvhdSave.Other.Year[Y]      / 1000.0;
        Scr8Data.MoveCost.PackUnpack[Y] :=
          IndustFund.Base[DataPage]^.MoveCost.PackUnpack.Year[Y] / 1000.0;
```

```
        Scr8Data.MoveCost.Freight[Y]     :=
          IndustFund.Base[DataPage]^.MoveCost.Freight.Year[Y]    / 1000.0;
        Scr8Data.MoveCost.Vehic[Y]       :=
          IndustFund.Base[DataPage]^.MoveCost.Vehic.Year[Y]      / 1000.0;
        Scr8Data.MoveCost.Driv[Y]        :=
          IndustFund.Base[DataPage]^.MoveCost.Driv.Year[Y]       / 1000.0;
        Scr8Data.MoveCost.LossRate[Y]    :=
          IndustFund.Base[DataPage]^.MoveCost.LossRate.Year[Y]   / 1000.0;
      END;
  END;

{*************************************************************************
 *  Name          : CopyScr8Info
 *
 *  Purpose       : Copy the data from the scr8data data structure into
 *                  the global variables used by the calculation routine.
 *
 *  Notes         : Alters the following global variable:
 *                     IndustFund
 *
 *  Invoked As    : CopyScr4Info
 *
 *  Return value  : -none-
 *
 *  Parameters
 *    Name              Access  Purpose
 *    ---------------   ------  ----------------------------------------
 *
 *  Date      Init  Modifications
 *  -------   ----  ------------------------------------------------------
 *  28Jan91   ASB   Creation
 *  12feb92   GJG   Added Screen8Data record to Method
 *************************************************************************}

PROCEDURE TScr8Dlg.CopyScr8Info (DataPage : Integer);

  VAR
    Y : Integer;

  BEGIN
    FOR Y := 1 to ABSMAXYEARS DO
      BEGIN
         IndustFund.Base[DataPage]^.OvhdCost.AdminPlan.Year[Y]  :=
           Scr8Data.OvhdCost.AdminPlan[Y]  * 1000.0;
         IndustFund.Base[DataPage]^.OvhdCost.Operating.Year[Y]  :=
           Scr8Data.OvhdCost.Operating[Y]  * 1000.0;
         IndustFund.Base[DataPage]^.OvhdCost.Mothball.Year[Y]   :=
           Scr8Data.OvhdCost.Mothball[Y]   * 1000.0;
         IndustFund.Base[DataPage]^.OvhdCost.Caretaker.Year[Y]  :=
           Scr8Data.OvhdCost.Caretaker[Y]  * 1000.0;
         IndustFund.Base[DataPage]^.OvhdCost.MaintSpace.Year[Y] :=
           Scr8Data.OvhdCost.MaintSpace[Y] * 1000.0;
```

```
        IndustFund.Base[DataPage]^.OvhdCost.Other.Year[Y]       :=
          Scr8Data.OvhdCost.Other[Y]       * 1000.0;
        IndustFund.Base[DataPage]^.OvhdSave.Operating.Year[Y]   :=
          Scr8Data.OvhdSave.Operating[Y]   * 1000.0;
        IndustFund.Base[DataPage]^.OvhdSave.Other.Year[Y]       :=
          Scr8Data.OvhdSave.Other[Y]       * 1000.0;
        IndustFund.Base[DataPage]^.MoveCost.PackUnpack.Year[Y] :=
          Scr8Data.MoveCost.PackUnpack[Y] * 1000.0;
        IndustFund.Base[DataPage]^.MoveCost.Freight.Year[Y]     :=
          Scr8Data.MoveCost.Freight[Y]     * 1000.0;
        IndustFund.Base[DataPage]^.MoveCost.Vehic.Year[Y]       :=
          Scr8Data.MoveCost.Vehic[Y]       * 1000.0;
        IndustFund.Base[DataPage]^.MoveCost.Driv.Year[Y]        :=
          Scr8Data.MoveCost.Driv[Y]        * 1000.0;
        IndustFund.Base[DataPage]^.MoveCost.LossRate.Year[Y]    :=
          Scr8Data.MoveCost.LossRate[Y]    * 1000.0;
      END;
    ChangeFlag := True;
  END;

{*************************************************************************
 *  Name          : TScr8Dlg.Init
 *
 *  Purpose       :
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18Jan91   ASB   Creation - reorganization of code
*************************************************************************}

CONSTRUCTOR TScr8Dlg.Init (DataPage : Integer);

  VAR
    L, Y     : INTEGER;
    P        : PView;
    rect     : TRect;

  BEGIN
    {*****************************************************************
     * Create a dialog box to display data choices in
     *****************************************************************}

    rect.Assign  ( 0, 0,78,22);
    TDialog.Init (rect, 'Screen Eight - Base Information (Unique Activity)');
    Options := Options OR ofCentered;
```

```
rect.Assign (2,1,42,2);
Insert(New(PStaticText,Init(rect,
  'Base:                                  ($K)')));
rect.Assign (9,1,29,2);
Insert(New(PStaticText,Init(rect,Screen1Data^.Base[DataPage].Name)));

rect.Assign(1,20,33,21);
Insert(New(PStaticText,Init(rect,'*' + FiscalYear(MaxYears) +
  ' value used in Beyond years')));

{*************************************************************
 * Years Header Across Top
 *************************************************************}

FOR Y := 1 TO MaxYears DO
  BEGIN
    rect.Assign (11+Y*10, 2, 15+Y*10, 3);
    Insert(New(PStaticText,Init(rect,FiscalYear(Y))));
  END;

{*************************************************************
 * Overhead Costs Titles
 *************************************************************}

rect.Assign  ( 2, 3,16, 4);
Insert(New(PStaticText,Init(rect,'OVHD  Adm/Plan')));

rect.Assign  ( 7, 4,17, 5);
Insert(New(PStaticText,Init(rect,'*Operating')));

rect.Assign  ( 8, 5,16, 6);
Insert(New(PStaticText,Init(rect,'Mothball')));

rect.Assign  ( 7, 6,17, 7);
Insert(New(PStaticText,Init(rect,'*Caretaker')));

rect.Assign  ( 7, 7,17, 8);
Insert(New(PStaticText,Init(rect,'*Maint Spc')));

rect.Assign  ( 7, 8,13, 9);
Insert(New(PStaticText,Init(rect,'*Other')));

{*************************************************************
 * Overhead Costs Fields
 *************************************************************}

For L := 3 to 8 DO
  For Y := 1 TO 6 DO
    BEGIN
      rect.Assign(8+Y*10, L, 17+Y*10, L + 1);
      P := New(PFInputLine,Init(rect,7,DPosRealSet,DReal,0,0.0,9999999.0));
      Insert (P);
```

```
      If (L=3) And (Y=1) Then
        Begin
          rect.Assign (1,4,6,5);
          Insert(New(PLabel,init(rect,'~C~OST',P)));
        End;
    END;

{*****************************************************************
 * Overhead Savings Titles
 *****************************************************************}

rect.Assign  ( 2,10,17,11);
Insert(New(PStaticText,Init(rect,'OVHD *Operating')));

rect.Assign  ( 7,11,13,12);
Insert(New(PStaticText,Init(rect,'*Other')));

{*****************************************************************
 * Overhead Savings Fields
 *****************************************************************}

For L := 10 to 11 DO
  For Y := 1 TO 6 DO
    BEGIN
      rect.Assign (8+Y*10, L, 17+Y*10, L + 1);
      P := New(PFInputLine,Init(rect,7,DPosRealSet,DReal,0,0.0,9999999.0));
      Insert (P);
      If (L=10) And (Y=1) Then
        Begin
          rect.Assign(1,11,6,12);
          Insert(New(PLabel,init(rect,'~S~AVE',P)));
        End;
    END;

{*****************************************************************
 * Moving Costs Titles
 *****************************************************************}

rect.Assign  ( 8,13,17,14);
Insert(New(PStaticText,Init(rect,'Pack/Unpa')));

rect.Assign  ( 2,14,15,15);
Insert(New(PStaticText,Init(rect,'COST  Freight')));

rect.Assign  ( 8,15,17,16);
Insert(New(PStaticText,Init(rect,'Vehic Mov')));

rect.Assign  ( 8,16,15,17);
Insert(New(PStaticText,Init(rect,'Driving')));

rect.Assign  ( 8,17,17,18);
Insert(New(PStaticText,Init(rect,'Loss Rate')));
```

```
    {*****************************************************************
     * Moving Costs Fields
     *****************************************************************}

    For L := 13 to 17 DO
      For Y := 1 TO 6 DO
        BEGIN
          rect.Assign (8+Y*10, L, 17+Y*10, L + 1);
          P := New(PFInputLine,Init(rect,7,DPosRealSet,DReal,0,0.0,9999999.0));
          Insert (P);
          If (L=13) And (Y=1) Then
            Begin
              rect.Assign(1,13,6,14);
              Insert(New(PLabel,init(rect,'~M~OVE',P)));
            End;
        END;

    {*************************************************************
     * "Previous" and "Cancel" buttons
     *************************************************************}

    rect.Assign  (33,19,45,21);
    Insert(New(PButton, Init(rect, '~N~ext', cmOK, bfNormal)));

    rect.Assign  (48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign  (63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; {TScr8Dlg.Init}

END. {Unit CScr8}
```

```
UNIT CSCR9;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT;

TYPE
  PScr9Dlg = ^TScr9Dlg;
  TScr9Dlg = OBJECT (TDialog)
               CONSTRUCTOR Init;
             END; {Object}

IMPLEMENTATION

{*************************************************************************
 *  Name          : TScr9Dlg.Init
 *
 *  Purpose       : Set up screen Nine
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18May92   AML   Creation
 *************************************************************************}

CONSTRUCTOR TScr9Dlg.Init;

  VAR
    L    : INTEGER;
    P    : PView;
    rect : TRect;

  BEGIN
    rect.Assign (0,0,78,22);
    TDialog.Init (rect,'Screen Nine - Explanatory Notes');
    Options := Options OR ofCentered;

    For L := 1 to 15 Do
      Begin
        rect.Assign (2, 2 + L, 56, 3 + L);
        P := New(PFInputLine,Init(rect,60,DCharSet,DString,0,0.0,0.0));
```

```
        Insert (P);
        If L = 1 Then
          Begin
            rect.Assign (1,2,42,3);
            Insert(New(PLabel,Init(rect,
              '~E~xplanatory Notes for Input Data report:', P)));
          End;
      End;

    rect.Assign (57, 3, 76, 15);
    P := New (PCheckBoxes, Init (rect,
      NewSItem ('~1~-Gen Scenario',
      NewSItem ('~2~-Distance',
      NewSItem ('~3~-Movement',
      NewSItem ('~4~-Static Info',
      NewSItem ('~5~-Dynamic Info',
      NewSItem ('~6~-Personl Info',
      NewSItem ('~7~-MilCon Info',
      NewSItem ('~8~-Unique Info',
      NewSItem ('SF1-Per~S~onnel',
      NewSItem ('SF2-~F~acilities',
      NewSItem ('SF3-~T~ransport',
      NewSItem ('SF4-~C~onstruct',
      nil)))))))))))))));
    Insert (P);
    rect.Assign (56,2,76,3);
    Insert(New(PLabel,Init(rect, 'Pages Foot~N~oted:', P)));


    {*****************************************************************
     * "OK" and "Cancel" buttons
     *****************************************************************}

    rect.Assign  (48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign  (63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; {TScr9Dlg.Init}

END. {Unit CScr9}
```

```
UNIT CSCR9;

{$F+,O+,S-,D+,L+}

INTERFACE

USES
  DIALOGS,
  VIEWS,
  OBJECTS,
  FINPUT;

TYPE
  PScr9Dlg = ^TScr9Dlg;
  TScr9Dlg = OBJECT (TDialog)
               CONSTRUCTOR Init;
             END; {Object}

IMPLEMENTATION

{*************************************************************************
 *  Name          : TScr9Dlg.Init
 *
 *  Purpose       : Set up screen Nine
 *
 *  Notes         :
 *
 *  Invoked As    :
 *
 *  Return value  :
 *
 *  Date      Init  Modifications
 *  -------   ----  ---------------------------------------------------
 *  18May92   AML   Creation
 *************************************************************************}

CONSTRUCTOR TScr9Dlg.Init;

  VAR
    L    : INTEGER;
    P    : PView;
    rect : TRect;

  BEGIN
    rect.Assign (0,0,78,22);
    TDialog.Init (rect,'Screen Nine - Explanatory Notes');
    Options := Options OR ofCentered;

    For L := 1 to 15 Do
      Begin
        rect.Assign (2, 2 + L, 56, 3 + L);
        P := New(PFInputLine,Init(rect,60,DCharSet,DString,0,0.0,0.0));
```

```
        Insert (P);
        If L = 1 Then
          Begin
            rect.Assign (1,2,42,3);
            Insert(New(PLabel,Init(rect,
              '~E~xplanatory Notes for Input Data report:', P)));
          End;
      End;

    rect.Assign (57, 3, 76, 15);
    P := New (PCheckBoxes, Init (rect,
      NewSItem ('~1~-Gen Scenario',
      NewSItem ('~2~-Distance',
      NewSItem ('~3~-Movement',
      NewSItem ('~4~-Static Info',
      NewSItem ('~5~-Dynamic Info',
      NewSItem ('~6~-Personl Info',
      NewSItem ('~7~-MilCon Info',
      NewSItem ('~8~-Unique Info',
      NewSItem ('SF1-Per~S~onnel',
      NewSItem ('SF2-~F~acilities',
      NewSItem ('SF3-~T~ransport',
      NewSItem ('SF4-~C~onstruct',
      nil)))))))))))))));
    Insert (P);
    rect.Assign (56,2,76,3);
    Insert(New(PLabel,Init(rect, 'Pages Foot~N~oted:', P)));


    {*****************************************************************
     * "OK" and "Cancel" buttons
     *****************************************************************}

    rect.Assign  (48,19,60,21);
    Insert(New(PButton, Init(rect, '~P~revious', cmYes, bfNormal)));

    rect.Assign  (63,19,75,21);
    Insert(New(PButton, Init(rect, '~D~one', cmNo, bfNormal)));
  END; {TScr9Dlg.Init}

END. {Unit CScr9}
```

```
Unit CSUP;
{$F+,O+,S-,D+,L+}
{$UNDEF  PROTO}   { <-- Define PROTO for Prototype marker '*P' on all reports  }

Interface

Uses
  CDEC,
  CRT,
  DOS,
  Drivers;

Function  AsOfDateTime                                  : String20;

Function  AtLeastZero  (Number       : Real)      : Real;

Function  BaseNState   (WhichBase    : Integer)   : String;

Procedure BeyondOut(Var Fil          : Text;
                        OutLabel     : String20;
                        Value        : YearAndBeyondRealRec;
                        Adjst        : Year1ToMaxRealRec;
                        Size,
                        DoBeyond     : Integer);

Procedure Center   (Var Fil          : Text;
                        OutLabel     : String);

Function  DirExists    (Path         : String)    : Boolean;

Procedure ErrorList    (ErrorBases   : Word);

Procedure ErrorOut     (OutLabel     : String);

Function  FileExists   (FileName     : String)    : Boolean;

Function  FiscalYear   (AbsYear      : Integer)   : String04;

Procedure FiscOut  (Var Fil          : Text;
                        Leading,
                        Size,
                        DoBeyond     : Integer);

Function  Min          (Num1, Num2   : Real)      : Real;

Function  Rept         (Stuff        : String;
                        Number       : Integer)   : String;

Procedure RunTimeError;

Function  StripFile    (FileName     : STRING)    : STRING;
```

```
Function  StripSpace   (Str          : String)    : String;

Function  TimeStr      (Year,
                        Month,
                        Day,
                        Hour,
                        Minute       : Integer)   : String20;

Procedure Title        (Var Fil      : Text;
                        ReportName   : String;
                        PageNumber   : Integer);

Function  UpperCase    (Stuff        : String)    : String;

Function  WithCommas   (Value        : Real;
                        DecPlaces    : Integer)   : String;

Function  Within       (Num1,
                        Num2,
                        Num3         : Real)      : Real;

Procedure YearOut  (Var Fil          : Text;
                        OutLabel     : String20;
                        Value,
                        Adjst        : Year1ToMaxRealRec;
                        Size,
                        DoBeyond     : Integer);

Function  YtoX         (Base, Power  : Real)      : Real;

Function  Zerocheck    (Num1, Num2   : Real)      : Real;

Implementation

{*************************************************************************
 *   Name           : AsOfDateTime
 *
 *   Purpose        : Used for date/time-stamping reports
 *
 *   Invoked As     : AsOfDateTime
 *
 *   Return Value   : String, containing date and time
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   27Mar92  AML   Creation
 *************************************************************************}
```

```
Function AsOfDateTime : String20;

  Var
    Year, Month, Day, Hour, Minute, Filler : Word;

  Begin
    GetDate (Year, Month, Day, Filler);
    GetTime (Hour, Minute, Filler, Filler);
    AsOfDateTime := TimeStr (Year, Month, Day, Hour, Minute);
  End;

{*************************************************************************
 *   Name           : AtLeastZero
 *
 *   Purpose        : Makes sure a value is at least zero
 *
 *   Notes          : Used to be a function MaxInteger with returned the greater
 *                    of two numbers.  It was almost always used to compare a
 *                    number to zero, so the function was specialized.
 *
 *   Invoked As     : AtLeastZero (Number)
 *
 *   Return value   : Integer
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   29May92   AML   Creation
 *************************************************************************}

Function AtLeastZero (Number : Real) : Real;

  Begin
    If Number > 0
      Then AtLeastZero := Number
      Else AtLeastZero := 0;
  End;

Function BaseNState (WhichBase : Integer) : String;

  Begin
    If Screen1Data^.Base[WhichBase].State = ''
      Then BaseNState := Screen1Data^.Base[WhichBase].Name
      Else BaseNState := Concat
        (Screen1Data^.Base[WhichBase].Name, ', ',
         Screen1Data^.Base[WhichBase].State);
  End;
```

```
{*************************************************************************
 *   Name           : BeyondOut
 *
 *   Purpose        : Outputs a report line (label, values for year and beyond),
 *                    doing all adjustments for ranges, inflation, etc.
 *
 *   Notes          : The YearOut procedure requires records of the type
 *                    Year1ToMaxRealRec.  Many records have a beyond field but
 *                    don't output them; therefore, BeyondOut, which takes
 *                    YearAndBeyondRealRec, optionally will not display the
 *                    Beyond field.
 *
 *   Invoked As     : BeyondOut (Fil, OutLable, Value, Adjst, Size, DoBeyond);
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   02Jun92   AML   Creation
 *************************************************************************

Procedure BeyondOut (Var Fil      : Text;
                         OutLabel : String20;
                         Value    : YearAndBeyondRealRec;
                         Adjst    : Year1ToMaxRealRec;
                         Size,
                         DoBeyond : Integer);

  Var
    Y : Integer;

  Begin
    Write (Fil, Blank:10, OutLabel);
    For Y := 1 to MaxYears DO
      Write (Fil, WithCommas (Within ( -(YtoX(10,Size-2)-1),
        Value.Year[Y] * Adjst.Year[Y],  (YtoX(10,Size-2)-1) ),0):Size);
    If DoBeyond = 0
      Then Writeln (Fil)
      Else Writeln (Fil, WithCommas (Within ( -(YtoX(10,Size-2) -1),
        Value.Beyond * Adjst.Year[MaxYears], (YtoX(10,Size-2)-1) ),0):Size);
  End;
```

```
{*****************************************************************************
 *   Name           : Center
 *
 *   Purpose        : Centers a line in a report
 *
 *   Invoked As     : Center (Fil, OutLabel)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   11Jun92   AML   Creation
 *****************************************************************************}

Procedure Center (Var Fil : Text; OutLabel : String);

  Begin
    Writeln (Fil, Blank:(40 - Length(OutLabel) Div 2), OutLabel);
  End;

{*****************************************************************************
 *   Name           : DirExists
 *
 *   Purpose        : Determines if a directory exists
 *
 *   Invoked As     : DirExists (Path)
 *
 *   Return value   : Boolean
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------------
 *   08Jul92   AML   Creation
 *****************************************************************************}

Function DirExists (Path : String):Boolean;

  Var
    F : Text;
    R : Boolean;

  Begin
    {$I-}
    Assign (F, Path + 'COBTEST.TMP');
    ReWrite (F);
    {$I+}
    R := (IOResult = 0);
```

```
    If R Then
      Begin
        Close (F);
        Erase (F);
      End;
    DirExists := R;
  End;

{*****************************************************************************
 *   Name           : ErrorList and ErrorOut
 *
 *   Purpose        : Error checking of scenario data
 *
 *   Notes          : Outputs a list of bases or a line of text to ERROR.RPT
 *                    (with an opening title, if necessary).
 *                  : These algorithms are part of the Inconsistent Data
 *                    Input/Output Tester (I.D.I.O.T.)
 *
 *   Invoked As     : ErrorList (ErrorBases) and ErrorOut (OutLabel)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   25Jun92   AML   Creation
 *****************************************************************************}

Procedure ErrorList (ErrorBases : Word);

  Var
    B : Integer;

  Begin
    For B := 0 To MaxBases Do
      If Odd(ErrorBases Div Round(YtoX(2,B)))
        Then ErrorOut ('     ' + BaseNState(B));
    ErrorOut ('');
  End;

Procedure ErrorOut (OutLabel : String);

  Var
    F : Text;

  Begin
    If FileExists (Info.ReportDir + 'Error.Rpt') Then
      Begin
        Assign (F, Info.ReportDir + 'Error.Rpt');
        Append (F);
      End
```

```
    Else
      Begin
        Assign (F, Info.ReportDir + 'Error.Rpt');
        ReWrite (F);
        Title (F, 'SCENARIO ERROR REPORT', -1);
        Writeln (F, Blank:10, 'Scenario File  : ', DataFileUsed);
        Writeln (F, Blank:10, 'Std Fctrs File : ', SFFileUsed);
        Writeln (F);
      End;
    Writeln (F, Blank:10, Outlabel);
    Close (F);
  End;

{*****************************************************************************
 *   Name           : FileExists
 *
 *   Purpose        : Determines if a file exists
 *
 *   Invoked As     : FileExists (FileName)
 *
 *   Return value   : Boolean
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   07Apr92   AML   Creation
 *****************************************************************************}

Function FileExists (FileName : String):Boolean;

  Var
    F : Text;
    R : Boolean;

  Begin
    Assign (F, FileName);
    {$I-}  Reset (F);  {$I+}
    R := (IOResult = 0);
    If R
      Then Close (F);
    FileExists := R;
  End;
```

```
{*************************************************************************
 *   Name           : FiscalYear
 *
 *   Purpose        : Returns, as a string, a Fiscal Year (19xx-21xx) based on
 *                    an absolute year (Year x)
 *
 *   Invoked As     : FiscalYear (Offset)
 *
 *   Return value   : String04
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   09Jun92   AML   Creation
 *************************************************************************}

Function FiscalYear (AbsYear : Integer) : String04;

  Var
    TempStr : String04;

  Begin
    Str (Screen1Data^.FY1 + AbsYear - 1, TempStr);
    FiscalYear := TempStr;
  End;

{*************************************************************************
 *   Name           : FiscOut
 *
 *   Purpose        : Outputs a row of years to head report tables.  Like
 *                    BeyondOut and YearOut, FiscOut also has parameters for
 *                    an amount of leading spaces, size of fields, and whether
 *                    or not to include a Beyond field.
 *
 *   Invoked As     : FiscOut (Fil, OutLable, Size, DoBeyond);
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   09Jun92   AML   Creation
 *************************************************************************}

Procedure FiscOut  (Var Fil      : Text;
                        Leading,
                        Size,
                        DoBeyond : Integer);
```

```
  Var
    Y : Integer;

  Begin
    Write (Fil, Blank:Leading);
    For Y := 1 To MaxYears Do
      Write (Fil, FiscalYear(Y):Size);
    If DoBeyond = 0
      Then Writeln (Fil)
      Else Writeln (Fil, 'Beyond':Size);
  End;

{*****************************************************************************
 *   Name           : Min
 *   Purpose        : Determines minimum of two numbers
 *   Invoked As     : Min (Num1, Num2)
 *   Return value   : INTEGER
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   03Jan91   AJD   Creation
 *   25Mar92   AML   Streamlining
 *****************************************************************************}

Function Min (Num1, Num2  : Real) : Real;

  Begin { Min }
    If Num1 < Num2
      Then Min := Num1
      Else Min := Num2;
  End;  { Min }

{*****************************************************************************
 *   Name           : Rept
 *   Purpose        : Returns a string with the specified number of repetitions
 *                    of another string.
 *   Invoked As     : Rept (Stuff, Number)
 *   Return value   : String
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   06May92   AML   Creation
 *****************************************************************************}
```

```
Function Rept (Stuff: String; Number : Integer): String;

  Var
    I : Integer;
    S : String;

  Begin
    S := '';
    For I := 1 To Number Do
      S := ConCat (S, Stuff);
    Rept := S;
  End;

{*****************************************************************************
 *   Name          : RunTimeError
 *
 *   Purpose       : Provide a friendlier exit in case of a runtime error.
 *
 *   Notes         : This procedure is never called directly, by is linked
 *                   into the exit chain.
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ---------------------------------------------------------
 *   25Jun92  AML   Creation
 *****************************************************************************}

Procedure RunTimeError;
  Const
    Hex : Array [0..15] of Char = '0123456789ABCDEF';
  Var
    R : Integer;
  Begin
    If ErrorAddr <> Nil Then
      Begin
        HideMouse;
        Reset (Input);
        TextColor (15);
        TextBackGround (7);
        GotoXY (20,5);
        Writeln ('┌──────────────────────────────────────────┐');
        For R := 6 To 20 Do
          Begin
            GotoXY (20, R);
            Writeln ('│', '│':40);
          End;
        GotoXY (20,21);
        Writeln ('└──────────────────────────────────────────┘');
        TextColor (0);
```

```
      GotoXY (23,6);
      Writeln ('TurboPascal Run-Time Error #',
        Chr (48 + ExitCode Div 100),
        Chr (48 + (ExitCode Mod 100) Div 10),
        Chr (48 + ExitCode Mod 10), ' has');
      GotoXY (23,7);
      Writeln ('just occurred at address $',
        Hex[Hi(Seg(ErrorAddr^)) Div 16],
        Hex[Hi(Seg(ErrorAddr^)) Mod 16],
        Hex[Lo(Seg(ErrorAddr^)) Div 16],
        Hex[Lo(Seg(ErrorAddr^)) Mod 16], ':',
        Hex[Hi(Ofs(ErrorAddr^)) Div 16],
        Hex[Hi(Ofs(ErrorAddr^)) Mod 16],
        Hex[Lo(Ofs(ErrorAddr^)) Div 16],
        Hex[Lo(Ofs(ErrorAddr^)) Mod 16]);
      GotoXY (23,8);
      Writeln ('while running COBRA v', Version, ' ...');
      GotoXY (23,10);
      Writeln ('For information, give these numbers');
      GotoXY (23,11);
      Writeln ('and an explanation of what you were');
      GotoXY (23,12);
      Writeln ('doing to:');
      GotoXY (30,14);
      Writeln ('R&K Engineering, Inc.');
      GotoXY (27,15);
      Writeln ('1733 King Street, Suite 320');
      GotoXY (30,16);
      Writeln ('Alexandria, VA  22314');
      GotoXY (34,17);
      Writeln ('(703)683-7100');
      GotoXY (32,18);
      Writeln ('FAX (703)519-9349');
      GotoXY (23,20);
      Writeln ('Press ENTER to exit COBRA to MS-DOS');
      Repeat
        Repeat Until KeyPressed
      Until ReadKey = Chr(13);
      TextBackGround (0);
      GotoXY (1,25);
      ClrEOL;
      GotoXY (1,24);
      ErrorAddr := Nil;
      ExitCode := 0;
      ExitProc := SavedExitProc;
    End;
End;
```

```
{*****************************************************************************
 *   Name           : StripFile
 *
 *   Purpose        : Removes path and extension from filename
 *                    ex. StripFile ('C:\DOS\MODE.EXE') = 'MODE'
 *
 *   Invoked As     : StripFile (FileName)
 *
 *   Return Value   : String
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ------------------------------------------------------
 *   07Apr92   AML   Creation
 *****************************************************************************

Function StripFile (FileName: String) : String;

  Var
    Temp : Integer;

  Begin
    Temp := Pos ('.', FileName);
    If Temp > 0
      Then Delete (FileName, Temp, Length (FileName) - Temp + 1);
    While (Pos ('\', FileName) > 0) Do
      Delete (FileName, 1, Pos ('\', FileName));
    StripFile := FileName;
  End;
```

```
{*************************************************************************
 *   Name           : StripSpace
 *
 *   Purpose        : Strips all spaces from end of string
 *
 *   Invoked As     : StripSpace(Str)
 *
 *   Notes          : As a function (rather than a procedure with a Var
 *                    parameter), StripSpace can handle different types of
 *                    strings (String20, String60, etc.)
 *
 *   Constraints
 *     Development: Turbo Pascal 6.0 w/TurboVision. MS-DOS 5.0+
 *     Production : MS-DOS 5.0
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   31JAN91  GJG   Creation
 *   09Apr92  AML   Rewrote smaller and faster, made into function
 *************************************************************************}

Function StripSpace (Str : String): String;

  Begin
    While (Str [Length (Str)] = ' ') Do
      Str [0] := Chr (Ord (Str [0]) - 1);
    While (Length(Str) > 0) And (Str[1] = ' ') Do
      Delete (Str,1,1);
    StripSpace := Str;
  End;

{*************************************************************************
 *   Name           : TimeStr
 *
 *   Purpose        : Puts provided Date and Time values into a string
 *
 *   Invoked As     : TimeStr (Year, Month, Day, Hour, Minute)
 *
 *   Note           : The multiplications by one are required to convert
 *                    WORD values into INTEGER values.
 *
 *   Return Value   : String, containing date and time
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   27Mar92  AML   Creation
 *************************************************************************}
```

```
Function  TimeStr (Year, Month, Day, Hour, Minute : Integer)  : String20;

  Var
    TempValue, Result : String;

  Begin
    Str (Hour, Result);
    If Hour < 10
      Then Result := Concat ('0', Result, ':')
      Else Result := Concat (Result, ':');
    If Minute < 10
      Then Result := Concat (Result, '0');
    Str (Minute, TempValue);
    Result := Concat (Result, TempValue, Blank);
    If Month < 10
      Then Result := Concat (Result, '0');
    Str (Month, TempValue);
    Result := Concat (Result, TempValue, '/');
    If Day < 10
      Then Result := Concat (Result, '0');
    Str (Day, TempValue);
    Result := Concat (Result, TempValue, '/');
    Str (Year, TempValue);
    TimeStr := Concat (Result, TempValue)
  End;
```

```
{*************************************************************************
 *   Name           : Title
 *
 *   Purpose        : Used for titling report pages
 *
 *   Notes          : Outputs a formfeed IF PageNumber is not ONE,
 *                  : Prints six blank lines (regardless of PageNumber),
 *                  : Prints title line (with ReportName, Cobra Version,
 *                    and Abs(PageNumber) IF Abs(PageNumber) is NOT ONE),
 *                  : Prints date line (current date and report date),
 *                  : Prints a blank line.
 *                  : If PageNumber is Negative, prints Group, Service,
 *                    Option Package Name, and another blank line.
 *
 *                  : A PageNumber of ZERO should be used if a formfeed is
 *                    desired, but a page number on the title line isn't.
 *
 *   Invoked As     : Title (Fil, ReportName, PageNumber)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   27Mar92   AML   Creation
 *************************************************************************}

Procedure Title (Var Fil : Text; ReportName : String; PageNumber : Integer);

  Var
    Count      : Integer;
    TempPage,
    TempResult : String;

  Begin
    If Abs(PageNumber) <> 1
      Then Write (Fil, Chr(12)); {Formfeed}
    For Count := 1 To 6 Do
      Writeln (Fil);
{$IFDEF PROTO}
    TempResult := Concat (ReportName, ' (COBRA v', Version, '*P)');
{$ELSE}
    TempResult := Concat (ReportName, ' (COBRA v', Version, ')');
{$ENDIF}
    If Abs(PageNumber) > 1 Then
      Begin
        Str (Abs(PageNumber), TempPage);
        TempResult := Concat (TempResult, ' - Page ', TempPage);
      End;
    Center (Fil, TempResult);
    Center (Fil, ConCat ('Data As Of ', Screen1Data^.DateLine,
```

```
                        ', Report Created ', AsOfDateTime));
    Writeln (Fil);
    If PageNumber < 0 Then
      Begin
        writeln (Fil, Blank:10, 'Group          : ', Screen1Data^.Group);
        writeln (Fil, Blank:10, 'Service        : ', Screen1Data^.Service);
        writeln (Fil, Blank:10, 'Option Package : ',
          Screen1Data^.OptionPkgName);
        writeln (Fil);
      End;
  End;

{*****************************************************************************
 *   Name          : UpperCase
 *
 *   Purpose       : Convert a string of text to UPPERCASE.
 *
 *   Invoked As    : UpperCase (Stuff)
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ---------------------------------------------------------
 *   02Jun92  AML   Creation
 *****************************************************************************}

Function UpperCase (Stuff : String): String;

  Var
    I : Integer;

  Begin
    For I := 1 To Length (Stuff) Do
      If (Stuff[I] >= 'a') And (Stuff[I] <= 'z')
        Then Stuff[I] := Chr (Ord (Stuff[I]) - 32);
    UpperCase := Stuff;
  End;
```

```
{*****************************************************************************
 *   Name           : WithCommas
 *
 *   Purpose        : Add commas to numbers for increased legibility.
 *
 *   Notes          : Allows user to select number of decimal places used.
 *                  : Special functions (integers only) use negative values
 *                    for #ofDecimalPlaces...
 *                    -1 causes plus signs to be placed before positive numbers
 *                    -2 causes parentheses to be placed around nonzero numbers
 *
 *   Invoked As     : WithCommas (Number, #ofDecimalPlaces)
 *
 *   Return value   : String
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date     Init  Modifications
 *   -------  ----  ---------------------------------------------------------
 *   25Mar92  AML   Creation
 *   26May92  AML   Added Special Functions
 *****************************************************************************}

Function WithCommas (Value : Real; DecPlaces : Integer) : String;

  Var
    Count,
    SpecFunc,
    TempValue   : Integer;
    Negative    : Boolean;
    Result,
    TempResult : String;

  Begin
    Negative := (Value < 0);
    Value := Abs (Value);
    If Value > 999999999
      Then Value := 999999999; {Numeric Overflow}
    Result := '';
    If DecPlaces < 0 Then
      Begin
        SpecFunc := DecPlaces;
        DecPlaces := 0;
      End
    Else SpecFunc := 0;
    Value := Value + (4 / YtoX (10, DecPlaces + 1));
    For Count := 2 DownTo 0 Do
      Begin
        TempValue := Trunc (Value / YtoX (10, Count * 3));
        Str (TempValue, TempResult);
```

```
      If ((TempValue < 100) And (Length (Result) > 0))
        Then TempResult := Concat ('0', TempResult);
      If ((TempValue < 10) And (Length (Result) > 0))
        Then TempResult := Concat ('0', TempResult);
      If TempValue = 0
        Then TempResult := '000';
      If Not ((TempResult = '000') And (Result = ''))
        Then Result := Concat (Result, TempResult, ',');
      Value := Value - (TempValue * YtoX (10, Count * 3))
    End;
  If Length (Result) > 0
    Then Delete (Result, Length (Result), 1);
  If Result = ''
    Then Result := '0';
  If DecPlaces > 0 Then
    Begin
      Str (Value:(DecPlaces + 1):(DecPlaces), TempResult);
      Delete (TempResult, 1, 1);
      Result := Concat (Result, TempResult)
    End;
  If Negative
    Then Result := Concat ('-', Result)
  Else If (SpecFunc = -1) And (Result <> '0')
    Then Result := ConCat ('+', Result);
  If (SpecFunc = -2)
    Then If (Result <> '0')
      Then Result := ConCat ('(', Result, ')')
      Else Result := ConCat (Result, Blank);
  WithCommas := Result;
End;
```

```
{*****************************************************************************
 *   Name           : Within
 *
 *   Purpose        : Makes sure one value (Num2) is within two other values
 *                    (Num1 and Num3)
 *
 *   Notes          : There used to be two functions, MaxReal and MinReal, which
 *                    returned the greater or lesser of two numbers.  Just about
 *                    all occurences of MaxReal and/or MinReal could be replaced
 *                    with Within.
 *
 *   Invoked As     : Within (Num1, Num2, Num3)
 *
 *   Return value   : Real
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   29May92   AML   Creation
 *****************************************************************************}

Function Within (Num1, Num2, Num3 : Real) : Real;

  Begin { Within }
    If Num2 < Num1
      Then Num2 := Num1
    Else If Num2 > Num3
      Then Num2 := Num3;
    Within := Num2
  End;  { Within }

{*****************************************************************************
 *   Name           : YearOut
 *
 *   Purpose        : Outputs a report line (label, values for the year),
 *                    doing all adjustments for ranges, inflation, etc.
 *
 *   Notes          : Has the option of printing a zero for Beyond years
 *
 *   Invoked As     : YearOut (Fil, OutLabel, Value, Adjst, Size, DoBeyond);
 *
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 5.0.
 *     Production  : MS-DOS 5.0.
 *
 *   Date      Init  Modifications
 *   -------   ----  ---------------------------------------------------------
 *   02Jun92   AML   Creation
 *****************************************************************************}
```

```
Procedure YearOut (Var Fil      : Text;
                       OutLabel : String20;
                       Value,
                       Adjst    : Year1ToMaxRealRec;
                       Size,
                       DoBeyond : Integer);

  Var
    Y : Integer;

  Begin
    Write (Fil, Blank:10, OutLabel);
    For Y := 1 to MaxYears DO
      Write (Fil, WithCommas (Within ( -(YtoX(10,Size-2)-1),
        Value.Year[Y] * Adjst.Year[Y],  (YtoX(10,Size-2)-1) ),0):Size);
    If DoBeyond = 0
      Then writeln (Fil)
      Else Writeln (Fil, 0:Size);
  End;

{*****************************************************************************
 *   Name           : YtoX
 *   Purpose        : Calculates Y to the X (Y = Base, X = Exponent/Power)
 *   Notes          : Standard Pascal has no ability to raise a number to a
 *                    power.  An easy way to do this for INTEGER numbers only
 *                    would be to have a loop simply multiply a number by itself
 *                    a specified number of times.  But since this function
 *                    needs to handle Real numbers, it was necessary to employ
 *                    the following relationship:
 *
 *                                          X     X * Ln (Y)
 *                                         Y  = E
 *
 *                    It is necessary to make sure that Y is never less than or
 *                    equal to zero because Ln (Y) would then be unndefined and
 *                    thus cause the function to "blow up."  Note that this
 *                    function does include a work-around to handle negative
 *                    Y's (bases).
 *
 *   Invoked As     : YtoX (Base, Power)
 *   Return value   : Real
 *   Constraints
 *     Development  : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production   : MS-DOS 3.30.
 *   Date     Init  Modifications
 *   -------  ----  ---------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   25Mar92  AML   Streamlining
 *****************************************************************************}
```

```
Function YtoX (Base, Power : Real) : Real;

  Var
    NegativeBase : Boolean;
    Result       : Real;

  Begin { YtoX }
    Result := 0.0;
    NegativeBase := (Base < 0.0);
    Base := Abs (Base);
    If Base > 0.01 Then
      Result := Exp (Power * Ln (Base));
    IF NegativeBase Then
      Result := -Result;
    YtoX := Result;
  End;  { YtoX }

{*****************************************************************************
 *   Name           : Zerocheck
 *   Purpose        : Returns Num1 / Num2 IF Num2 <> 0; otherwise 0
 *   Invoked As     : Zerocheck (Num1, Num2)
 *   Return value   : Real
 *   Constraints
 *     Development : Turbo Pascal 6.0.  MS-DOS 3.30.
 *     Production  : MS-DOS 3.30.
 *   Date     Init  Modifications
 *   -------  ----  ------------------------------------------------------
 *   03Jan91  AJD   Creation
 *   25MAr92  AML   Streamlining
 *****************************************************************************}

Function Zerocheck (Num1, Num2 : Real) : Real;

  Begin { Zerocheck }
    If Abs (Num2) < 0.001
      Then Zerocheck := 0.0
      Else Zerocheck := Num1 / Num2
  End;  { Zerocheck }

End.  { UNIT }
```

```
UNIT FINPUT;

{$F+,O+,S-,D+,L+}

{*************************************************************************
 * FINPUT - Version 1.2
 *
 * This unit implements a derivative of TInputLine that supports several
 * data types dynamically.  It also provides formatted input for all the
 * numerical types, keystroke filtering and uppercase conversion, field
 * justification, and range checking.
 *
 * When the field is initialized, many filtering and uppercase converions
 * are implemented pertinent to the particular data type.
 *
 * The CheckRange and ErrorHandler methods should be overridden if the
 * user wants to implement then.
 *
 * Original Author : Allen Bauer (CIS ID: 76066,3202)
 *                   Ver 1.1 (c) 1990 Allen Bauer
 *
 * Fixes & Updates by Andrew Burton (CIS ID: 73227, 2606)
 *
 * Update 1.2 : Added MinValue, MaxValue fields, and automatic range-
 *    checking support.  If a field is called with Init() value for min
 *    >= Init() value for max, then range-checking is disabled on that
 *    field (works to limits of the type) -- if min < max, then the field
 *    will force values in that range.  Also added DResponse character set
 *    to allow 'Y' and 'N' entries.
 *
 * 02feb92  GJG : Modified HandleEvent to handle Up, Down, Enter key remapping.
 *************************************************************************}

INTERFACE

USES
  OBJECTS,
  DRIVERS,
  DIALOGS;

TYPE
  VKeys        = SET OF CHAR;

  PFInputLine = ^TFInputLine;
  TFInputLine = OBJECT(TInputLine)
                    ValidKeys             : VKeys;
                    DataType, Decimals    : BYTE;
                    imMode                : WORD;
                    Validated, ValidSent  : BOOLEAN;
                    MinValue, MaxValue    : REAL;
```

```
      CONSTRUCTOR Init (VAR Bounds  : TRect;
                            AMaxLen : INTEGER;
                            ChrSet  : VKeys;
                            DType,
                            Dec     : BYTE;
                            min,
                            max     : REAL);
      CONSTRUCTOR Load (VAR S       : TStream);

      FUNCTION    CheckRange : BOOLEAN;                    VIRTUAL;
      FUNCTION    DataSize   : WORD;                       VIRTUAL;

      PROCEDURE   Draw;                                    VIRTUAL;
      PROCEDURE   ErrorHandler;                            VIRTUAL;
      PROCEDURE   GetData       (VAR Rec);                 VIRTUAL;
      PROCEDURE   HandleEvent   (VAR Event : TEvent);      VIRTUAL;
      PROCEDURE   SetData       (VAR Rec);                 VIRTUAL;
      PROCEDURE   Store         (VAR S     : TStream);
    END;

CONST
  imLeftJustify  = $0001;
  imRightJustify = $0002;
  imConvertUpper = $0004;

  DString        = 0;
  DChar          = 1;
  DReal          = 2;
  DByte          = 3;
  DShortInt      = 4;
  DInteger       = 5;
  DLongInt       = 6;
  DWord          = 7;
  DDate          = 8;
  DTime          = 9;

  DRealSet       : VKeys = [#1..#31, '+', '-', '0'..'9', '.', 'E', 'e'];
  DPosRealSet    : VKeys = [#1..#31, '0'..'9', '.'];
  DSignedSet     : VKeys = [#1..#31, '+', '-', '0'..'9'];
  DUnSignedSet   : VKeys = [#1..#31, '0'..'9'];
  DCharSet       : VKeys = [#1..#31, ' '..'~'];
  DUpperSet      : VKeys = [#1..#31, ' '..'`', '{'..'~'];
  DAlphaSet      : VKeys = [#1..#31, 'A'..'Z', 'a'..'z'];
  DFileNameSet   : VKeys = [#1..#31, '!', '#'..')', '-'..'.', '0'..'9',
                                     '@'..'Z', '^'..'{', '}'..'~'];
  DPathSet       : VKeys = [#1..#31, '!', '#'..')', '-'..'.', '0'..':',
                                     '@'..'Z', '^'..'{', '}'..'~', '\'];
  DFileMaskSet   : VKeys = [#1..#31, '!', '#'..'*', '-'..'.', '0'..':',
                                     '?'..'Z', '^'..'{', '}'..'~', '\'];
  DDateSet       : VKeys = [#1..#31, '0'..'9', '/'];
  DTimeSet       : VKeys = [#1..#31, '0'..'9', ':'];
  DResponseSet   : VKeys = [#1..#31, 'Y', 'y', 'N', 'n'];
```

```
  DStrategySet   : VKeys = [#1..#31, '0'..'6'];

  cmValidateYourself = 2000;

PROCEDURE RegisterFInputLine;

  CONST
      RFInputLine : TStreamRec = (
                                   ObjType : 20000;
                                   VmtLink : Ofs(typeof(TFInputLine)^);
                                   Load    : @TFInputLine.Load;
                                   Store   : @TFinputLine.Store
                                  );

IMPLEMENTATION

USES
   VIEWS,
   MSGBOX,
   STRFMT,
   DOS;

FUNCTION CurrentDate : STRING;

   VAR
      Year,
      Month,
      Day,
      DOW     : WORD;
      DateStr : STRING[10];

   BEGIN
      GetDate (Year,Month,Day,DOW);
      DateStr := SFLongint(Month,2) + '/' + SFLongInt(Day,2) + '/'
               + SFLongInt(Year mod 100,2);
      FOR DOW := 1 TO length(DateStr) DO
         IF DateStr[DOW] = ' ' THEN
            DateStr[DOW] := '0';
      CurrentDate := DateStr;
   END;

FUNCTION CurrentTime : STRING;

   VAR
      Hour,
      Minute,
      Second,
      Sec100  : WORD;
      TimeStr : STRING[10];
```

```
   BEGIN
      GetTime(Hour,Minute,Second,Sec100);
      TimeStr := SFLongInt(Hour,2) + ':' + SFLongInt(Minute,2) + ':'
               + SFLongInt(Second,2);
      FOR Sec100 := 1 TO length(TimeStr) DO
         IF TimeStr[Sec100] = ' ' THEN
            TimeStr[Sec100] := '0';
      CurrentTime := TimeStr;
   END;

PROCEDURE RegisterFInputLine;

   BEGIN
      RegisterType(RFInputLine);
   END;

CONSTRUCTOR TFInputLine.Init (VAR Bounds  : TRect;
                                  AMaxLen : INTEGER;
                                  ChrSet  : VKeys;
                                  DType,
                                  Dec     : BYTE;
                                  min,
                                  max     : REAL);

   BEGIN
      IF (DType IN [DDate,DTime]) AND (AMaxLen < 8) THEN
         AMaxLen := 8;

      TInputLine.Init(Bounds,AMaxLen);

      ValidKeys := ChrSet;
      DataType  := DType;
      Decimals  := Dec;
      Validated := TRUE;
      ValidSent := FALSE;

      CASE DataType OF
         DReal      : BEGIN
                         imMode := imRightJustify;
                         IF min >= max THEN
                            BEGIN
                               MinValue := -1.7E38;
                               MaxValue :=  1.7E38;
                            END
                         ELSE
                            BEGIN
                               MinValue := min;
                               MaxValue := max;
                            END;
                      END;
```

```
DByte     : BEGIN
               imMode := imRightJustify;
               IF min >= max THEN
                  BEGIN
                     MinValue :=   0.0;
                     MaxValue := 255.0;
                  END
               ELSE
                  BEGIN
                     MinValue := min;
                     MaxValue := max;
                  END;
            END;
DInteger  : BEGIN
               imMode := imRightJustify;
               IF min >= max THEN
                  BEGIN
                     MinValue := -32768.0;
                     MaxValue :=  32767.0;
                  END
               ELSE
                  BEGIN
                     MinValue := min;
                     MaxValue := max;
                  END;
            END;
DLongInt  : BEGIN
               imMode := imRightJustify;
               IF min >= max THEN
                  BEGIN
                     MinValue := -2147483648.0;
                     MaxValue :=  2147483647.0;
                  END
               ELSE
                  BEGIN
                     MinValue := min;
                     MaxValue := max;
                  END;
            END;
DShortInt : BEGIN
               imMode := imRightJustify;
               IF min >= max THEN
                  BEGIN
                     MinValue := -128.0;
                     MaxValue :=  127.0;
                  END
               ELSE
                  BEGIN
                     MinValue := min;
                     MaxValue := max;
                  END;
            END;
```

```
        DWord      : BEGIN
                        imMode := imRightJustify;
                        IF min >= max THEN
                           BEGIN
                              MinValue :=     0.0;
                              MaxValue := 65535.0;
                           END
                        ELSE
                           BEGIN
                              MinValue := min;
                              MaxValue := max;
                           END;
                     END;
        DChar,
        DString,
        DDate,
        DTime      : BEGIN
                        imMode   := imLeftJustify;
                        MinValue :=  0.0; { MinValue & MaxValue will not }
                        MaxValue :=  0.0; { be used with str-based types }
                     END;

      END;
      IF ValidKeys = DUpperSet THEN
         imMode := imMode OR imConvertUpper;
      EventMask := EventMask OR evMessage;
   END;

CONSTRUCTOR TFInputLine.Load (VAR S : TStream);

   BEGIN
      TInputLine.Load(S);
      S.Read (ValidKeys, sizeof(VKeys));
      S.Read (DataType,  sizeof(BYTE));
      S.Read (Decimals,  sizeof(BYTE));
      S.Read (imMode,    sizeof(WORD));
      S.Read (Validated, sizeof(BOOLEAN));
      S.Read (ValidSent, sizeof(BOOLEAN));
      S.Read (MinValue,  sizeof(REAL));
      S.Read (MaxValue,  sizeof(REAL));
   END;

PROCEDURE TFInputLine.Store(VAR S: TStream);

BEGIN
  TInputLine.Store(S);
  S.Write(ValidKeys, sizeof(VKeys));
  S.Write(DataType,  sizeof(BYTE));
  S.Write(Decimals,  sizeof(BYTE));
  S.Write(imMode,    sizeof(WORD));
  S.Write(Validated, sizeof(BOOLEAN));
  S.Write(ValidSent, sizeof(BOOLEAN));
```

```
  S.Write(MinValue,  sizeof(REAL));
  S.Write(MaxValue,  sizeof(REAL));
END;

PROCEDURE TFInputLine.HandleEvent (VAR Event : TEvent);

   VAR
      NewEvent : TEvent;

   BEGIN
      CASE Event.What OF
         evKeyDown    : BEGIN
                           { Reassign keys Up, Down, Enter }
                           CASE Event.KeyCode OF
                              kbUp    : Event.KeyCode := kbShiftTab;
                              kbDown  : Event.KeyCode := kbTab;
                              kbEnter : Event.KeyCode := kbTab;
                           END; {end case Event.KeyCode}
                           IF (imMode AND imConvertUpper) <> 0 THEN
                              Event.CharCode := upcase(Event.CharCode);
                           IF NOT(Event.CharCode IN [#0..#31]) THEN
                              BEGIN
                                 Validated := FALSE;
                                 ValidSent := FALSE;
                              END;
                           IF (Event.CharCode <> #0) AND NOT
                              (Event.CharCode IN ValidKeys)
                             THEN ClearEvent(Event);
                        END; {end evKeyDown case begin}
         evBroadcast : BEGIN
                           IF (Event.Command = cmReceivedFocus)    AND
                              (Event.InfoPtr <> @Self)             AND
                              ((Owner^.State AND sfSelected) <> 0) AND
                              NOT (Validated)                      AND
                              NOT (ValidSent) THEN
                              BEGIN
                                 NewEvent.What    := evBroadcast;
                                 NewEvent.InfoPtr := @Self;
                                 NewEvent.Command := cmValidateYourself;
                                 PutEvent (NewEvent);
                                 ValidSent        := TRUE;
                              END;
                           IF (Event.Command = cmValidateYourself) AND
                              (Event.InfoPtr = @Self) THEN
                              BEGIN
                                 IF NOT CheckRange THEN
                                    BEGIN
                                       ErrorHandler;
                                       Select;
                                    END
```

```
                                    ELSE
                                       Validated := TRUE;
                                    ValidSent := FALSE;
                                    ClearEvent (Event);
                                 END;
                           END; {end case Event.What}
      END; {end BEGIN}
      TInputLine.HandleEvent (Event);
   END; {TFInputLine.HandleEvent}

PROCEDURE TFInputLine.GetData (VAR Rec);

   VAR
      Code : INTEGER;

   BEGIN
      CASE DataType OF
         Dstring,
         DDate,
         DTime      : TInputLine.GetData(Rec);
         DChar      : char(Rec) := Data^[1];
         DReal      : val(Data^, REAL(Rec)      , Code);
         DByte      : val(Data^, BYTE(Rec)      , Code);
         DShortInt  : val(Data^, SHORTINT(Rec)  , Code);
         DInteger   : val(Data^, INTEGER(Rec)   , Code);
         DLongInt   : val(Data^, LONGINT(Rec)   , Code);
         DWord      : val(Data^, WORD(Rec)      , Code);
      END;
   END;

PROCEDURE TFInputLine.SetData (VAR Rec);

   BEGIN
      CASE DataType OF
         DString,
         DDate,
         DTime      : TInputLine.SetData(Rec);
         DChar      : Data^ := CHAR(Rec);
         DReal      : Data^ := SFDReal(REAL(Rec),MaxLen,Decimals);
         DByte      : Data^ := SFLongInt(BYTE(Rec),MaxLen);
         DShortInt  : Data^ := SFLongInt(SHORTINT(Rec),MaxLen);
         DInteger   : Data^ := SFLongInt(INTEGER(Rec),MaxLen);
         DLongInt   : Data^ := SFLongInt(LONGINT(Rec),MaxLen);
         DWord      : Data^ := SFLongInt(WORD(Rec),MaxLen);
      END;
      SelectAll (TRUE);
   END;

FUNCTION TFInputLine.DataSize: WORD;
```

```
   BEGIN
      CASE DataType OF
         DString,
         DDate,
         DTime     : DataSize := TInputLine.DataSize;
         DChar     : DataSize := sizeof(CHAR);
         DByte     : DataSize := sizeof(BYTE);
         DShortInt : DataSize := sizeof(SHORTINT);
         DInteger  : DataSize := sizeof(INTEGER);
         DLongInt  : DataSize := sizeof(LONGINT);
         DWord     : DataSize := sizeof(WORD);
         DReal     : DataSize := sizeof(REAL);
      END;
   END;

PROCEDURE TFInputLine.Draw;

   VAR
      RD   : REAL;
      Code : INTEGER;

   BEGIN
      IF NOT ((State AND sfFocused) <> 0) THEN
         CASE DataType OF
            DReal       : BEGIN
                             IF Data^ = '' THEN
                                Data^ := SFDReal(0.0,MaxLen,Decimals)
                             ELSE
                                BEGIN
                                   val(Data^, RD, Code);
                                   Data^ := SFDReal(RD,MaxLen,Decimals);
                                END;
                          END;

            DByte,
            DShortInt,
            DInteger,
            DLongInt,
            DWord       : IF Data^ = '' THEN
                             Data^ := SFLongInt(0,MaxLen);
            DDate       : IF Data^ = '' THEN
                             Data^ := CurrentDate;
            DTime       : IF Data^ = '' THEN
                             Data^ := CurrentTime;
         END;

      IF State AND (sfFocused+sfSelected) <> 0 THEN
         BEGIN
            IF (imMode AND imRightJustify) <> 0 THEN
               WHILE (length(Data^) > 0) AND (Data^[1] = ' ') DO
                  delete(Data^,1,1);
         END
```

```
      ELSE
         BEGIN
            IF ((imMode AND imRightJustify) <> 0) AND (Data^ <> '') THEN
               WHILE (length(Data^) < MaxLen) DO
                  insert(' ',Data^,1);
            IF (imMode AND imLeftJustify) <> 0 THEN
               WHILE (length(Data^) > 0) AND (data^[1] = ' ') DO
                  delete(Data^,1,1);

         END;
      TInputLine.Draw;
   END;

FUNCTION TFInputLine.CheckRange : BOOLEAN;

   VAR
      MH,
      DM,
      YS      : LONGINT;
      RealMH  : REAL;
      Code    : INTEGER;
      MHs,
      DMs,
      YSs     : STRING[2];
      Delim   : CHAR;
      Ok      : BOOLEAN;

   BEGIN
      Ok := TRUE;
      CASE datatype OF
         DDate,
         DTime       : BEGIN
                          IF DataType = DDate THEN
                             Delim := '/'
                          ELSE Delim := ':';
                          IF pos(Delim,Data^) > 0 THEN
                             BEGIN
                                MHs := copy(Data^,1,pos(Delim,Data^));
                                DMs := copy(Data^,pos(Delim,Data^)+1,2);
                                delete(Data^,pos(Delim,Data^),1);
                                YSs := copy(Data^,pos(Delim,Data^)+1,2);
                                IF length(MHs) < 2 THEN
                                   MHs := '0' + MHs;
                                IF length(DMs) < 2 THEN
                                   DMs := '0' + DMs;
                                IF length(YSs) < 2 THEN
                                   YSs := '0' + YSs;
                                Data^ := MHs + DMs + YSs;
                             END;
```

```
                    IF (length(Data^) >= 6) AND (pos(Delim,Data^) = 0) THEN
                       BEGIN
                          val(copy(Data^,1,2), MH, Code);
                          IF Code <> 0 THEN
                             MH := 0;
                          val(copy(Data^,3,2), DM, Code);
                          IF Code <> 0 THEN
                             DM := 0;
                          val(copy(Data^,5,2), YS, Code);
                          IF Code <> 0 THEN
                             YS := 0;
                          IF DataType = DDate THEN
                             BEGIN
                                IF (MH > 12) OR (MH < 1) OR
                                   (DM > 31) OR (DM < 1)
                                 THEN Ok := FALSE;
                             END
                          ELSE
                             BEGIN
                                IF (MH > 23) OR (MH < 0) OR
                                   (DM > 59) OR (DM < 0) OR
                                   (YS > 59) OR (YS < 0)
                                 THEN Ok := FALSE;
                             END;
                          insert(Delim,Data^,5);
                          insert(Delim,Data^,3);
                       END
                    ELSE
                       Ok := FALSE;
                 END;

      DByte,
      DShortint,
      DInteger,
      DWord      : BEGIN
                      val(Data^, MH, Code);
                      IF (Code <> 0) OR (MH < MinValue) OR (MH > MaxValue)
                        THEN Ok := FALSE
                 END;
      DReal      : BEGIN
                      val(Data^, RealMH, Code);
                      IF (Code <> 0) OR (RealMH < MinValue) OR
                         (RealMH > MaxValue)
                        THEN Ok := FALSE
                 END;
   END;
   CheckRange := Ok;
END;
```

```
PROCEDURE TFInputLine.ErrorHandler;

   VAR
      MsgString : STRING[80];
      str1,
      str2      : STRING[20];
      Code      : INTEGER;
      Event     : TEvent;
      TempWord  : Word;

   BEGIN
      CASE DataType OF
         DDate       : MsgString :=
           ' Invalid Date Format!  Enter Date as MM/DD/YY ';
         DTime       : MsgString :=
           ' Invalid Time Format!  Enter Time as HH:MM:SS ';
         DByte,
         DShortInt,
         DInteger,
         DWord,
         DReal       : BEGIN
                          Str(MinValue:0:Decimals, str1);
                          Str(MaxValue:0:Decimals, str2);
                          MsgString := ' Number must be between '
                            + str1 + ' and ' + str2 + ' ';
                       END;
      END;
      TempWord := MessageBox(MsgString, NIL, mfError + mfOkButton);
   END;

END.
```

```
UNIT STRFMT;

{$F+,O-,S-,D+,L+}

INTERFACE

FUNCTION SLongInt  (L: longint)                                  : STRING;
FUNCTION SFLongInt (L: longint; Siz: INTEGER)                    : STRING;
FUNCTION SReal     (R: REAL)                                     : STRING;
FUNCTION SFReal    (R: REAL;    Siz: INTEGER)                    : STRING;
FUNCTION SFDReal   (R: REAL;    Siz: INTEGER; Dec: INTEGER)      : STRING;

IMPLEMENTATION

FUNCTION SLongInt;

  VAR
    RStr : ^STRING;

  BEGIN
    asm
    mov  sp,bp
    push ss
    push word ptr [bp+0ah]
  END;
  str(L, RStr^);
END;

FUNCTION SFLongInt;

  VAR
    RStr : ^STRING;

  BEGIN
    asm
    mov  sp,bp
    push ss
    push word ptr [bp+0ch]
  END;
  str(L:Siz, RStr^);
END;

FUNCTION SReal;

  VAR
    RStr : ^STRING;

  BEGIN
    asm
    mov  sp,bp
    push ss
    push word ptr [bp+0ch]
```

```
  END;
  str(R, RStr^);
END;

FUNCTION SFReal;

  VAR
    RStr : ^STRING;

  BEGIN
    asm
    mov  sp,bp
    push ss
    push word ptr [bp+0eh]
  END;
  str(R:Siz, RStr^);
END;

FUNCTION SFDReal;

  VAR
    RStr : ^STRING;

  BEGIN
    asm
    mov  sp,bp
    push ss
    push word ptr [bp+10h]
  END;
  str(R:Siz:Dec, RStr^);
END;

END.
```