

US EPA ARCHIVE DOCUMENT

CARES

Cumulative and Aggregate Risk Evaluation System

Code Manual



March 20, 2002
CARES 1.0

CropLife America
1156 Fifteenth Street, N.W., Suite 400
Washington, DC 20005
Phone: 202-296-1585
Web Site: www.CropLifeAmerica.org

Copyright Notice

Copyright © 2002 CropLife America

Authors

Contributing authors for code are:

Muhilan Pandian, Harry Reed, and David Gonzales
infoscientific.com
2275 Corporate Circle, Suite 220
Henderson, NV 89074
Email: muhilan@infoscientific.com
Web Site: www.infoscientific.com

Design

Design and layout by:

David S. Farrier, Ph.D.
Summit Research Services
68911 Open Field Dr.
Montrose, CO 81401
Tel: 970-249-1389
Email: DFarrier@SummitPK.com
Web Site: www.SummitPK.com

Trademark Acknowledgements

Microsoft and Windows are registered trademarks of Microsoft Corporation. Notitia is a registered trademark of infoscientific.com.

Name Change

CropLife America was formerly known as the American Crop Protection Association

Table of Contents



1.0 Overview	1
1.1 Purpose and Scope	1
1.2 Program Description.....	1
1.3 Programming Standards	1
2.0 Code 2	
2.1 Residential – Dermal – During Application – Unit Exposure, Area Treated	2
2.2 Residential – Dermal – During Application – Unit Exposure, Amount of Formulation Used	3
2.3 Residential – Dermal – Post Application – Transfer Coefficient, Residue.....	4
2.4 Residential – Dermal – Post Application – Transfer Coefficient, Area Treated.....	5
2.5 Residential – Dermal – Post Application – Transfer Factor, Residue.....	7
2.6 Residential – Dermal – Post Application – Transfer Factor, Area Treated	10
2.7 Residential – Dermal – Post Application – Fraction Transferred.....	13
2.8 Residential – Dermal – Post Application – Flux Rate	14
2.9 Residential – Dermal – Post Application – Water Concentration.....	15
2.10 Residential – Dermal – Post Application – Film Thickness.....	16
2.11 Residential – Ingestion – Post Application – Granules/Pellets (Formulation).....	17
2.12 Residential – Ingestion – Post Application – Grass/Plants.....	18
2.13 Residential – Ingestion – Post Application – Soil.....	19
2.14 Residential – Ingestion – Post Application – Paint Chips	20
2.15 Residential – Ingestion – Post Application – Water Concentration	21
2.16 Residential – Ingestion – Post Application – Flux Rate	22
2.17 Residential – Ingestion – Post Application – Hand-to-Mouth – Mass Balance.....	23
2.18 Residential – Ingestion – Post Application – Hand-to-Mouth – Fraction Transferred 25	
2.19 Residential – Ingestion – Post Application – Hand-to-Mouth – EPA SOP.....	26
2.20 Residential – Inhalation – During Application – Unit Exposure, Area Treated.....	27
2.21 Residential – Inhalation – During Application – Unit Exposure, Amount of Formulation Used	28
2.22 Residential – Inhalation – Post Application – Air Concentration, Specified	29
2.23 Residential – Inhalation – Post Application – Air Concentration, Calculated	30
2.24 Residential – Exposure Calculations	31
2.25 Residential – Event Allocation	50
2.25 Dietary – Exposure Calculations.....	76

1.0 Overview

1.1 Purpose and Scope

The purpose of this manual is to corroborate and illustrate the claim that CARES is available as an “open source” program. Since there are near-term expected enhancements to the program, it is premature to publish the full source description at this time. Therefore, this manual is intended to illustrate the content and organization of the final source code manual.

Ultimately, the main use of the code manual is two-fold. First, it will allow for open inspection of core algorithms underlying internal calculations for quality assurance inspection. Second, it will enable programmers to understand the basic structure in order to augment or enhance certain features and functions or even add and integrate new modular components.

1.2 Program Description

CARES is an exposure calculation and risk analysis program covering both aggregate (multiple source) and cumulative (multiple chemical) assessment of dietary, residential, and water exposure pathways. The program is written in Visual Basic 6.0 and is designed in a modular fashion using object oriented programming methodology. Furthermore, the program is designed to fit as a subset into the framework of Notitia™, a general data management, integration, analysis, and reporting framework.

1.3 Programming Standards

The programming standards for the CARES project are adopted in most part from Practical Standards for Visual Basic (Microsoft Press, 2000). Visual Basic is a Microsoft proprietary programming environment.

2.0 Code

2.1 Residential – Dermal – During Application – Unit Exposure, Area Treated

```
'
' Algorithm - Dermal-101 (F11101)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    sim = sim + 1

    Dim vntUnitExp As Variant
    Dim vntApplication As Variant
    Dim vntAreaTreated As Variant
    Dim vntRefDur As Variant
    Dim vntBodyWt As Variant

    Dim rst01 As ADODB.Recordset
    Dim rst02 As ADODB.Recordset

    Set rst01 = out1
    Set rst02 = out2

    ' do calcs
    vntUnitExp = oParam.oProd.GetDistributionValue("s6")
    vntApplication = oParam.oProd.GetDistributionValue("s3")

    vntAreaTreated = Settings.Setting("s3").Value
    vntRefDur = Settings.Setting("s5").Value
    vntBodyWt = oParam.rBodyWeight * Settings.Setting("s6").Value

    Dim vntExposure As Variant
    vntExposure = vntUnitExp * vntApplication * vntAreaTreated / vntRefDur
* vntBodyWt)

    ' write intermediate results
    rst01.AddNew
    rst01.Fields.Item("SIM").Value = sim
    rst01.Fields.Item("UnitExp").Value = vntUnitExp
    rst01.Fields.Item("Applicatio").Value = vntApplication
    rst01.Fields.Item("AreaTreate").Value = vntAreaTreated
    rst01.Fields.Item("RefDur").Value = vntRefDur
    rst01.Fields.Item("BodyWt").Value = vntBodyWt
    rst01.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    Algorithm = Exposure
End Function
```

2.2 Residential – Dermal – During Application – Unit Exposure, Amount of Formulation Used

```

'
' Algorithm - Dermal-102 (F11102)
'

Public Function Algorithm(oParam As clsParamBlock) As Double

    sim = sim + 1

    Dim rstO1 As ADODB.Recordset
    Dim rstO2 As ADODB.Recordset

    Set rstO1 = out1
    Set rstO2 = out2

    Dim vntUnitExp As Variant
    Dim vntApplication As Variant
    Dim vntAmtFrmUsed As Variant
    Dim vntRefDur As Variant
    Dim vntBodyWt As Variant

    ' do calcs
    vntUnitExp = oParam.oProd.GetDistributionValue("s6")
    vntApplication = oParam.oProd.GetDistributionValue("s4")
    vntAmtFrmUsed = oParam.oProd.GetDistributionValue("s2")

    vntRefDur = Settings.Setting("s5").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure As Variant
    vntExposure = vntUnitExp * vntApplication * vntAmtFrmUsed / (vntRefDur
* vntBodyWt)

    ' write intermediate results
    rstO1.AddNew
    rstO1.Fields.Item("SIM").Value = sim
    rstO1.Fields.Item("UnitExp").Value = vntUnitExp
    rstO1.Fields.Item("Applicatio").Value = vntApplication
    rstO1.Fields.Item("AmtFrmUsed").Value = vntAmtFrmUsed
    rstO1.Fields.Item("RefDur").Value = vntRefDur
    rstO1.Fields.Item("BodyWt").Value = vntBodyWt
    rstO1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = Exposure
End Function

```


2.3 Residential – Dermal – Post Application – Transfer Coefficient, Residue

```

'
' Algorithm - Dermal-103 (F11103)
'

Public Sub Algorithm2(oParam As clsParamBlock)

    Dim vntTransRes As Variant
    Dim vntTransCoeff As Variant
    Dim vntExpDur As Variant
    Dim vntFracTrans As Variant
    Dim vntBodyWt As Variant

    Dim rstO1 As ADODB.Recordset
    Dim rstO2 As ADODB.Recordset
    Dim rstO3 As ADODB.Recordset

    Set rstO1 = out1
    Set rstO2 = out2
    Set rstO3 = out3

    sim = sim + 1

    ' do calcs
    vntTransRes = oParam.oProd.GetDistributionValue("s8")

    vntTransCoeff = Settings.Setting("s2").Value()
    vntFracTrans = Settings.Setting("s4").Value()
    vntExpDur = Settings.Setting("s4").Value()

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure As Variant
    Dim vntHand As Variant

    vntExposure = vntTransRes * vntTransCoeff * vntExpDur / vntBodyWt
    vntHand = vntTransRes * vntTransCoeff * vntFracTrans * vntExpDur

    ' write intermediate results
    rstO1.AddNew
    rstO1.Fields.Item("SIM").Value = sim
    rstO1.Fields.Item("TransRes").Value = vntTransRes
    rstO1.Fields.Item("TransCoeff").Value = vntTransCoeff
    rstO1.Fields.Item("ExpDur").Value = vntExpDur
    rstO1.Fields.Item("FracTrans").Value = vntFracTrans
    rstO1.Fields.Item("BodyWt").Value = vntBodyWt

    rstO1.Update

    ' write final results
    rstO2.AddNew
    rstO2.Fields.Item("Sim") = sim
    rstO2.Fields.Item("Exposure") = vntExposure
    rstO2.Update

    ' write hand results
    rstO3.AddNew
    rstO3.Fields.Item("Sim") = sim
    rstO3.Fields.Item("Hand") = vntHand
    rstO3.Update

    DoEvents

    oParam.rExposure = vntExposure
    oParam.rHand = vntHand
End Sub

```

2.4 Residential – Dermal – Post Application – Transfer Coefficient, Area Treated

```

'
' Algorithm - Dermal-104 (F11104)
'

Public Sub Algorithm2(oParam As clsParamBlock)

    sim = sim + 1

    Dim vntApplication As Variant
    Dim vntFracAI As Variant
    Dim vntTransRes As Variant
    Dim vntTransCoeff As Variant
    Dim vntDuration As Variant
    Dim vntBodyWt As Variant
    Dim vntFracHand As Variant

    Dim rstO1 As ADODB.Recordset
    Dim rstO2 As ADODB.Recordset
    Dim rstO3 As ADODB.Recordset

    Set rstO1 = out1
    Set rstO2 = out2
    Set rstO3 = out3

    ' do calcs
    vntApplication = oParam.oProd.GetDistributionValue("s3")
    vntFracAI = oParam.oProd.GetDistributionValue("s5")

    vntTransCoeff = Settings.Setting("s2").Value()
    vntFracHand = Settings.Setting("s3").Value()
    vntDuration = Settings.Setting("s4").Value()

    vntBodyWt = oParam.rBodyWeight

    vntTransRes = vntApplication * vntFracAI * 100

    Dim vntExposure As Variant
    vntExposure = vntTransRes * vntTransCoeff * vntDuration / vntBodyWt

    Dim vntHand As Variant
    vntHand = vntTransRes * vntTransCoeff * vntDuration * vntFracHand

    ' write intermediate results
    rstO1.AddNew
    rstO1.Fields.Item("SIM").Value = sim
    rstO1.Fields.Item("Applicatio").Value = vntApplication
    rstO1.Fields.Item("FracAI").Value = vntFracAI

    rstO1.Fields.Item("TransRes").Value = vntTransRes
    rstO1.Fields.Item("TransCoeff").Value = vntTransCoeff
    rstO1.Fields.Item("Duration").Value = vntDuration
    rstO1.Fields.Item("BodyWt").Value = vntBodyWt
    'O1.Update

    ' write final results
    rstO2.AddNew
    rstO2.Fields.Item("Sim") = sim
    rstO2.Fields.Item("Exposure") = vntExposure
    rstO2.Fields.Item("Hand") = vntHand
    rstO2.Update

    ' write hand data
    rstO3.AddNew
    rstO3.Fields.Item("Sim") = sim

```

```
rst03.Fields.Item("Hand") = vntHand  
rst03.Update  
  
DoEvents  
  
oParam.rExposure = vntExposure  
oParam.rHand = vntHand  
End Sub
```

2.5 Residential – Dermal – Post Application – Transfer Factor, Residue

```

\
' Algorithm - Dermal-105 (F11105)
,

Public Sub Algorithm2(oParam As clsParamBlock)

    Dim vntTransFactorHU, vntTransFactorHC, vntTransFactorUbU,
vntTransFactorUbC
    Dim vntTransFactorLbU, vntTransFactorLbC, vntTransFactorFU,
vntTransFactorFC
    Dim vntSurfaceAreaHU, vntSurfaceAreaHC, vntSurfaceAreaUbU,
vntSurfaceAreaUbC
    Dim vntSurfaceAreaLbU, vntSurfaceAreaLbC, vntSurfaceAreaFU,
vntSurfaceAreaFC
    Dim vntClothFactorU, vntClothFactorC
    Dim vntTransRes, vntRefDur, vntBodyWt

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset
    Dim o3 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2
    Set o3 = out3

    Outputs.Enable True          ' enable outputs

    sim = sim + 1

    ' get distribution values

    vntTransFactorHU = Settings.Setting("s2").Value()
    vntTransFactorHC = Settings.Setting("s3").Value()
    vntTransFactorUbU = Settings.Setting("s4").Value()
    vntTransFactorUbC = Settings.Setting("s5").Value()
    vntTransFactorLbU = Settings.Setting("s6").Value()
    vntTransFactorLbC = Settings.Setting("s7").Value()
    vntTransFactorFU = Settings.Setting("s8").Value()
    vntTransFactorFC = Settings.Setting("s9").Value()

    Dim dblHandsU As Double, dblHandsC As Double
    Dim dblUpperU As Double, dblUpperC As Double
    Dim dblLowerU As Double, dblLowerC As Double
    Dim dblFeetU As Double, dblFeetC As Double

    Call Module1.GetAreas(oParam.oPop, dblHandsU, dblHandsC, dblUpperU,
dblUpperC, dblLowerU, dblLowerC, dblFeetU, dblFeetC)

    Dim dblSurfaceArea As Double
    dblSurfaceArea = Surface_A * (oParam.rBodyHeight ^ Surface_B) *
(oParam.rBodyWeight ^ Surface_C)

    vntSurfaceAreaHU = dblHandsU * dblSurfaceArea
    vntSurfaceAreaHC = dblHandsC * dblSurfaceArea
    vntSurfaceAreaUbU = dblUpperU * dblSurfaceArea
    vntSurfaceAreaUbC = dblUpperC * dblSurfaceArea
    vntSurfaceAreaLbU = dblLowerU * dblSurfaceArea
    vntSurfaceAreaLbC = dblLowerC * dblSurfaceArea
    vntSurfaceAreaFU = dblFeetU * dblSurfaceArea
    vntSurfaceAreaFC = dblFeetC * dblSurfaceArea

    vntClothFactorU = Settings.Setting("s18").Value()
    vntClothFactorC = Settings.Setting("s19").Value()

    vntTransRes = oParam.oProd.GetDistributionValue("s8")

```

```

vntRefDur = Settings.Setting("s22").Value()
vntBodyWt = oParam.rBodyWeight 'Settings.Setting("s23").Value()

' do calcs

Dim vntSum, vntSumU, vntSumC
vntSumU = 0
vntSumU = vntSumU + vntTransFactorHU * vntSurfaceAreaHU *
vntClothFactorU
vntSumU = vntSumU + vntTransFactorUbU * vntSurfaceAreaUbU *
vntClothFactorU
vntSumU = vntSumU + vntTransFactorLbU * vntSurfaceAreaLbU *
vntClothFactorU
vntSumU = vntSumU + vntTransFactorFU * vntSurfaceAreaFU *
vntClothFactorU

vntSumC = 0
vntSumC = vntSumC + vntTransFactorHC * vntSurfaceAreaHC *
vntClothFactorC
vntSumC = vntSumC + vntTransFactorUbC * vntSurfaceAreaUbC *
vntClothFactorC
vntSumC = vntSumC + vntTransFactorLbC * vntSurfaceAreaLbC *
vntClothFactorC
vntSumC = vntSumC + vntTransFactorFC * vntSurfaceAreaFC *
vntClothFactorC

vntSum = vntSumU + vntSumC

Dim vntExposure
vntExposure = vntSum * vntTransRes / (vntRefDur * vntBodyWt)

' calculate hand exposure
Dim vntHand, vntHand1, vntHand2
vntHand1 = vntTransFactorHC * vntSurfaceAreaHC * vntClothFactorC
vntHand2 = vntTransFactorHU * vntSurfaceAreaHU * vntClothFactorU
vntHand = (vntHand1 + vntHand2) * vntTransRes

' write intermediate results
o1.AddNew
o1.Fields.Item("SIM").Value = sim
o1.Fields.Item("TranFacHU").Value = vntTransFactorHU
o1.Fields.Item("TranFacHC").Value = vntTransFactorHC
o1.Fields.Item("TranFacUbU").Value = vntTransFactorUbU
o1.Fields.Item("TranFacUbC").Value = vntTransFactorUbC
o1.Fields.Item("TranFacLbU").Value = vntTransFactorLbU
o1.Fields.Item("TranFacLbC").Value = vntTransFactorLbC
o1.Fields.Item("TranFacFU").Value = vntTransFactorFU
o1.Fields.Item("TranFacFC").Value = vntTransFactorFC

o1.Fields.Item("SurAreaHU").Value = vntSurfaceAreaHU
o1.Fields.Item("SurAreaHC").Value = vntSurfaceAreaHC
o1.Fields.Item("SurAreaUbU").Value = vntSurfaceAreaUbU
o1.Fields.Item("SurAreaUbC").Value = vntSurfaceAreaUbC
o1.Fields.Item("SurAreaLbU").Value = vntSurfaceAreaLbU
o1.Fields.Item("SurAreaLbC").Value = vntSurfaceAreaLbC
o1.Fields.Item("SurAreaFU").Value = vntSurfaceAreaFU
o1.Fields.Item("SurAreaFC").Value = vntSurfaceAreaFC

o1.Fields.Item("ClothFacU").Value = vntClothFactorU
o1.Fields.Item("ClothFacC").Value = vntClothFactorC

o1.Fields.Item("TransRes").Value = vntTransRes
o1.Fields.Item("RefDur").Value = vntRefDur
o1.Fields.Item("BodyWt").Value = vntBodyWt

' write final results
o2.AddNew
o2.Fields.Item("Sim") = sim
o2.Fields.Item("Exposure") = vntExposure
o2.Fields.Item("Hand") = vntHand

```

```
' write hand data
O3.AddNew
O3.Fields.Item("Sim") = sim
O3.Fields.Item("Hand") = vntHand

DoEvents

oParam.rExposure = vntExposure
oParam.rHand = vntHand
End Sub
```

2.6 Residential – Dermal – Post Application – Transfer Factor, Area Treated

```

\
' Algorithm - Dermal-106 (F11106)
,
Public Sub Algorithm2(oParam As clsParamBlock)

    Dim vntTransFactorHU, vntTransFactorHC, vntTransFactorUbU,
vntTransFactorUbC
    Dim vntTransFactorLbU, vntTransFactorLbC, vntTransFactorFU,
vntTransFactorFC
    Dim vntSurfaceAreaHU, vntSurfaceAreaHC, vntSurfaceAreaUbU,
vntSurfaceAreaUbC
    Dim vntSurfaceAreaLbU, vntSurfaceAreaLbC, vntSurfaceAreaFU,
vntSurfaceAreaFC
    Dim vntClothFactorU, vntClothFactorC
    Dim vntTransRes, vntRefDur, vntBodyWt
    Dim vntApplication, vntFracDislodged

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset
    Dim o3 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2
    Set o3 = out3

    sim = sim + 1

    ' get distribution values
vntApplication = oParam.oProd.GetDistributionValue("s3")
vntFracDislodged = oParam.oProd.GetDistributionValue("s5")

vntTransFactorHU = Settings.Setting("s3").Value()
vntTransFactorHC = Settings.Setting("s4").Value()
vntTransFactorUbU = Settings.Setting("s5").Value()
vntTransFactorUbC = Settings.Setting("s6").Value()
vntTransFactorLbU = Settings.Setting("s7").Value()
vntTransFactorLbC = Settings.Setting("s8").Value()
vntTransFactorFU = Settings.Setting("s9").Value()
vntTransFactorFC = Settings.Setting("s10").Value()

    Dim dblHandsU As Double, dblHandsC As Double
    Dim dblUpperU As Double, dblUpperC As Double
    Dim dblLowerU As Double, dblLowerC As Double
    Dim dblFeetU As Double, dblFeetC As Double

    Call Module1.GetAreas(oParam.oPop, dblHandsU, dblHandsC, dblUpperU,
dblUpperC, dblLowerU, dblLowerC, dblFeetU, dblFeetC)

    Dim dblSurfaceArea As Double
    dblSurfaceArea = Surface_A * (oParam.rBodyHeight ^ Surface_B) *
(oParam.rBodyWeight ^ Surface_C)

vntSurfaceAreaHU = dblHandsU * dblSurfaceArea
vntSurfaceAreaHC = dblHandsC * dblSurfaceArea
vntSurfaceAreaUbU = dblUpperU * dblSurfaceArea
vntSurfaceAreaUbC = dblUpperC * dblSurfaceArea
vntSurfaceAreaLbU = dblLowerU * dblSurfaceArea
vntSurfaceAreaLbC = dblLowerC * dblSurfaceArea
vntSurfaceAreaFU = dblFeetU * dblSurfaceArea
SvntSurfaceAreaFC = dblFeetC * dblSurfaceArea

vntClothFactorU = Settings.Setting("s19").Value()
vntClothFactorC = Settings.Setting("s20").Value()

vntRefDur = Settings.Setting("s21").Value()

```

```

vntBodyWt = oParam.rBodyWeight

' do calcs
vntTransRes = Application * FracDislodged * 100#

Dim vntSum, vntSumU, vntSumC
vntSumU = 0
vntSumU = vntSumU + vntTransFactorHU * vntSurfaceAreaHU *
vntClothFactorU
vntSumU = vntSumU + vntTransFactorUbU * vntSurfaceAreaUbU *
vntClothFactorU
vntSumU = vntSumU + vntTransFactorLbU * vntSurfaceAreaLbU *
vntClothFactorU
vntSumU = vntSumU + vntTransFactorFU * vntSurfaceAreaFU *
vntClothFactorU

vntSumC = 0
vntSumC = vntSumC + vntTransFactorHC * vntSurfaceAreaHC *
vntClothFactorC
vntSumC = vntSumC + vntTransFactorUbC * vntSurfaceAreaUbC *
vntClothFactorC
vntSumC = vntSumC + vntTransFactorLbC * vntSurfaceAreaLbC *
vntClothFactorC
vntSumC = vntSumC + vntTransFactorFC * vntSurfaceAreaFC *
vntClothFactorC

vntSum = vntSumU + vntSumC

Dim vntExposure
vntExposure = vntSum * vntTransRes / (vntRefDur * vntBodyWt)

' calculate hand exposure
Dim vntHand, vntHand1, vntHand2
vntHand1 = vntTransFactorHC * vntSurfaceAreaHC * vntClothFactorC
vntHand2 = vntTransFactorHU * vntSurfaceAreaHU * vntClothFactorU
vntHand = (vntHand1 + vntHand2) * vntTransRes

' write intermediate results
o1.AddNew
o1.Fields.Item("SIM").Value = sim
o1.Fields.Item("Applicatio").Value = vntApplication
o1.Fields.Item("Dislodged").Value = vntFracDislodged
o1.Fields.Item("TransRes").Value = vntTransRes

o1.Fields.Item("TranFacHU").Value = vntTransFactorHU
o1.Fields.Item("TranFacHC").Value = vntTransFactorHC
o1.Fields.Item("TranFacUbU").Value = vntTransFactorUbU
o1.Fields.Item("TranFacUbC").Value = vntTransFactorUbC
o1.Fields.Item("TranFacLbU").Value = vntTransFactorLbU
o1.Fields.Item("TranFacLbC").Value = vntTransFactorLbC
o1.Fields.Item("TranFacFU").Value = vntTransFactorFU
o1.Fields.Item("TranFacFC").Value = vntTransFactorFC

o1.Fields.Item("SurAreaHU").Value = vntSurfaceAreaHU
o1.Fields.Item("SurAreaHC").Value = vntSurfaceAreaHC
o1.Fields.Item("SurAreaUbU").Value = vntSurfaceAreaUbU
o1.Fields.Item("SurAreaUbC").Value = vntSurfaceAreaUbC
o1.Fields.Item("SurAreaLbU").Value = vntSurfaceAreaLbU
o1.Fields.Item("SurAreaLbC").Value = vntSurfaceAreaLbC
o1.Fields.Item("SurAreaFU").Value = vntSurfaceAreaFU
o1.Fields.Item("SurAreaFC").Value = vntSurfaceAreaFC

o1.Fields.Item("ClothFacU").Value = vntClothFactorU
o1.Fields.Item("ClothFacC").Value = vntClothFactorC

o1.Fields.Item("RefDur").Value = vntRefDur
o1.Fields.Item("BodyWt").Value = vntBodyWt
o1.Update

' write final results

```



```
o2.AddNew
o2.Fields.Item("Sim") = sim
o2.Fields.Item("Exposure") = vntExposure
o2.Fields.Item("Hand") = vntHand
o2.Update

' write hand data
O3.AddNew
O3.Fields.Item("Sim") = sim
O3.Fields.Item("Hand") = vntHand
O3.Update

DoEvents

oParam.rExposure = vntExposure
oParam.rHand = vntHand
End Sub
```

2.7 Residential – Dermal – Post Application – Fraction Transferred

```

'
' Algorithm - Dermal-107 (F11107)
'
Public Sub Algorithm2(oParam As clsParamBlock)

    Dim vntAmtFrmUsed, vntFracAIForm, vntFracDislodged
    Dim vntFracTrans, vntRefDur, vntBodyWt, vntFracHand

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset
    Dim O3 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2
    Set O3 = out3

    sim = sim + 1

    ' do calcs
    vntAmtFrmUsed = oParam.oProd.GetDistributionValue("s9")
    vntFracAIForm = oParam.oProd.GetDistributionValue("s10")
    vntFracDislodged = oParam.oProd.GetDistributionValue("s5")

    vntFracTrans = Settings.Setting("s4").Value()
    vntFracHand = Settings.Setting("s5").Value()
    vntRefDur = Settings.Setting("s6").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntTransRes
    vntTransRes = vntAmtFrmUsed * vntFracAIForm * vntFracDislodged

    Dim vntExposure
    vntExposure = vntTransRes * vntFracTrans / (vntRefDur * vntBodyWt)

    Dim vntHand
    vntHand = vntTransRes * vntFracHand / vntRefDur

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("AmtFrmUsed").Value = vntAmtFrmUsed
    o1.Fields.Item("FracAIForm").Value = vntFracAIForm
    o1.Fields.Item("Dislodged").Value = vntFracDislodged
    o1.Fields.Item("FracTrans").Value = vntFracTrans
    o1.Fields.Item("RefDur").Value = vntRefDur
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Fields.Item("FracHand").Value = vntFracHand
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Fields.Item("Hand") = vntHand
    o2.Update

    ' write hand data
    O3.AddNew
    O3.Fields.Item("Sim") = sim
    O3.Fields.Item("Hand") = vntHand
    O3.Update

    DoEvents

    oParam.rExposure = vntExposure
    oParam.rHand = vntHand
End Sub

```

2.8 Residential – Dermal – Post Application – Flux Rate

```

'
' Algorithm - Dermal108 (F11108)
'
Public Sub Algorithm2(oParam As clsParamBlock)

    Dim vntFluxRateAI, vntSurfaceArea, vntExpDur
    Dim vntBodyWt, vntFracTrans

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset
    Dim o3 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2
    Set o3 = out3

    sim = sim + 1

    ' do calcs
    vntFluxRateAI = Settings.Setting("s1").Value()
    vntSurfaceArea = Settings.Setting("s2").Value()
    vntExpDur = Settings.Setting("s3").Value()
    vntFracTrans = Settings.Setting("s4").Value()

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntFluxRateAI * vntSurfaceArea * vntExpDur / (vntBodyWt
* 10000 * 24)

    Dim vntHand
    vntHand = vntFluxRateAI * vntSurfaceArea * vntFracTrans * vntExpDur /
(10000 * 24)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("FluxRateAI").Value = vntFluxRateAI
    o1.Fields.Item("ExpDur").Value = vntExpDur
    o1.Fields.Item("FracTrans").Value = vntFracTrans
    o1.Fields.Item("SurfArea").Value = vntSurfaceArea
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Fields.Item("Hand") = vntHand
    o2.Update

    ' write hand data
    o3.AddNew
    o3.Fields.Item("Sim") = sim
    o3.Fields.Item("Hand") = vntHand
    o3.Update

    DoEvents

    oParam.rExposure = vntExposure
    oParam.rHand = vntHand
End Sub

```

2.9 Residential – Dermal – Post Application – Water Concentration

```

\
'Algorithm - Dermal-109 (F11109)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    sim = sim + 1

    Dim vntConcAIWater, vntPermCoeff, vntSurfArea, vntExpDur, vntBodyWt

    ' do calcs
    vntConcAIWater = oParam.oProd.GetDistributionValue("s11")
    vntPermCoeff = oParam.oProd.GetDistributionValue("s12")

    vntExpDur = Settings.Setting("s3").Value()

    SurfArea = Surface_A * (oParam.rBodyHeight ^ Surface_B) *
(oParam.rBodyWeight ^ Surface_C)

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntConcAIWater * vntPermCoeff * vntSurfArea * vntExpDur
/ (vntBodyWt * 1000000)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("ConcAI").Value = vntConcAIWater
    o1.Fields.Item("PermCoeff").Value = vntPermCoeff
    o1.Fields.Item("ExpDur").Value = vntExpDur
    o1.Fields.Item("SurfArea").Value = vntSurfArea
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.10 Residential – Dermal – Post Application – Film Thickness

```

\
' Algorithm - Dermal-110 (F11110)
'
Public Sub Algorithm2(oParam As clsParamBlock)

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset
    Dim O3 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2
    Set O3 = out3

    sim = sim + 1

    Dim vntDensityForm, vntFracAIForm, vntFilmThick, vntSurfArea,
vntRefDur, vntBodyWt, vntFracHand

    ' do calcs
    vntDensityForm = oParam.oProd.GetDistributionValue("s13")
    vntFracAIForm = oParam.oProd.GetDistributionValue("s10")

    vntFilmThick = Settings.Setting("s3").Value()
    vntFracHand = Settings.Setting("s4").Value()
    vntRefDur = Settings.Setting("s5").Value()
    vntSurfArea = Settings.Setting("s6").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntDensityForm * vntFracAIForm * vntFilmThick *
vntSurfArea / (vntRefDur * vntBodyWt)

    Dim vntHand
    vntHand = vntDensityForm * vntFracAIForm * vntFilmThick * vntSurfArea
* vntFracHand / vntRefDur

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("DensForm").Value = vntDensityForm
    o1.Fields.Item("FracAIForm").Value = vntFracAIForm
    o1.Fields.Item("FilmThick").Value = vntFilmThick
    o1.Fields.Item("SurfArea").Value = vntSurfArea
    o1.Fields.Item("RefDur").Value = vntRefDur
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Fields.Item("FracHand").Value = vntFracHand
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Fields.Item("Hand") = vntHand
    o2.Update

    ' write hand data
    O3.AddNew
    O3.Fields.Item("Sim") = sim
    O3.Fields.Item("Hand") = vntHand
    O3.Update

    DoEvents

    oParam.rExposure = vntExposure
    oParam.rHand = vntHand
End Sub

```

2.11 Residential – Ingestion – Post Application – Granules/Pellets (Formulation)

```

\
' Algorithm - Ingestion-101 (F12101)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntFracAIForm, vntIngestionRate, vntBodyWt

    sim = sim + 1

    ' do calcs
    vntFracAIForm = oParam.oProd.GetDistributionValue("s10")

    vntIngestionRate = Settings.Setting("s2").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntIngestionRate * vntFracAIForm / vntBodyWt

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("FracAIForm").Value = vntFracAIForm
    o1.Fields.Item("IngestRate").Value = vntIngestionRate
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.12 Residential – Ingestion – Post Application – Grass/Plants

```

\
' Algorithm - Ingestion-102 (F12102)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim vntApplication, vntGroundCover, vntDislodgeable, vntIngestionRate,
vntBodyWt

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    sim = sim + 1

    ' do calcs
vntApplication = oParam.oProd.GetDistributionValue("s3")

vntGroundCover = Settings.Setting("s2").Value()
vntDislodgeable = Settings.Setting("s3").Value()
vntIngestionRate = Settings.Setting("s4").Value()
vntBodyWt = oParam.rBodyWeight

    Dim vntResidue
vntResidue = vntApplication * vntDislodgeable / vntGroundCover

    Dim vntExposure
vntExposure = vntResidue * vntIngestionRate / vntBodyWt

    ' write intermediate results
o1.AddNew
o1.Fields.Item("SIM").Value = sim
o1.Fields.Item("ApplicAI").Value = vntApplication
o1.Fields.Item("GndCover").Value = vntGroundCover
o1.Fields.Item("Dislodge").Value = vntDislodgeable
o1.Fields.Item("IngestRate").Value = vntIngestionRate
o1.Fields.Item("BodyWt").Value = vntBodyWt
o1.Update

    ' write final results
o2.AddNew
o2.Fields.Item("Sim") = sim
o2.Fields.Item("Exposure") = vntExposure
o2.Fields.Item("Residue") = vntResidue
o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.13 Residential – Ingestion – Post Application – Soil

```

\
' Algorithm - Ingestion-103 (F12103)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntApplication, vntSoilDensity, vntSoilThickness, vntDislodgeable
    Dim vntIngestionRate, vntBodyWt

    ' do calcs
    sim = sim + 1

    vntApplication = oParam.oProd.GetDistributionValue("s3")

    vntSoilDensity = Settings.Setting("s2").Value()
    vntSoilThickness = Settings.Setting("s3").Value()
    vntDislodgeable = Settings.Setting("s4").Value()
    vntIngestionRate = Settings.Setting("s5").Value()

    vntBodyWt = oParam.rBodyWeight

    Dim vntResidue
    vntResidue = vntApplication * vntDislodgeable / (vntSoilDensity *
vntSoilThickness)

    Dim vntExposure
    vntExposure = vntResidue * vntIngestionRate / vntBodyWt

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("AreaTreat").Value = vntApplication
    o1.Fields.Item("SoilDens").Value = vntSoilDensity
    o1.Fields.Item("SoilThick").Value = vntSoilThickness
    o1.Fields.Item("Dislodge").Value = vntDislodgeable
    o1.Fields.Item("IngestRate").Value = vntIngestionRate
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Fields.Item("Residue") = vntResidue
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```


2.14 Residential – Ingestion – Post Application – Paint Chips

```

\
'Algorithm - Ingestion-104 (F12104)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntFracAIPaint, vntFracAIAvail, vntIngestionRate, vntBodyWt

    sim = sim + 1

    ' do calcs
    vntFracAIPaint = Settings.Setting("s1").Value()
    vntFracAIAvail = Settings.Setting("s2").Value()
    vntIngestionRate = Settings.Setting("s3").Value()

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntIngestionRate * vntFracAIPaint * vntFracAIAvail /
vntBodyWt

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("FracPaint").Value = vntFracAIPaint
    o1.Fields.Item("FracAvail").Value = vntFracAIAvail
    o1.Fields.Item("IngestRate").Value = vntIngestionRate
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.15 Residential – Ingestion – Post Application – Water Concentration

```

\
' Algorithm - Ingestion-105 (F12105)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntWaterConc, vntIngestionRate, vntExpDur, vntBodyWt

    ' do calcs
    sim = sim + 1

    vntWaterConc = oParam.oProd.GetDistributionValue("s11")

    vntIngestionRate = Settings.Setting("s2").Value()
    vntExpDur = Settings.Setting("s3").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntWaterConc * vntIngestionRate * vntExpDur /
(vntBodyWt)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("WaterConc").Value = vntWaterConc
    o1.Fields.Item("IngestRate").Value = vntIngestionRate
    o1.Fields.Item("ExpDur").Value = vntExpDur
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.16 Residential – Ingestion – Post Application – Flux Rate

```

\
' Algorithm - Ingestion-106 (F12106)
'
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntFluxRateAI, vntSurfArea, vntBodyWt

    ' do calcs
    sim = sim + 1

    vntFluxRateAI = Settings.Setting("s1").Value()
    vntSurfArea = Settings.Setting("s2").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntFluxRateAI * vntSurfArea / vntBodyWt

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("FluxRateAi").Value = vntFluxRateAI
    o1.Fields.Item("SurfArea").Value = vntSurfArea
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.17 Residential – Ingestion – Post Application – Hand-to-Mouth – Mass Balance

```

\
' Algorithm - Ingestion-107 (F12107)
'
Public Function AlgorithmBW(oParam As clsParamBlock) As Double

    sim = sim + 1

    Dim vntContactFreq, vntTransEff, vntExpDur, vntBodyWt, vntSurfAreaU,
vntSurfAreaHM

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    ' do calcs

    vntContactFreq = Settings.Setting("s2").Value()
    vntTransEff = Settings.Setting("s3").Value()
    vntExpDur = Settings.Setting("s4").Value()
    vntSurfAreaHM = Settings.Setting("s6").Value()

    Dim dblSurfaceArea As Double
    Dim dblHandsU As Double, dblHandsC As Double
    Dim dblUpperU As Double, dblUpperC As Double
    Dim dblLowerU As Double, dblLowerC As Double
    Dim dblFeetU As Double, dblFeetC As Double

    Call Module1.GetAreas(oParam.oPop, dblHandsU, dblHandsC, dblUpperU,
dblUpperC, dblLowerU, dblLowerC, dblFeetU, dblFeetC)
    dblSurfaceArea = Surface_A * (oParam.rBodyHeight ^ Surface_B) *
(oParam.rBodyWeight ^ Surface_C)
    vntSurfAreaU = dblHandsU * dblSurfaceArea

    vntBodyWt = oParam.rBodyWeight 'Settings.Setting("s7").Value()

    Dim vntTransFac
    Dim N%, vntSum

    vntSum = 0
    For N = 1 To Int(vntContactFreq * vntExpDur)
        vntSum = vntSum + ((1 - vntTransEff) ^ (N - 1))
    Next N

    If vntSum = 0 Then
        vntTransFac = vntTransEff
    Else
        vntTransFac = vntTransEff * vntSum
    End If
    Dim vntExposure
    vntExposure = (oParam.rHand * vntTransFac / vntBodyWt) * (vntSurfAreaU
/ vntSurfAreaHM)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("HandExp").Value = oParam.rHand 'Handexp
    o1.Fields.Item("ContFreq").Value = vntContactFreq
    o1.Fields.Item("TransEff").Value = TvntTransEff
    o1.Fields.Item("ExpDur").Value = vntExpDur
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Fields.Item("TransFac").Value = vntTransFac
    o1.Fields.Item("SurfAreaU").Value = vntSurfAreaU
    o1.Fields.Item("SurfAreaHM").Value = vntSurfAreaHM

```

```
o1.Update

' write final results
o2.AddNew
o2.Fields.Item("Sim") = sim
o2.Fields.Item("Exposure") = vntExposure
o2.Fields.Item("TransFac") = vntTransFac
o2.Update

DoEvents

'Set I1 = Nothing          ' free up Recordset

AlgorithmBW = vntExposure
End Function
```

2.18 Residential – Ingestion – Post Application – Hand-to-Mouth – Fraction Transferred

```

\
' Algorithm - Ingestion-108 (F12108)
'
Public Function AlgorithmBW(oParam As clsParamBlock) As Double

    sim = sim + 1

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntFracTrans, vntSurfAreaU, vntSurfAreaHM, vntBodyWt

    ' do calcs

    vntFracTrans = Settings.Setting("s2").Value()
    vntSurfAreaHM = Settings.Setting("s4").Value()

    Dim dblSurfaceArea As Double
    Dim dblHandsU As Double, dblHandsC As Double
    Dim dblUpperU As Double, dblUpperC As Double
    Dim dblLowerU As Double, dblLowerC As Double
    Dim dblFeetU As Double, dblFeetC As Double

    Call Module1.GetAreas(oParam.oPop, dblHandsU, dblHandsC, dblUpperU,
    dblUpperC, dblLowerU, dblLowerC, dblFeetU, dblFeetC)
    SurfaceArea = Surface_A * (oParam.rBodyHeight ^ Surface_B) *
(oParam.rBodyWeight ^ Surface_C)
    vntSurfAreaU = dblHandsU * dblSurfaceArea

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = (oParam.rHand * vntFracTrans / vntBodyWt) *
(vntSurfAreaU / vntSurfAreaHM)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("HandExp").Value = oParam.rHand 'Handexp
    o1.Fields.Item("FracTrans").Value = vntFracTrans
    o1.Fields.Item("SurfAreaU").Value = vntSurfAreaU
    o1.Fields.Item("SurfAreaHM").Value = vntSurfAreaHM
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    AlgorithmBW = vntExposure
End Function

```

2.19 Residential – Ingestion – Post Application – Hand-to-Mouth – EPA SOP

```

\
' Algorithm - Ingestion-109 (F12109)
,
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    Dim vntApplication, vntDislodgeable, vntContactFreq, vntTransEff,
vntExpDur, vntSurfArea, vntBodyWt

    ' do calcs
    sim = sim + 1

    vntApplication = oParam.oProd.GetDistributionValue("s3")
    vntDislodgeable = oParam.oProd.GetDistributionValue("s5")

    vntContactFreq = Settings.Setting("s3").Value()
    vntTransEff = Settings.Setting("s4").Value()
    vntExpDur = Settings.Setting("s5").Value()
    vntSurfArea = Settings.Setting("s6").Value()
    vntBodyWt = oParam.rBodyWeight

    Dim vntTransRes
    vntTransRes = vntApplication * vntDislodgeable * 100

    Dim vntExposure
    vntExposure = vntTransRes * vntContactFreq * vntSurfArea * vntTransEff
* vntExpDur / vntBodyWt

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("AppArea").Value = vntApplication
    o1.Fields.Item("Dislodge").Value = vntDislodgeable
    o1.Fields.Item("ContFreq").Value = vntContactFreq
    o1.Fields.Item("TransEff").Value = vntTransEff
    o1.Fields.Item("ExpDur").Value = vntExpDur
    o1.Fields.Item("SurfArea").Value = vntSurfArea
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.20 Residential – Inhalation – During Application – Unit Exposure, Area Treated

```

\
' Algorithm - Inhalation-101 (F13101)
,
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim vntApplication, vntAreaTreated, vntUnitExp, vntRefDur, vntBodyWt

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    sim = sim + 1

    ' do calcs
    vntApplication = oParam.oProd.GetDistributionValue("s3")
    vntUnitExp = oParam.oProd.GetDistributionValue("s7")

    vntAreaTreated = Settings.Setting("s2").Value()
    vntRefDur = Settings.Setting("s4").Value()

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntUnitExp * vntApplication * vntAreaTreated /
(vntRefDur * vntBodyWt)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("Applic").Value = vntApplication
    o1.Fields.Item("AreaTreat").Value = vntAreaTreated
    o1.Fields.Item("UnitExp").Value = vntUnitExp
    o1.Fields.Item("RefDur").Value = vntRefDur
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```


2.21 Residential – Inhalation – During Application – Unit Exposure, Amount of Formulation Used

```

\
' Algorithm - Inhalation-102 (F13102)
,

Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim vntApplication, vntAmtFormUsed, vntUnitExp, vntRefDur, vntBodyWt

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    sim = sim + 1

    ' do calcs
    vntApplication = oParam.oProd.GetDistributionValue("s4")
    vntAmtFormUsed = oParam.oProd.GetDistributionValue("s2")
    vntUnitExp = oParam.oProd.GetDistributionValue("s7")

    vntRefDur = Settings.Setting("s4").Value()

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntUnitExp * vntApplication * vntAmtFormUsed /
(vntRefDur * vntBodyWt)

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("Applic").Value = vntApplication
    o1.Fields.Item("AmtUsed").Value = vntAmtFormUsed
    o1.Fields.Item("UnitExp").Value = vntUnitExp
    o1.Fields.Item("RefDur").Value = vntRefDur
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.22 Residential – Inhalation – Post Application – Air Concentration, Specified

```

\
' Algorithm - Inhalation-103 (F13103)
,
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim vntAirConcAI, vntExpDur, vntInhalationRate, vntBodyWt

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    sim = sim + 1

    ' do calcs
    vntAirConcAI = oParam.oProd.GetDistributionValue("s1")

    vntExpDur = Settings.Setting("s2").Value()

    Dim b As Double, C As Double, BMR As Double
    GetBNF oParam.oPop, b, C
    BMR = b * oParam.rBodyWeight + C
    vntInhalationRate = 0.05625 * BMR

    vntBodyWt = oParam.rBodyWeight

    Dim vntExposure
    vntExposure = vntAirConcAI * vntInhalationRate * vntExpDur / vntBodyWt

    ' write intermediate results
    o1.AddNew
    o1.Fields.Item("SIM").Value = sim
    o1.Fields.Item("AirConcAI").Value = vntAirConcAI
    o1.Fields.Item("ExpDur").Value = vntExpDur
    o1.Fields.Item("InhaleRate").Value = vntInhalationRate
    o1.Fields.Item("BodyWt").Value = vntBodyWt
    o1.Update

    ' write final results
    o2.AddNew
    o2.Fields.Item("Sim") = sim
    o2.Fields.Item("Exposure") = vntExposure
    o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.23 Residential – Inhalation – Post Application – Air Concentration, Calculated

```

\
' Algorithm - Inhalation-104 (F13104)
,
Public Function Algorithm(oParam As clsParamBlock) As Double

    Dim vntAmtFormUsed, vntFracAIForm, vntFormDens, vntDilFac, vntVolume,
vntExpDur, vntInhalationRate, vntBodyWt

    Dim o1 As ADODB.Recordset
    Dim o2 As ADODB.Recordset

    Set o1 = out1
    Set o2 = out2

    sim = sim + 1

    ' do calcs
vntAmtFormUsed = oParam.oProd.GetDistributionValue("s2")
vntFracAIForm = oParam.oProd.GetDistributionValue("s10")

    vntFormDens = Settings.Setting("s3").Value()
vntDilFac = Settings.Setting("s4").Value()
vntVolume = Settings.Setting("s5").Value()
vntExpDur = Settings.Setting("s6").Value()

    Dim b As Double, C As Double, BMR As Double
GetBNF oParam.oPop, b, C
BMR = b * oParam.rBodyWeight + C
vntInhalationRate = 0.05625 * BMR

    vntBodyWt = oParam.rBodyWeight

    Dim vntAirConc
vntAirConc = vntAmtFormUsed * vntFracAIForm * vntFormDens * vntDilFac
/ vntVolume

    Dim vntExposure
vntExposure = vntAirConc * vntInhalationRate * vntExpDur / vntBodyWt

    ' write intermediate results
o1.AddNew
o1.Fields.Item("SIM").Value = sim
o1.Fields.Item("AmtUsed").Value = vntAmtFormUsed
o1.Fields.Item("FracAIForm").Value = vntFracAIForm
o1.Fields.Item("FormDens").Value = vntFormDens
o1.Fields.Item("DilFac").Value = vntDilFac
o1.Fields.Item("Volume").Value = vntVolume
o1.Fields.Item("ExpDur").Value = vntExpDur
o1.Fields.Item("InhaleRate").Value = vntInhalationRate
o1.Fields.Item("BodyWt").Value = vntBodyWt
o1.Fields.Item("AirConc").Value = vntAirConc
o1.Update

    ' write final results
o2.AddNew
o2.Fields.Item("Sim") = sim
o2.Fields.Item("Exposure") = vntExposure
o2.Update

    DoEvents

    Algorithm = vntExposure
End Function

```

2.24 Residential – Exposure Calculations

```

'
' Support code for REx residential functions
'

Option Explicit

' for Body surface area determinations
Public Const Surface_A As Double = 0.0235
Public Const Surface_B As Double = 0.4225
Public Const Surface_C As Double = 0.5145

Public Const cRouteDermal As Integer = 1
Public Const cRouteIngFood As Integer = 2
Public Const cRouteIngHTM As Integer = 3
Public Const cRouteIngWater As Integer = 4
Public Const cRouteInhale As Integer = 5

Public Enum eDuringPost
    eDuring = 1
    ePost = 2
End Enum

Public Enum eProfCons
    eProf = 1      ' professional product
    eCons = 2     ' consumer product
End Enum

Public Enum eSex
    eMale = 0
    eFemale = 1
End Enum

' get population list
Public Function GetPop() As Dictionary

    Dim dPop As Dictionary

    '1st see if population has been specified in a SharedVariable(...)

    Dim vRefPop As Variant

    Dim vGid As Variant
    If SharedVariableExists("InputPop", "Group") = False Then
        MsgBox "Please select a population file?", vbExclamation, "No
population group found!"
        Exit Function
    Else
        vGid = GetSharedVariable("InputPop", "Group") ' open population
group
    End If

    Set dPop = New Dictionary

    Dim ids As iDataset
    Set ids = GetSystemLibrary.OpenDataset(CStr(vGid))

    Dim r As adodb.Recordset
    Dim id As iData

    ' we need to go through the catalog for the dataset and find
    ' a table of type t16101, This will be the table we want to use
    ' we'll assume (for now) that there is only 1 pop table per file
    Dim sTableName As String
    Dim dCat As Dictionary
    Set dCat = ids.GetCatalog

```

```

Dim vItem
For Each vItem In dCat.Keys
    Dim dItem As Dictionary
    Set dItem = dCat.Item(vItem)
    If LCase(dItem.Item("tabletype")) = "t16101" Then
        sTableName = CStr(vItem)
        Exit For
    End If
Next vItem
If sTableName = "" Then
    Err.Raise 42, , "No pop table!!!!"
End If

Set id = ids.OpenTable(sTableName)

If id Is Nothing Then
    MsgBox Err.Description, vbCritical, "EventAllocator(GetPopulation
- INTERNAL ERROR)"
    Exit Function
End If

Set r = id.Recordset()

Do While Not r.EOF
    Dim Cares_id As String
    Cares_id = r.Fields.Item("1").Value()

    Dim vWeights As Variant
    vWeights = GetWeights(r)

    Dim vHeights As Variant
    vHeights = GetHeights(r)

    Dim oPopInfo As clsPopInfo
    Set oPopInfo = New clsPopInfo

    oPopInfo.vWeights = vWeights
    oPopInfo.vHeights = vHeights
    oPopInfo.nAge = r.Fields.Item("82").Value()
    oPopInfo.sID = Cares_id

    If Not dPop.Exists(Cares_id) Then
        dPop.Add Cares_id, oPopInfo
    Else
        MsgBox "Cares_id=" & Cares_id & " already exists in
collection", vbCritical, "REX-1(GetPop() - INTERNAL ERROR)"
    End If
    r.MoveNext

    DoEvents
Loop

r.Close
Set id = Nothing
Set ids = Nothing

Set GetPop = dPop
End Function

Public Function GetChem() As Dictionary

    Dim dSel As Dictionary
    If SharedVariableExists("InputChem", "all") = False Then
        MsgBox "Please select desired Chemical/CAS#'s", vbExclamation, "No
Chemical/CAS# selection founc!"
        Exit Function
    Else
        Set dSel = GetSharedVariable("InputChem", "all") ' get selected
chemicals
    End If

```

```

Dim dChem As Dictionary
Set dChem = New Dictionary

Dim n As Long
n = 0

Dim vCas As Variant
For Each vCas In dSel.Keys
    If vCas <> "*" Then
        n = n + 1
        dChem.Add vCas, dSel.Item(vCas)      ' CAS, Chemical name
    End If
Next vCas

Set GetChem = dSel
End Function

Public Function GetProductInfo2(dChem As Dictionary, vScenario As Variant)
As Dictionary
    Dim dProd As Dictionary

    Dim vA As Variant
    If SharedVariableExists("InputRes", "Products") = False Then
        MsgBox "Please select a product file?", vbExclamation, "No product
filefound!"
        Exit Function
    Else
        vA = GetSharedVariable("InputRes", "Products")      ' open population
group
    End If

    Dim vGid As Variant
    Dim vName As Variant
    vGid = vA(0)      ' file refid
    vName = vA(1)      ' table name

    Dim ids As iDataset
    Set ids = GetSystemLibrary.OpenDataset(CStr(vGid))

    Dim r As adodb.Recordset
    Dim id As iData

    Set id = ids.OpenTable(CStr(vName))

    If id Is Nothing Then
        MsgBox Err.Description, vbCritical, "REX-1(GetProduct - INTERNAL
ERROR)"
        Exit Function
    End If

    Set dProd = New Dictionary

    Set r = id.Recordset()
    Do While Not r.EOF
        Dim ProductId, Name, CAS, Form, WtFrac, Scenario, ProfCons, Market

        Scenario = r.Fields.Item("Scenario").Value

        If vScenario = Scenario Then
            Dim oProdInfo As clsProdInfo
            Set oProdInfo = New clsProdInfo

            oProdInfo.ProductId = r.Fields.Item("ProductID").Value
            oProdInfo.Name = r.Fields.Item("Name").Value
            oProdInfo.CAS = r.Fields.Item("CAS").Value
            oProdInfo.Form = r.Fields.Item("Form").Value
            oProdInfo.WtFrac = r.Fields.Item("Wt Frac").Value
            oProdInfo.Scenario = r.Fields.Item("Scenario").Value
            oProdInfo.ProfCons = r.Fields.Item("Prof/Cons").Value
            oProdInfo.Market = r.Fields.Item("Market").Value
        End If
    Loop
End Function

```

```

        ProductId = oProdInfo.ProductId
        CAS = oProdInfo.CAS

        If dProd.Exists(ProductId) Then
            Err.Raise 42, , "Duplicate Product ID!!!!!"
        End If

        ' only add if chemical has been selected .. and if appropriate
scenario
        If dChem.Exists(CAS) Then
            dProd.Add ProductId, oProdInfo
        End If

        ' set up distributions
        Dim vSet As Variant
        For Each vSet In oProdInfo.ProdSettings.Settings.Keys
            oProdInfo.InitializeDistribution CStr(vSet), "Type", 1
            oProdInfo.InitializeDistribution CStr(vSet), "Single",
0.01
        Next vSet

        InitProdDist r, oProdInfo
    End If

    r.MoveNext
Loop

    r.Close
    Set id = Nothing
    Set ids = Nothing

    Set GetProductInfo2 = dProd
End Function

Public Function Decay(n As Long, Optional f As Double = 0.1) As Double
    Decay = (1 - f) ^ n
End Function

Public Function GetEventDays2(rEvents As adodb.Recordset, sCaresId As
String, vScenario As Variant) As Dictionary
    Set GetEventDays2 = Nothing

    Dim sScenario As String
    Dim dScenarios As Dictionary

    If SharedVariableExists("InputRes", "scenarios") = True Then
        Set dScenarios = GetSharedVariable("InputRes", "scenarios")
        sScenario = dScenarios(CLng(vScenario))
    Else
        MsgBox "No scenario dictionary found?", vbExclamation,
"REX1(Module1:GetEventDays)"
        Exit Function
    End If

    ' is Current scenario in the list of events?
    Dim bInEvents As Boolean
    bInEvents = False

    Dim fld As adodb.Field
    For Each fld In rEvents.Fields
        If LCase(fld.Name) = LCase(sScenario) Then
            bInEvents = True
            Exit For
        End If
    Next fld

    If bInEvents = False Then Exit Function

    ' get all potential events for this person

```

```

rEvents.Filter = ""
rEvents.MoveFirst

Dim sFilter As String
sFilter = "[CARES ID] = '" & sCaresId & "'"
rEvents.Filter = sFilter

If rEvents.RecordCount = 0 Then
    rEvents.Filter = ""
    Exit Function
End If

Dim C As Dictionary
Set C = New Dictionary
Do While Not rEvents.EOF
    Dim sProd As String
    sProd = rEvents.Fields.Item(sScenario).Value
    If sProd <> "?" And sProd <> "-1" Then
        ' entry consists of "Day => ProductID"
        C.Add rEvents.Fields.Item("Day").Value, sProd
    End If
    rEvents.MoveNext
Loop

Set GetEventDays2 = C

rEvents.Filter = ""
rEvents.MoveFirst

End Function

' !!! TODO: Fold into ExpCalcs
Public Function HToMExpCalcsBW(clsme As Object, irs As adodb.Recordset,
ors As adodb.Recordset, vScenario As Variant, vRoute As Variant,
vDuringPost As Variant) As Variant

    Dim oParam As New clsParamBlock

    irs.Filter = "Hand <> 0"

    Dim f1 As New frmWait          ' wait form

    Dim dPop As Dictionary
    Set dPop = GetPop()          ' get selected population
    If dPop Is Nothing Then
        HToMExpCalcsBW = "010:Flxxxx:Couldn't get Population input"
        Exit Function
    End If

    ' if we get here we need to allocate a result array for each
day, person, chemical
    f1.Caption = "Calculating H-To-M exposure values - Please Wait"
    f1.Label1.Caption = "CARES ID"
    f1.Label3.Caption = "Event Day"
    f1.Show

    Dim bReInit As Boolean        ' true when new group has been found

    bReInit = True

    Dim aResults(1 To 365) As Single
    Dim dpDays As New Dictionary
    Dim bDoOutput As Boolean     ' true when output is necessary

    Dim vPopLast, vChemLast
    vPopLast = ""
    vChemLast = ""

    ' loop is ineffecient. ReOrganize - later
    Do While Not irs.EOF

```



```

Dim vPop, vChem, vDay

vPop = irs.Fields.Item("Cares id").Value
vChem = irs.Fields.Item("CAS").Value
vDay = irs.Fields.Item("Day").Value

oParam.vChem = vChem
oParam.vDay = vDay

Dim vWeights As Variant
Dim vHeights As Variant

Dim oPopInfo As clsPopInfo
Set oPopInfo = dPop.Item(vPop)

Set oParam.oPop = oPopInfo

vWeights = oPopInfo.vWeights      ' get body weight array
vHeights = oPopInfo.vHeights      ' get body height array

Dim nAge As Integer
nAge = oPopInfo.nAge                ' get persons age
oParam.nAge = nAge

f1.Label2.Caption = vPop

Dim meNf As NotitiaFunction
Set meNf = clsme

Dim bRunMe As Boolean
bRunMe = False
bDoOutput = False

Select Case meNf.FunctionName
    ' no output for ingestion 107,8,9 if Age > 3
    Case "F12107", "F12108", "F12109"
        If nAge <= 3 Then
            NotitiaLog.WriteLine meNf.FunctionName & " : No output
for CARES ID: " & vPop & " (Age=" & nAge & ")"
            bRunMe = True
        End If
    Case Else
        bRunMe = True
End Select

' clear out results array. a ReDim may be faster
Dim n%
If bReInit = True Then
    For n = 1 To 365
        aResults(n) = 0
    Next n
    dpDays.RemoveAll                ' for day/event/product info

    bReInit = False                ' has been reinitialized
End If

f1.Label4.Caption = vDay

If bRunMe = True Then
    oParam.rBodyWeight = vWeights(GetMonth(vDay))
    oParam.rBodyHeight = vHeights(GetMonth(vDay))

    oParam.rHand = irs.Fields.Item("Hand").Value

    oParam.rExposure = clsme.AlgorithmBW(oParam)

    ' now do decay calcs
    bDoOutput = True

    If vDuringPost = eDuring Then
        aResults(vDay) = aResults(vDay) + oParam.rExposure

```

```

Else
    Dim nDecay As Long
    For nDecay = vDay To 365
        aResults(nDecay) = aResults(nDecay) +
(oParam.rExposure * Decay(nDecay - vDay))
    Next nDecay
End If

'record product use day
dpDays.Add CLng(vDay), irs.Fields.Item("Product").Value
End If
If fl.bCancelled = True Then
    fl.Hide
    HToMExpCalcsBW = "014:Flxxxx:User Cancelled"
    Exit Function
End If

vPopLast = vPop
vChemLast = vChem

irs.MoveNext

If Not irs.EOF Then

    vPop = irs.Fields.Item("Cares id").Value
    vChem = irs.Fields.Item("CAS").Value
    vDay = irs.Fields.Item("Day").Value

    If (vPop <> vPopLast And vPopLast <> "") Or _
        (vChem <> vChemLast And vChemLast <> "") Then

        bReInit = True ' group has changed write output etc

        If bDoOutput = True And bRunMe = True Then
            CARESOutputHand ors, aResults, vPop, vChem, dpDays,
vScenario, vRoute, vDuringPost
        End If
        bDoOutput = False ' output has been done

    End If
Else
    bDoOutput = True
End If

DoEvents
Loop

If bDoOutput = True And bRunMe = True Then
    CARESOutputHand ors, aResults, vPop, vChem, dpDays, vScenario,
vRoute, vDuringPost
End If

fl.Hide

irs.Filter = ""

If irs.RecordCount > 0 Then
    irs.MoveFirst
End If

HToMExpCalcsBW = 0
End Function

Public Function ExpCalcs(clsme As Object, ors As adodb.Recordset,
vScenario As Variant, vRoute As Variant, vDuringPost As Variant) As
Variant
    Dim dPop As New Dictionary
    Dim dChem As New Dictionary
    Dim dProd As New Dictionary

    Dim n As Integer

```

```

Dim NF As NotitiaFunction
Set NF = clsme

Dim f1 As New frmWait      ' wait form

Set dPop = GetPop()      ' get selected population
If dPop Is Nothing Then
    ExpCalcs = "010:Flxxxx:Couldn't get Population input"
    Exit Function
End If

Set dChem = GetChem()    ' get selected chemicals
If dChem Is Nothing Then
    ExpCalcs = "011:Flxxxx:Couldn't get Chemical input"
    Exit Function
End If

Set dProd = GetProductInfo2(dChem, vScenario) ' get product info
If dProd Is Nothing Then
    NotitiaLog.WriteLine "No products found for scenario " & vScenario
    ExpCalcs = 0
    Exit Function
End If

' get event allocator input
If SharedVariableExists("Events", "DayUse") = False Then
    ExpCalcs = "013:Flxxx:Couldn't get Event Allocation data"
    Exit Function
End If

Dim r1 As adodb.Recordset
Set r1 = GetSharedVariable("Events", "DayUse")

r1.Filter = ""
If r1.RecordCount <= 0 Then
    ExpCalcs = "014:Flxxxx:No records in Event Allocation data"
    Exit Function
End If
r1.MoveFirst

' if we get here we need to allocate a result array for each
day, person, chemical
f1.Caption = "Calculating exposure values - Please Wait"
f1.Label1.Caption = "CARES ID"
f1.Label3.Caption = "Event Day"
f1.Show

Dim aResults As Variant

Dim vPop, vChem, vDay

Dim dpDays As New Dictionary

Dim oParam As clsParamBlock

For Each vPop In dPop.Keys

    f1.Label2.Caption = vPop
    f1.Label2.Refresh

    f1.Label4.Caption = ""
    f1.Label4.Refresh

    Dim oPopInfo As clsPopInfo
    Set oPopInfo = dPop.Item(vPop)

    Dim vWeights As Variant
    Dim vHeights As Variant
    vWeights = oPopInfo.vWeights      ' get body weight array
    vHeights = oPopInfo.vHeights     ' get body height array

```

```

Dim dEvents As Dictionary
Set dEvents = GetEventDays2(r1, CStr(vPop), CStr(vScenario))

If Not dEvents Is Nothing Then

    dpDays.RemoveAll      ' for day/event/product info

    ' 1st get products used for person for entire year
    For Each vDay In dEvents.Keys

        fl.Label4.Caption = vDay
        fl.Label4.Refresh

        ' get productid that person used for this day/scenario
        Dim sProdId As String
        sProdId = dEvents.Item(vDay)

        ' get prodinfo for product
        Dim oProd As clsProdInfo
        If dProd.Exists(sProdId) Then
            Set oProd = dProd.Item(sProdId)
        End If

        If Not oProd Is Nothing Then
            ' determine if we should run Algorithm
            If RunOK(NF, oPopInfo, oProd, vScenario, vRoute,
vDuringPost) = True Then

                ' build a new Parameter block for the algorithm
                Set oParam = New clsParamBlock
                Set oParam.oPop = oPopInfo
                Set oParam.oProd = oProd

                oParam.vScenario = vScenario
                oParam.vDuringPost = vDuringPost
                oParam.vRoute = vRoute
                oParam.nAge = oPopInfo.nAge
                oParam.vDay = vDay
                oParam.rBodyWeight = vWeights(GetMonth(vDay)) '
get body weight for this day
                oParam.rBodyHeight = vHeights(GetMonth(vDay)) '
get body height for this day

                dpDays.Add CLng(vDay), oParam
            End If
        End If

    Next vDay      ' onto next event day

    ' 2nd now do exposure calcs ...
    Dim dResults As New Dictionary

    ' get parameter block for each event
    For Each vDay In dpDays.Keys
        fl.Label4.Caption = vDay
        fl.Label4.Refresh

        Set oParam = dpDays.Item(vDay)

        ' get results array for products' CAS
        If Not dResults.Exists(oParam.oProd.CAS) Then
            ReDim aResults(1 To 365) As Single
            For n = 1 To 365
                aResults(n) = 0
            Next n
            dResults.Item(oParam.oProd.CAS) = Empty
        Else
            aResults = dResults.Item(oParam.oProd.CAS)
        End If
    End If

```

```

        ' at this point aResults either is a new empty array or
...
        ' ... already contains exposure values for the CAS

        ' do exposure calculation
        oParam.rExposure = clsme.Algorithm(oParam)

        ' now do decay calcs
        If vDuringPost = eDuring Then
            aResults(vDay) = aResults(vDay) + oParam.rExposure
        Else
            Dim nDecay As Long
            For nDecay = vDay To 365
                aResults(nDecay) = aResults(nDecay) +
(oParam.rExposure * Decay(nDecay - vDay))
            Next nDecay
        End If

        ' save CAS exposure back into CAS dictionary
        dResults.Item(oParam.oProd.CAS) = aResults
    Next vDay

    ' now for each CAS that person was exposed to write output
    For Each vChem In dResults
        aResults = dResults.Item(vChem)
        CARESOutputNew ors, aResults, vPop, vChem, dpDays,
vScenario, vRoute, vDuringPost
    Next vChem

    dResults.RemoveAll

    End If      ' if NO events for person

    DoEvents
    If f1.bCancelled = True Then
        f1.Hide
        ExpCalcs = "014:F1xxxx:User Cancelled"
        Exit Function
    End If

NextvPop:
    Next vPop

    f1.Hide

    ExpCalcs = 0
End Function

'!!! TODO: Fold into ExpCalcs
Public Function ExpCalcs2(clsme As Object, ors As adodb.Recordset,
vScenario As Variant, vRoute As Variant, vDuringPost As Variant) As
Variant
    Dim dPop As New Dictionary
    Dim dChem As New Dictionary
    Dim dProd As New Dictionary

    Dim n As Integer

    Dim NF As NotitiaFunction
    Set NF = clsme

    Dim f1 As New frmWait      ' wait form

    Set dPop = GetPop()      ' get selected population
    If dPop Is Nothing Then
        ExpCalcs2 = "010:F1xxxx:Couldn't get Population input"
        Exit Function
    End If

```

```

Set dChem = GetChem()           ' get selected chemicals
If dChem Is Nothing Then
    ExpCalcs2 = "011:Flxxxx:Couldn't get Chemical input"
    Exit Function
End If

Set dProd = GetProductInfo2(dChem, vScenario) ' get product info
If dProd Is Nothing Then
    ExpCalcs2 = 0
    Exit Function
End If

' get event allocator input
If SharedVariableExists("Events", "DayUse") = False Then
    ExpCalcs2 = "013:Flxxxx:Couldn't get Event Allocation data"
    Exit Function
End If

Dim r1 As adodb.Recordset
Set r1 = GetSharedVariable("Events", "DayUse")

r1.Filter = ""
If r1.RecordCount <= 0 Then
    ExpCalcs2 = "014:Flxxxx:No records in Event Allocation data"
    Exit Function
End If
r1.MoveFirst

' if we get here we need to allocate a result array for each
day, person, chemical
f1.Caption = "Calculating exposure values - Please Wait"
f1.Label1.Caption = "CARES ID"
f1.Label3.Caption = "Event Day"
f1.Show

Dim aResults As Variant           ' exposure output
Dim aHResults As Variant         ' hand output

Dim vPop, vChem, vDay

Dim dpDays As New Dictionary

Dim oParam As clsParamBlock

For Each vPop In dPop.Keys
    f1.Label2.Caption = vPop
    f1.Label2.Refresh

    f1.Label4.Caption = ""
    f1.Label4.Refresh

    Dim vWeights As Variant
    Dim vHeights As Variant

    Dim oPopInfo As clsPopInfo
    Set oPopInfo = dPop.Item(vPop)

    vWeights = oPopInfo.vWeights   ' get body weight array
    vHeights = oPopInfo.vHeights  ' get body height array

    ' get events for this person/scenario
    Dim dEvents As Dictionary
    Set dEvents = GetEventDays2(r1, CStr(vPop), CStr(vScenario))

    If Not dEvents Is Nothing Then

        dpDays.RemoveAll          ' for day/event/product info

        For Each vDay In dEvents.Keys
            f1.Label4.Caption = vDay
            f1.Label4.Refresh
        
```

```

Dim sProdId As String
sProdId = dEvents.Item(vDay)

Dim oProd As clsProdInfo
If dProd.Exists(sProdId) Then
    Set oProd = dProd.Item(sProdId)
End If

If Not oProd Is Nothing Then
    If RunOK(NF, oPopInfo, oProd, vScenario, vRoute,
vDuringPost) = True Then
        ' build a new Parameter block for the algorithm

        Set oParam = New clsParamBlock
        Set oParam.oPop = oPopInfo
        Set oParam.oProd = oProd

        oParam.vScenario = vScenario
        oParam.vDuringPost = vDuringPost
        oParam.vRoute = vRoute
        oParam.nAge = oPopInfo.nAge
        oParam.vDay = vDay
        oParam.rBodyWeight = vWeights(GetMonth(vDay)) '
get body weight for this day
        oParam.rBodyHeight = vHeights(GetMonth(vDay)) '
get body height for this day

        'record product use day
        dpDays.Add CLng(vDay), oParam
    End If
End If

DoEvents
If f1.bCancelled = True Then
    f1.Hide
    ExpCalcs2 = "014:F1xxxx:User Cancelled"
    Exit Function
End If

Next vDay

Dim dResults As New Dictionary

' 2nd now do exposure calcs ...

' get parameter block for each event
For Each vDay In dpDays.Keys
    f1.Label4.Caption = vDay
    f1.Label4.Refresh

    Set oParam = dpDays.Item(vDay) ' get parameter block for
event

    ' get results array for products' CAS
    If Not dResults.Exists(oParam.oProd.CAS) Then
        ReDim aResults(1 To 365) As Single
        ReDim aHResults(1 To 365) As Single
        For n = 1 To 365
            aResults(n) = 0
            aHResults(n) = 0
        Next n
        dResults.Item(oParam.oProd.CAS) = Empty
    Else
        Dim vArray As Variant
        vArray = dResults.Item(oParam.oProd.CAS)

        aResults = vArray(0)
        aHResults = vArray(1)
    End If

```

```

        ' do exposure calculation
        Call clsme.Algorithm2(oParam) ' execute algorithm

        ' now do decay calcs
        If vDuringPost = eDuring Then
            aResults(vDay) = aResults(vDay) + oParam.rExposure
        Else
            Dim nDecay As Long
            For nDecay = vDay To 365
                aResults(nDecay) = aResults(nDecay) +
(oParam.rExposure * Decay(nDecay - vDay))
            Next nDecay
        End If
        aHResults(vDay) = oParam.rHand

        dResults.Item(oParam.oProd.CAS) = Array(aResults,
aHResults)
    Next vDay

    For Each vChem In dResults
        vArray = dResults.Item(oParam.oProd.CAS)
        aResults = vArray(0)
        aHResults = vArray(1)
        CARESOutput2New ors, aResults, aHResults, vPop, vChem,
dpDays, vScenario, vRoute, vDuringPost
    Next vChem

    dResults.RemoveAll

    End If      ' if No events for this person

    DoEvents
    If f1.bCancelled = True Then
        f1.Hide
        ExpCalcs2 = "014:F1xxxx>User Cancelled"
        Exit Function
    End If

    Next vPop      ' go onto the next person

    f1.Hide

    ExpCalcs2 = 0
End Function

Private Function CARESOutputHand(ors As adodb.Recordset, aResults() As
Single, vPop As Variant, vChem As Variant, dpDays As Dictionary, vScenario
As Variant, vRoute As Variant, vDuringPost As Variant) As Boolean
    Dim nDay As Long
    For nDay = 1 To 365
        Dim rExposure As Single
        rExposure = aResults(nDay)

        If rExposure <> 0 Then
            Dim O4 As adodb.Recordset
            Set O4 = ors

            O4.AddNew
            O4.Fields.Item("Cares ID").Value = vPop
            O4.Fields.Item("Day").Value = nDay
            O4.Fields.Item("CAS").Value = vChem
            O4.Fields.Item("Scenario").Value = vScenario
            If dpDays.Exists(nDay) Then
                O4.Fields.Item("Product").Value = dpDays.Item(nDay)
            Else
                O4.Fields.Item("Product").Value = ""
            End If
            O4.Fields.Item("Route").Value = vRoute
            O4.Fields.Item("DuringPost").Value = vDuringPost ' post
            O4.Fields.Item("Exposure").Value = rExposure
        End If
    Next nDay
End Function

```



```

        O4.Update

        DoEvents

    End If
Next nDay

    CARESOutputHand = True
End Function

Private Function CARESOutputNew(ors As adodb.Recordset, aResults As
Variant, vPop As Variant, vChem As Variant, dpDays As Dictionary,
vScenario As Variant, vRoute As Variant, vDuringPost As Variant) As
Boolean
    Dim nDay As Long
    For nDay = 1 To 365
        Dim rExposure As Single
        rExposure = aResults(nDay)

        If rExposure <> 0 Then
            Dim O4 As adodb.Recordset
            Set O4 = ors

            O4.AddNew
            O4.Fields.Item("Cares ID").Value = vPop
            O4.Fields.Item("Day").Value = nDay
            O4.Fields.Item("CAS").Value = vChem
            O4.Fields.Item("Scenario").Value = vScenario
            If dpDays.Exists(nDay) Then
                Dim oParam As clsParamBlock
                Set oParam = dpDays.Item(nDay)
                O4.Fields.Item("Product").Value = oParam.oProd.Name
            'dpDays.Item(nDay)
            Else
                O4.Fields.Item("Product").Value = ""
            End If
            O4.Fields.Item("Route").Value = vRoute
            O4.Fields.Item("DuringPost").Value = vDuringPost ' post
            O4.Fields.Item("Exposure").Value = rExposure
            O4.Update

            DoEvents

        End If
    Next nDay

    CARESOutputNew = True
End Function

Private Function CARESOutput2New(ors As adodb.Recordset, aResults As
Variant, aHResults As Variant, vPop As Variant, vChem As Variant, dpDays
As Dictionary, vScenario As Variant, vRoute As Variant, vDuringPost As
Variant) As Boolean
    Dim nDay As Long
    For nDay = 1 To 365
        Dim rExposure As Single
        rExposure = aResults(nDay)

        If rExposure <> 0 Then
            Dim O4 As adodb.Recordset
            Set O4 = ors

            O4.AddNew
            O4.Fields.Item("Cares ID").Value = vPop
            O4.Fields.Item("Day").Value = nDay
            O4.Fields.Item("CAS").Value = vChem
            O4.Fields.Item("Scenario").Value = vScenario
            If dpDays.Exists(nDay) Then
                Dim oParam As clsParamBlock
                Set oParam = dpDays.Item(nDay)
                O4.Fields.Item("Product").Value = oParam.oProd.Name

```

```

Else
    O4.Fields.Item("Product").Value = ""
End If
O4.Fields.Item("Route").Value = vRoute
O4.Fields.Item("DuringPost").Value = vDuringPost ' post
O4.Fields.Item("Exposure").Value = rExposure

If aHResults(nDay) <> 0 Then
    O4.Fields.Item("Hand").Value = aHResults(nDay)
End If

O4.Update

DoEvents

End If
Next nDay

CAREOutput2New = True
End Function

Public Function GetRunDescription(BaseDesc As String) As String
If Not SharedVariableExists("RunSpec", "all") Then
    GetRunDescription = BaseDesc & " (" & CStr(Now) & ")"
Else
    Dim d As Dictionary
    Set d = GetSharedVariable("RunSpec", "all")
    If Not d.Exists("RunSpec(Short)") Then
        GetRunDescription = BaseDesc & " (" & CStr(Now) & ")"
    Else
        Dim RunDesc As String
        RunDesc = d.Item("RunSpec(Short)")
        GetRunDescription = RunDesc & " (" & CStr(Now) & ")"
    End If
End If
End Function

Private Function GetWeights(r As adodb.Recordset) As Variant
Dim vWeight As Variant
ReDim vWeight(1 To 12)

Dim n As Integer
For n = 1 To 12 ' for fields Weight_1 to Weight_12
    Dim sFld As String
    sFld = "Weight_" & n
    vWeight(n) = r.Fields.Item(sFld).Value
Next n

GetWeights = vWeight
End Function

Private Function GetHeights(r As adodb.Recordset) As Variant
Dim vHeight As Variant
ReDim vHeight(1 To 12)

Dim n As Integer
For n = 1 To 12 ' for fields Height_1 to Height_12
    Dim sFld As String
    sFld = "Height_" & n
    vHeight(n) = r.Fields.Item(sFld).Value
Next n

GetHeights = vHeight
End Function

Private Function GetMonth(vDay As Variant)
Dim dEpoch As Date
dEpoch = CDate("Jan 1, 1990") ' the EPOCH ...

Dim dDay As Date
dDay = DateAdd("d", vDay, dEpoch) ' ... add vDays to epoch ...

```

```

        GetMonth = Month(dDay)           ' ... and get the month (1-12)
    End Function

Private Function InitProdDist(r As adodb.Recordset, oProd As clsProdInfo)
As Boolean
    ' since the parameters in data table are not what the ...
    ' ... nDistribution wants we have to map 'em to appropriate names

    Dim vBase As Variant
    For Each vBase In Array( _
        Array("Air Conc AI", "s1"), _
        Array("Amt Form Vol", "s2"), _
        Array("App AI Area Treat", "s3"), _
        Array("App Amt AI Used", "s4"), _
        Array("Frac AI Dislodge", "s5"), _
        Array("Unit Exp Dermal", "s6"), _
        Array("Unit Exp Inhalation", "s7"), _
        Array("Trans Res", "s8"))

        ' get distribution type
        Dim vType As Variant
        vType = r.Fields.Item(vBase(0) & " T").Value

        If Not IsNull(vType) And IsNumeric(vType) Then

            ' set distribution type
            oProd.InitializeDistribution CStr(vBase(1)), "Type", vType

            Dim vExt As Variant
            For Each vExt In Array( _
                Array("P1", "P1"), _
                Array("P2", "P2"), _
                Array("P3", "P3"), _
                Array("P4", "P4"))

                ' element 0 is the tail end of the ADO field name of the
parameter
                ' element 1 is a pseudo parameter used by the mapping
routine

                Dim sFld As String
                sFld = vBase(0) & " " & vExt(0)

                Dim vValue As Variant
                vValue = r.Fields.Item(sFld).Value

                If Not IsNull(vValue) And IsNumeric(vValue) Then
                    Dim sParamName As String

                    sParamName = GetDistParamName(CInt(vType),
CStr(vExt(1)))

                    oProd.InitializeDistribution CStr(vBase(1)),
sParamName, vValue
                End If

            Next vExt
        End If

    Next vBase

End Function

' this function maps pseudo parameter names to something that
nDistribution likes
Private Function GetDistParamName(nType As Integer, sPseudoParam As
String)
    Dim sRes As String

    Select Case nType

```

```

Case 0
  Select Case sPseudoParam
    Case "P1"
      sRes = "Default"
    Case Else
      MsgBox "Distribution type 0 takes only 1 parameter"
      sRes = ""
  End Select
Case 1
  Select Case sPseudoParam
    Case "P1"
      sRes = "Single"
    Case Else
      MsgBox "Distribution type 1 takes only 1 parameter"
      sRes = ""
  End Select
Case 2
  Select Case sPseudoParam
    Case "P1"
      sRes = "Param1" & "Mu"
    Case "P2"
      sRes = "Param2" & "Sigma"
    Case "P3"
      sRes = "Min"
    Case "P4"
      sRes = "Max"
  End Select
Case 3
  Select Case sPseudoParam
    Case "P1"
      sRes = "Param1" & "Zeta"
    Case "P2"
      sRes = "Param2" & "Sigma"
    Case "P3"
      sRes = "Min"
    Case "P4"
      sRes = "Max"
  End Select
Case 4
  Select Case sPseudoParam
    Case "P1"
      sRes = "Param1" & "Min"
    Case "P2"
      sRes = "Param2" & "Likely"
    Case "P3"
      sRes = "Param3" & "Max"
    Case Else
      MsgBox "Distribution type 4 takes only 3 parameters"
      sRes = ""
  End Select
Case 5
  Select Case sPseudoParam
    Case "P1"
      sRes = "Param1" & "Min"
    Case "P2"
      sRes = "Param2" & "Max"
    Case Else
      MsgBox "Distribution type 5 takes only 2 parameters"
      sRes = ""
  End Select
Case Else
End Select
GetDistParamName = sRes

End Function

Public Sub GetBNF(oPop As clsPopInfo, ByRef B As Double, ByRef C As
Double)

  Dim nSex As eSex
  nSex = oPop.nSex

```

```

Select Case oPop.nAge
  Case 0 To 2
    If nSex = eMale Then
      B = 0.249
      C = -0.127
    Else
      B = 0.244
      C = -0.13
    End If
  Case 3 To 9
    If nSex = eMale Then
      B = 0.095
      C = 2.11
    Else
      B = 0.085
      C = 2.033
    End If
  Case 10 To 17
    If nSex = eMale Then
      B = 0.074
      C = 2.754
    Else
      B = 0.056
      C = 2.898
    End If
  Case 18 To 29
    If nSex = eMale Then
      B = 0.063
      C = 2.896
    Else
      B = 0.062
      C = 2.036
    End If
  Case 30 To 59
    If nSex = eMale Then
      B = 0.048
      C = 3.653
    Else
      B = 0.034
      C = 3.538
    End If
  Case Is > 60
    If nSex = eMale Then
      B = 0.049
      C = 2.459
    Else
      B = 0.038
      C = 2.755
    End If
End Select

End Sub

' determine if calcs should be run for this algorithm/person/product
Private Function RunOK(n As NotitiaFunction, oPop As clsPopInfo, oProd As
clsProdInfo, vScenario, vRoute, vDuringPost) As Boolean
  Select Case LCase(n.FunctionName)
    Case "f11101", "f11102", "f13101", "f13102" ' for Der 101/102, Inh
101/102
      If oPop.nAge < 18 Then
        RunOK = False
        Exit Function
      End If

      If oProd.ProfCons = eProf Then
        RunOK = False
        Exit Function
      End If

    Case "f12107", "f12108", "f12109" ' for Ing 107/108/109

```

```
        If oPop.nAge < 1 Or oPop.nAge > 5 Then
            RunOK = False
            Exit Function
        End If

        Case Else
    End Select

    RunOK = True
End Function

'
' Get body area coefficients for surface area determination
Public Sub GetAreas(oPop As clsPopInfo, HandsU As Double, HandsC As
Double, UpperU As Double, UpperC As Double, LowerU As Double, LowerC As
Double, FeetU As Double, FeetC As Double)
    Dim nSex As eSex
    nSex = oPop.nSex

    Select Case nSex
        Case eMale
            HandsU = 5.2
            HandsC = 0

            UpperU = 13.7
            UpperC = 43.1

            LowerU = 12.7
            LowerC = 18.3

            FeetU = 7
            FeetC = 0
        Case eFemale
            HandsU = 5.1
            HandsC = 0

            UpperU = 13.4
            UpperC = 42.7

            LowerU = 12.8
            LowerC = 19.5

            FeetU = 6.5
            FeetC = 0
        Case Else
            NotitiaLog.MessageBox "Unknown sex <" & nSex & "> for ID <" &
oPop.sID & ">", "REx1(module1:GetAreas)", vbExclamation
    End Select
End Sub
```

2.25 Residential – Event Allocation

```

'
' Event Allocator (F15101)
'   Original Author: Muhilan Pandian
'

Option Explicit

Private g_dChem As Dictionary           ' for chemical selection
Private g_dScenarioProd As Dictionary   ' for scenario product info

' Public
'   Days, NumberScenarios
'   Cooccur(NumberScenarios,NumberScenarios), DayUse(Days,
NumberScenarios),
'   DayWeek(NumberScenarios,7), Season(NumberScenarios,12),
UseNo(NumberScenarios)
' Private
'   CountDayOfWeek, DayMinus, DayPlus, FlagDone, MaxUseNo, RandomCompare,
RandomValue, UseDay
'   CorProd(NumberScenario,NumberScenarios), DayMod(7,NumberScenarios),
'   ProbUse(Days,NumberScenarios), SeasMod(Days,NumberScenarios)
' Temporary
'   TotalTemp(NumberScenarios), Total
' Counters
'   CountDay, CountDayNorm, CountEffDay, CountProd, CountProdCon,
CountUse, Scenario,
'   Scenario1, Scenario2
Public Sub EventsAllocate(days As Long, _
                        NumberScenarios As Long, _
                        CoOccur() As Variant, _
                        DayWeek() As Variant, _
                        Season() As Variant, _
                        DayUse() As Variant, _
                        UseNo() As Variant, _
                        EffDaysUse() As Variant)

    Dim SeasMod() As Single
    Dim DayMod() As Single
    Dim CorProd() As Single
    Dim ProbUse() As Single 'Probability of product use
    Dim TotalTemp() As Single

    ReDim SeasMod(days, NumberScenarios)
    ReDim DayMod(7, NumberScenarios)
    ReDim CorProd(NumberScenarios, NumberScenarios)
    ReDim ProbUse(days, NumberScenarios)
    ReDim TotalTemp(NumberScenarios)

    ' Initialize SeasMod, DayMod, ProbUse, DayUse
    Dim Scenario As Variant
    For Scenario = 1 To NumberScenarios
        Dim CountDay As Variant
        For CountDay = 1 To days
            If (CountDay < 31) Then
                SeasMod(CountDay, Scenario) = Season(Scenario, 1) / 31
            ElseIf (CountDay < 59) Then
                SeasMod(CountDay, Scenario) = Season(Scenario, 2) / 28
            ElseIf (CountDay < 90) Then
                SeasMod(CountDay, Scenario) = Season(Scenario, 3) / 31
            ElseIf (CountDay < 120) Then
                SeasMod(CountDay, Scenario) = Season(Scenario, 4) / 30
            ElseIf (CountDay < 151) Then
                SeasMod(CountDay, Scenario) = Season(Scenario, 5) / 31
            ElseIf (CountDay < 181) Then
                SeasMod(CountDay, Scenario) = Season(Scenario, 6) / 30
            
```

```

ElseIf (CountDay < 212) Then
  SeasMod(CountDay, Scenario) = Season(Scenario, 7) / 31
ElseIf (CountDay < 243) Then
  SeasMod(CountDay, Scenario) = Season(Scenario, 8) / 31
ElseIf (CountDay < 273) Then
  SeasMod(CountDay, Scenario) = Season(Scenario, 9) / 30
ElseIf (CountDay < 304) Then
  SeasMod(CountDay, Scenario) = Season(Scenario, 10) / 31
ElseIf (CountDay < 334) Then
  SeasMod(CountDay, Scenario) = Season(Scenario, 11) / 30
ElseIf (CountDay < 365) Then
  SeasMod(CountDay, Scenario) = Season(Scenario, 12) / 31
End If
If (CountDay < 8) Then DayMod(CountDay, Scenario) =
DayWeek(Scenario, CountDay)
  ProbUse(CountDay, Scenario) = 1# / days
  DayUse(CountDay, Scenario) = -1
Next CountDay
Next Scenario

' Initialize CorProd
Dim Scenario1 As Variant

For Scenario1 = 1 To NumberScenarios
  Dim Scenario2 As Variant
  For Scenario2 = 1 To NumberScenarios
    CorProd(Scenario1, Scenario2) = CoOccur(Scenario1, Scenario2)
  Next Scenario2
Next Scenario1

' EventsAllocate-----
Randomize

'Apply modifiers: Season and Day of week
'StartDay = Day of week value for Jan 1
Dim StartDay As Variant
StartDay = Int((7# * Rnd) + 1)

Dim CountProd As Variant
For CountProd = 1 To NumberScenarios

  For CountDay = 1 To days
    ProbUse(CountDay, CountProd) = ProbUse(CountDay, CountProd) *
SeasMod(CountDay, CountProd)
    Dim CountDayOfWeek As Variant
    If (CountDay = 1) Then
      CountDayOfWeek = StartDay
    Else
      CountDayOfWeek = CountDayOfWeek + 1
    End If
    If (CountDayOfWeek = 8) Then CountDayOfWeek = 1
    ProbUse(CountDay, CountProd) = ProbUse(CountDay, CountProd) *
DayMod(CountDayOfWeek, CountProd)
  Next CountDay
Next CountProd

'Renormalize use probability
'Determine maximum value for no of uses across products (=MaxUseNo)
Dim MaxUseNo As Variant
MaxUseNo = 0
For CountProd = 1 To NumberScenarios
  If (UseNo(CountProd) > MaxUseNo) Then MaxUseNo = UseNo(CountProd)
  TotalTemp(CountProd) = 0#
  For CountDay = 1 To days
    TotalTemp(CountProd) = TotalTemp(CountProd) + ProbUse(CountDay,
CountProd)
  Next CountDay
Next CountProd
For CountProd = 1 To NumberScenarios
  For CountDay = 1 To days

```



```

        ProbUse(CountDay, CountProd) = ProbUse(CountDay, CountProd) /
TotalTemp(CountProd)
    Next CountDay
    Next CountProd

    'Loop on MaxUseNo
    Dim CountUse As Variant
    For CountUse = 1 To MaxUseNo
        For CountProd = 1 To NumberScenarios
            'Check for no of product use value not = 0
            If (UseNo(CountProd) <> 0) Then
                'Determine Day of use for product CountProd and other
products
                'Once Day of use is set, subtract 1 from UseNo
                'RandomCompare = Accumulator for checking against random
value
                'FlagDone = Flag to check if Day of use assigned (0=No,
1=Yes)
                'UseDay = For tracking assigned Day of use for concurrent
uses
                Dim RandomValue, RandomCompare, FlagDone
                RandomValue = Rnd
                RandomCompare = 0#
                FlagDone = 0
                'Determine Day of Use for product CountProd
                For CountDay = 1 To days
                    RandomCompare = RandomCompare + ProbUse(CountDay,
CountProd)
                    If (RandomValue <= RandomCompare And FlagDone = 0) Then
                        DayUse(CountDay, CountProd) = 0
                        UseNo(CountProd) = UseNo(CountProd) - 1

                        Dim UseDay As Variant
                        UseDay = CountDay
                        FlagDone = 1

                        'Renormalize use probability (ProbUse) for product
CountProd
                        Dim CountEffDay As Variant
                        For CountEffDay = 0 To EffDaysUse(CountProd)
                            Dim DayPlus, DayMinus
                            DayPlus = UseDay + CountEffDay
                            If (DayPlus <= days) Then
                                ProbUse(DayPlus, CountProd) =
ProbUse(DayPlus, CountProd) * CountEffDay / EffDaysUse(CountProd)
                            End If
                            DayMinus = UseDay - CountEffDay
                            If (DayMinus > 0) Then
                                ProbUse(DayMinus, CountProd) =
ProbUse(DayMinus, CountProd) * CountEffDay / EffDaysUse(CountProd)
                            End If
                        Next CountEffDay

                        Dim Total
                        Total = 0#

                        Dim CountDayNorm As Variant
                        For CountDayNorm = 1 To days
                            Total = Total + ProbUse(CountDayNorm, CountProd)
                        Next CountDayNorm
                        For CountDayNorm = 1 To days
                            ProbUse(CountDayNorm, CountProd) =
ProbUse(CountDayNorm, CountProd) / Total
                        Next CountDayNorm
                    End If
                Next CountDay

                'Determine concurrent uses for other products with product j
                Dim CountProdCon As Variant
                For CountProdCon = 1 To NumberScenarios 'CountProdCon =
Counter for concurrent product

```

```

        If (CountProdCon <> CountProd And _
            UseNo(CountProdCon) <> 0 And _
            ProbUse(UseDay, CountProdCon) > 0# And _
            Rnd <= CorProd(CountProdCon, CountProd)) Then
            DayUse(UseDay, CountProdCon) = 0
            UseNo(CountProdCon) = UseNo(CountProdCon) - 1
            'Renormalize use probability (ProbUse) for product
CountProdCon
            For CountEffDay = 0 To EffDaysUse(CountProdCon)
                DayPlus = UseDay + CountEffDay
                If (DayPlus <= days) Then
                    ProbUse(DayPlus, CountProdCon) =
ProbUse(DayPlus, CountProdCon) * CountEffDay / EffDaysUse(CountProdCon)
                End If
                DayMinus = UseDay - CountEffDay
                If (DayMinus > 0) Then
                    ProbUse(DayMinus, CountProdCon) =
ProbUse(DayMinus, CountProdCon) * CountEffDay / EffDaysUse(CountProdCon)
                End If
            Next CountEffDay
            Total = 0#
            For CountDayNorm = 1 To days
                Total = Total + ProbUse(CountDayNorm,
CountProdCon)
            Next CountDayNorm
            For CountDayNorm = 1 To days
                ProbUse(CountDayNorm, CountProdCon) =
ProbUse(CountDayNorm, CountProdCon) / Total
            Next CountDayNorm
        End If
    Next CountProdCon
    End If
Next CountProd
Next CountUse

End Sub
' Support Routines -----

Public Function InitializeOutputs(rst1 As ADO.DB.Recordset, rst3 As
ADO.DB.Recordset, Scenarios As Dictionary) As Boolean

    ' make a template RST for output
    Set rst1 = Nothing
    Set rst1 = New ADO.DB.Recordset
    rst1.Fields.Append "Cares Id", adVarChar, 50
    rst1.Fields.Append "Day", adInteger

    Dim vKey As Variant
    For Each vKey In Scenarios.Keys ' 'Key' = Scenario #
        If vKey <> "" Then
            Dim fld_name As String
            fld_name = Scenarios.Item(vKey)
            rst1.Fields.Append fld_name, adVarChar, 50
        End If
    Next vKey
    rst1.Open ' open rst for output...

    ' prepare for product generation
    Set g_dChem = GetChem() ' get selected chemicals
    If g_dChem Is Nothing Then
        NotitiaLog.MessageBox "No Chemical information available!",
"REx2(Module1:InitializeOutputs)", vbExclamation
        InitializeOutputs = False
        Exit Function
    End If

    Set g_dScenarioProd = Nothing
    Set g_dScenarioProd = New Dictionary
    For Each vKey In Scenarios.Keys ' 'Key' = Scenario #
        If vKey <> "" Then

```

```

        Dim dProd As Dictionary
        Set dProd = GetProductInfo(g_dChem, Key) ' get product info

        If dProd Is Nothing Then
            NotitiaLog.MessageBox "No Product information available
for scenario <" & Scenarios.Item(Key) & ">!",
"REx2(Module1:InitializeOutputs)", vbExclamation
        Else
            g_dScenarioProd.Add Key, dProd
        End If
    End If
Next Key
InitializeOutputs = True
End Function

Public Function InitializeConstantArrays(dScenarios As Dictionary, _
    ids As iDataset, _
    Settings As nSettings, _
    NumberScenarios As Long, _
    days As Long, _
    CoOccur() As Variant, _
    DayWeek() As Variant, _
    Season() As Variant, _
    UseNo() As Variant, _
    EffDaysUse() As Variant, _
    Optional Force As Boolean = False,
-
    Optional ReloadPeriod As Boolean =
True, _
    Optional ReloadUse As Boolean =
True) As Boolean

    Static Tag As Date ' if tag changes then we need to re-initialize the
static arrays

    Dim bForce As Boolean
    bForce = Force

    ' get 'tag' from dScenarios
    If dScenarios.Item("Tag") = Tag And Force = False Then
        InitializeConstantArrays = True ' no need to re-initialize
static arrays
        Exit Function
    Else
        bForce = True
    End If

    Tag = dScenarios.Item("Tag") ' update tag

    InitializeConstantArrays = False ' assume the worst

    ' redimension arrays...
    If ReloadPeriod = True Or bForce = True Then
        ReDim CoOccur(NumberScenarios, NumberScenarios)
        ReDim DayWeek(NumberScenarios, 7)
        ReDim Season(NumberScenarios, 12)

        ' fill arrays from db tables ...
        FillArrayFromTable1 Season, ids, "T15101-1", dScenarios, "Season"
        FillArrayFromTable1 DayWeek, ids, "T15101-2", dScenarios,
"DayWeek"
        FillArrayFromTable2 CoOccur, ids, "T15101-3", dScenarios,
"CoOccur"

        Module1.FillSettingsFromArrays Settings, dScenarios, Season,
DayWeek, CoOccur
    End If

    If ReloadUse = True Or bForce = True Then
        SetUseDefaults Settings, dScenarios, UseNo, EffDaysUse
    End If

```

```

        InitializeConstantArrays = True
    End Function

    Public Function InitializeStaticArrays(dScenarios As Dictionary, _
        ids As iDataset, _
        Settings As nSettings, _
        NumberScenarios As Long, _
        days As Long, _
        CoOccur() As Variant, _
        DayWeek() As Variant, _
        Season() As Variant, _
        DayUse() As Variant, _
        UseNo() As Variant, _
        EffDaysUse() As Variant, _
        Optional Force As Boolean = False)

    As Boolean

        Static Tag As Date ' if tag changes then we need to re-initialize the
        static arrays

        ' get 'tag' from dScenarios

        Dim bForce As Boolean
        bForce = Force
        If dScenarios.Item("Tag") = Tag And Force = False Then
            InitializeStaticArrays = True ' no need to re-initialize static
arrays
            Exit Function
        Else
            bForce = True
        End If

        Tag = dScenarios.Item("Tag") ' update tag

        InitializeStaticArrays = False ' assume the worst

        ' redimension arrays...
        ReDim DayUse(days, NumberScenarios) ' output array?
        ReDim UseNo(NumberScenarios)
        ReDim EffDaysUse(NumberScenarios)

        ' reinitialise use array from settings
        FillUseArraysFromSettings Settings, dScenarios, UseNo, EffDaysUse

        InitializeStaticArrays = True
    End Function

    Public Function FillArrayFromMatrix(Settings As nSettings, MatAlias As
    String, dst()) As Boolean
        Dim rs As Long, CS As Long, rd As Long, cd As Long

        FillArrayFromMatrix = False ' assume the worst

        ' get setting matrix...
        Dim sm As nSetting
        Set sm = Settings.Setting(MatAlias)

        If sm.GetType() <> "matrix" Then Exit Function
        rs = sm.GetParam("Rows")
        CS = sm.GetParam("Cols")

        If Not IsArray(dst) Then Exit Function
        rd = UBound(dst, 1) - LBound(dst, 1) + 1 - 1 ' because option base
= 0
        cd = UBound(dst, 2) - LBound(dst, 2) + 1 - 1 ' because option base
= 0

        ' is/are src/dst the same dimension...

```

```

If rs <> rd Or CS <> cd Then Exit Function

Dim r As Long, C As Long
For r = 1 To rs
    For C = 1 To CS
        dst(r, C) = sm.Value(r:=r, C:=C)    ' copy src => dst
    Next C
Next r

FillArrayFromMatrix = True    ' assume the worst

End Function

Public Function FillArrayFromObjectMatrix1d(Settings As nSettings,
MatAlias As String, dst()) As Boolean
    Dim rs As Long, CS As Long, rd As Long, cd As Long

    FillArrayFromObjectMatrix1d = False    ' assume the worst

    ' get setting matrix...
    Dim sm As nSetting
    Set sm = Settings.Setting(MatAlias)

    If sm.GetType() <> "matrix" Then Exit Function
    rs = sm.GetParam("Rows")
    CS = sm.GetParam("Cols")

    If Not IsArray(dst) Then Exit Function
    rd = UBound(dst, 1) - LBound(dst, 1) + 1 - 1    ' because option base
= 0
    cd = 1    ' 1 1d matrix has 1 col

    ' is/are src/dst the same dimension...
    If rs <> rd Or CS <> cd Then Exit Function

    Dim r As Long, C As Long
    For r = 1 To rs
        dst(r) = sm.Value(r:=r, C:=1).Value()    ' copy src => dst
    Next r

    FillArrayFromObjectMatrix1d = True    ' assume the worst

End Function

' get all possible scenarios from Meta file .
Public Function GetMetadata(ids As iDataset) As Boolean
    GetMetadata = False    ' just in case

    ' get input dataset to initialize arrays from
    Dim dl As iLibrarian
    Set dl = GetSystemLibrary()    ' get data library

    Dim C As Collection
    Set C = dl.FindMeta("s15101", , True)
    If C.Count <> 1 Then
        MsgBox "Error finding Meta for F15101...", vbInformation,
"EventAllocation1(GetMetaData)"
        Exit Function
    End If

    Dim c2
    c2 = Split(C.Item(1), "|")

    Dim Refid As String
    Refid = c2(LibExField.Refid)

    Set ids = dl.OpenDataset(Refid)

    GetMetadata = True
End Function

```

```

Public Function WriteOutput(Cares_ID As String, _
                           dScenarios As Dictionary, _
                           Settings As nSettings, _
                           rst1 As ADODB.Recordset, _
                           rst2 As ADODB.Recordset, _
                           rst3 As ADODB.Recordset, _
                           rst4 As ADODB.Recordset, _
                           rst6 As ADODB.Recordset, _
                           days As Long, _
                           CoOccur() As Variant, _
                           DayWeek() As Variant, _
                           Season() As Variant, _
                           DayUse() As Variant, _
                           UseNo() As Variant, _
                           EffDaysUse() As Variant, _
                           dProb As Dictionary) As Boolean

    WriteOutput = False

    Write1 rst1, Cares_ID, dScenarios, dProb, DayUse, days, rst6

    WriteOutput = True
End Function

Public Sub Write1(rst As ADODB.Recordset, id As String, Scenarios As
Dictionary, Prob As Dictionary, DayUse() As Variant, days As Long, rst6 As
ADODB.Recordset)

    '
    ' if recordset already has records in it then just overwrite the
existing data
    ' else creat a new entry
    '
    Dim vDayUse() As Variant
    vDayUse = DayUse

    Dim o1 As ADODB.Recordset
    Set o1 = rst

    Dim o6 As ADODB.Recordset
    Set o6 = rst6

    Dim i%
    For i = 1 To days
        '1st turn off any scenarios that the id won't be using
        Dim nFld, N
        N = 1

        For Each nFld In Scenarios.Keys
            If nFld <> "*" Then ' ignore Tag field
                If Prob.Exists(id) Then
                    Dim bUse As Boolean
                    Dim dUse As Dictionary
                    Set dUse = Prob.Item(id)
                    If Not dUse.Exists(nFld) Then
                        Err.Raise 42, , "Scenario " & nFld & " non extant
in selected P() table!!"
                    Else
                        bUse = dUse.Item(nFld)
                    End If
                    If vDayUse(i, N) = 0 Then ' EA sez to use it
                        If bUse = False Then ' scenario P() sez No
                            vDayUse(i, N) = -1
                        End If
                    End If
                Else
                    Err.Raise 42, , "ID " & id & " non extant in
Write1:Prob!!!!"
                End If
                N = N + 1
            End If
        End If
    End If

```

```

        Next nFld
    Next i

    'dim i%
    For i = 1 To days
        '2nd determine if record is all -1's. If so, don't write it!
        N = 1

        Dim nSum As Integer
        nSum = 0
        For Each nFld In Scenarios.Keys
            If nFld <> "*" Then ' ignore Tag field
                nSum = nSum + vDayUse(i, N)
                N = N + 1
            End If
        Next nFld

        If nSum <> -(Scenarios.Count - 1) Then

            o1.AddNew
            o1.Fields.Item("Cares id").Value = id

            o1.Fields.Item("Day").Value = i

            N = 1
            For Each nFld In Scenarios.Keys
                If nFld <> "*" Then

                    If vDayUse(i, N) <> -1 Then
                        ' now generate a random product

                        ' 1st get list of products for this scenario
                        Dim dProdInfo As Dictionary
                        Set dProdInfo = g_dScenarioProd.Item(nFld)
                        If dProdInfo.Count > 0 Then
                            ' 2nd now get a random product suitable for
this scenario

                            Dim oPi As clsProdInfo
                            Set oPi = GetRndProduct(CLng(nFld), dProdInfo)

                            o1.Fields.Item(Scenarios.Item(nFld)).Value =
oPi.ProductId

                            o6.AddNew
                            o6.Fields.Item("Cares id").Value = id
                            o6.Fields.Item("Day").Value = i
                            o6.Fields.Item("Scenario").Value = CInt(nFld)
                            o6.Fields.Item("Product").Value =
oPi.ProductId

                            o6.Fields.Item("CAS").Value = oPi.CAS

                        Else
                            o1.Fields.Item(Scenarios.Item(nFld)).Value =
"?"

                        End If
                    Else
                        o1.Fields.Item(Scenarios.Item(nFld)).Value =
CStr(vDayUse(i, N))
                    End If

                    N = N + 1
                End If
            Next nFld

            End If
        Next i

        Set o1 = Nothing
    End Sub

```

```

Private Function FillArrayFromTable1(vArray() As Variant, ids As iDataset,
Table As String, dScenarios As Dictionary, Name As String) As Boolean
' fill arrays from db tables ...
Dim id As iData
Dim r As ADODB.Recordset

' fill Season from T15101-1
Set id = ids.OpenTable(Table)
If id Is Nothing Then
MsgBox "Oops...", vbInformation, "REx2(FillArrayFromTable:" &
Name & "1)"
End If
Set r = id.OpenRST()
If r Is Nothing Then
MsgBox "Oops..." & vbCrLf & id.LastErrorMessage, vbInformation,
"REx2(FillArrayFromTable:" & Name & "2)"
End If

Dim N As Integer
N = 1
Dim Key As Variant
For Each Key In dScenarios.Keys ' go through all selected
scenarios
Dim Scenario As String
Scenario = dScenarios.Item(Key)
If Key <> "*" Then
r.Filter = "Scenario = '" & Scenario & "'"

Select Case r.RecordCount
Case 0
MsgBox "Oops... Scenario <" & Scenario & "> Not
found!", vbInformation, "REx2(FillArrayFromTable:" & Name & "3)"
Case 1 ' fall through
Case Else
MsgBox "Oops... recordcount=" & r.RecordCount,
vbInformation, "REx2(FillArrayFromTable:" & Name & "4)"
End Select

Dim n2 As Integer
For n2 = 1 To UBound(vArray, 2)
vArray(N, n2) = r.Fields.Item(n2).Value
Next n2
N = N + 1
End If

Next Key

Set r = Nothing
Set id = Nothing
End Function

Private Function FillArrayFromTable2(vArray() As Variant, ids As iDataset,
Table As String, dScenarios As Dictionary, Name As String) As Boolean
' fill arrays from db table ... selectively
Dim id As iData
Dim r As ADODB.Recordset

' fill Season from T15101-1
Set id = ids.OpenTable(Table)
If id Is Nothing Then
MsgBox "Oops...", vbInformation, "REx2(FillArrayFromTable:" &
Name & "1)"
End If
Set r = id.OpenRST()
If r Is Nothing Then
MsgBox "Oops..." & vbCrLf & id.LastErrorMessage, vbInformation,
"REx2(FillArrayFromTable:" & Name & "2)"
End If

Dim N As Integer

```



```

N = 1
Dim Key As Variant
For Each Key In dScenarios.Keys          ' go through all selected
scenarios
    Dim Scenario As String
    Scenario = dScenarios.Item(Key)
    If Key <> "*" Then
        r.Filter = "Scenario = '" & Scenario & "'"

        Select Case r.RecordCount
            Case 0
                MsgBox "Oops... Scenario <" & Scenario & "> Not
found!", vbInformation, "REx2(FillArrayFromTable:" & Name & "3)"
            Case 1
                ' fall through
            Case Else
                MsgBox "Oops... recordcount=" & r.RecordCount,
vbInformation, "REx2(FillArrayFromTable:" & Name & "4)"
        End Select

        ' we want to fill from fields that are only in dScenarios
        Dim Key2, n2
        n2 = 1
        For Each Key2 In dScenarios.Keys
            If Key2 <> "*" Then
                Dim Item2 As String
                Item2 = dScenarios.Item(Key2)
                vArray(N, n2) = r.Fields.Item(Item2).Value
                n2 = n2 + 1
            End If
        Next Key2
        N = N + 1
    End If

Next Key

Set r = Nothing
Set id = Nothing
End Function

Public Sub FillSettingsFromArrays(Settings As nSettings, Scenarios As
Dictionary, Season() As Variant, DayWeek() As Variant, CoOccur() As
Variant)
    Settings.SetParam "m1", "|Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep,
Oct, Nov, Dec", "Values"
    Settings.SetParam "m2", "|Sun, Mon, Tue, Wed, Thu, Fri, Sat", "Values"

    ' dimension m1 (Season)
    Settings.SetParam "m1", Scenarios.Count - 1, "Rows"
    Settings.SetParam "m1", 12, "Cols"

    ' dimension m2 (Day of week)
    Settings.SetParam "m2", Scenarios.Count - 1, "Rows"
    Settings.SetParam "m2", 7, "Cols"

    ' dimension m3 (coOccur)
    Settings.SetParam "m3", Scenarios.Count - 1, "Rows"
    Settings.SetParam "m3", Scenarios.Count - 1, "Cols"

    Dim Key, Item, v, r, C

    ' fill m1 with data
    For C = 0 To 12
        'For r = 1 To Scenarios.Count - 1
        r = 1
        For Each Key In Scenarios.Keys
            If Key <> "*" Then
                If C = 0 Then
                    v = Scenarios.Item(Key)
                Else
                    v = Season(r, C)
                End If
            End If
        Next Key
    Next C

```

```

        Settings.Setting("m1").Value r:=r, C:=C, value1:=v
        r = r + 1
    End If
'Next r
Next Key
Next C

' fill m2 with defaults
For C = 0 To 7
    r = 1
    For Each Key In Scenarios.Keys
        If Key <> "*" Then

            If C = 0 Then
                v = Scenarios.Item(Key)
            Else
                v = DayWeek(r, C)
            End If
            Settings.Setting("m2").Value r:=r, C:=C, value1:=v
            r = r + 1
        End If
    Next Key
Next C

' fill m3 with defaults
For C = 0 To Scenarios.Count - 1 'list1.ListCount
    r = 1
    For Each Key In Scenarios.Keys
        If Key <> "*" Then
            If C = 0 Then
                v = Scenarios.Item(Key)
                Settings.Setting("m3").Value r:=C, C:=r, value1:=v
            Else
                v = CoOccur(r, C)
            End If
            Settings.Setting("m3").Value r:=r, C:=C, value1:=v
            r = r + 1
        End If
    Next Key
Next C

End Sub

Public Sub FillArraysFromSettings(Settings As nSettings, Scenarios As
Dictionary, Season() As Variant, DayWeek() As Variant, CoOccur() As
Variant)
    Dim r, C, v

    ' update Season (m1)
    For r = 1 To Settings.Setting("m1").GetParam("Rows")
        For C = 1 To Settings.Setting("m1").GetParam("Cols")
            v = Settings.Setting("m1").Value(r:=r, C:=C)
            Season(r, C) = v
        Next C
    Next r

    ' update DayWeek (m2)
    For r = 1 To Settings.Setting("m2").GetParam("Rows")
        For C = 1 To Settings.Setting("m2").GetParam("Cols")
            v = Settings.Setting("m2").Value(r:=r, C:=C)
            DayWeek(r, C) = v
        Next C
    Next r

    ' update CoOccur (m3)
    For r = 1 To Settings.Setting("m3").GetParam("Rows")
        For C = 1 To Settings.Setting("m3").GetParam("Cols")
            v = Settings.Setting("m3").Value(r:=r, C:=C)
            CoOccur(r, C) = v
        Next C
    Next r

```

```

End Sub

Public Function GetPop(dPop As Dictionary) As Boolean
    GetPop = False

    Dim vRefPop As Variant

    Dim vGid As Variant
    If SharedVariableExists("InputPop", "Group") = False Then
        MsgBox "Group id not found? Have you selected a population file?",
vbExclamation, "No population group found!"
        Exit Function
    Else
        vGid = GetSharedVariable("InputPop", "Group") ' open population
group
    End If

    Dim ids As iDataset
    Set ids = GetSystemLibrary.OpenDataset(CStr(vGid))

    dPop.RemoveAll

    Dim r As ADODB.Recordset
    Dim id As iData

    ' we need to go through the catalog for the dataset and find
    ' a table of type t16101, This will be the table we want to use
    ' we'll assume (for now) that there is only 1 pop table per file
    Dim sTableName As String
    Dim dCat As Dictionary
    Set dCat = ids.GetCatalog

    Dim vItem
    For Each vItem In dCat.Keys
        Dim dItem As Dictionary
        Set dItem = dCat.Item(vItem)
        If LCase(dItem.Item("tabletype")) = "t16101" Then
            sTableName = CStr(vItem)
            Exit For
        End If
    Next vItem
    If sTableName = "" Then
        Err.Raise 42, , "No pop table!!!!"
    End If

    Set id = ids.OpenTable(sTableName)

    If id Is Nothing Then
        MsgBox Err.Description, vbCritical, "EventAllocator(GetPopulation
- INTERNAL ERROR)"
        Exit Function
    End If

    Set r = id.Recordset()

    Do While Not r.EOF
        Dim Cares_ID As String
        Cares_ID = r.Fields.Item("1").Value()

        If Not dPop.Exists(Cares_ID) Then
            dPop.Add Cares_ID, Empty
        Else
            MsgBox "Cares_id=" & Cares_ID & " already exists in
collection", vbCritical, "EventAllocator(GetPopulation - INTERNAL ERROR)"
        End If
        r.MoveNext

        DoEvents
    Loop

```

```

        r.Close
        Set id = Nothing
        Set ids = Nothing

        GetPop = True
    End Function

' determine scenario P() for each person in selected population
Public Function GetP(dProb As Dictionary, dScenarios As Dictionary, dPop
As Dictionary) As Boolean
    GetP = False      ' assume the worst

    Randomize

    If SharedVariableExists("InputRes", "Probabilities") = False Then
        MsgBox "Scenario P() data not found? Have you selected a scenario
P() file?", vbExclamation, "No scenario probability file found!"
        Exit Function
    End If

    ' now read scenario P() file (if available)
    Dim vP As Variant
    vP = GetSharedVariable("InputRes", "Probabilities")
    Dim sRefid As String
    Dim sTableName As String
    sRefid = vP(0)
    sTableName = vP(1)

    Dim dP As New Dictionary      ' for 'raw' Scenario P()'s
    dP.RemoveAll

    Dim ids As iDataset
    Set ids = GetSystemLibrary().OpenDataset(sRefid)
    If ids Is Nothing Then
        Err.Raise 42, , "GetP() - INTERNAL ERROR - Cannot open RefID=" &
sRefid
    End If

    Dim id As iData
    Set id = ids.OpenTable(sTableName, , , adUseServer)
    If id Is Nothing Then
        Err.Raise 42, , "GetP() - INTERNAL ERROR - Cannot open Table=" &
sTableName & "/" & sRefid
    End If

    Dim r As ADODB.Recordset
    Set r = id.Recordset
    Do While Not r.EOF
        Dim nScenario As Long
        nScenario = r.Fields.Item("Scenario #").Value
        Dim nP As Double
        nP = r.Fields.Item("P").Value

        If dP.Exists(nScenario) Then
            MsgBox "Duplicate Scenario (" & nScenario & ") found in P()
table (" & sTableName & ")", vbExclamation, "Duplicate Scenario # in P()
table!"
            Exit Function
        End If

        If dScenarios.Exists(nScenario) Then      ' only add to list if it a
selected scenario
            dP.Add nScenario, nP
        End If
        r.MoveNext
    Loop
    r.Close
    Set id = Nothing
    Set ids = Nothing

    ' now for each person iterate through all selected scenarios

```

```

' and determine if the scenario is applicable for the person

dProb.RemoveAll      ' clear out result(s) list

Dim vPop As Variant
For Each vPop In dPop.Keys
  Dim sID As String
  sID = vPop          ' cares id

  Dim dUse As Dictionary
  Set dUse = New Dictionary

  Dim vScen As Variant
  For Each vScen In dP.Keys
    nScenario = vScen      ' scenario #
    nP = dP.Item(vScen)    ' scenario P()

    Dim nRnd As Double
    nRnd = Rnd              ' get a "random" # [0..1)

    Dim bUse As Boolean
    If nRnd <= nP Then
      bUse = True
    Else
      bUse = False
    End If
    If dUse.Exists(nScenario) Then
      Err.Raise 42, , "GetP() - INTERNAL ERROR - Duplice
scenarios in dUse !!!"
    Else
      dUse.Add nScenario, bUse
    End If
  Next vScen

  ' add id to scenario use list
  If dProb.Exists(vPop) Then
    Err.Raise 42, , "GetP() - INTERNAL ERROR - Duplice population
ID found (" & vPop & ")"
  Else
    dProb.Add vPop, dUse
  End If
Next vPop

GetP = True
End Function

Public Function GetRunDescription(BaseDesc As String) As String
  If Not SharedVariableExists("RunSpec", "all") Then
    GetRunDescription = BaseDesc & " (" & CStr(Now) & ")"
  Else
    Dim d As Dictionary
    Set d = GetSharedVariable("RunSpec", "all")
    If Not d.Exists("RunSpec(Short)") Then
      GetRunDescription = BaseDesc & " (" & CStr(Now) & ")"
    Else
      Dim RunDesc As String
      RunDesc = d.Item("RunSpec(Short)")
      GetRunDescription = RunDesc & " (" & CStr(Now) & ")"
    End If
  End If
End Function

'
'=====
'
Public Sub FillSettingsFromUseArrays(Settings As nSettings, Scenarios As
Dictionary, UseNo() As Variant, EffDaysUse() As Variant)
  Dim Key, Item, v, r, C

```

```

Dim sCol As String      ' column header string
sCol = ""

For Each Key In Scenarios
  If Key <> "*" Then
    If sCol = "" Then
      sCol = Scenarios.Item(Key)
    Else
      sCol = sCol & "," & Scenarios.Item(Key)
    End If
  End If
Next Key

' dimension m4 (UseNo)
Settings.SetParam "m4", 1, "Cols"
Settings.SetParam "m4", Scenarios.Count - 1, "Rows"

' dimension m5 (EffDaysUse)
Settings.SetParam "m5", 1, "Cols"
Settings.SetParam "m5", Scenarios.Count - 1, "Rows"

Settings.SetParam "m4", "|Uses\" & sCol, "Values"
Settings.SetParam "m5", "|Days\" & sCol, "Values"

' fill m4 with data
For r = 1 To Scenarios.Count - 1
  v = UseNo(r)
  Settings.Setting("m4").Value r:=r, C:=1, value1:=v
Next r

' fill m5 with data
For r = 1 To Scenarios.Count - 1
  v = EffDaysUse(r)
  Settings.Setting("m5").Value r:=r, C:=1, value1:=v
Next r

End Sub

Public Sub FillUseArraysFromSettings(Settings As nSettings, Scenarios As
Dictionary, UseNo() As Variant, EffDaysUse() As Variant)
  Dim r, C, v

  ' update UseNo (m4)
  For r = 1 To Settings.Setting("m4").GetParam("Rows")
    v = Settings.Setting("m4").Value(r:=r, C:=1)
    UseNo(r) = v
  Next r

  ' update EffDaysUse (m5)
  For r = 1 To Settings.Setting("m5").GetParam("Rows")
    v = Settings.Setting("m5").Value(r:=r, C:=1)
    EffDaysUse(r) = v
  Next r

End Sub

Private Sub SetUseDefaults(Settings As nSettings, Scenarios As Dictionary,
UseNo() As Variant, EffDaysUse() As Variant)
  ' initialize uses to some default
  ReDim UseNo(Scenarios.Count - 1)
  ReDim EffDaysUse(Scenarios.Count - 1)

  Dim N As Long
  For N = 1 To Scenarios.Count - 1
    UseNo(N) = 6          ' 6 times / year
    EffDaysUse(N) = 30   ' 30 days effective
  Next N
  FillSettingsFromUseArrays Settings, Scenarios, UseNo, EffDaysUse
End Sub

```

```

Public Function GetRndProduct(nScenario As Long, dProdInfo As Dictionary)
As clsProdInfo
    Dim dProd As New Dictionary
    dProd.RemoveAll

    Dim AccMkt As Double          ' accumulated Market P()'s
    AccMkt = 0

    ' 1st find all products available for this scenario
    Dim vProd As Variant
    For Each vProd In dProdInfo.Keys
        Dim oPi As clsProdInfo
        Set oPi = dProdInfo.Item(vProd)

        dProd.Add vProd, oPi.Market

        AccMkt = AccMkt + oPi.Market
    Next vProd

    If dProd.Count = 0 Then
        NotitiaLog.MessageBox "dProd.Count=0",
"REX1(Module1:GetRndProduct2) - INTERNAL ERROR"
        Set GetRndProduct = Nothing ' no prods with associated CAS
        Exit Function
    End If

    ' now we have a list of compatible products and their P()'s
    ' generate a rnd# and pick a product according to market share
    Dim nRnd As Double
    nRnd = Rnd                    ' get a random # [0..1)

    Dim AccumP As Double         ' accumulated Market P()
    AccumP = 0

    For Each vProd In dProd.Keys
        Dim nMarket As Double
        nMarket = dProd.Item(vProd)
        AccumP = AccumP + nMarket

        If nRnd < (AccumP / AccMkt) Then
            Set GetRndProduct = dProdInfo.Item(vProd)
            Exit Function
        End If
    Next vProd

    'if we get here then what happened?
    Set GetRndProduct = Nothing 'Empty
End Function

Public Function GetProductInfo(g_dChem As Dictionary, vScenario As
Variant) As Dictionary
    Dim dProd As Dictionary

    Dim vA As Variant
    If SharedVariableExists("InputRes", "Products") = False Then
        MsgBox "Please select a product file?", vbExclamation, "No product
filefound!"
        Exit Function
    Else
        vA = GetSharedVariable("InputRes", "Products") ' open population
group
    End If

    Dim vGid As Variant
    Dim vName As Variant
    vGid = vA(0)          ' file refid
    vName = vA(1)        ' table name

    Dim ids As iDataset
    Set ids = GetSystemLibrary.OpenDataset(CStr(vGid))

```

```

Dim r As ADODB.Recordset
Dim id As iData

Set id = ids.OpenTable(CStr(vName))

If id Is Nothing Then
    MsgBox Err.Description, vbCritical, "REx-1(GetProduct - INTERNAL
ERROR)"
    Exit Function
End If

Set dProd = New Dictionary

Set r = id.Recordset()
Do While Not r.EOF
    Dim ProductId, Name, CAS, Form, WtFrac, Scenario, ProfCons, Market

    Scenario = r.Fields.Item("Scenario").Value

    If vScenario = Scenario Then
        Dim oProdInfo As clsProdInfo
        Set oProdInfo = New clsProdInfo

        oProdInfo.ProductId = r.Fields.Item("ProductID").Value
        oProdInfo.Name = r.Fields.Item("Name").Value
        oProdInfo.CAS = r.Fields.Item("CAS").Value
        oProdInfo.Form = r.Fields.Item("Form").Value
        oProdInfo.WtFrac = r.Fields.Item("Wt Frac").Value
        oProdInfo.Scenario = r.Fields.Item("Scenario").Value
        oProdInfo.ProfCons = r.Fields.Item("Prof/Cons").Value
        oProdInfo.Market = r.Fields.Item("Market").Value

        ProductId = oProdInfo.ProductId
        CAS = oProdInfo.CAS

        If dProd.Exists(ProductId) Then
            Err.Raise 42, , "Duplicate Product ID!!!!!"
        End If

        ' only add if chemical has been selected .. and if appropriate
scenario
        If g_dChem.Exists(CAS) Then
            dProd.Add ProductId, oProdInfo
        End If

    End If

    r.MoveNext
Loop

r.Close
Set id = Nothing
Set ids = Nothing

Set GetProductInfo = dProd
End Function

Public Function GetChem() As Dictionary
    Dim dSel As Dictionary
    If SharedVariableExists("InputChem", "all") = False Then
        MsgBox "Please select desired Chemical/CAS#'s", vbExclamation, "No
Chemical/CAS# selection founc!"
        Exit Function
    Else
        Set dSel = GetSharedVariable("InputChem", "all") ' get selected
chemicals
    End If

    Dim g_dChem As Dictionary
    Set g_dChem = New Dictionary

```



```

    Dim N As Long
    N = 0

    Dim vCAS As Variant
    For Each vCAS In dSel.Keys
        If vCAS <> "*" Then
            N = N + 1
            g_dChem.Add vCAS, dSel.Item(vCAS)           ' CAS, Chemical name
        End If
    Next vCAS

    Set GetChem = dSel
End Function

' Notitia Function Wrapper-----
-----
'
' F15101 = REX event allocator
'
' 2/8/2001 - HWR - The Epoch
' 5/2/2001 - HWR - Folded into REX 2
' 7/5/2001 - HWR - Much stuff added
'

Option Explicit

Implements NotitiaFunction

Private Inputs As iIOArray
Private outputs As iIOArray
Public Settings As nSettings

Private sID As String

Private sim As Long

Private iuf As iUserFile

Private or2 As iRecordset
Private or4 As iRecordset
Private or6 As iRecordset

Private out1 As ADODB.Recordset
Private out2 As ADODB.Recordset
Private out3 As ADODB.Recordset
Private out4 As ADODB.Recordset
Private out6 As ADODB.Recordset

Public ids As iDataset           ' Meta file dataset...

Public Scenarios As Dictionary
Public dScenarios As Dictionary
Public dPop As New Dictionary
Public dProb As New Dictionary   ' scenario use list

Private NumberScenarios As Long
Public days As Long

Private CoOccur() As Variant
Private DayUse() As Variant
Private DayWeek() As Variant
Private Season() As Variant
Private UseNo() As Variant
Private EffDaysUse() As Variant

Private f1 As New frm15101

Private Sub Class_Initialize()
    Set Inputs = New iIOArray
    Set outputs = New iIOArray

```

```

Set Settings = New nSettings

Settings.Compatibility = NotitiaFunction_FunctionType()
sID = GetGuid() 'getguid()

sim = 0

' initialize settings

' statics
Settings.AddLocal "m1", "Seasonal Use", "matrix"
Settings.AddLocal "m2", "Day of Week Use", "matrix"
Settings.AddLocal "m3", "Co-Occurrence", "matrix"

' invisibles
Settings.AddLocal "m4", "Number of Uses", "matrix", True
Settings.AddLocal "m5", "Efficacy Period", "matrix", True

Settings.AddLocal "c1", "Days", "constant"
Settings.SetParam "c1", 365, "Value"

' initialize inputs
Inputs.Add "inpl", "*", "*", Nothing, "(Scenarios)", "Selected
scenarios"

Module1.GetMetadata ids          ' get metadata for F15101

' see of Scenarios variable has been set...
If SharedVariableExists("InputRes", "scenarios") = True Then
    Set dScenarios = GetSharedVariable("inputRes", "scenarios")

    NumberScenarios = dScenarios.Count - 1

    Module1.InitializeConstantArrays dScenarios, _
                                     ids, _
                                     Settings, _
                                     NumberScenarios, _
                                     days, _
                                     CoOccur, _
                                     DayWeek, _
                                     Season, _
                                     UseNo, _
                                     EffDaysUse, _
                                     True, True ' force
(re)initialization
'Else
'MsgBox "dScenarios not found!", vbInformation, "F15101..."
End If

' initialize outputs
Set iuf = New iUserFile
iuf.Create "s15101"
iuf.Name = "Output CARES Event Allocator (" & CStr(Now) & ")"

iuf.Description = GetRunDescription("CARES Event Allocator")
iuf.Disposition = "output"

Set or6 = iuf.GetIRST("t15101-6")
Set out6 = or6.Recordset()
outputs.Add "out1", "rst-1", "DayUse", Nothing, "S15101(T15101)",
"F15101 DayUse data" ' , True
outputs.Add "out6", "iuf-1", "EA Output", iuf, "-NO-TYPE-", "F15101
output data"

SetSharedVariable "Events", "DayUse", Nothing

days = 365
End Sub

Private Sub Class_Terminate()

```

```

On Error GoTo Oops

Set Inputs = Nothing
Set outputs = Nothing
Set Settings = Nothing

Set out1 = Nothing
Set out2 = Nothing
Set out3 = Nothing
Set out4 = Nothing
Set out6 = Nothing

Set or2 = Nothing
Set or4 = Nothing
Set or6 = Nothing      ' the only one that is being used
Set dScenarios = Nothing
Exit Sub

Oops:
MsgBox "Error terminating REx2:F15101", vbInformation, Err.Description
Err.Clear
End Sub

Private Sub NotitiaFunction_About()
    DoAbout Me
End Sub

Private Function NotitiaFunction_AlterSettings(NewSettings As nSettings)
As Boolean
    NotitiaFunction_AlterSettings = True
End Function

Private Function NotitiaFunction_APIComments() As String
    NotitiaFunction_APIComments = ""      ' nothing special
End Function

Private Function NotitiaFunction_APIVersion() As String
    NotitiaFunction_APIVersion = "0.1.3"
End Function

Private Function NotitiaFunction_Category() As String
    NotitiaFunction_Category = "1"
End Function

Private Function NotitiaFunction_CheckInputs() As Boolean
    NotitiaFunction_CheckInputs = True   ' assume OK
    ' we need 1 input at this time

    Dim obj As Variant
    ' check for scenarios dictionary
    Inputs.FindByInternalAlias "inpl", obj
    If IsEmpty(obj) Then
        NotitiaFunction_CheckInputs = False
        Exit Function
    End If
End Function

Private Function NotitiaFunction_CheckSettings() As Boolean
    NotitiaFunction_CheckSettings = True
End Function

Private Function NotitiaFunction_CopyRight() As String
    NotitiaFunction_CopyRight = ""
End Function

Private Function NotitiaFunction_FunctionClassification() As String
    NotitiaFunction_FunctionClassification = "Model"
End Function

Private Function NotitiaFunction_FunctionDescription() As String
    NotitiaFunction_FunctionDescription = "101: Event Allocation"

```

```

End Function

Private Function NotitiaFunction_GetIconName() As String
    NotitiaFunction_GetIconName = "Event101"
End Function

Private Function NotitiaFunction_ReceiveMessage(MsgId As String, nfFrom As
NotitiaCore2.NotitiaFunction, Msg As Variant) As Variant

End Function

Private Function NotitiaFunction_Report(Optional ReportLevel As Variant)
As Variant

End Function

Private Function NotitiaFunction_Restore(strEnv As String) As Boolean
    NotitiaFunction_Restore = True
End Function

Private Function NotitiaFunction_Save() As String
    NotitiaFunction_Save = ""
End Function

Private Function NotitiaFunction_Special() As String
    NotitiaFunction_Special = ""
End Function

Private Function NotitiaFunction_SpecialFeatures() As Variant
    Dim sfa(0 To 0) As Variant
    '           Caption           Run Level
    sfa(0) = Array("View/Edit Arrays", "ViewEdit")

    NotitiaFunction_SpecialFeatures = sfa
End Function

Private Function NotitiaFunction_StaticID() As String
    NotitiaFunction_StaticID = "{8CF23441-FAB3-11d3-BBE6-0000C06EB2D1}"
End Function

Private Function NotitiaFunction_DynamicID() As String
    NotitiaFunction_DynamicID = sID
End Function

Private Function NotitiaFunction_FunctionLongDescription() As String
    NotitiaFunction_FunctionLongDescription = "101: Event Allocation"
End Function

Private Function NotitiaFunction_FunctionName() As String
    NotitiaFunction_FunctionName = "F15101"
End Function

Private Function NotitiaFunction_FunctionSubClassification() As String
    NotitiaFunction_FunctionSubClassification = "who knows"
End Function

Private Function NotitiaFunction_FunctionType() As String
    NotitiaFunction_FunctionType = "F15101"
End Function

Private Function NotitiaFunction_FunctionVersion() As String
    NotitiaFunction_FunctionVersion = "0.1"
End Function

Private Function NotitiaFunction_GetInputs() As iIOArray
    Set NotitiaFunction_GetInputs = Inputs
End Function

Private Function NotitiaFunction_GetOutputs() As iIOArray
    Set NotitiaFunction_GetOutputs = outputs
End Function

```

```

Private Function NotitiaFunction_GetInputTypes() As String
    NotitiaFunction_GetInputTypes = NFGetInputTypes(Me)
End Function

Private Function NotitiaFunction_GetOutputTypes() As String
    NotitiaFunction_GetOutputTypes = NFGetOutputTypes(Me)
End Function

Private Function NotitiaFunction_GetSettings() As nSettings
    Set NotitiaFunction_GetSettings = Settings
End Function

Private Function NotitiaFunction_GetHelp() As String
End Function

Private Function NotitiaFunction_Initialize(Optional How As Variant) As Boolean
    NotitiaFunction_Initialize = True
End Function

Private Function NotitiaFunction_Run(Optional RunLevel As Variant) As Variant
    NotitiaFunction_Run = "009:F15101:Unspecified error"

    If Not IsMissing(RunLevel) Then
        If RunLevel = "ViewEdit" Then
            F15101_SetUp Me
            NotitiaFunction_Run = 0
            Exit Function
        End If
    End If

    If NotitiaFunction_CheckInputs = False Then
        NotitiaFunction_Run = "001:F15101:No scenario input found"
        Exit Function
    End If

    ' has scenarios already been found as a global?
    If dScenarios Is Nothing Then
        Inputs.FindByInternalAlias "inp1", dScenarios ' get scenario
dictionary
        If dScenarios Is Nothing Then
            NotitiaFunction_Run = "002:F15101:No scenario input found"
            Exit Function
        End If
    End If

    If dScenarios.Count = 0 Then
        NotitiaFunction_Run = "003:F15101:No scenarios selected"
        Exit Function
    End If

    ' get population input
    If GetPop(dPop) = False Then
        NotitiaFunction_Run = "004:F15101:Couldn't get population"
        Exit Function
    End If

    ' process Scenario P() values
    If GetP(dProb, dScenarios, dPop) = False Then
        NotitiaFunction_Run = "005:F15101:Error processing scenario P()
data"
        Exit Function
    End If

    iuf.ClearAll

    Module1.InitializeOutputs out1, out3, dScenarios

    NumberScenarios = dScenarios.Count - 1

```

```

Module1.InitializeConstantArrays dScenarios, _
                                ids, _
                                Settings, _
                                NumberScenarios, _
                                days, _
                                CoOccur, _
                                DayWeek, _
                                Season, _
                                UseNo, _
                                EffDaysUse, _
                                False      ' don't force

(re)initialization

'1st get population table...
Dim fw As New frmWait
fw.Show

Dim N As Long
N = 0

Dim vPop As Variant
For Each vPop In dPop.Keys

    fw.Label4.Caption = vPop

    N = N + 1
    fw.Label2.Caption = N & "/" & dPop.Count
    DoEvents

Module1.InitializeStaticArrays dScenarios, _
                                ids, _
                                Settings, _
                                NumberScenarios, _
                                days, _
                                CoOccur, _
                                DayWeek, _
                                Season, _
                                DayUse, _
                                UseNo, _
                                EffDaysUse, _
                                True      ' force (re)initialization

Module1.EventsAllocate _
                                days, _
                                NumberScenarios, _
                                CoOccur, _
                                DayWeek, _
                                Season, _
                                DayUse, _
                                UseNo, _
                                EffDaysUse

Module1.WriteOutput _
                                CStr(vPop), _
                                dScenarios, _
                                Settings, _
                                out1, _
                                out2, _
                                out3, _
                                out4, _
                                out6, _
                                days, _
                                CoOccur, _
                                DayWeek, _
                                Season, _
                                DayUse, _
                                UseNo, _
                                EffDaysUse, _
                                dProb

```

```

        DoEvents
        If fw.bUserCancelled = True Then
            NotitiaFunction_Run = "005:F15101:User Cancelled"
            fw.Hide
            Exit Function
        End If

    Next vPop

    fw.Hide

    outputs.SetByInternalAlias "out1", out1
    SetSharedVariable "Events", "DayUse", out1

    iuf.Flush , True

    NotitiaFunction_Run = 0
End Function

Private Sub NotitiaFunction_SetGlobalSettings(GSettings As nSettings)
End Sub

Private Sub NotitiaFunction_SetInputs(i As iIOArray)
End Sub

Private Function NotitiaFunction_Status(Optional StatusLevel As Variant)
As Variant
    NotitiaFunction_Status = "OK"
End Function

Private Function NotitiaFunction_Terminate(Optional How As Variant) As
Variant
End Function

Private Function NotitiaFunction_UseClassification() As String
    NotitiaFunction_UseClassification = "Don't know"
End Function

Public Sub F15101_SetUp(cls As F15101)

    If dScenarios Is Nothing Or SharedVariableExists("InputRes",
"scenarios") = True Then

        Set dScenarios = GetSharedVariable("inputRes", "scenarios")

        NumberScenarios = dScenarios.Count - 1

        Module1.InitializeConstantArrays dScenarios, _
            ids, _
            Settings, _
            NumberScenarios, _
            days, _
            CoOccur, _
            DayWeek, _
            Season, _
            UseNo, _
            EffDaysUse, _
            False, True, True ' force

(re)initialization

    End If

    Set f1.MyClass = cls
    f1.Init
    f1.Show vbModal

    If f1.ApplyChanges = True Then

```

```
        Module1.FillArraysFromSettings Settings, dScenarios, Season,  
DayWeek, CoOccur  
        Module1.FillUseArraysFromSettings Settings, dScenarios, UseNo,  
EffDaysUse  
        End If  
  
End Sub  
  
Public Sub ReloadDefaults(ReloadPeriod As Boolean, ReloadUse As Boolean)  
    Module1.InitializeConstantArrays dScenarios, _  
        ids, _  
        Settings, _  
        NumberScenarios, _  
        days, _  
        CoOccur, _  
        DayWeek, _  
        Season, _  
        UseNo, _  
        EffDaysUse, _  
        True, _  
        ReloadPeriod, _  
        ReloadUse  
  
End Sub
```


2.25 Dietary – Exposure Calculations

```

'
' Dietary Module
'
Option Explicit

Private Type tFoodCons
    Key As Long
    FC As Long
    CS As Byte
    FF As Byte
    CM As Byte
    CAMT As Double
End Type

Private Type tFoodMatch
    id As String
    days(1 To 365) As Long
End Type

Dim r1 As ADODB.Recordset

Dim ids1 As iDataset
Dim ids2 As iDataset
Dim ids3 As iDataset

Private out1 As ADODB.Recordset
Private out2 As ADODB.Recordset
Private out6 As ADODB.Recordset
Private out7 As ADODB.Recordset

Public bUserCancelled As Boolean

Dim dPop As Dictionary
Private dPx As Dictionary
Private aFoodMatch() As tFoodMatch
Private dMap As Dictionary
lives
Private nIdLimit As Long
l==all)
Private dLimit As Dictionary
Private dRAC As Dictionary
Private dCCCache As Dictionary

Private nMaxExp As Long

Private Const nMaxExpA As Long = 2
Private Const nMaxExpB As Long = 0

Private Type tMaxExp
    Exp As Double
    Key As String
End Type

Private aMaxExp() As tMaxExp
Private nMaxExpSize As Long

Private Sub Command1_Click()
    DoIt
End Sub

Public Sub DoIt()
    ReadT18103 dCCCache
    ReadFoodMatch aFoodMatch

    GetPopulation
    CalculateResidue

```

```

End Sub

' for a given food commodity code, or RAC, get the pesticide residue
information associated with food
Public Sub initialize(in1 As iDataset, _
                    in2 As iDataset, _
                    in3 As iDataset, _
                    o1 As ADODB.Recordset, _
                    o2 As ADODB.Recordset, _
                    o6 As ADODB.Recordset, _
                    o7 As ADODB.Recordset, _
                    dP As Dictionary, _
                    dM As Dictionary, _
                    nLimit As Long, nMaxX As Long)

    Set ids1 = in1
    Set ids2 = in2
    Set ids3 = in3

    Set out1 = o1
    Set out2 = o2
    Set out6 = o6 ' for N max output
    Set out7 = o7 ' for daily output

    bUserCancelled = False

    Set dPx = dP
    Set dMap = dM

    nIdLimit = nLimit
    nMaxExp = nMaxX

    nMaxExpSize = 0
    If nMaxExp <> 0 Then
        nMaxExpSize = (nMaxExpA * nMaxExp) + nMaxExpB
        ReDim aMaxExp(1 To nMaxExpSize) ' 2 * N so we can redim stuff
    End If

    Set dRAC = Nothing
    Set dRAC = New Dictionary
End Sub

Private Sub CalculateResidue()
    Set dLimit = Nothing
    Set dLimit = New Dictionary ' initialize discrete id limit checking

    Dim N As Long
    N = 1

    Dim person As Variant
    For Each person In dPop.Keys ' go through each population

        Dim Residues As New Dictionary ' CAS residues for this person
        Residues.RemoveAll

        Dim ResMax As New Dictionary
        ResMax.RemoveAll

        If N Mod 10 = 0 Then
            Label1.Caption = N & " / " & dPop.Count
            Label1.Refresh
        End If

        N = N + 1

        Dim CCGs As New Dictionary
        Dim DayCCGs As New Dictionary ' consumed code groups for
each day of year

```

```

        GetCCGFromID person, CCGs, DayCCGs ' a list of ConsCodesGroups
that the person ate

        ' caclulate aggregate residues
Dim RACCACHE As New Dictionary          ' temporary CCG/RAC cache
RACCACHE.RemoveAll                     ' clear CCG/RAC cache

        ' now calculate discrete residues...
DoDiscrete person, DayCCGs, RACCACHE, Residues, ResMax

WriteADOOutput person, Residues, out1, ResMax ' aggregate output

DoEvents
If bUserCancelled = True Then Exit Sub

Next person

If nMaxExp > 0 Then
    Dim Fake As tMaxExp
    AddToExpBuffer Fake, True

    WriteMaxExp out6
End If

Labell.Caption = "done"

dLimit.RemoveAll
Set dLimit = Nothing
End Sub

Private Sub DoDiscrete(person As Variant, DayCCGs As Dictionary, RACCACHE
As Dictionary, Residues As Dictionary, ResMax As Dictionary)
    Dim DayAgg As New Dictionary
    DayAgg.RemoveAll

    Dim vDay

    For Each vDay In DayCCGs.Keys()          ' go through all consumed
groups
        Dim CCG As String
        CCG = DayCCGs.Item(vDay)

        Dim RACs As Dictionary
        Set RACs = GetRACFromCCG(CCG, RACCACHE)

        If bUserCancelled Then Exit Sub

        Dim CASTotal As Double
        CASTotal = 0

        Dim RACex As Variant
        For Each RACex In RACs.Keys()        ' go through all RAC's
            Dim vA As Variant
            vA = Split(RACex, "|")          ' split into RAC/CS/FF/CM

            Dim RAC As Long
            RAC = vA(0)                    ' RAC/Commodity code
            Dim CS As Integer
            CS = vA(1)                    ' cook status
            Dim FF As Integer
            FF = vA(2)                    ' food form
            Dim CM As Integer
            CM = vA(3)                    ' cook method

            Dim RACamt As Double
            RACamt = RACs.Item(RACex)      ' amount of RACex consumed

            Dim CASSs As Dictionary
            Set CASSs = GetCASFromRAC(RAC, CS, FF, CM) ' get list of CAS's
associated with RAC

```

```

Dim CAS As Variant
For Each CAS In CASs.Keys() ' all CASs for RAC
  Dim CASAmt As Double
  Dim CASResidue As Double
  CASResidue = GetPxMappedFCTMF(RAC, CStr(CAS), CS, FF, CM)

  CASAmt = RACamt * CASResidue / 1000

  ' accumulate daily outputs
  Dim sKey As String
  sKey = vDay & "|" & CAS

  If DayAgg.Exists(sKey) Then
    DayAgg.Item(sKey) = DayAgg.Item(sKey) + CASAmt
  Else
    DayAgg.Item(sKey) = CASAmt
  End If

  ' now write CAS residues
  WriteADODayOutput person, vDay, RAC, CAS, CS, FF, CM,
RACamt, CASAmt, CASResidue, out2

  DoEvents
  If bUserCancelled = True Then Exit Sub

  ' accumulate CAS residues for this person
  If Not Residues.Exists(CAS) Then
    Residues.Add CAS, CASAmt
  Else
    Residues.Item(CAS) = Residues.Item(CAS) + CASAmt
  End If
  ' determine max CAS amount
  If Not ResMax.Exists(CAS) Then
    ResMax.Item(CAS) = CASAmt
  Else
    ResMax.Item(CAS) = Max(ResMax.Item(CAS), CASAmt)
  End If

  ' accumulate unique RACs
  If Not dRAC.Exists(RAC) Then
    dRAC.Add RAC, RAC
  End If

Next CAS

Next RACex

Next vDay

WriteADOAggOutput person, DayAgg, out7 ' day aggregate output
End Sub

Private Function GetCASFromRAC(RAC As Variant, CS As Variant, FF As
Variant, CM As Variant) As Dictionary
  Dim dEmpty As New Dictionary
  Set GetCASFromRAC = dEmpty ' assume the worst

  Dim sKey As String
  sKey = RAC & "+" & CS & "+" & FF & "+" & CM

  If dMap.Exists(sKey) Then
    Dim vA, vRAC, vCS, vFF, vCM

    vA = dMap.Item(sKey)
    vRAC = vA(0)
    vCS = vA(1)
    vFF = vA(2)
    vCM = vA(3)

    Dim sKey2 As String

```

```

        sKey2 = vRAC & "+" & vCS & "+" & vFF & "+" & vCM
        If dPx.Exists(sKey2) Then
            Set GetCASFromRAC = dPx.Item(sKey2)
        End If

        Exit Function
    Else
        MsgBox "CAS/RAC/...( " & sKey & " ) not found in dMap()!",
vbExclamation, "F17101(Test4:GetCASFromRAC() - INTERNAL ERROR"
    End If

End Function

'
' Get RAC (Raw Agricultural Commodity (food) Code) based upon consumption
group
Private Function GetRACFromCCG(ConsCode As Variant, ByRef Cache As
Dictionary) As Dictionary
    Set GetRACFromCCG = Nothing

    ' is CCG already in cache? If so, return it. Else add it
    ' considering everything caching is probably useless here
    If Cache.Exists(ConsCode) Then
        Set GetRACFromCCG = Cache.Item(ConsCode)
        Exit Function
    End If

    Dim Hid As Long      ' household id
    Dim person As Long
    Dim DayCode As Long

    Hid = ConsCode / 1000          ' household id
    person = (ConsCode / 10) Mod 100 ' person #
    DayCode = ConsCode Mod 10      ' food consumed code group #

    ' query t18103 array to get commodity codes for each consumption code

    Dim cRes As Collection
    Set cRes = SearchT18103(dCCCache, CLng(ConsCode))

    ' for given consumption code group store all associated commodity
codes and amounts
    Dim fed As Dictionary          ' foods eaten Dictionary
    Set fed = New Dictionary

    ' fed will consist of a list of clsCommodityCode's that describes what
foods are in the ConsCode group

    Dim vRes As Variant
    If Not cRes Is Nothing Then
        For Each vRes In cRes

            Dim CC As Long
            Dim CS As Integer
            Dim FF As Integer
            Dim CM As Integer
            Dim Amt As Double

            SplitAssign vRes, "|", CC, CS, FF, CM, Amt

            Dim RACex As String      ' RAC (extended)
            RACex = CC & "|" & CS & "|" & FF & "|" & CM

            If Not fed.Exists(RACex) Then
                fed.Add RACex, Amt
            Else
                MsgBox "Multiple Commodity (RAC) code <" & RACex & "> found
for CCG < " & ConsCode & ">"
            End If

            DoEvents
        End For
    End If
End Function

```

```

        If bUserCancelled Then Exit Function

        Next vRes
    End If

    Cache.Add ConsCode, fed          ' add CCG to temporary cache

    Set GetRACFromCCG = fed          ' return RACex (CC|CS|FF|CM), Amt

End Function

Private Sub GetCCGFromID(Cares_ID As Variant, Aggregate As Dictionary,
Discrete As Dictionary)
    Aggregate.RemoveAll

    Dim id As iData
    Dim r As ADODB.Recordset

    Dim Eaten As Dictionary
    Set Eaten = Aggregate 'New Dictionary

    Dim N As Integer      ' 0..365(6)

    If Not (Discrete Is Nothing) Then
        Discrete.RemoveAll
    End If

    Dim nPos As Long
    nPos = SearchFoodMatch(aFoodMatch, CStr(Cares_ID))

    If nPos <> -1 Then
        Dim tfm As tFoodMatch
        tfm = aFoodMatch(nPos)

        Dim nDay As Integer
        For nDay = 1 To 365

            Dim ConsVal As Long      ' consumption value
            ConsVal = tfm.days(nDay)

            ' keep count of how many ConsCode's were eaten by this person
            If Eaten.Exists(ConsVal) Then
                Eaten.Item(ConsVal) = Eaten.Item(ConsVal) + 1
            Else
                Eaten.Add ConsVal, 1    ' 1 occurrence of cons code
            End If

            ' keep tract of all food consumption groups eaten each day
            If Not (Discrete Is Nothing) Then
                N = N + 1
                Discrete.Add N, ConsVal
            End If

            DoEvents

            If bUserCancelled Then Exit Sub
        Next nDay
    Else
        NotitiaLog.WriteLine "Cares id <" & Cares_ID & "> not found in Food
Match data..."
    End If
End Sub

Private Sub GetPopulation()
    Labell1.Caption = "Obtaining population information"

    dPop.RemoveAll

    Dim r As ADODB.Recordset
    Dim id As iData

```

```

' we need to go through the catalog for the dataset and find
' a table of type t16101, This will be the table we want to use
' we'll assume (for now) that there is only 1 pop table per file
Dim sTableName As String
Dim dCat As Dictionary
Set dCat = ids1.GetCatalog
Dim vItem
For Each vItem In dCat.Keys
    Dim dItem As Dictionary
    Set dItem = dCat.Item(vItem)
    If LCase(dItem.Item("tabletype")) = "t16101" Then
        sTableName = CStr(vItem)
        Exit For
    End If
Next vItem
If sTableName = "" Then
    Err.Raise 42, , "No pop table!!!!"
End If

Set id = ids1.OpenTable(sTableName)

If id Is Nothing Then
    MsgBox Err.Description, vbCritical, "GetPopulation"
    Exit Sub
End If

Set r = id.Recordset()

Do While Not r.EOF
    Dim Cares_ID As String
    Cares_ID = r.Fields.Item("1").Value()

    If Not dPop.Exists(Cares_ID) Then
        dPop.Add Cares_ID, 0 ' Empty
    Else
        NotitiaLog.MessageBox "Cares_id=" & Cares_ID & " already
exists in collection", "Dietary(GetPopulation) - data consistence error",
vbInformation
    End If
    r.MoveNext

    DoEvents
    If bUserCancelled = True Then Exit Sub
Loop

End Sub

Private Sub Command3_Click() ' cancel
    bUserCancelled = True
End Sub

Private Sub Form_Load()
    Labell.Caption = ""

    Set dPop = New Dictionary

    bUserCancelled = False
End Sub

Private Sub WriteADOOutput(person As Variant, res As Dictionary, rst As
ADODB.Recordset, ResMax As Dictionary)

    Dim CAS As Variant
    For Each CAS In res.Keys
        rst.AddNew

        rst.Fields.Item("Cares_id").Value = person ' oerson ID
        rst.Fields.Item("CAS").Value = CAS
        rst.Fields.Item("Amount").Value = res.Item(CAS)
        rst.Fields.Item("AnnAvg").Value = res.Item(CAS) / 365
    Next CAS

```

```

        rst.Fields.Item("ExpMax").Value = ResMax.Item(CAS)
    rst.Update
    Next CAS
End Sub

Private Sub WriteADODayOutput(vPerson As Variant, vDay As Variant, RAC As
Variant, CAS As Variant, CS As Variant, FF As Variant, CM As Variant,
RACamt As Variant, CASamt As Variant, CASResidue As Variant, rst As
ADODB.Recordset)

    If rst Is Nothing Then Exit Sub

    If nMaxExp > 0 Then
        Dim sKey As String
        sKey = vPerson & vbTab & _
            vDay & vbTab & _
            CAS & vbTab & _
            RAC & vbTab & _
            CS & vbTab & _
            FF & vbTab & _
            CM & vbTab & _
            RACamt & vbTab & _
            CASResidue

        Dim tExp As tMaxExp
        tExp.Exp = CASamt
        tExp.Key = sKey

        AddToExpBuffer tExp
    End If

    ' has ID output limit been reached?
    If nIdLimit <> -1 And _
        dLimit.Count >= nIdLimit And _
        Not dLimit.Exists(vPerson) Then

        Exit Sub
    Else
        If Not dLimit.Exists(vPerson) Then
            dLimit.Add vPerson, vPerson
        End If
    End If

    rst.AddNew

    rst.Fields.Item("Cares_id").Value = vPerson           ' oerson ID
    rst.Fields.Item("day").Value() = vDay
    rst.Fields.Item("CAS").Value = CAS
    rst.Fields.Item("Food").Value = RAC
    rst.Fields.Item("CASExp").Value = CASamt
    rst.Fields.Item("CS").Value = CS
    rst.Fields.Item("FF").Value = FF
    rst.Fields.Item("CM").Value = CM
    rst.Fields.Item("RACamt").Value = RACamt
    rst.Fields.Item("CASResidue").Value = CASResidue

    rst.Update
End Sub

Public Sub Cleanup()
    Set out1 = Nothing
    Set out2 = Nothing
    Set out6 = Nothing
End Sub

' return the CAS residue value via a P%() calculation
Private Function GetPx(RAC As Long, CAS As String, CS As Integer, FF As
Integer, CM As Integer) As Double
    Dim nRnd As Integer
    nRnd = Random(0, 100)           ' get an integer between 0..100

```



```

Dim sKey As String
sKey = RAC & "+" & CS & "+" & FF & "+" & CM

' get P%(nRnd) for RAC/CAS/CS/FF/CM

Dim vArray As Variant
vArray = dPx.Item(sKey).Item(CAS).Item("Px")

GetPx = vArray(nRnd)
End Function

Private Function GetPxMapped(RAC As Long, CAS As String, CS As Integer, FF
As Integer, CM As Integer) As Double
    GetPxMapped = 0

    Dim sKey As String
    sKey = RAC & "+" & CS & "+" & FF & "+" & CM

    If dMap Is Nothing Then Exit Function
    If dMap.Count = 0 Then Exit Function

    Dim nRAC As Long          ' mapped RAC
    If dMap.Exists(sKey) Then
        nRAC = dMap.Item(sKey)

        GetPxMapped = GetPx(nRAC, CAS, CS, FF, CM)
        Exit Function
    Else
        MsgBox "CAS/RAC/..." & sKey & " not found in dMap()",
vbExclamation, "F17101(Test4:GetPxMapped() - INTERNAL ERROR"
        End If
    End Function

Private Function GetPxMappedFCTMF(RAC As Long, CAS As String, CS As
Integer, FF As Integer, CM As Integer) As Double
    GetPxMappedFCTMF = 0

    Dim sKey As String
    sKey = RAC & "+" & CS & "+" & FF & "+" & CM

    If dMap Is Nothing Then Exit Function
    If dMap.Count = 0 Then Exit Function

    Dim nRAC As Long          ' mapped RAC
    Dim nCS As Integer
    Dim nFF As Integer
    Dim nCM As Integer
    Dim nFCT As Double        ' FCT for RAC
    Dim nMF1 As Double        ' MF1 for RAC
    Dim nMF2 As Double        ' MF2 for RAC

    If dMap.Exists(sKey) Then
        Dim vArray As Variant
        vArray = dMap.Item(sKey)
        If Not IsArray(vArray) Then
            MsgBox "CAS/RAC map entry not array! <" & sKey & ">",
vbExclamation, "F17101(Test4:GetPxMappedFCT() - INTERNAL ERROR"
            Exit Function
        End If

        nRAC = vArray(0)      ' RAC
        nCS = vArray(1)      ' CS
        nFF = vArray(2)      ' FF
        nCM = vArray(3)      ' CM
        nFCT = vArray(4)     ' FCT
        nMF1 = vArray(5)     ' MF1
        nMF2 = vArray(6)     ' mf1

        If Rnd > nFCT Then

```

```

        GetPxMappedFCTMF = 0 ' Residue is 0 if RND falls outside of
fct
    Else
        GetPxMappedFCTMF = GetPx(nRAC, CAS, nCS, nFF, nCM) * nMF1 *
nMF2
    End If
    Exit Function
Else
    MsgBox "CAS/RAC not found in dMap(!)", vbExclamation,
"F17101(Test4:GetPxMappedFCTMF() - INTERNAL ERROR"
    End If
End Function

```

```

Private Function Random(Min As Integer, Max As Integer) As Integer
    Random = Int((Max - Min + 1) * Rnd + Min)
End Function

```

```

Private Function ReadT18103(dCache As Dictionary) As Boolean

    Set dCache = Nothing
    Set dCache = New Dictionary

    Dim id As IData
    Set id = ids3.OpenTable("t18103", , , adUseServer,
adLockBatchOptimistic) ' food cons extra
    If id Is Nothing Then
        MsgBox Err.Description, vbCritical, "ReadInAll"
        Exit Function
    End If

    Dim r As ADODB.Recordset
    Set r = id.Recordset()
    r.MoveFirst

    Dim nRecs As Long
    nRecs = r.RecordCount()

    Labell1.Caption = "Obtaining " & nRecs & " food consumption records"

    Dim i As Long
    i = 1
    Do While Not r.EOF
        Dim Key As Long
        Dim FC As Long
        Dim CS As Byte
        Dim FF As Byte
        Dim CM As Byte
        Dim Amt As Double

        Dim FCId As Long
        Dim person As Long
        Dim day As Long

        FCId = r.Fields.Item("1").Value ' fcid
        person = r.Fields.Item("2").Value ' person
        day = r.Fields.Item("3").Value ' day

        Key = (FCId * 1000) + (person * 10) + day

        Dim Key2 As String

        FC = r.Fields.Item("4").Value ' food code
        CS = r.Fields.Item("5").Value ' cook status
        FF = r.Fields.Item("6").Value ' food form
        CM = r.Fields.Item("7").Value ' cook method
        Amt = r.Fields.Item("8").Value ' Amount

        Key2 = FC & "|" & CS & "|" & FF & "|" & CM & "|" & Amt
    End Do
End Function

```

```

        Dim C As Collection
        If dCache.Exists(Key) Then
            Set C = dCache.Item(Key)
        Else
            Set C = New Collection
            dCache.Add Key, C
        End If
        C.Add Key2

        DoEvents

        r.MoveNext
    Loop

    r.Close
    Set id = Nothing

End Function

Private Function SearchT18103(dCache, Key As Long) As Collection
    If Not dCache.Exists(Key) Then
        Set SearchT18103 = Nothing
        Exit Function
    End If

    Dim C As Collection
    Set C = dCache.Item(Key)
    Set SearchT18103 = C
End Function

Private Function ReadFoodMatch(a() As tFoodMatch) As Boolean
    Labell.Caption = "Redging 'food match' data"
    ' open all tables and get the max records

    Dim nMax As Long
    nMax = 0

    Dim id As New iData
    id.DataSource = ids2.DataSource

    Dim Table As Variant
    For Each Table In Array("January", "February", "March", "April",
        "May", "June", "July", "August", "September", "October", "November",
        "December")

        id.CommandText = CStr(Table)
        id.CommandType = adCmdTable
        id.OpenRST

        Dim r As ADODB.Recordset
        Set r = id.Recordset()

        Dim nRecs As Long
        nRecs = r.RecordCount()
        id.CloseRst
        Set r = Nothing

        If nRecs > nMax Then nMax = nRecs

    Next Table

    ReDim a(1 To nMax)

    Dim i As Long

    Dim nDaysSoFar As Long
    nDaysSoFar = 0

    Dim td1 As String, td2 As String, fd1 As String, fd2 As String
    Dim FieldDelimiters As Variant

```

```

FieldDelimiters = id.GetDelimiters()
td1 = FieldDelimiters(0)
td2 = FieldDelimiters(1)
fd1 = FieldDelimiters(2)
fd2 = FieldDelimiters(3)

For Each Table In Array("January", "February", "March", "April",
"May", "June", "July", "August", "September", "October", "November",
"December")
    Label1.Caption = "Table: " & Table
    Label1.Refresh

    i = 1

    id.CommandText = "select * from " & td1 & Table & td2 & " order by
" & fd1 & "CARES_ID" & fd2
    id.CommandType = adCmdText
    id.OpenRST

    Dim nDayInMonth As Integer

    Set r = id.Recordset()
    Do While Not r.EOF
        Dim sID As String
        sID = r.Fields.Item("Cares_id").Value

        If a(i).id <> "" Then
            If a(i).id <> sID Then
                NotitiaLog.MessageBox "id sync error for table <" &
Table & "> i=" & i & " id=" & a(i).id, "ReadFoodMatch()", vbExclamation
            End If
        Else
            a(i).id = sID
        End If

        nDayInMonth = 1
        Dim fld As ADODB.Field
        For Each fld In r.Fields

            Dim ConsVal As Long          ' consumption value

            If IsNumeric(fld.Name) Then
                ConsVal = r.Fields.Item(fld.Name).Value()

                a(i).days(nDaysSoFar + nDayInMonth) = ConsVal
                nDayInMonth = nDayInMonth + 1
            End If
        Next fld

        r.MoveNext
        i = i + 1

        DoEvents
        If bUserCancelled Then Exit Function
    Loop

    nDaysSoFar = nDaysSoFar + r.Fields.Count - 1

    id.CloseRst
    Set r = Nothing

Next Table

Set id = Nothing

End Function

Private Function SearchFoodMatch(a() As tFoodMatch, id As String) As Long
    Dim low As Long
    low = LBound(a)

```

```

Dim high As Long
high = UBound(a)

Dim middle As Long

Do While (low <= high)

    middle = (low + high) / 2

    If (id = a(middle).id) Then
        SearchFoodMatch = middle
        Exit Function
    End If

    If (id < a(middle).id) Then
        high = middle - 1 ' search low end of array
    Else
        low = middle + 1 ' search high end of array
    End If
Loop

SearchFoodMatch = -1 ' search key not found

End Function

Private Function Max(a As Variant, b As Variant)
    If a > b Then
        Max = a
    Else
        Max = b
    End If
End Function

Public Function WriteRAC(out5 As ADODB.Recordset) As Boolean
    WriteRAC = False ' just in case

    On Error GoTo Oops

    Dim vRAC As Variant
    For Each vRAC In dRAC.Keys
        out5.AddNew
        out5.Fields.Item("Food").Value = vRAC
        out5.Update
    Next vRAC

    WriteRAC = True
    Exit Function

Oops:
    NotitiaLog.MessageBox Err.Description, "Dietary1(Test4:WriteRAC)",
vbExclamation
    Err.Clear
    Exit Function
End Function

' ann an exposure key to the MAX EXP buffer
Private Function AddToExpBuffer(tValue As tMaxExp, Optional bSort As
Boolean = False, Optional bReset As Boolean = False) As Boolean
    Static nCurrent As Long ' current pointer into vMaxExp
    Static nRecs As Long

    If bReset = True Then
        ReDim aMaxExp(1 To nMaxExp) ' remove everything
        ReDim aMaxExp(1 To nMaxExpSize)
        nCurrent = 0
        nRecs = 0
    End If

    If nCurrent = 0 Then nCurrent = 1 ' since we're 1's based...

```

```

' only add a record if not forced to sort
If bSort <> True Then
    aMaxExp(nCurrent) = tValue

    nCurrent = nCurrent + 1
    nRecs = nRecs + 1
End If

' time to sort & compact?
If nCurrent > nMaxExpSize Or bSort = True Then
    ' yep

    ' so, sort it ....
    If nMaxExpSize < 500 Then ' arbitrary limit, but must be >=3 for
HeapSort
        QuicksortExp aMaxExp, 1, nMaxExpSize
    Else
        HeapsortExp aMaxExp
    End If

    ' ... and remove all but 1st nMaxExp entries
    ReDim Preserve aMaxExp(1 To nMaxExp) ' remove everything past
N ....
    ReDim Preserve aMaxExp(1 To nMaxExpSize) ' ... and restore proper
size

    If nCurrent > nMaxExpSize Or bSort = True Then
        nCurrent = nMaxExp + 1
    End If

End If

End Function

Private Sub QuicksortExp(List() As tMaxExp, Min As Long, Max As Long)
    Dim med_value As tMaxExp

    Dim hi As Long
    Dim lo As Long
    Dim i As Long

    ' If the list has no more than 1 element, it's sorted.
    If Min >= Max Then Exit Sub

    ' Pick a dividing item.
    i = Int((Max - Min + 1) * Rnd + Min)
    med_value = List(i)

    ' Swap it to the front so we can find it easily.
    List(i) = List(Min)

    ' Move the items smaller than this into the left
    ' half of the list. Move the others into the right.
    lo = Min
    hi = Max
    Do
        ' Look down from hi for a value < med_value.
        Do While List(hi).Exp < med_value.Exp
            hi = hi - 1
            If hi <= lo Then Exit Do
        Loop
        If hi <= lo Then
            List(lo) = med_value
            Exit Do
        End If

        ' Swap the lo and hi values.
        List(lo) = List(hi)

        ' Look up from lo for a value >= med_value.
        lo = lo + 1
    
```

```

Do While List(lo).Exp >= med_value.Exp
    lo = lo + 1
    If lo >= hi Then Exit Do
Loop
If lo >= hi Then
    lo = hi
    List(hi) = med_value
    Exit Do
End If

' Swap the lo and hi values.
List(hi) = List(lo)
Loop

' Sort the two sublists
QuicksortExp List(), Min, lo - 1
QuicksortExp List(), lo + 1, Max
End Sub

Private Sub HeapsortExp(x() As tMaxExp)
    Dim i As Long
    Dim j As Long
    Dim k As Long
    Dim y As tMaxExp    'Variant

    Dim N As Long
    N = UBound(x)
    If N < 3 Then Exit Sub

    ' create initial heap
    For k = 2 To N
        ' insert x(k) into existing heap of size k-1
        i = k
        y = x(k)
        j = i / 2
        Do While j > 0
            If y.Exp > x(j).Exp Then GoTo L1    ' descending sort
            x(i) = x(j)
            i = j
            j = i / 2
        Loop
L1:    x(i) = y
    Next k

    ' we remove x(1) and place it in its proper position
    ' in the array. We then adjust the heap
    For k = N To 2 Step -1
        y = x(k)
        x(k) = x(1)
        ' readjust heap of order k-1
        ' move y down the heap for proper position
        i = 1
        j = 2

        If x(3).Exp < x(2).Exp And k - 1 >= 3 Then j = 3    ' descending
sort

        ' j is the larger son of i
        ' in the heap of size k-1
        Do While j <= k - 1
            If x(j).Exp >= y.Exp Then GoTo L2    ' descending
            x(i) = x(j)
            i = j
            j = 2 * i
            If j + 1 <= k - 1 Then
                If x(j + 1).Exp < x(j).Exp Then    ' descending
                    j = j + 1
                End If
            End If
        Loop
L2:    x(i) = y

```

```

Next k

End Sub

Private Sub WriteMaxExp(rst As ADODB.Recordset)

    If rst Is Nothing Then Exit Sub

    Dim N As Long
    For N = 1 To nMaxExp
        Dim sKey As String
        sKey = aMaxExp(N).Key

        Dim rExp As Double
        rExp = aMaxExp(N).Exp

        Dim vPerson, vDay, CAS, RAC, CS, FF, CM, RACamt, CASResidue
        SplitAssign sKey, vbTab, vPerson, vDay, CAS, RAC, CS, FF, CM,
        RACamt, CASResidue

        rst.AddNew

        rst.Fields.Item("Cares_id").Value = vPerson          ' oerson ID
        rst.Fields.Item("day").Value() = vDay
        rst.Fields.Item("CAS").Value = CAS
        rst.Fields.Item("Food").Value = RAC
        rst.Fields.Item("CASExp").Value = rExp

        rst.Fields.Item("CS").Value = CS
        rst.Fields.Item("FF").Value = FF
        rst.Fields.Item("CM").Value = CM
        rst.Fields.Item("RACamt").Value = RACamt
        rst.Fields.Item("CASResidue").Value = CASResidue

        rst.Update

    Next N
End Sub

Private Sub WriteADOAggOutput(person As Variant, DayAgg As Dictionary, rst
As ADODB.Recordset)
    If rst Is Nothing Then Exit Sub

    Dim v
    For Each v In DayAgg.Keys
        Dim vDay, vCAS
        SplitAssign v, "|", vDay, vCAS

        If DayAgg.Item(v) <> 0 Then
            rst.AddNew

            rst.Fields.Item("Cares_id").Value = person          ' person ID
            rst.Fields.Item("Day").Value = vDay
            rst.Fields.Item("CAS").Value = vCAS
            rst.Fields.Item("Exposure").Value = DayAgg.Item(v)

            rst.Update
        End If
    Next v
End Sub

```



```

'
' Dietary Module support code
'

Option Explicit

Public Const cRouteDermal As Integer = 1
Public Const cRouteIngFood As Integer = 2
Public Const cRouteIngHTM As Integer = 3
Public Const cRouteIngWater As Integer = 4
Public Const cRouteInhale As Integer = 5

Dim ids1 As iDataset
Dim ids2 As iDataset
Dim ids3 As iDataset

' return true if we should run the dietary module(s)
Public Function DietaryEnabled() As Boolean
' see if "dietary" option has been selected in the source selector
If SharedVariableExists("InputOptions", "ExpMedia") = True Then
    Dim d As Dictionary
    Set d = GetSharedVariable("InputOptions", "ExpMedia")
    If Not d.Exists("dietary") Then
        DietaryEnabled = False
        Exit Function
    End If
End If

    DietaryEnabled = True ' enable if no source selection found
End Function

Public Function CalculateResidues(in1 As iDataset, _
                                in2 As iDataset, _
                                in3 As iDataset, _
                                out1 As ADODB.Recordset, _
                                out2 As ADODB.Recordset, _
                                out3 As ADODB.Recordset, _
                                out4 As ADODB.Recordset, _
                                out5 As ADODB.Recordset, _
                                out6 As ADODB.Recordset, _
                                out7 As ADODB.Recordset, _
                                nLimit As Long, _
                                nMaxExp As Long, _
                                RndSeed As Double) As Boolean

    CalculateResidues = False

    Rnd = CDbl(RndSeed)
    Randomize CDbl(RndSeed)

    Dim dChem As Dictionary
    If SharedVariableExists("InputChem", "all") = False Then
        MsgBox "No Chemicals selected? Please make selected from 'Chemical
input' screen.", vbExclamation, "No Chemicals selected!"
        Exit Function
    Else
        Set dChem = GetSharedVariable("InputChem", "all")
    End If

    ' write CAS output
    Dim vCAS As Variant
    For Each vCAS In dChem.Keys
        Dim vCASName As String
        vCASName = dChem.Item(vCAS)

        If vCAS <> "*" Then
            out3.AddNew
            out3.Fields.Item("CAS").Value = vCAS
            out3.Fields.Item("Source").Value = 1 ' dietary
            out3.Fields.Item("Route").Value = cRouteIngFood '
        ingestion, food
    
```

```

        out3.Update
    End If

Next vCAS

Dim dResidue As Dictionary
Set dResidue = GetSharedVariable("DietaryOptions", "Residue")

If dResidue Is Nothing Then
    MsgBox "No Chemical residue data???", vbExclamation,
"F17101(Module1:CalculateResidues 1 - INTERNAL ERROR)"
    Exit Function
End If
If dResidue.Count = 0 Then
    MsgBox "No Chemical residue data???", vbExclamation,
"F17101(Module1:CalculateResidues 2 - INTERNAL ERROR)"
    Exit Function
End If

Dim dMap As New Dictionary
Dim dFactors As New Dictionary

Dim ffm As New frmFoodMap
ffm.initialize ids3, dResidue, dMap, dFactors
ffm.Show vbModal

If dMap.Count = 0 Then
    MsgBox "No Food => Residue map established", vbExclamation, "Food
=> Residue map required"
    Exit Function
End If

PleaseWait True, "Building P%() arrays"
DoEvents

' now build P%()'s
Dim dPx As Dictionary          ' for storing RAC/CAS Px() data
Set dPx = BuildPxArraysZero(dChem, dResidue, dFactors, out4)
PleaseWait False
DoEvents

If dPx Is Nothing Then
    MsgBox "No P%() array???", vbExclamation,
"F17101(Module1:CalculateResidues - INTERNAL ERROR)"
    Exit Function
End If

DoEvents
' check to see if Chemical input screen has been run. Return false if
' hasn't been run
'

Dim t4 As New Test4
t4.initialize in1, in2, in3, out1, out2, out6, out7, dPx, dMap,
nLimit, nMaxExp

t4.Show          ' show dialog.form
t4.DoIt          ' do residue calculations
t4.Hide         ' all done...
t4.WriteRAC out5 ' write RACs
t4.CleanUp      ' clean up recordset references

If t4.bUserCancelled = True Then
    CalculateResidues = False
Else
    CalculateResidues = True
End If

Set dChem = Nothing

```

```

End Function

' set up Fake inputs to test function
Public Function SetupFakeIO(cls As NotitiaFunction) As Boolean
    SetupFakeIO = False

    ' let's try to 1st get inputs from Shared variables
    Dim vFoodMatch, vRefPop, vFoodCons
    Dim oResidue As Object

    vFoodMatch = GetSharedVariable("DietaryOptions", "FoodMatch")
    vRefPop = GetSharedVariable("DietaryOptions", "RefPop")
    vFoodCons = GetSharedVariable("DietaryOptions", "FoodCons")

    If SharedVariableExists("DietaryOptions", "Residue") Then
        Set oResidue = GetSharedVariable("DietaryOptions", "Residue")
    End If

    If IsEmpty(vFoodMatch) Or IsEmpty(vRefPop) Or IsEmpty(vFoodCons) Or
oResidue Is Nothing Then
        If IsEmpty(vFoodMatch) Then
            MsgBox "Have you run the 'Dietary input' module?", vbQuestion,
"No 'Food Match' data found!"
            Exit Function
        End If
        If IsEmpty(vRefPop) Then
            MsgBox "Have you run the 'Dietary input' module?", vbQuestion,
"No 'Population' data found!"
            Exit Function
        End If
        If IsEmpty(vFoodCons) Then
            MsgBox "Have you run the 'Dietary input' module?", vbQuestion,
"No 'Food Consumption' data found!"
            Exit Function
        End If
        If oResidue Is Nothing Then
            MsgBox "Have you run the 'Dietary input' module?", vbQuestion,
"No 'Chemical Residue' data found!"
            Exit Function
        End If
    Else
        ' if we get here then all required inputs have been found
        Set ids1 = GetSystemLibrary.OpenDataset(CStr(vRefPop))
        Set ids2 = GetSystemLibrary.OpenDataset(CStr(vFoodMatch))
        Set ids3 = GetSystemLibrary.OpenDataset(CStr(vFoodCons))
    End If

    Dim Inputs As iIOArray
    Set Inputs = cls.GetInputs()
    Inputs.SetByInternalAlias "in 1", ids1
    Inputs.SetByInternalAlias "in 2", ids2
    Inputs.SetByInternalAlias "in 3", ids3
    Inputs.SetByInternalAlias "d 0", oResidue

    SetupFakeIO = True

End Function

Public Sub CopyRecordset(src As ADODB.Recordset, dst As ADODB.Recordset)
    dst.AddNew

    Dim fld As ADODB.Field
    For Each fld In src.Fields
        dst.Fields.Item(fld.Name).Value = src.Fields.Item(fld.Name).Value
    Next fld

    dst.Update
End Sub

Public Function Floor(x) As Double
    Floor = Int(x)

```

```

End Function

Public Function Ceil(x) As Double
  If x = Int(x) Then
    Ceil = x
  Else
    Ceil = Int(x) + 1
  End If
End Function

Public Function GetPercentiles(Data() As Double, Optional N As Integer =
100) As Double()

  Dim Px() As Double
  ReDim Px(0 To N) As Double

  Quicksort Data, LBound(Data), UBound(Data)

  Dim lvd As Double
  lvd = Data(LBound(Data))

  Dim hvd As Double
  hvd = Data(UBound(Data))

  Dim k As Long
  Dim NumPercentile As Long
  NumPercentile = N

  Dim DataLen As Long
  DataLen = UBound(Data) - LBound(Data) + 1

  Dim Interval As Double
  Dim Percentile As Double
  Dim Location As Double
  Dim LowValue As Double
  Dim HighValue As Double
  Dim hv As Double
  Dim lv As Double

  Dim slope As Double
  Dim Constant As Double

  Interval = 100 / NumPercentile
  For k = 0 To NumPercentile - 1
    Percentile = k * Interval / 100

    Location = Percentile * DataLen

    LowValue = Floor(Location)
    If LowValue = 0 Then ' since we're not 0-based
      lv = 0
    Else
      lv = Data(LowValue)
    End If

    HighValue = Ceil(Location)
    If HighValue = 0 Then ' since we're not 0-based
      hv = 0
    Else
      hv = Data(HighValue)
    End If

    slope = hv - lv

    Constant = Location - LowValue

    Px(k) = lv + slope * Constant
  Next k

```

```

    GetPercentiles = Px
End Function

Sub Quicksort(List() As Double, Min As Long, Max As Long)
    Dim med_value As Double
    Dim hi As Long
    Dim lo As Long
    Dim i As Long

    ' If the list has no more than 1 element, it's sorted.
    If Min >= Max Then Exit Sub

    ' Pick a dividing item.
    i = Int((Max - Min + 1) * Rnd + Min)
    med_value = List(i)

    ' Swap it to the front so we can find it easily.
    List(i) = List(Min)

    ' Move the items smaller than this into the left
    ' half of the list. Move the others into the right.
    lo = Min
    hi = Max
    Do
        ' Look down from hi for a value < med_value.
        Do While List(hi) >= med_value
            hi = hi - 1
            If hi <= lo Then Exit Do
        Loop
        If hi <= lo Then
            List(lo) = med_value
            Exit Do
        End If

        ' Swap the lo and hi values.
        List(lo) = List(hi)

        ' Look up from lo for a value >= med_value.
        lo = lo + 1
        Do While List(lo) < med_value
            lo = lo + 1
            If lo >= hi Then Exit Do
        Loop
        If lo >= hi Then
            lo = hi
            List(hi) = med_value
            Exit Do
        End If

        ' Swap the lo and hi values.
        List(hi) = List(lo)
    Loop

    ' Sort the two sublists
    Quicksort List(), Min, lo - 1
    Quicksort List(), lo + 1, Max
End Sub

Public Function BuildPxArraysZero(dChem As Dictionary, dRes As Dictionary,
dFactors As Dictionary, out4 As ADODB.Recordset) As Dictionary
    Dim dPx As New Dictionary

    Dim ids4 As iDataset
    Dim id4 As New iData

    Dim sRefID As String
    Dim sTableName As String
    Dim vRes As Variant, vItem As Variant
    For Each vRes In dRes.Keys
        vItem = dRes.Item(vRes)

```

```

SplitAssign vItem, "|", sRefID, sTableName

'1st get all unique RAC/CAS codes from residue data
Set ids4 = GetSystemLibrary.OpenDataset(sRefID)

Dim sSQL As String
sSQL = "SELECT DISTINCT [RAC], [CAS], [CS], [FF], [CM] From [" &
sTableName & "] GROUP BY [Rac], [Cas], [CS], [FF], [CM];"

id4.CloseRst
id4.DataSource = ids4.DataSource
id4.CommandText = sSQL
id4.CommandType = adCmdText

Dim r4 As ADODB.Recordset
Set r4 = id4.OpenRST()

If r4.RecordCount <= 0 Then
    MsgBox "No residue data?", vbExclamation,
"F17101 (Module1:BuildPxArrays)"
    Exit Function
End If

' if we get here then we have some unique RAC/CAS combos
Do While Not r4.EOF
    Dim RAC As Long 'String
    Dim CAS As String
    Dim CS, FF, CM

    CAS = r4.Fields.Item("CAS").Value

    ' has chemical (CAS) been previously selected (in chem screen)
    ?
    If DoCAS(dChem, CAS) = True Then

        RAC = r4.Fields.Item("RAC").Value

        CS = r4.Fields.Item("CS").Value
        FF = r4.Fields.Item("FF").Value
        CM = r4.Fields.Item("CM").Value

        If IsNull(CS) Or CS = "" Then CS = 0
        If IsNull(FF) Or FF = "" Then FF = 0
        If IsNull(CM) Or CM = "" Then CM = 0

        Dim sKey As String
        sKey = RAC & "+" & CS & "+" & FF & "+" & CM

        ' RAC/CS/FF/CM not previously found. Create entry for
        it...
        If Not dPx.Exists(sKey) Then
            dPx.Add sKey, New Dictionary
        End If

        ' get (possibly just created) RAC/CS/FF/CM dictionary
        Dim dRAC As Dictionary
        Set dRAC = dPx.Item(sKey)

        If Not dRAC.Exists(CAS) Then
            Dim dDst As Dictionary
            Set dDst = New Dictionary
            dDst.Add "Px", Empty ' for Px(array)
            dDst.Add "Raw", Empty ' for 'raw' data
            dDst.Add "Src", vItem ' src of this data
            dRAC.Add CAS, dDst
        End If

    End If

End If

r4.MoveNext

```

```

        Loop
        id4.CloseRst
        Set ids4 = Nothing
        DoEvents
    Next vRes

' at this point we have an empty RAC/CAS 'array' set up
' now. Go through data and store RAC/CAS values

Dim vKey, vRAC, vCAS
For Each vKey In dPx.Keys
    For Each vCAS In dPx.Item(vKey).Keys
        Set dDst = dPx.Item(vKey).Item(vCAS)

        Dim vA, vCS, vFF, vCM

        SplitAssign vKey, "+", vRAC, vCS, vFF, vCM

        vItem = dDst.Item("Src")
        SplitAssign vItem, "|", sRefID, sTableName

        ' now let's get all Residue data for RAC/CAS/CS/FF/CM
        Set ids4 = Nothing
        Set ids4 = GetSystemLibrary.OpenDataset(sRefID)

        Dim DataSource As String
        DataSource = ids4.DataSource
        Set ids4 = Nothing

        id4.DataSource = DataSource 'ids4.DataSource
        id4.CommandType = adCmdText
        id4.CommandText = "select * from [" & sTableName & "] where
RAC=" & vRAC & " and CAS='" & vCAS & "' and CS=" & vCS & " and FF=" & vFF
& " and CM=" & vCM
        Set r4 = id4.OpenRST()
        If r4.RecordCount <= 0 Then
            MsgBox "r4.recordcount = " & r4.RecordCount
            GoTo next_vCas
        End If

        Dim nZeros As Long
        nZeros = 0

        If dFactors.Exists(vKey) Then
            nZeros = dFactors.Item(vKey)(7)
        End If

        Dim a() As Double
        ReDim a(1 To r4.RecordCount() + nZeros)

        Dim nIdx As Long
        nIdx = 1
        Do While Not r4.EOF
            a(nIdx) = r4.Fields.Item("Residue Value").Value
            nIdx = nIdx + 1
            r4.MoveNext
        Loop
        id4.CloseRst

        ' now all 'nZeros' 0 records
        Dim N As Long
        For N = 1 To nZeros
            a(nIdx) = 0
            nIdx = nIdx + 1
        Next N

        dDst.Item("Px") = GetPercentiles(a) ' save Px(RAW)
        dDst.Item("Raw") = a ' save RAW data
    Next vCAS
Next vKey
(sorted)

```

```

        WritePercentiles dDst.Item("Px"), vCAS, vRAC, vCS, vFF, vCM,
out4

        DoEvents
next_vCas:
    Next vCAS
    Next vKey

    Set BuildPxArraysZero = dPx

' DumpPx dPx, "Px() dump..."
End Function

Public Sub DumpPx(dPx As Dictionary, Optional sTitle As String = "")
    If sTitle <> "" Then NotitiaLog.WriteLine sTitle

    Dim vRAC, vCAS, dSrc
    For Each vRAC In dPx.Keys
        For Each vCAS In dPx.Item(vRAC).Keys
            Set dSrc = dPx.Item(vRAC).Item(vCAS)
            NotitiaLog.WriteLine "RAC=" & vRAC & " CAS=" & vCAS

            Dim a
            a = dSrc.Item("Raw")
            NotitiaLog.WriteLine Space(4) & "RAW..."
            Dim N
            NotitiaLog.WriteText Space(6)
            For N = LBound(a) To UBound(a)
                NotitiaLog.WriteText Format(a(N), "0.000E-00") & " "
            Next N
            NotitiaLog.WriteLine ""

            a = dSrc.Item("Px")
            NotitiaLog.WriteLine Space(4) & "Px..."

            NotitiaLog.WriteText Space(6)
            For N = LBound(a) To UBound(a)
                NotitiaLog.WriteText Format(a(N), "0.000E-00") & " "
            Next N
            NotitiaLog.WriteLine ""
        Next vCAS
    Next vRAC

End Sub

' has CAS been selected.
Public Function DoCAS(dChem As Dictionary, sCAS As String) As Boolean
    DoCAS = False
    If dChem Is Nothing Then Exit Function ' no chemical dictionary?!

    If dChem.Exists(sCAS) Then
        DoCAS = True ' it exists!!!
    End If
End Function

Public Function GetRunDescription(BaseDesc As String) As String
    If Not SharedVariableExists("RunSpec", "all") Then
        GetRunDescription = BaseDesc & " (" & CStr(Now) & ")"
    Else
        Dim d As Dictionary
        Set d = GetSharedVariable("RunSpec", "all")
        If Not d.Exists("RunSpec(Short)") Then
            GetRunDescription = BaseDesc & " (" & CStr(Now) & ")"
        Else
            Dim RunDesc As String
            RunDesc = d.Item("RunSpec(Short)")
            GetRunDescription = RunDesc & " (" & CStr(Now) & ")"
        End If
    End If
End Function

```



```
' write out percentile info .....
Public Function WritePercentiles(vPx As Variant, vCAS As Variant, vRAC As
Variant, vCS As Variant, vFF As Variant, vCM As Variant, out4 As
ADODB.Recordset) As Boolean
    out4.AddNew

    out4.Fields.Item("CAS").Value = vCAS
    out4.Fields.Item("Food").Value = vRAC
    out4.Fields.Item("CS").Value = vCS
    out4.Fields.Item("FF").Value = vFF
    out4.Fields.Item("CM").Value = vCM

    Dim N As Integer

    For N = LBound(vPx) To UBound(vPx)
        Dim sP As String
        sP = "p" & CStr(N)
        out4.Fields.Item(sP).Value = vPx(N)
    Next N

    out4.Update
End Function
```