APPENDIX G. Visual Basic Pseudocode and Fortran Subroutines

This section presents the Visual Basic pseudocode for both interfaces and the FORTRAN code for the water quality component developed for WATFLOOD. The full source code will take several hundreds of pages to be printed and defies any rational attempt to be included as a hardcopy printout. So what is presented here is a pseudocode that contains the subroutines and functions with a brief description of the process involved. For the main procedures, the full code is included. The source code in full extent resides in ASCII text files for each interface.

Table G1. Visual Basic File Structure of the Interfaces (●AGNPS / ■WATFLOOD)

Main Forms:		LANDATA.FRM	●■	AGNPSGR.BAS	●
INTERAGN.FRM	●	RESUME.FRM	●	+ WATFLDGR.BAS	■
+ INTERWAT.FRM	■	SENSANAL.FRM	●	DATACTL.BAS	●■
GRIDMAKE.FRM	●■	SENSINPU.FRM	●	DRAWFLOW.BAS	●■
AGNPSCDB.FRM	●	SENSNORM.FRM	●	LAYERDB.BAS	●■
+ WATFCDB.FRM	■	SENSRANK.FRM	●	LAYERDLL.BAS	●■
CONTROL.FRM	●■	**Additonal Forms**:		GLOBAL.BAS	●■
INITIAL.FRM	●■	SOIL.FRM	●	MAIN.BAS	●■
COLLECT.FRM	●■	FERTILIZ.FRM	●■	DEMLIB-A.BAS	●
FLOWDIR.FRM	●■	PESTDB.FRM	●■	+ DEMLIB-W.BAS	■
DATA.FRM	●■	PESTICID.FRM	●■	**Declarations & VBX:**	
EXPORTAS.FRM	●■	POINTSOU.FRM	●	WIN30API.BAS	●■
EDRANGES.FRM	●■	FEEDLOT.FRM	●	GLOBDEC.BAS	●■
DISPLEG.FRM	●■	NONFEED.FRM	●	MSOLE2.VBX	●■
DISPRANG.FRM	●■	IMPOUND.FRM	●	MSOUTLIN.VBX	●■
DISPGIO.FRM	●■	ADDEROS.FRM	●	CMDIALOG.VBX	●■
OUTPUT.FRM	●	CHANNEL.FRM	●	THREED.VBX	●■
SUMMARY.FRM	●	**Common Files:**		SPIN.VBX	●■
TRACE.FRM	●	AGNPSLIB.BAS	●	RMWIN4.VBX	●■
DUPGRID.FRM	●■	+ WATFLIB.BAS	■	ANIBUTTON.VBX	●■

VISUAL BASIC PSEUDOCODE

(Subroutines and functions with brief description)

Program Name: InterAGN
Module: INTERAGN.FRM

Sub CheckSerialNumber ()
‘ Verify serial number from distribution disk
‘ GetINIInfo : pHex : PutIniInfo

Sub cmdAnalScen_Click ()
‘ Activates the Analysis and Scenarios toolbar

Sub cmdASCII_Click ()
‘ Activates the Export ASCII file Procedure
‘ optMode(0):: ExportASCII.Show

Sub cmdCollect_Click ()
‘ Activates the Collect Data Toolbar

Sub cmdCollDat_Click ()
‘ Activates the Collect Data: Collect.Show

Sub cmdCreaDB_Click ()
‘ Activates the Create Structure:
‘ CreaDBAGNPS.Show

Sub cmdDispInput_Click ()
‘ Activates the Graphical Display Procedure
‘ DisplayGIO.Show

Sub cmdDupGrid_Click ()
‘ Activates the Duplicate Grid: DupGrid.Show

Sub cmdEditGrid_Click ()
‘ Activates the Grid Editor: EditGrid.Show

Sub cmdEditRanges_Click ()
‘ Activates the Range Editor: EditRanges.Show

Sub cmdEditSum_Click ()
‘ Activates the Cell Editor: DataForm.Show

Sub cmdExit_Click ()
‘ Exit the AGNPS Interface: Unload Me

Sub cmdFlowDir_Click ()
‘ Activates Flow Direction Editor: FlowDir.Show

Sub cmdMake_Click ()
‘ Activates the Grid Maker Toolbar

Sub cmdMakeGrid_Click ()
‘ Activates the Grid Maker: GridMaker.Show

Sub cmdModata_Click ()
‘ Activates the Landcover Editor: Landata.Show

Sub cmdResTable_Click ()
‘ Activates the Output Display: OutputForm.Show

Sub cmdResults_Click ()
‘ Activates the Graphic Display Toolbar

Sub cmdResume_Click ()
‘ Activates the Output Summary Form:
‘ ResumeRes.Show

Sub cmdRun_Click ()
‘ Activates the Run AGNPS Procedure
‘ optMode(1) : ExportASCII.Show

Sub cmdRunAGNPS_Click ()
‘ Activates the Run AGNPS Toolbar

Sub cmdSaveGrid_Click ()
‘ Creates an empty Database file

Sub cmdSensit_Click ()
‘ Activate the Sensitivity Analysis Form
‘ Sensitivity.Show

Sub cmdShell_Click ()
‘ Activates the AGNPS Shell

Sub cmdWatDat_Click ()
‘ Activates the Initial Data Form: InitialData.Show

Sub Form_Load ()
‘ Initializes Program by calling:
‘ CheckSerialNumber : ReadINIFile

Function GetINIInfo$ (section$, parm$)
' Gets a Field From intaglic.INI

Function makecheck% (a$)
' Calculate checksum for a string

Sub mnuAnalScen_Click ()
‘ Access through menu: cmdAnalScen_Click

Sub mnuAnalScenar_Click (Index As Integer)
‘ Access through menu:
‘ Case 1: cmdDupGrid_Click
‘ Case 2: cmdModata_Click
‘ Case 3: cmdSensit_Click

Sub mnuCollect_Click ()
‘ Access through menu: cmdCollect_Click

Sub mnuCollectEdit_Click (Index As Integer)
‘ Access through menu:
‘ Case 1: cmdWatDat_Click
‘ Case 2: cmdCollDat_Click
‘ Case 3: cmdFlowDir_Click
‘ Case 4: cmdEditSum_Click

Sub mnuDispInpOut_Click (Index As Integer)
‘ Access through menu:
‘ Case 1: cmdEditRanges_Click
‘ Case 2: cmdDispInput_Click
‘ Case 3: cmdResTable_Click

Sub mnuMake_Click ()
‘ Access through menu: cmdMake_Click

Sub mnuMakeEdit_Click (Index As Integer)
‘ Access through menu
‘ Case 1: cmdSaveGrid_Click
‘ Case 2: cmdMakeGrid_Click
‘ Case 3: cmdCreaDB_Click
‘ Case 4: cmdEditGrid_Click

Sub mnuResults_Click ()
‘ Access through menu: cmdResults_Click

Sub mnuRunAGNPS_Click ()
‘ Access through menu: cmdRunAGNPS_Click

Sub mnuRunModel_Click (Index As Integer)
‘ Access through menu
‘ Case 1: cmdASCII_Click
‘ Case 2: cmdRun_Click
‘ Case 3: cmdShell_Click

Function phex$ (i%, w%)
' Convert i to a fixed width hex string

Sub putlnilnfo (sect$, parm$, Val As Variant)
' These writes to the intaglic.INI

Program Name: InterAGN
Module: GRIDMAKE.FRM

Sub addbtn_Click ()
‘ Calculates and rotate the grid points and
‘ Adds the grid to the active database by calling:
‘ CheckGridName
‘ CreateLL_GridRectTable for Lat/Long grids
‘ CreateUTM_GridRectTable for UTM grids

Sub addcheck_Click (Value As Integer)
‘ If checked allows grid to be saved in the selected
‘ database. It will also allow to change active file by:
‘ cmdLoadDB

Sub Check3D1_Click (Value As Integer)
‘Sends Command to interact with RAISON
‘ SendCommand "[EV_MPCHLL]ON" -checked
‘ SendCommand "[EV_MPCHLL]OFF" -unchecked

Function CheckGridName (grNm As String)
' This function check to see if the gridname exists
' in the layer database. Returns true if name exists.

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1

Sub colour_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub Command3D1_Click ()
‘ Draws the Grid
‘ DrawGrid or DrawUTMGrid

Sub Command3D2_Click ()
‘ Closes current form
‘ Unload Me

Sub DrawGrid () and Sub DrawUTMGrid ()
‘ Sends Command to interact with RAISON
‘ a = "[PR_REDRAW]"
‘ Calculates and rotate the grid points and
‘ Draws the grid polygon and text
‘ a = "[MP_LLPOLY]" + parameters"
‘ a = a + "[MP_LLRTEXT]" + parameters”

Sub Eas_Change ()
‘ Updates UTM values from eastern change by:
‘ UpdateFromUTM

Sub FillUTMValues ()
‘ Fill UTM values from grid position by:
‘ ConvertGeotoUTM

Sub Form_Activate ()
‘Sends Command to interact with RAISON

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub grdSize_Change ()
‘ Updates UTM values from grid size change by:
‘ UpdateFromUTM

Sub Nor_Change ()
‘ Updates UTM values from northern change by:
‘ UpdateFromUTM

Sub SendCommand (ddecommand$)
‘Sends Command to RAISON
‘ form.object.LinkExecute ddecommand$

Sub Spin1_Spin Up/Down () and Spin2
‘ Increase/Decrease number of rows and columns

Sub Text2_LinkNotify ()
‘ Notify for RAISON link: and gets zone and datum when clicking in map: FillUTMValues

Sub UpdateFromUTM ()
‘ Update from lat/lon to UTM: ConvertUTMtoGeo

Program Name: InterAGN
Module: AGNPSCDB.FRM

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1

Sub Command3D1_Click ()

‘ Creates tables for the AGNPS structure by:
‘ CreateAGNPSTables & WriteClassesScheme

Sub Command3D2_Click ()
‘ Closes current form: Unload Me

Sub FillinClasses ()
' Selects lookup table for grouping landuse and
' Gets available classes schemes to fill combo box
‘ GetLookupDBName

Sub Form_Load ()
‘ Initializes form: FillinClasses

Sub GetLookupDBName ()
‘ Get lookup table file. Uses Dialog Box Action 1

Sub gridname_Click ()
‘ Selects and opens gridname from combo box and
‘ Verifies if tables exist if not prompts to create

Sub WriteClassesScheme ()
‘ Creates table for selected classes scheme

Program Name: InterAGN
Module: CONTROL.FRM

Sub AreaThres_Change ()
‘ Sets the area for automatic cell selection

Sub chkAutoSelect_Click (Value As Integer)
‘ Access the autoselection options

Sub cmdAutoSelect_Click ()
' Verify error layer & name shed: VerifyMatch
' Calculate grid intersection: FillPolyData
' Write to General Cell and Mark/Unmark Cell:
‘ MarkUnmarkAreas
‘ Delete from summary: RemoveSummaryData

Sub cmdDrawMaps_Click ()
‘ Draw selected layer
‘ SendCommand "[PR_CURWIN]"
‘ a = "[LY_DRAW]" + parameters"

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1
Sub cmdLoadMaps_Click ()
‘ Open available layer file. Uses Dialog Box Action 1
‘ FindGridTblLayers

Sub Command3D1_Click ()
‘ Refresh RAISON map
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + parameters”

Sub Command3D2_Click ()
‘ Closes current form: Unload Me

Function FillinGrids (dbname$) As Integer
‘ Fills the gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub GridAddCell_Click (Value As Integer)
‘ Send command to add cell

Sub GridDeleteCell_Click (Value As Integer)
‘ Send command to remove cell

Sub gridname_Click ()
‘ Selects gridname from combo box and
‘ Verifies if tables exist if not prompts to create

Sub GridSubCell_Click (Value As Integer)
‘ Send command to subdivide cell

Sub lblGridColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub lblWshdColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub MarkUnmarkAreas (dbGridName, GrdName)
‘ Based on summary results keep or remove cell

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub RemoveSummaryData (db$, gridname$)
‘ Query Database and remove all records

Sub SendCommand (ddecommand$)
‘Sends Command to RAISON

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:
‘ If EditGrid.DeleteCell : AGNPS_DeleteGridCell
‘ If EditGrid.SubCel : AGNPS_SubdivideCell
‘ If EditGrid.AddCell : AGNPS_AddGridRectCell

Sub VerifyMatch (Resp)
‘Verify that selected layer correspond to Watershed

Program Name: InterAGN
Module: INITIAL.FRM

Sub CalculateCellArea (db, grdn As String)
‘ Calculates cell area based on origin grid values

Sub CalculateEI ()
' Calculate EI value as function of Precipitation and
' Storm Type.....function is:
’ EI = c(dur^n)(prec^m)

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1

Sub cmdRain_Click ()
‘ Provides guide values for duration and intensity

Sub cmdSaveDB_Click ()
' Verify data and save input in DB: VerifyInput

Sub Command3D1_Click ()
‘ Refresh RAISON map
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + parameters”

Sub EnabledSelection ()
‘ Enables selection or automatic calculation of EI

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids
‘ FindGrids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Selects gridname, reads from database &
‘ Calculates Cell Area and number of cells in grid:
‘ Call CalculateCellArea

Sub InitializeForm ()
‘ Initialize form and reset values with defaults

Sub optEIVal_Click (Index%, Value%)
‘ Checks for selections and calculates EI values:
‘ EnabledSelection
‘ If optEIVal(1) : CalculateEI

Sub optHydCal_Click (Index%, Value%)
‘ Checks for hydrology options and defaults

Sub RainIntensity_Click ()
‘ Select Case for Rain Intensity Box

Sub SendCommand (ddecommand$)
‘Sends Command to RAISON

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:

Sub txtDuration_Change ()
‘ Duration Input - Send to calculate EI by
‘ CalculateEI

Sub txtPrecipitation_Change ()
‘ Precipitation Input - Send to calculate EI by
‘ If optEIVal(1) : CalculateEI

Sub txtStormType_Click ()
‘ Storm type selection - Send to calculate EI by
‘ If optEIVal(1) : CalculateEI

Sub VerifyInput ()
' Validate Input Data (Return RetryInput%)

Program Name: InterAGN
Module: COLLECT.FRM

Sub cmdCollect_Click ()
' Main procedure to collect data from maps
' Calculate grid intersection: Call FillPolyData
' Write to General Cell depending on lookup table
‘ Call WriteSoilDependent
' Delete grid summary table & compact database
‘ Call RemoveSummaryData
‘ CompactDatabase

Sub cmdDispLegend_Click ()
' Display legend form...: DispLegend.Show

Sub cmdDraw_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub cmdDrawElev_Click ()
‘ Draw selected layer (elevation or flow
‘ SendCommand "[PR_CURWIN]"
‘ a = "[LY_DRAW]" + parameters"

Sub cmdDrawMaps_Click ()
‘ Draw selected layer
‘ SendCommand "[PR_CURWIN]"
‘ a = "[LY_DRAW]" + parameters"

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdFlowDir_Click ()
‘ Calculates data from DEM file
‘ Call DoAGNPSFillData
Sub cmdLegendDEM_Click ()
' Set variables to display general numeric ranges

Sub cmdLoadDB_Click ()
‘ Open a Database File. Uses Dialog Box Action 1

Sub cmdLoadElev_Click ()
‘ Open a DEM File. Use Dialog Box Action 1

Sub cmdLoadMaps_Click ()
‘ Opens a Map File and allow selection of layer
‘ Uses Dialog Box Action 1

Sub cmdReduceDem_Click ()
' Add polygons in grid BOX to a new temporal
‘ database for DEM layers (DEM & PtDem):
‘ Call GetBoxCheckUTM
‘ e = NarrowRegionLayer

‘ Call CopyLayerCharacteristics

Sub cmdReduceMap_Click ()
' Add polygons in grid BOX to a new temporal
‘ database for MAP layers (Soil & Landuse):
‘ Call GetBoxCheckUTM

Sub comboFieldLook_Click ()
‘ Fills combo box for available fields

Sub comboLayerName_Click ()
‘ Fills combo box for available layers

Sub CopyLayerCharacteristics ()
‘ Copy the characteristics table for the database
‘ e = CopyLayersChar

Sub DrawFlow (dbnameE As String)
‘ Draw flow directions: DrawFlowArrow

Sub DrawFlowArrow()
‘ Calculate position point for arrow
‘ Send command to draw line in RAISON
‘ a = "[LY_DRAWARROW]" + “parameters”

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub FillInGridSummary()
' Fill Lookup value in Grid Summary

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub GetBoxCheckUTM()
' Select the Minimum x,y and Maximum x,y for the
‘ active grid = BOX and check for UTM

Sub GetLandDependantValues ()
' Get land values from lookup using map code
Sub GetLanduseSummaryValues ()
' Get land summary from table using map code

Sub GetLookupDBName ()
‘ Locate Lookup Database (lookup.mdb)

Sub GetPercentageCounters (ActiveFile)
' Resets & gets percent counter from record table

Sub GetSoilDependantValues ()
'Get soil type from lookup table using map code

Sub gridname_Click ()
‘ Select gridname, reads from database

Sub lblColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub lblGridColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub optDisplay_Click()
‘ Sets display options for DEM file

Sub optLookTable_Click ()
' Select lookup table to use for model data...

Sub RemoveSummaryData (db$, gridname$)
‘ Query Database and remove all records

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON
‘ form.object.LinkExecute ddecommand$

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:

Sub VerifyLookupMatch (Resp)
‘ Verifies that soil process is to be done first
‘ Check that layer match the active lookup table

Sub WaterValues ()
' Assign defaults for water to general cell table &
' Assign defaults for water to channel table

Sub WriteLandDependent ()
' Write landuse values on General Cell Data
‘ Reads percentage on summary table
‘ Call GetLandDependantValues
‘ Compares with limits and writes in cell table

Sub WriteLanduseSummary ()
' Gets classes types...
' Extract values from summary table...
' Write the values in the Landuse Summary table
Sub WriteSoilDependent ()
' Write soil dependent values on General Cell Data:
Call GetSoilDependantValues
‘ Compare with limits & writes in cell and soil table

Program Name: InterAGN
Module: FLOWDIR.FRM

Sub CheckStatus ()
‘ Check elevation status according to direction

Sub cmdCheck_Click ()
‘ Produce text report of elevation status in all cells
‘ Deletes old text reports and open editor to view

Sub cmdLoadDB_Click ()
‘ Open a Database File. Use Dialog Box Action 1

Sub Command3D1_Click ()
‘ Refresh RAISON map
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + parameters”

Sub Command3D2_Click ()
‘ Closes current form: Unload Me

Sub Command3D3_Click ()
‘ Draws flow direction: Call DrawFlow

Sub DrawFlow (dbnameE As String)
‘ Draw flow directions: DrawFlowArrow

Sub DrawFlowArrow()
‘ Calculate position point for arrow
‘ Send command to draw line in RAISON
‘ a = "[LY_DRAWARROW]" + “parameters”

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Selects gridname, reads from database &
' For river elevation update...Verifies if field exists

Sub lblcolorold_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub lblDir_Click (Index As Integer)
‘ Click in direction, change picture & assign the
‘ Flow Direction, Receiving Cell & Elevation

Sub lblElevFrom_DblClick ()
‘ Modify the elevation value for the selected cell &
‘ Save the elevation and recheck the status
Sub lblColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub ReadCellValues ()
‘ Reads flow direction, receiving cell & elevation
‘ for the selected cell and check status

Sub SelectCell_Click (Value As Integer)
‘ Clicks on map and select cell and grid number
‘ SendCommand "[EV_MPCHLL]ON"
‘ Returns selected grid number

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON
‘ form.object.LinkExecute ddecommand$
‘ And changes color of selected cell
‘ ddestr = "[LY_DROBJUSERDEF]" + “params”

Program Name: InterAGN
Module: DATA.FRM

Sub AGNPSSpread_Change (Col#, Row#)
‘ Edits spreadsheet & changes database table
' Function ChangeGridValue

Sub AGNPSSpread_Click (Col#, Row#)
' Avoids editing water cells

Sub AGNPSSpread_RightClick()
‘ Edit yellow range. Launch additional forms

Sub CellList_Click ()
' Select cell number from list & goto cell on spread

Sub cmdAddSpecific_Click ()
' Select cells according to specific criteria
‘ If optSpecify(0) "Soil_Texture"
‘ If optSpecify(1) "Fert_Ind"
‘ If optSpecify(2) "Pest_Ind"
‘ If optSpecify(3) "Point_Ind"
‘ If optSpecify(4) "Add_Erosion"
‘ If optSpecify(5) "Impound_Ind"
‘ For selected options add cell to list & to spread
‘ Finally read grid values from database
‘ CellList.AddItem (selGridNumber)
‘ SpreadAddCell
‘ ReadGridValues

Sub cmdCancel_Click ()
‘ Cancels search by criteria

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1
Sub cmdUpdate_Click ()
' Initialize Spreadsheet and updates from list by:
‘ selGridNumber
‘ SpreadAddCell
‘ ReadGridValues

Sub Command3D1_Click ()
‘ Refresh RAISON map
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + parameters”

Sub Command3D2_Click ()
‘ Closes current form: Unload Me

Sub Command3D3_Click ()
' Select All of the cells in the grid
‘ CellList.AddItem (selGridNumber)
‘ SpreadAddCell
‘ ReadGridValues

Sub Command3D4_Click ()
' Initialize Spreadsheet: CellList.Clear : InitSpread()

Sub Command3D5_Click ()
' Initialize spreadsheet list and combo boxes
‘ CellList.Clear : InitSpread() : Add Items

Function FillinGrids (dbname$) As Integer

‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Selects gridname, reads from database &
' Initializes the spreadsheet

Function InitSpread () As Integer
‘ Initializes the spreadsheet variables names

Sub lblGridColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub ReadGridValues ()
‘ Reads grid values from database
‘ Uses spreadsheet names as table fields

Sub SelectCell_Click (Value As Integer)
‘ Clicks on map and select cell and grid number
‘ SendCommand "[EV_MPCHLL]ON"
‘ Returns selected grid number

Sub SpreadAddCell ()
' Adds Column to Spread from ListCell and
' Prevents block of aditional info cells to be edited
Sub Text2_LinkNotify ()
‘ Notify for RAISON link:
‘ If selected : Removes cell from list & spread
‘ Else add cell to list & to spreadsheet

Program Name: InterAGN
Module: EXPORTAS.FRM

Sub CheckMode ()
‘ Checks mode for export data or run the model

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdExport_Click ()
‘ Open file for output & writes formated data
‘ WriteInitialData & WriteCellData

Sub cmdFile_Click ()
‘ Select ASCII export file. Use Dialog Box Action 2

Sub cmdLoadDB_Click ()
‘ Open a Database File. Uses Dialog Box Action 1

Sub cmdRun_Click ()
‘ If run mode: activates the model executable file
‘ First exports data in model format
‘ WriteInitialData & WriteCellData
‘ Then run the model by activating shell command
‘ When terminated import the model ouput: Import

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Selects gridname, reads from database &
' For runing date...Verifies if field exists

Sub Import ()
‘ Opens temporal model output file to read and
‘ Write into the current database
‘ Watershed Summary..WriteWatershedSummary
' Sediment Analysis...WriteSedimentAnalysis
' Hydrology & Sediments...WriteHydroSediments
' Nutrients...WriteNutrients
' Pesticide...RWPesticide
' If Source_Deposition table does’nt exists, create
‘ Call ImportBinary_SRC_DEP

Sub ImportBinary_SRC_DEP ()
' Reads binary .SRC..Cell Potential Contributions
' Reads and writes the SRC file
' Reads binary .DEP file...Deposition Cell Values
' Reads and writes the DEP file
Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub RWAddErosValues (selGridNumber)
' Read Add_Erosion from Add_Erosion Table
' Do numeric format...and Writes to ASCII file

Sub RWChanValues (selGridNumber)
' Read Channel values from Channel Table
' Do numeric format...and Writes to ASCII file

Sub RWFeedlotValues (selGridNumber)
' Read Feedlot values from Feedlot Table
' Do numeric format...and Writes to ASCII file

Sub RWFertValues (selGridNumber)
' Read Fertilizer values from Fertilizer Table
' Do numeric format...and Writes to ASCII file

Sub RWGenCellValues (selGridNumber)
' Read Grid values from General Table
' Do numeric format...and Writes to ASCII file
‘ Soil_Texture : RWSoilValues (selGridNumber)
‘ Fert_Ind : RWFertValues (selGridNumber)
‘ Pest_Ind : RWPestValues (selGridNumber)
‘ NonFeed : RWNonFeedValues (selGridNumber)
‘ Feedlot : RWFeedlotValues (selGridNumber)
‘ Add_Eros : RWAddErosValues (selGridNumber)
‘ Imp_Ind : RWImpoundValues (selGridNumber)
‘ RWChanValues (selGridNumber)

Sub RWHydroValues (selGridNumber)
‘ Import hydrology output and store in database

Sub RWImpoundValues (selGridNumber)
' Read Impoundment values from Impoundment
' Do numeric format...and Writes to ASCII file

Sub RWNonFeedValues (selGridNumber)
' Read Non-Feedlot values fom NonFeedlot Table
' Do numeric format...and Writes to ASCII file

Sub RWNutrients (selGridNumber)
‘ Import nutrients output and store in database

Sub RWPesticide ()
' Read pesticide for the cells & write in PesticideR

Sub RWPestValues (selGridNumber)
' Read Pesticide values from Pesticide Table
' Do numeric format...and Writes to ASCII file

Sub RWSedimValues (selGridNumber)
‘ Import sediment output and store in database

Sub RWSoilValues (selGridNumber)
' Read Soil values from Soil Table...
' Do numeric format...and Writes to ASCII file...
Sub WriteCellData ()
' Send to Read/Write values with selGridNumber
‘ RWGenCellValues (selGridNumber)

Sub WriteHydroSediments ()
' Read/write hydrology and soil loss for all cells:
‘ RWHydroValues (selGridNumber)
‘ RWSedimValues (selGridNumber)

Sub WriteInitialData ()
' Read from Database...for Initial Watershed Data
' Do numeric format...and Writes to ASCII file

Sub WriteNutrients ()
‘ Import by reading nutrients
' Read NUTRIENTS and write in Nutrients table

Sub WriteSedimentAnalysis ()
‘ Import by reading sediments
' Read SEDIMENT and write in Sediment Analysis

Sub WriteWatershedSummary ()
‘ Import initial and summary results
' Read INITIAL and write in Watershed Summary
‘ Updates running time. If required, create field.

Program Name: InterAGN
Module: EDRANGES.FRM

Sub blue_Change ()
‘ Changes blue color value

Sub CalcRanges ()
‘ Calculate Ranges based on increment

Sub chkExtremes_Click (Value As Integer)
‘ Allow spliting ranges in the extremes values

Sub chkRanges_Click (Value As Integer)
‘ Allow edition of ranges

Sub cmdColorScale_Click ()
‘ Process the color ramp with current color values

Sub cmdCreateNew_Click ()
‘ Create new legend for selected field.
‘ Save current ranges information to new table

Sub cmdDefRang_Click ()
‘ Uses default ranges & colors to display legends

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1
Sub cmdSave_Click ()
' Deletes all of the current characteristics...
‘ Then Add Characteristics for selected values

Sub comboFieldLook_Click ()
‘ Selects field to use

Sub comboLegend_Click ()
' Sort the range list : SortRange
‘ Selects legend to use

Sub DeleteCharAllMultIndex ()
' Delete all from characteristic table

Function FillinGrids (dbname$) As Integer
‘ Fill gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub green_Change ()
‘ Changes green color value

Sub gridname_Click ()
‘ Select gridname & Initialize form

Sub InitializeForm ()
‘ Reset form by erasing labels and setting controls

Sub optLayer_Click (Index%, Value As Integer)
‘ Selects layer to use:
‘ Input options:
"General_Cell" "Soil" "Fertilizer" "Pesticide"
‘ Output options:
"Hydrology" "Sediments" "Nutrients" "PesticideR"

Sub Picture2_MouseMove ()
‘ Displays color for active range

Sub ReadCellValues ()
‘ Reads cell values from database
‘ Search maximum, minimum and extreme values

Sub red_Change ()
‘ Changes red color value

Sub SortRange ()
‘ Sort range to get RngTo and RngFrom values

Sub txtNoRanges_Change ()
‘ Changes the number of ranges to use

Sub txtRngMax_Change ()
‘ Changes the maximum value

Sub txtRngMin_Change ()
‘ Changes the minimum value

Program Name: InterAGN Module: DISPLEG.FRM

Sub ColorBox_Click (Index As Integer)
‘ Sets Color by using Dialog Box Action 3

Sub Form_Load ()
' Display legend form...

Sub GetLandCover (MCode, LookText)
' Gets landuse from lookup table using map code

Sub GetSoilTexture (MCode, LookText)
'Gets soil type from lookup table using map code

Program Name: InterAGN Module: DISPRANG.FRM

Sub Form_Load ()
' Display range form...
' Sort the range list : SortRange

Sub SortRange ()
‘ Sort range to get RngTo and RngFrom values

Program Name: InterAGN Module: DISPGIO.FRM

Sub cmdDraw_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub cmdDrawLayer_Click ()
‘ Draw selected layer
‘ SendCommand "[PR_CURWIN]"
‘ a = "[LY_DRAW]" + parameters"

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdLegend_Click ()
' Set variables to display numeric ranges
‘ DisplayRanges.Show

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1

Sub cmdRedraw_Click ()
‘ Refresh map : SendCommand "[PR_REDRAW]"

Sub comboFieldLook_Click ()
‘ Selects field to use

Sub comboTable_Click ()
‘ Selects table to use

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Select gridname, reads from database

Sub lblGridColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub optLayer_Click (Index%, Value As Integer)
‘ Selects layer to use Input/Output tables

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:

Program Name: InterAGN Module: OUTPUT.FRM

Sub CellList_Click ()
' Select cell number and go to cell on spread

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1

Sub comboTable_Click ()
‘ Initialize Spreadsheet and selects table to use

Sub Command3D1_Click ()
‘ Draw selected Grid

‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub Command3D2_Click ()
‘ Closes current form: Unload Me

Sub Command3D3_Click ()
' Select All of the cells in the grid
‘ CellList.AddItem (CStr(selGridNumber))
‘ SpreadAddCell
‘ ReadGridValues

Sub Command3D4_Click ()
' Initialize Spreadsheet and clears all

Sub Command3D5_Click ()
‘ Display summary table : Summary.Show

Function FillinGrids (dbname$) As Integer
‘ Fill gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Select gridname, reads from database &
' Initialize Spreadsheet

Function InitSpread () As Integer
‘ Reset spreadsheet depending on table name:
‘ Hydrology : Sediments : Nutrients : Pesticides

Sub lblGridColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub ReadGridValues ()
‘ Reads grid values from database
‘ Uses spreadsheet names as table fields

Sub SelectCell_Click (Value As Integer)
‘ Clicks on map and select cell and grid number
‘ SendCommand "[EV_MPCHLL]ON"

Sub SpreadAddCell ()
' Adds Column to Spread from ListCell and
' Prevents block of aditional info cells to be edited

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:
‘ If selected, remove cell from list and spreadsheet
‘ Else add cell to list and to spreadsheet
‘ CellList.Addtem : SpreadAddCell :
ReadGridValues

Program Name: InterAGN
Module: SUMMARY.FRM

Sub Form_Load ()
‘ Initialize form and read summary data and output:
' Read from the Watershed Summary Table...
' Read from the Sediment Analysis Table...
‘ Fills all text labels with variable values...

Program Name: InterAGN
Module: TRACE.FRM

Sub blue_Change ()
‘ Changes blue color value
Sub CalculateContributions ()
' Select options and read data
‘ If Hydrology then Amount is‘ returned directly :
‘ DefineSourceOptions
' Calculate Sediment Yields. Returns cell_yield,
‘ cell_gen, cell_flowin. Then Calculate Delivery
‘ Ratios and percentage of contribution
' Receives Amount returned and display results

Sub CalculateDeliveryRatios ()
' Calculate Cell Delivery Ratios: Cell_Yield,
Cell_GenIn, Cell_FlwIn
' Aggregate Downstream Delivery Ratio for draining
cells (recursive)
' Calculate Downstream Delivery Ratio & Amounts

Sub CalculateSedimentYields ()
' Calculate Yields and Flowing in values
' Accumulate amounts in for receiving cells (rec)
' Copy Results to Generic Variables

Sub CalculateSedNutrientYields ()
' Calculate Yields and Flowing in values
' Accumulate amounts in for receiving cells (rec)

Sub CalculateSolNutrientYields ()
' Calculate Yields and Flowing in values
' Accumulate amounts in for receiving cells (rec)

Sub chkSpaDisp_Click (Value As Integer)
‘ Access spatial display options

Sub cmdCalculate_Click ()
‘ Sends to : CalculateContributions

Sub cmdLoadDB_Click ()
‘ Open a Database File. Use Dialog Box Action 1

Sub cmdColorScale_Click ()
‘ Process the color ramp with current color values

Sub Combo1_Click ()
‘ Controls combo selection for options

Sub Combo2_Click ()
‘ Selects source and initialize spread :
‘ InitBrfResultSpread

Sub comboTable_Click ()
‘ Selects table to use

Sub Command3D1_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub Command3D2_Click ()
‘ Closes current form: Unload Me
Sub Command3D3_Click ()
‘ Draw traced flow : Call DrawFlow

Sub Command3D4_Click ()
' Initialize Spreadsheet : InitAccountSpread
' Redraw Grid
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub Command3D5_Click ()
' Select cells on spreadsheet that fall inside range
' Change color of selected cells...
‘ ddestr = "[LY_DRGRIDCELLUSERDEF]"

Sub DefineSourceOptions ()
‘ Define source options for contributions:
' Hydrology : pass values from Spread to Amount()
‘ Depending on combo options:
' Read data from spreadsheet.
‘ Redimension variables...
' Select rows for input spreadsheet data...

Sub DrawFlow (dbnameE As String)
‘ Draw flow directions for the grid
‘ DrawFlowArrow

Sub DrawFlowArrow()
‘ Calculate position point for arrow
‘ Send command to draw line in RAISON
‘ a = "[LY_DRAWARROW]" + “parameters”

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub FixBrfResultSpread ()
' Set floating for sort and prevent block to be edited

Sub Form_Load ()
‘ Initializes form and test for RAISON link:
' Initialize Spreadsheets : InitSpread : Call InitAccountSpread

Sub green_Change ()
‘ Changes green color value

Sub gridname_Click ()
‘ Selects gridname, reads from database &
' Initialize Spreadsheet

Sub InitAccountSpread ()
‘ Initializes account spreadsheet & format cells

Sub InitBrfResultSpread ()
‘ Initializes results spreadsheet and format cells

Sub InitCombos ()
‘ Resets combos contents and initialize spread InitBrfResultSpread
Function InitSpread () As Integer
‘ Reset spreadsheet depending on table name:
‘ Hydrology : Sediments : Nutrients

Sub lblColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub ReadGridValues ()
‘ Reads grid values from database

Sub red_Change ()
‘ Changes red color value

Sub SelectCell_Click (Value As Integer)
‘ Clicks on map and select cell and grid number
‘ SendCommand "[EV_MPCHLL]ON"

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON

Sub SpreadAddCell ()
' Adds Column to Spread from ListCell

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:
' Change color of selected cell...
‘ ddestr = "[LY_DRGRIDCELLUSERDEF]"

Sub TraceContributions ()
‘ Create snapshot of receiving cell by query
‘ From cells on query result:
‘ SpreadAddCell : ReadGridValues
‘ Call DrawFlow(

Sub txtNoRanges_Change ()
‘ Changes number of ranges on spatial display

Sub txtRngMax_Change ()
‘ Changes the maximum value

Sub txtRngMin_Change ()
‘ Changes the minimum value

Program Name: InterAGN
Module: DUPGRID.FRM

Sub cmdDuplicate_Click ()
' Checks if grid already exists...

' If not exists then duplicate grid...CreateCopyGrid
‘ When done copy grid input data...CopyGridData

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default

Sub CopyGridData (db, grid, NewGrd)
‘ Assign table names:
' For all the grid and nongrid tables...copy data

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form

Sub gridname_Click ()
‘ Selects and opens gridname

Sub NewGrdName_Change ()
‘ Checks for valid grid name & enable duplication

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Program Name: InterAGN
Module: LANDATA.FRM

Sub CellList_Click ()
' Select cell number and go to cell on spread

Sub ChangeBackColor ()
‘ Changes back color of rows in spreadsheet

Sub cmbFrom_Click ()
‘ Selects landclass to change % from

Sub cmbTo_Click ()
‘ Selects landclass to change % to

Sub cmdApply_Click ()
‘ Proceed to change % of landcover “from-to”
‘ ChangeBack - Calculate changes - ResetBack

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default
‘ Uses Dialog Box Action 1

Sub cmdRecalc_Click ()
‘ Calculate land values based on class grouping
‘ Call GetTypicalValues
' Calculate for active cells with typical values
' Write values in database for the active cells
‘ Call WriteLandParameters

Sub cmdUpdate_Click ()
' Initialize Spreadsheet and from cells in list:
‘ SpreadAddCell
‘ ReadGridValues

Sub Command3D1_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub Command3D2_Click ()
‘ Closes current form: Unload Me

Sub Command3D3_Click ()
' Select All of the cells in the grid

Sub Command3D4_Click ()
' Initialize Spreadsheet

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form

Sub GetTypicalValues ()
' Reads Classes Scheme from Initial Data...
' Gets classes types...

Sub gridname_Click ()
‘ Selects and opens gridname : Initialize Spreadsheet

Function InitSpread () As Integer
' Reads Classes Scheme from Initial Data...
' Gets classes types...
' Add classes types according to lookup and selection...

Sub lblGridColor_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub optChange_Click (Index%, Value%)
‘ Changes percentage by option or text input

Sub ReadGridValues ()
‘ Reads grid values from database
‘ Use spreadsheet column names as field names

Sub ResetBackColor ()
' Reset FROM & TO cells colors to spread back

Sub SelectCell_Click (Value As Integer)
‘ Clicks on map and select cell and grid number
‘ SendCommand "[EV_MPCHLL]ON"
‘ Returns selected grid number

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON

Sub SpreadAddCell ()
' Adds Column to Spread from ListCell

Sub SummarySpread_Change (Col#, Row#)
‘ Change value on spread & save on database

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:
' Change color of selected cell...
‘ ddestr = "[LY_DRGRIDCELLUSERDEF]"

Sub WriteLandParameters ()
' Open database and tables.....
‘ With the values calculated from the typical data:
' Write the values in the General Cell table for the selected grid...

Program Name: InterAGN
Module: RESUME.FRM

Sub cmdAddAll_Click ()
' Initialize Spreadsheet : InitSpread

Sub cmdClearAll_Click ()
' Initialize Spreadsheet : InitSpread

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default

Function FillinGrids (dbname$) As Integer
‘ Fill gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and spreadsheet
‘ Clears combo & fill with grids on file: FillinGrids

Sub gridname_Click ()
‘ Select gridname : Initialize Spreadsheet

Function InitSpread () As Integer
‘ Reset spreadsheet depending on table name:
‘ Column names on spreadsheet as fields names

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub ReadResults (gridn)
' Read from the Watershed Summary Table...

Sub SpreadAddCell ()
' Adds Column to Spread from ListCell
' Prevents block to be edited (data) and (results)
' Resets position

Program Name: InterAGN
Module: SENSANAL.FRM

Sub CalculateVarGrad()
‘ Calculate mean normalized sensitivity gradient

Sub cmdDraw_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdLoadDB_Click ()
‘ Opens a Database File and assigns it as default

Sub cmdNormGrad_Click ()
‘ VerifySensTbl - If table does not exist; create it
' Open Prepare Sensitivity Input Window
‘ SensInput.Show 1 (modal 1)

Sub cmdRankGrad_Click ()
‘ Open Ranki window : NewSensRankGrad.Show

Sub cmdRunSens_Click ()
' Exports ASCII data and prepares BAT file...
' Run DOS batch file...
' Import output for batch file: ImportOutput
‘ Calculate variation and write in table...
‘ WriteDateSensRun

Sub ExportData ()
‘ Open file for output & write data in model format
‘ For perturbed variables read parameters &
‘ percentages from sensitivity table:
‘ WriteInitialData & WriteCellData
' Prepare BAT file

Function FillinGrids (dbname$) As Integer
‘ Fills gridname combo box with available grids

Sub Form_Load ()
‘ Initializes form and test for RAISON link:

Sub gridname_Click ()
‘ Selects and opens gridname : InitializeForm
' Read from Watershed Summary Table and Sensitivity Table...

Sub ImportOutput ()
' Read base case output data from watershed summary table....
' Read parameters and percentages from summary table..until EOF
' Calculate variation and gradients...write to sensitivity table

Sub InitializeForm ()
‘ Resets label values

Sub lblGridColor_Click ()

‘ Sets Color by using Dialog Box Action 3

Sub OpenDBFile ()
‘ Opens Database File and Sets Default Name

Sub RWAddErosValues (f1, f2, Par, Low, High)
' Reads Add_Erosion from Add_Erosion Table
' Does the Numeric Format for file...
' No sensitivity parameters for Additional Erosion
' Writes ASCII files...1 for Low / 2 for High

Sub RWChanValues (f1, f2, Par, Low, High)
' Reads Channel Values From Channel Table...
' Does the Numeric Format for file...
' Calculate Low and High for Channel Data
' Writes ASCII files...1 for Low / 2 for High

Sub RWFeedlotValues (f1, f2, Par, Low, High)
' Reads Feedlot Values From Feedlot Table
' Does the Numeric Format for file
' No sensitivity parameters for Feedlots
' Writes ASCII files...1 for Low / 2 for High

Sub RWFertValues(f1, f2, Par, Low, High)
' Reads Fertilizer Values From Fertilizer Table...
' Does the Numeric Format for file...
' Calculate Low and High for Soil Data
' Writes ASCII files...1 for Low / 2 for High

Sub RWGenCellValues (f1, f2, Par, Low, High)
' Reads Grid Values From General Table...
' Does the Numeric Format for file...
' Calculate Low and High for Cell Data
' Writes ASCII files...1 for Low / 2 for High
‘ Soil_Texture : RWSoilValues (selGridNumber)
‘ Fert_Ind : RWFertValues (selGridNumber)
‘ Pest_Ind : RWPestValues (selGridNumber)
‘ NonFeed : RWNonFeedValues (selGridNumber)
‘ Feedlot : RWFeedlotValues (selGridNumber)
‘ Add_Eros : RWAddErosValues (selGridNumber)
‘ Impnd_Ind : RWImpoundValues (selGridNumber)
‘ RWChanValues (selGridNumber)

Sub RWImpoundValues (f1, f2, Par, Low, High)
' Reads Impoundment Values From Impoundment
' Does the Numeric Format for file...
' No sensitivity parameters for Impoundments...
' Writes ASCII files...1 for Low / 2 for High

Sub RWNonFeedValues (f1, f2, Par, Low, High)
' Read Non-Feedlot values from NonFeedlot
' Does the Numeric Format for file...
' No sensitivity parameters for Non-feedlots...
' Writes ASCII files...1 for Low / 2 for High
Sub RWPestValues(f1, f2, Par, Low, High)
' Reads Pesticide Values From Pesticide Table...
' Does the Numeric Format for file...
' No sensitivity parameters for pesticide...
' Writes ASCII files...1 for Low / 2 for High

Sub RWSoilValues (f1, f2, Par, Low, High)
' Reads Soil Values From Soil Table...
' Does the Numeric Format for file...
' Calculate Low and High for Soil Data
' Writes ASCII files...1 for Low / 2 for High

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON

Sub Text2_LinkNotify ()
‘ Notify for RAISON link:
' Change color of selected cell...
‘ ddestr = "[LY_DRGRIDCELLUSERDEF]"
‘ SendCommand ddestr

Sub VerifyFieldExist ()
' For LastSensRunDate update...Verifies if exists
' If field does not exist, then create it

Sub VerifySensTbl ()
' Check if Sensitivity table exists, if not create it...
‘ CreateAGNPSSens(dbname, gridname)

Sub WriteCellData (f1, f2, Par, Low, High)
' Send to Read/Write cell data..Call RWGenCellValues

Sub WriteInitialData (f1, f2, Par, Low, High)
' Read from Database...for Initial Watershed Data
' Does the Numeric Format for file...
' Calculate Low and High for Initial Data
' Writes ASCII files...1 for Low / 2 for High

Program Name: InterAGN
Module: SENSINPU.FRM

Sub cmbGroup_Click ()
‘ Select variables according to data group:
‘ Initial Data, General Cell, Soil, Fertilizer and Channel

Sub cmdAdd_Click ()
' If an item is selected, add it to spreadsheet
‘ Call SpreadAddParam

Sub cmdAddAll_Click ()
' Add all listed parameters to spreadsheet
‘ Call SpreadAddParam

Sub cmdCancel_Click ()
‘ Closes current form: Unload Me
Sub cmdClear_Click ()
' Initialize Spreadsheet: InitParamSpread

Sub cmdDel_Click ()
' If an item is selected, delete it from spreadsheet
‘ Use delete action from spreadsheet...

Sub cmdDelAll_Click ()
' Delete all listed parameters from spreadsheet

‘ Use delete action from spreadsheet...

Sub cmdGlobal_Click ()
‘ Assign % of variation to all listed variables...

Sub cmdSave_Click ()
' Verify that all LowPct and HighPct have values...
' Save to Database and Exit...
‘ Call RemovSensitivityData
‘ Call WriteSensitivityData

Sub Form_Load ()
' Initialize combo Group: InitComboGroup
' Initialize Spreadsheet: InitParamSpread
' Read Sensitivity Table: ReadSensTable

Sub InitComboGroup ()
' Initialize combo Group

Sub InitParamSpread ()
' Initialize Spreadsheet

Sub optPct_Click (Index%, Value As Integer)
‘ Assign perturbation percentage by case or input

Sub ParamList_DblClick ()
' Add/Remove Parameter to spreadsheet
‘ Call SearchExists(Index, Add)
‘ If Add : Call SpreadAddParam
‘ Else use delete action from spreadsheet...

Sub Params_Change (Col#, Row As Long)
‘ Change percentage value and check if it’s valid

Sub PertPerc_Change ()
‘ Change percentage value on input box

Sub PertPerc_Click ()
' Reset optPct values to none...

Sub ReadSensTable ()
' Read Sensitivity Table...
‘ Call SpreadAddParam

Sub RemovSensitivityData (db$, gridname$)
‘ Query database and remove data from table

Sub SearchExists (Index%, Add%)
' Search if already has being added to spread
Sub SpreadAddParam (Par, Low, High)
'Adds Column to Spread from ListCell

Sub WriteSensitivityData (db$, gridname$)
‘ Write sensitivity input in database sensitivity table

Program Name: InterAGN
Module: SENSNORM.FRM

Sub Check1_MouseMove
‘ Display database and grid being passed

Sub Check2_Click ()
‘ Display legend options (load and make visible)

Sub cmbVarsOut_Click ()
‘ Select output variables according to field names
‘ ReadOutputValues
‘ DrawGraphic

Sub cmdAll_Click ()
‘ Mark all available parameters

Sub cmdClose_Click ()
‘ Close legend options and - DrawGraphic

Sub cmdExit_Click ()
‘ Closes current form: Unload Me

Sub cmdNone_Click ()
‘ Unmark all available parameters

Sub DrawGraphic ()
' Draw the Normalized Gradients

Sub Form_DragDrop (Src , X, Y)
‘ Drag rectangle to zoom on selected area

Sub Form_Resize ()
‘ Resizes form, controls and set new positions

Sub InitGraph ()
' Reset Graph Settings

Sub lblBackCol_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub Picture1_MouseDown
' Reset zoom to original setting with right button
' Prepare for dragging
‘ Picture1.DrawMode = 10 NOT_XOR_PEN

Sub Picture1_MouseMove
' Start dragging a rectangle

Sub Picture1_MouseUp
' End drag Picture1.DrawMode = 13 COPY_PEN
Sub ReadOutputValues ()
' Read LOW and HIGH values for the output

Sub ReadSensitivityTable ()
' Read Sensitivity Table...Assign random colors

Sub Scroll_Change ()
‘ Move options up/down according to scroll control

Program Name: InterAGN
Module: SENSRANK.FRM

Sub Check1_MouseMove
‘ Display database and grid being passed

Sub Check2_Click ()
‘ Display legend options (load and make visible)

Sub cmbVarsOut_Click ()
‘ Select output variables according to field names
‘ ReadOutputValues
‘ DrawGraphic

Sub cmdAll_Click ()
‘ Mark all available parameters

Sub cmdClose_Click ()
‘ Close legend options and - DrawGraphic

Sub cmdExit_Click ()
‘ Closes current form : Unload Me

Sub cmdNone_Click ()
‘ Unmark all available parameters

Sub cmdOptions_Click ()
‘ Make options visible

Sub Command1_Click ()
‘ Draw rank graphic : DrawGraphic

Sub DrawGraphic ()
' Draw the Mean Normalized Gradients

Sub Form_Resize ()
‘ Resize form, controls and calculate new positions

Sub InitGraph ()
' Reset Graph Settings

Sub lblBackCol_Click ()
‘ Sets Color by using Dialog Box Action 3

Sub Picture1_MouseDown
' Zoom out-left(1) in-right(2)
‘ When done redraw : DrawGraphic

Sub ReadOutputValues ()
' Read GRAD values for the output variable...

Sub ReadSensitivityTable ()
' Read Sensitivity Table...Assign random colors

Sub Scroll_Change ()
‘ Move legend up and down according to scroll

Program Name: InterAGN Module: SOIL.FRM

Sub cmdApply_Click ()
' Verify data input except for water selection
' Write values on database for selected or all cells

Sub cmdCancel_Click ()
‘ Closes current form: Unload Me

Sub cmdDefaults_Click ()
‘ Reset to default values

Sub DispWaterValues ()
' Spreedsheet Display of Water Values for all...

Sub Form_Load ()
‘ Initialize form and read data : ReadCellData

Sub ReadCellData ()
‘ Read soil related data...

Sub SoilType_Click ()
‘ Change the soil type for the selected cells

Sub VerifyInput ()
' Validate Input Data (Return RetryInput%)

Sub WaterValues
‘ Change Channel_Ind, Fert_Ind, Pest_Ind, LandSlope to 0...
‘ Change SCS_No to 100, Manning's n to 0.99, SlopeLength to 0...
‘ Change K_factor, C_factor, P_factor, SurfCond, COD_factor to 0...
' Assign defaults for water to channel table...

Program Name: InterAGN Module: FERTILIZ.FRM

Sub cmdApply_Click ()
' Verify data input except for OFF
' Write values on database for selected or all cells

Sub cmdCancel_Click ()
‘ Closes current form: Unload Me
Sub optFertInd_Click (Index, Value As Integer)
' Null values for fertilization if off selected...

Sub optLevel_Click (Index, Value As Integer)
' Default values for fertilization rates...

Sub ReadCellData ()
‘ Read fertilizer related data...

Sub VerifyInput ()
' Validation of Input Data (Return RetryInput%)

Program Name: InterAGN Module: PESTDB.FRM

Sub cmdAccept_Click ()
‘ Loads pesticide data for selected pesticide type

Sub cmdCancel_Click ()
‘ Closes current form: Unload Me

Sub Form_Load ()
‘ Access pesticide database for list of Available Pesticides

Sub GetPestDBName ()
‘ Locate Pesticide Database (pestic.mdb)
CMDialog1.Filename = "*.mdb"

Sub List1_Click ()
‘ Read from database ("Pesticide Names")

Sub List2_Click ()
‘ Read from database ("Pesticide Tradename")

Program Name: InterAGN
Module: PESTICID.FRM

Sub cmdApply_Click ()
' Verify data input except for OFF
' Write values on database for selected or all cells

Sub cmdCancel_Click ()
‘ Closes current form : Unload Me

Sub cmdDefaults_Click ()
‘ Select Pesticide Type and access the pesticide DB
‘ PestDB.Show

Sub cmdDefltPest_Click ()
‘Reset to default values

Sub Form_Load ()
‘ Initializes form and fill combo with pesticide type

Sub optPestInd_Click (Index%, Value As Integer)
' Null values for pesticides if off selected...

Sub optTimeAp_Click (Index, Value)
‘ Enable/disable options. Depends: application time

Sub ReadCellData ()
‘ Read pesticide related data...

Sub VerifyInput ()
' Validate Input Data (Return RetryInput%)
' Check by groups depending on time of application

Program Name: InterAGN
Module: FEEDLOT.FRM

Sub AddPntScr_Click (Value As Integer)
‘ Add point source and send command
‘ SendCommand "[EV_MPCHLL]ON"

Sub CellList_Click ()
‘ Select cell from listbox & make it the active one

Sub ClearValues ()
' Clear values from data form

Sub cmdAccept_Click ()
' Verify data input & write values for selected cells

Sub cmdAdd_Click ()
' Can't fill in any data until "apply" or create the data points: GetValues : CellList_Click

Sub cmdClear_Click ()
‘ Clear cell list

Sub cmdDel_Click ()
‘ Delete Layer Object

Sub cmdDeleteData_Click ()
‘ Delete data in cell(s)

Sub cmdDraw_Click ()
‘ Draw Grid - SendCommand "[PR_REDRAW]"
‘ a = "[LY_DRAWGRID]" + “parameters"

Sub cmdFillZ_Click ()
' Fill with zeros the rest of the data...

Sub cmdPropag_Click ()
' Verify data input and proceed with changes...
' Change all source values with the displayed data...

Sub cmdShowCells_Click ()
‘ Send command to display point sources
‘ ddestr = "[LY_DRAWUSERDEF]" + “parameters"

Sub cmdUpdate_Click ()
' Verify data input and write form values to database

Sub Form_Load ()
' Define variables before the form will work..
Identify
‘ DDE share established by source application

Sub GetValues ()
‘ Get Layer Object Values

Sub optBuffer_Click (Index, Value As Integer)
‘ Enable/disable options for buffer area selection

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON

Sub Text2_LinkNotify ()
‘ Notify RAISON & change color of selected cell
' Remove cell from list/map or add cell to list/map

Sub VerifyInput ()
' Validate Input Data (Return RetryInput as Integer)

Program Name: InterAGN
Module: NONFEED.FRM

Sub AddPntScr_Click (Value As Integer)
‘ Add point source and send command
‘ SendCommand "[EV_MPCHLL]ON"

Sub CellList_Click ()
‘ Select cell from listbox and make it the active one

Sub ClearValues ()
' Clear values from data form

Sub cmdAccept_Click ()
' Verify data input & write values for selected cells

Sub cmdAdd_Click ()
' Can't fill in any data until "apply" or create the data points: GetValues : CellList_Click

Sub cmdClear_Click ()
‘ Clear cell list

Sub cmdDel_Click ()
‘ Delete Layer Object

Sub cmdDeleteData_Click ()
‘ Delete data in cell(s)

Sub cmdDraw_Click ()
‘ Draw Grid: SendCommand "[PR_REDRAW]"

Sub cmdPropag_Click ()
' Verify data input and proceed with changes...
Sub cmdShowCells_Click ()
‘ Send command to display point sources
‘ ddestr = "[LY_DRAWUSERDEF]"

Sub cmdUpdate_Click ()
' Verify data input & write form values to database

Sub Form_Load ()
' Define variables before the form will work. Identify
‘ DDE share established by source application

Sub GetValues ()
‘ Get Layer Object Values

Sub optEnter_Click (Index, Value As Integer)
‘ Selects if source enters at top or bottom of cell

Sub SendCommand (ddecommand$)
‘ Sends Command to RAISON

Sub Text2_LinkNotify ()
‘ Notify RAISON & changes color of selected cell
' Remove cell from list/map or add cell to list/map

Sub VerifyInput ()
' Validate Input Data (Return RetryInput as Integer)

Program Name: InterAGN
Module: IMPOUND.FRM

Sub AddPntScr_Click (Value As Integer)
‘ Add point source and send command
‘ SendCommand "[EV_MPCHLL]ON"

Sub CellList_Click ()
‘ Select a cell from the listbox and make it the active one...

Sub ClearValues ()
' Clear values from data form

Sub cmdAccept_Click ()
' Verify data input & write values for selected cells

Sub cmdAdd_Click ()
' Can't fill data until "apply" or create the data points.
‘ GetValues : CellList_Click

Sub cmdClear_Click ()
‘ Clear cell list

Sub cmdDel_Click ()
‘ Delete Layer Object

Sub cmdDeleteData_Click ()
‘ Delete data in cell(s)

Sub cmdDraw_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"

Sub cmdPropag_Click ()
' Verify data input and proceed with changes...

Sub cmdShowCells_Click ()
‘ Send command to display point sources
‘ ddestr = "[LY_DRAWUSERDEF]"
‘ SendCommand ddestr

Sub cmdUpdate_Click ()
' Verify data input & write form values to database

Sub Form_Load ()
' Define variables before the form will work. Identify
‘ DDE share established by the source application

Sub GetValues ()
‘ Get Layer Object Values

Sub Text2_LinkNotify ()
‘ Notify RAISON & changes color of selected cell
' Remove cell from list/map or add cell to list/map

Sub VerifyInput ()
' Validate Input Data (Return RetryInput%)

Program Name: InterAGN
Module: ADDEROS.FRM

Sub AddPntScr_Click (Value As Integer)
‘ Add point source and send command
‘ SendCommand "[EV_MPCHLL]ON"

Sub CellList_Click ()
‘ Select cell from listbox and make it the active one

Sub ClearValues ()
' Clear values from data form

Sub cmdAccept_Click ()
' Verify data input & write values for selected cells

Sub cmdClear_Click ()
‘ Clear cell list

Sub cmdDel_Click ()
‘ Delete Layer Object

Sub cmdDeleteData_Click ()
‘ Delete data in cell(s)

Sub cmdDraw_Click ()
‘ Draw selected Grid
‘ SendCommand "[PR_REDRAW]"
Sub cmdPropag_Click ()
' Verify data input and proceed with changes...

Sub cmdShowCells_Click ()
‘ Send command to display point sources
‘ ddestr = "[LY_DRAWUSERDEF]"
‘ SendCommand ddestr

Sub cmdUpdate_Click ()
' Verify data input and write form values to database

Sub Form_Load ()
' Define variables before the form will work. Identify
‘ DDE share established by the source application

Sub GetValues ()
‘ Get Layer Object Values

Sub optErosType_Click (Index%, Value%)
‘ Select type of additional erosion

Sub Text2_LinkNotify ()
‘ Notify RAISON & changes color of selected cell
' Remove cell from list/map or add cell to list/map

Sub VerifyInput ()
' Validation of Input Data (Return RetryInput%)

Program Name: InterAGN
Module: CHANNEL.FRM

Sub cmdApply_Click ()
' Verify data input
' Write values on database for selected or all cells

Sub cmdCancel_Click ()
‘ Closes current form : Unload Me

Sub cmdDeflt_Click ()
‘ Reset to default values

Sub Command3D1_Click ()
‘ Mark for all particles sizes

Sub Command3D2_Click ()
‘ Unmark for all particles sizes

Sub Form_Load ()
' Skip if input data is Null or Reads from Initial Watershed Data
' Assigning Labels and Enable/Disable form fields
' AGNPS independent of geomorphic or not, or
' Geomorphic or not independentl of peak method, or
' SCS-TR55 and Geomorphic, and
' SCS-TR55 and Non-Geomorphic
‘ Finally : ReadCellData
Sub optChanType_Click (Index%, Value%)
‘ Select channel type

Sub optUseDecay_Click (Index%, Value%)
‘ Enable options if default decay is selected

Sub ReadCellData ()
‘ Read channel related data...

Sub VerifyInput ()
' Validation of Input Data (Return RetryInput%)
' General Channel and Use Decay or not...or
' Geomorphic or not independently of peak method...or
' SCS-TR55 and Geomorphic....or
' SCS-TR55 and Non-Geomorphic.
' Check by groups depending on selected options

Program Name: InterAGN
Module: AGNPSLIB.BAS

Function AddPointSource (dbname$, layerName$, gridName$, centre%, GridNumber&, X#, y#) As Long
' Passes back the objectid of the pointsource
' x is lon, y is lat. If centre is true then the point is centred in the

Function AGNPS_CellCnt% (db As Database, gridName$, total&, level1count&)
‘ Counts the number of cells in a grid

Function CreateAGNPSAddEros (db, grid) As Int
‘ Creates the Additional Erosion table

Function CreateAGNPSChannel (db, gridID%, gridname) As Integer
‘ Creates the Channel Information table

Function CreateAGNPSdb (dbname$) As Integer
‘ Creates the layer database

Function CreateAGNPSFdlt (dbname$, gridName) As Integer
‘ Creates the Feedlot table

Function CreateAGNPSFert (db, gridid, gridname) As Int
‘ Creates the Fertilizer table

Function CreateAGNPSGen (db, gridid, gridname) As Int
‘ Creates the General Cell Data table

Function CreateAGNPSHydro (db, gridid, gridname) As Int
‘ Creates the Hydrology table
Function CreateAGNPSImpound (db, gridName)
‘ Creates the Impoundment table

Function CreateAGNPSinit (db, gridName)
' Creates the Intial Watershed table

Function CreateAGNPSNonFdlt (db, gridName)
‘ Creates the Non-feedlot table

Function CreateAGNPSNutri (db, grdid, gname)
‘ Creates the Nutrients table

Function CreateAGNPSPest (db, grdid, gname)
‘ Creates the Pesticide table

Function CreateAGNPSPestic (db, grdid, gname)
‘ Creates the Pesticide Results table

Function CreateAGNPSSedim (db, gid, gname)
‘ Creates the Sediment Results table

Function CreateAGNPSSedimAnal (db gName)
' Creates sediment analysis table (model results)

Function CreateAGNPSSens (db, gridName)
' Creates the sensitivity table (% of variation)

Function CreateAGNPSSoil (db, gid, gname)
‘ Creates the Soil table

Function CreateAGNPSSrcDep (db, gid, gname)
‘ Creates the Source_Deposition table

Function CreateAGNPSTables (Form, db, gridid, gridName) as Integer
‘ Controls the creation of AGNPS Tables:
‘ CreateAGNPSinit(dbname, gridName)
‘ CreateAGNPSGen(dbname, gridid, gridName)
‘ CreateAGNPSSoil(dbname, gridid, gridName)
‘ CreateAGNPSFert(dbname, gridid, gridName)
‘ CreateAGNPSPest(dbname, gridid, gridName)
‘ CreateAGNPSNonFdlt(dbname, gridName)
‘ CreateAGNPSFdlt(dbname, gridName)
‘ CreateAGNPSAddEros(dbname, gridName)
‘ CreateAGNPSImpound(dbname, gridName)
‘ CreateLanduseSum(dbname, gridid, gridName)
‘ CreateAGNPSChannel(dbnm, gridid, gridName)
‘ CreateAGNPSWatSum(dbname, gridName)
‘ CreateAGNPSSedimAnal(dbname, gridName)
‘ CreateAGNPSHydro(dbname, gridid, gridName)
‘ CreateAGNPSSedim(dbname, gridid, gridName)
‘ CreateAGNPSNutri(dbname, gridid, gridName)
‘ CreateAGNPSPestic(dbname, gridid, gridName)
‘ CreateAGNPSSrcDep(dbnm, gridid, gridName)

Function CreateAGNPSWatSum (db, gridName)
' Creates the watershed summary table

Function CreateLanduseSum% (Form, db, gridid, gridName)
‘ Creates the Landuse summary table

Function ExtractNextParameter (CmdString As String, Value As Variant, param As String)
' Extracts next parameter up to separator char '|' a '\|' is interperated
‘ as a double | in the parameter a '[' signals the end of the command
‘ and will not move beyond that point.
‘ use '\[' to pass a '[' as a parameter

Function FindGridPointData% (db As Database, gridLayerID%, gridObjectID&, tableID%, objectIDArray&())
' Get the grid object and put it into a VB array
' Open up the type table and scan for points

Function GetCentrePnt (db, gridid%, GridNumber&, X#, y#, utmflag%, grzone&, grdatum&) As Integer
‘ Finds centre points of grid element

Sub ReadINIFile ()
‘ Read file for name of executable and database

Sub WriteINIFile ()
‘ Write file for name of executable and database

Program Name: InterAGN
Module: AGNPSGR.BAS

Function AGNPS_AddGridRectCell (db As Database, gridID As Integer, lat As Double, lon As Double, NewGridNumber As Long, isutm%)
' Adds a rectangular grid cell to the existing grid. Assumes that the
‘ grid number starts at 1 and every grid after is incremented by 1.
' Returns: the new grid cell number added or LAYER_ERROR
' Grid cells can not be negative numbers

Function AGNPS_CheckValidGridCellNumber (db As Database, gridID As Integer, ByVal gridNumber As Long)
' Check whether or not the specified cell exists.
' Returns SUCCESS if it exists.

Function AGNPS_DeleteGridCell (db As Database, gridID As Integer, gridNumber As Long) As Integer
' Delete grid cell from all related grid tables
' Returns SUCCESS or LAYER_ERROR.

Function AGNPS_FitGridCellNumber (db As Database, gridID, NewGridNumber As Long)
' Adds one to all grid cell numbers greater than or
' equal the newGridNumber. In this way it keeps
' all the grid cell numbers unique.

Function AGNPS_NextGridNumber (gridID As Long, cellNumber As Long) As Long
' Returns: the next unique grid number.

Function AGNPS_RemoveGridCellNumber (db As Database, gridID, gridNumber As Long)
' Removes grid cell number from the grid table &
' renumbers the grid cells greater than it.

Function AGNPS_SubdivideCell (db As Database, gridID, gridNumber As Long)
' Subdivides a grid cell into four separate cells.

Program Name: InterAGN Module: DRAWFLOW.BAS

Sub Intrsct2Lines (m1#, b1#, m2#, b2#, x#, y#)
‘ Intersect two lines

Sub LineFromPoints (x1#, y1#, x2#, y2#, slope#, yIntercept#)
‘ Calculate line when points are known

Sub midPointOf2Points (strtX, strtY, endX, endY, midX, midY)
‘ Calculate mid point in line given 2 points

Sub MidPointOfRect (plyg() As Layer, x#, y#)
‘ Calculate mid point of a rectanble polygon by:
‘ Intesrsec2Lines
‘ LineFromPoints
‘ midPointOf2Points

Program Name: InterAGN Module: LAYERDB.BAS

Function AddCharacteristic (db, layid, varFld$, multindex$, FrmValIndex$, ToValIndex$)r
' Adds Characteristics to the characteristic table specified by layerID.

Function AddCollectionObject (db As Database, collectionID As Integer, layerid As Integer, objectID As Long, drawingorder As Long)
' Adds a collection object to the database.
' A collection is layerID, objectID & drawingOrder.
' The drawingOrder is a priority, so one can set a preference to the order of drawing.
' Returns: SUCCESS if the object was added
Function AddDependencyObject (db, lyrid, obID, DepLyrID%, DependencyObjectID&)
' Adds a dependency object: The layerID and objectID act as a parent to the DependencyLayerID and the DependencyObjectID which acts like a child. Returns: SUCCESS if the object was added, failure if it already exists.

Function AddGridObject (db, gid, gridNumber, numvert%, LAYERPOINT As Integer)
' Adds a grid object to the database. A grid object for example is a grid cell. Object can be a rectangular or polygonal object. Returns: ObjectId

Function AddLayertoLayerSummary (db, tName, gid, lyrtype, application$, appinter As Integer, Caption As String, boolTable As Integer)
' Adds the layer information of a layer to the LayerSummary table. Returns: layer id of new layer

Function AddLL_GridRectCell (db As Database, gridid As Integer, Lat As Double, lon As Double, newGridNumber As Long)
' Adds a rectangular grid cell to the existing grid. Assumes that the Grid is numbered starting at 1
' and every grid there after is incremented by 1.
' Grid cells can not be negative numbers Returns: the new grid cell number or LAYER_ERROR

Function AddLL_GridRectToLayerSummary (db, tabName$, application$, appinter, Caption$, gridLat, gridLon, rows, cols, angle, GridWidth)
' Adds rectangular grid information to the layer summary table. Also updates the grid rect table.
' Returns: layer id of newly added grid layer.

Function AddObject (db, layerid, varFld$, varvalue As Variant, objectType%, numvert&, LAYERPOINT As Integer, Caption As String)

' Adds an object to a given layer. This can be point, line or polygon. Will update only one value field of Layer table. Returns: object id of new object.

Function AddObjectMultData (db As Database, layerid%, varFld() As String, varvalue() As Variant, objectType%, numvert&, LYRPOINT As Integer, Caption As String) As Long
' Adds an object that has multiple values associated with it to a given layer. This object can be a point, line or polygon. Returns: objectId of new object.

Function AddObjectNoData (db As Database, layerid%, objectType%, numvert&, LAYERPOINT As Integer, Caption As String)
' Adds an object to a given layer. This can be point, line or polygon. No value fields of the Layer table will be updated. Returns: object id

Function AddPolyHoleObjectNoData (db, lyrid, numvert&, LAYERPOINT, Caption, LevelArray() As Long, VertArray() As Long, numpolys#)
' Adds an object to layer. This can only be a polygon with holes. No value fields of the Layer table will be updated. Returns: object id

Function AddUTM_GridRectCell (db As Database, gridid%, gridinX#, gridinY As Double, newGridNumber As Long) As Long
' Adds a rectangular grid cell to the existing grid. Assumes that the Grid is numbered starting at 1
' and every grid there after is incremented by 1.
' Returns: the new grid cell number added.

Function AddUTM_GridRecttoLayerSummary (db, tableName$, application$, appinter%, GridWidth#, layertype As Integer)
' Adds rectangular grid information to the layer summary table. Also updates the grid rect table.
' Returns: layer id of newly added grid layer.

Function ChangeCharacteristic (db, layerid%, varFld$, multindex$, FromValueIndex$, fillpattern%, pointsz%)
' Change characteristics in characteristics table.
' Returns: SUCCESS if SUCCESS.

Function ChangeCharacteristicRngs (db As Database, layerid%, varFld$, multindex$, FromValueIndex$, ToValueIndex$) As Integer
' Change the characteristics ranges in the characteristics table.
' Returns: SUCCESS if SUCCESS.

Function ChangeCollectionDrawingOrder (db As Database, collectionID As Integer, layerid As Integer, objectID As Long, draworder#)
' Allows the user to change/add the drawingOrder of a given collection object.
' Returns: SUCCESS if the object existed

Function ChangeGridMultValue (db, layerid As Integer, gridNumber As Long, varFld() As String, VarData() As Variant) As Integer
' Searches the specified grid layer and changes multiple values. If the function finds the object it changes it Returns: SUCCESS if value existed

Function ChangeGridValue (db As Database, layerid As Integer, gridNumber As Long, varFld As String, VarData As Variant)
' Searches the specified grid layer and changes value to given value. If the function finds a value it changes it, otherwise it adds a new record to the database. Field name defaults to "Value" if none is given. Returns: SUCCESS if the value existed in the database.

Function ChangeGridValueGivenGridObjectID (db As Database, layerid As Integer, gridObjID As Long, varFld As String, VarData As Variant)
' Searches the specified grid layer and changes value to given value. If the function finds a value it changes it, otherwise it adds a new record to the database. Field name defaults to "Value" if none is given. Returns: SUCCESS if the value existed in the database.

Function ChangeLayerGridbyName (db As Database, layer$, gridid%) As Integer
' Changes layer that a grid is associated with.

Function ChangeMultValue (db As Database, layerid As Integer, objectID As Long, varFld() As String, VarData() As Variant) As Integer
' Changes multiple values for a given object.
' Returns: SUCCESS if the values were changed, failure if there was no match.

Function ChangeValue (db As Database, layerid As Integer, objectID As Long, varFld As String, VarData As Variant) As Integer
' Changes the value of one field of an object.
' Returns: SUCCESS if the value was changed and failure if there was no match.

Function CheckValidGridCellNumber (db As Database, gridid As Integer, ByVal gridNumber As Long) As Integer
'Check whether or not the specified grid cell exists.Returns SUCCESS if it exists.

Function CreateCollectionTable (dbname$, collection As String, collcomment As String)
' This procedure creates a collection table given the open database and the collection name.
' The database is passed to this procedure closed.
Rturns: layerID of the newly added layer.

Function CreateLayerDatabase (dbname As String) As Integer
' Creates a basic layer database.

Function CreateLayerTable (db, newVarTab$, isgridtable As Integer, isUTM As Integer, zone appinter%, Caption$, boolTable%)
' Adds a new layer table.

Function CreateLL_GridRectLayer (dbname As String, gridName$, application$, appinter%, gridCols%, GridWidth#, GridAngle#) As Integer
' Creates a rectangular grid layer and and puts it in the database. Numbers grids start at 1 and increments each grid by 1 there after. This is only used to create Lat Lon layer tables.
' Returns: layer id of newly added physical grid.

Function CreateLL_GridRectTable (dbname As String, newgridtable As String, app$, appinter%, angle#, GridWidth#) As Integer
' Adds a new rectangular grid table to database.

Function CreateLL_LayerTable (db As String, newVarTable As String, isgridtable As Integer)
' Adds a new layer table.
' This is only used to create Lat Lon layer tables.

Function CreateNewFieldVariable (dbname As String, tbl As String, fld() As Field) As Integer
' Adds a field(s) to a layer table. This can be any layer table; grid or not grid.

Function CreateNewIndx (dbname$, tblname$, indx() As Index) As Integer
'This is a generic function tocreate new indexes

Function CreateParentTable% (db, layID)
' This creates a table to keep track of the parents of objects. This is only used when the created table is the results of polygonal boolean operations.

Function CreateTheDatabase (dbname$)
' Simply creates a new empty database.

Function CreateTheTable (dbname As String, tblname, fld() As Field, indx() As Index)
' Adds a table to the database given the table, fields and indexes. Assumes table does not exist.

Function CreateUTM_GridRectLayer (dbname As String, gridName$, application$, appinter, Integer, datum As Integer)
' Creates a rectangular grid layer and and puts it in the database. Number grids start at 1 and increments each grid by 1 there after.
' Returns: layer id of newly added physical grid.

Function CreateUTM_LayerTable (db As String, newVarTable As String, isgridtable As Integer, Caption$, boolTable%) As Integer
' Adds a new layer table.
' This is only used to create UTM layer tables.

Function DeleteCharacteristic (db As Database, layerid%, varFld$, multindex$, FromValueIndex$, ToValueIndex$)
' Delete the characteristics from the characteristic table. Returns: SUCCESS if SUCCESS.

Function DeleteCollectionObject (db As Database, collectionID As Integer, layerid As Integer, objectID As Long) As Integer
' Deletes a collection object from the collection table. Returns: SUCCESS if the object was deleted, failure if did not exist.

Function DeleteCollectionTable% (db As Database, collection$)
'This function deletes a collection table.

Function DeleteDependencyObject (db As Database, layerid%, objectID&, DependLayerID%, DependObjectID&)
' Deletes a dependency object from the dependencies table. Returns: SUCCESS if the object was deleted, failure if did not exist.

Function DeleteGridCell (db As Database, gridid As Integer, gridNumber As Long)
' Deletes a Grid Cell from the database.
' The following tables are affected : Position, Type, GridSummary, associated Grid Table and all Layer Tables associated with this grid.
' Returns: SUCCESS if SUCCESS.

Function DeleteGridLayerObject (db , layerid As Integer, gridNumber As Long)
' Deletes a Grid layer from the database.
' layerid is id of layer that holds data not the gridid

Function DeleteLayerbyName (db, layer$)
'Deletes an entire layer from a layerdatabase
'This operation can not be undone.

Function DeleteLayerObject (db As Database, layerid As Integer, objectID As Long)
' Deletes an Object from a database along with all the existing data associated with that object.
' Given - the open database, layerid.
' Returns: SUCCESS if SUCCESS.

Function DoesFieldExist (tbl As Table, fld As String) As Integer
'This function checks to see if a field exists in a given table.

Function DoesParentExist (db As Database, layerid As Integer) As Integer
'This tests if a parents table exists for a given layer.

Function DoesTableExist (db As Database, tblname As String) As Integer
' This tests if a given table exists in the database.

Function FindCollections (db As Database, colltbl() As String) As Integer
' Fills array with names of all the collection tables.
' Returns: number of collection tables in the DB

Function FindGridIDGivenLayerID (db As Database, layerid As Integer) As Integer
' Given the layerid
' Returns: the gridId as an integer, if not found returns FAILURE
Function FindGridNumberGivenGridObjectID (db As Database, gridid As Integer, gridObID)
' Returns: the grid number given the grid objectID.

Function FindGridNumberGivenObjectID (db As Database, layerid As Integer, objectID As Long)
'Given the layerid, objectid.
'Returns : the gridnumber as an integer.

Function FindGridObjectIDGivenGridNumber (db As Database, gridid As Integer, gridNum)
' Given the gridID and the grid number.
' Returns: the gridObjectID of the object.

Function FindGridObjectIDGivenObjectID (db As Database, layerid As Integer, objectID As Long)
'Given the open databasae, layerid and ObjectID.
'Returns the GridObjecID.

Function FindGrids (db, gridTbl() As String)
'Given the database. Returns : all the Grids that exist in the file

Function FindGridTblLayers (db As Database, ByVal gridid As Long, GridLayerTbls)
' Fills the string array GridLayerTbls with the names of all the grid layers associated with a particuluar grid. Returns: the number of layers in the array.

Function FindGridTblTypeLayers (db As Database, gridid As Integer, GridLayerTbls() As String, GridLayerType() As Integer) As Integer
' Given the gridID, fills in the names of the tables associated with it, and the array of the type of tables they are. A specialized version of GridTblLayers. Returns: the number of layers.

Function FindLayersAndTypesNonGrid (db As Database, layers() As String, layertypes)
' Given the database, the layers() array is filled with the names of the non-grid layer (layers not associated with a grid) and also the layerstype() array is filled with the tables type.
' Note: doesn't return collection tables!
' Returns: the number of layers.

Function FindObjectIDGivenGridObjectID (db As Database, layerid As Integer, gridObjectID)
'Given the open databasae, layerid and ObjectID.
'Returns the ObjectID.

Function FitGridCellNumber (db As Database, gridid As Integer, newGridNumber As Long)
' Adds one to all grid cell numbers greater than or ' equal to the newGridNumber. In this way it keeps all the grid cell numbers unique.

Function GenerateRangeList (db As Database, tblname As String, varname$, multname$, RngFrom() As String, RngTo() As String)
'This function creates a list of ranges for a given thematic map. The tblname is the name of the layer, varname is the name of the data field and multname is the thematic map or characteristic. Two arrays RngFrom and Rngto are returned.

Function GenerateTableList (db As Database, tblname() As String) As Integer
'This function created a list of tables given a DB. An array of tablename, tblname(), is returned.

Function GenerateVarList (db As Database, tblname As String, vars() As String) As Integer
'This function creates a list of data fields for a given layer anem. Tblname is the name of the layer, nd vars() is the array of field names.

Function GenerateVarMultList (db As Database, tblname As String, varname$, mults() As String)
'This function creates a list of thematic maps or characteristics for a given layer (tblname) and data field (varname). An array called mults() is returned.

Function GetApplication (db As Database, LayerName As String, appinter As Integer)
' Given the layer name, gives the application and the appInter. Returns: the application name.

Function GetGridLayerObject (db, layerid As Integer, objectID As Long, gridid As Integer, gridObjectID As Long, varFld As String, varvalue As Variant, valflag As Integer)
' Given the objectid and the layerid. Default value field is "Value". Returns : gridobjid, gridid & value

Fn GetGridMultValueByLayerIDAndGridN (db, layerid As Integer, gridNumber As Long, varFld() As String, varvalue() As Variant)
' Given the gridnumber and the layerid and array of Value fields and an array of values are returned as a variants. Returns : FAILURE if fails

Fn GetGridMultValueByLayerNameAndGridN (db, LayerN$, gridObjectID As Long, varFld() As String, varvalue() As Variant)
' Given the gridobjectid and the layerid and array of Value fields and an array of values are eturned as a variants. This function is faster than GetGridMultValueByLayerIdAndGridObjID.

Function GetGridObjectLabel (db As Database, gridid As Integer, gridObjectID As Long, gridvalue As Variant) As Integer
' Retrieves the grid object label (i.e. Grid number)
' Returns: SUCCESS if the function was success
Fn GetGridValueByLayerIDAndGridNumber (db, layerid, gridNumber, varFld, varvalue)
' Given the gridnumber, the layerid and the name of the value field the value is returned as a variant. The default is "Value". Returns : the value of the grid cell from the correct database, as a variant

Fn GetGridValueByLayerNameAndGridObjID (db As Database, LayerName As String, gridObjID As Long, varFld As String, varval)
' Given the gridobjectid, the layername and the name of the value field the value is returned as a variant. The default "Value". This function is faster than GetGridValueByLayerIdAndGridObjID.
' Returns : the value of the grid cell from the DB

Function GetLayerObjectFirstPnt (db, layerid As Integer, objectID As Long, pnt)
'This function returns first point of a given object.

Function GetLayerObject (db, lyrid, objID, PnAr, NumVer, objType, LvlArray, VerArr(), numpol)
' Given the open layer database, the layerid and the object this function will return the points, object type of the given object. Returns: If the object does not exists a value of -1 is returned.

Function GetLayerObjectChar (db, layerid, vrfld, multindex, varvalue, fillpattern, pointsz%)
' Retrieves the layer object characteristics based on Value field selected and a value The condition is greater than or equal to the value passed in. This function will work for both polygonal layers and grid layers since the structure of the Characteristic table is the same. Returns: SUCCESS

Function GetLayerObjectMultValue (db, layerid As Integer, objectID As Long, varFld(), varval)
' Retrieves the layer object values. Returns as many of the object's values as requested. Returns: SUCCESS if the function was success

Function GetLayerObjectValue (db, layerid As Integer, objID, varFld As String, varval)
' Retrieves the layer object value. Only one value is returned, if the field is not specified then then default is 'Value'. Returns: SUCCESS

Function GetObjectType (db, layerid, objectID&, objectType, numvert, Caption, numpolys)
' Returns the all data in the type table associated with an object. Return the objecttype or ERROR

Function GetParents (db As Database, layerid%, objectID&, PLayerID() As Integer, PObjID)
'This function returns a given objects parents. Objects can at most have two parents.
'PLayerID and PObjID are the arrays returned.
Function GetUTMData (db As Database, layerid As Integer, zone As Long, datum As Long)
' Retrieves the layer's or grid's UTM data
' Returns: SUCCESS if the function was success

Function GridCell (db As Database, gridTbl As String, X As Double, y As Double) As Long
' Given the (x,y) point - Returns: the grid cell number that the point is inside.

Function gridObjectID (db As Database, gridTbl As String, X As Double, y As Double)
' Given the (x,y) point - Returns: the grid object ID that the point is inside.

Function GridStatus (db As Database, layerid%)
'This returns the grid status of a given layer:
' GRIDTABLE - a grid
' GRIDLAYER - a layer associated with a grid
' LAYERTABLE - a layer is not associated with grid

Function HALL2VBDblArrS (pntArr, nvert, sArr)
' Converts huge array to visual basic double array

Function LayerName (db, ByVal layerid)
' Given the layerId - Returns name of the layer.

Function LayerTableID (db, tbl As String)
' Given the layer name - Returns: the layer id

Function NextGridNumber (db, gridid)
' Returns: the next unique grid number.

Function NextUniqueID (db, layerid As Integer)
' Given the layer - Returns: the next available id.

Function RemoveGridCellNumber (db As Database, gridid As Integer, gridNumber)
' Removes grid cell number from the grid table, and renumbers the grid cells greater than it.

Function RotatePoint (oX As Double, oY As Double, X As Double, y As Double, angle)
' Given the origin point (oX, oY), and the point to be rotated (x, y), this rotates the point an angle about the origin. The result is back into the point (x, y).

Function StringToVar (sampleVar, convString$)
' Ttake variant as string and convert to data type.

Function SubdivideCell (db, gridid, gridNumber)
' Subdivides a grid cell into four separate cells.

Function VB2CFileNameString$ (fileName$)
‘ Convert VB to C file name string for external use

Function VBArrToHugeArr (X() As lyrPntType, y)
‘ Convert VB to huge array to pass between dlls

Program Name: InterAGN
Module: MAIN.BAS

Function ConvertGeotoUTM (originLat#, originLon#, datum&, zone&, Gn#, Ge#)
‘ This converts the geographic lat/lon to UTM

Function ConvertUTMtoGeo (Gn#, Ge#, datum&, zone&, originLat#, originLon#)
‘ This converts the geographic UTM to lat/lon

Function CopyLayersChar (db As Database, thelayername$, newdb, newlayername$)
‘ This copies the characteristics table from one layer

Function CreateCopyGrid (db As Database, layername$, newlayername$) As Integer
‘ Create structure and copy old to new grid

Function CreateCopyLayer (db, layername$, newdb, newdbname$, newlayername$, flds)
‘ Creates a copy of a layer into a new one.

Sub FillPolyData (dbGridName$, gridName$, dbTableName$, tableName$, Forma$)
‘ Main procedure for extracting layer values
' Resets percent counter...

Function NarrowRegionLayer (dbname$, newdatabase%, minx#, miny#, maxx#, maxy#)
‘ Narrows the region of interest to a bounded box

Program Name: InterAGN
Module: DEMLIB-A.BAS

Sub DoAGNPSFillData (GridDb, GridlayerID%, DEMdb As Database, DEMLayerID%)
'Get records for percent counting...
‘ Call DoAGNPSFillFlowDirection()
‘ Call DoAGNPSFillSlope()
‘ Call DoAGNPSReceivingCell

Sub DoAGNPSFillFlowDirection (GridDb, GrdlyrID, DEMdb, DEMLayerID%, gAng#)
' Find delta from DEM file:
‘ Call FindDeltaFromDem()
‘ FlowDir = FindGridFlowDirection()

Sub DoAGNPSFillSlope (GridDb, GridlayerID, DEMdb, DEMLayerID%, zone&, datum&)
' For river elevation update...Verifies if field exists
' Find the length of the grid
' Find our bounding rectangle
' Create snapshot of elements in bound rectangle
‘ Slope = FindGridSlope()
' Write River Elevation using variable: elevation

Sub DoAGNPSReceivingCell (db As Database, GridlayerID%)
‘ Calculate receiving cell using flow direction value
‘ Call GetFlowArrow()
‘ Call PointForRecievingCell(x1, y1, x2, y2, tx, ty)

Function FindAngle# (x1#, y1#, x2#, y2#)
' This finds an angle where the +x axis is zero
' and the angle rotates counter-clockwise

Function FindBoundingRectangle# (vbArr#(), nVert&, minx#, miny#, maxx#, maxy#)
‘ Finds bounding rectangle por a given polygon

Sub FindDeltaFromDem (DEMdb, DeltaValue#)
‘ Finds the delta value from the DEM file

Function FindDEMsInGrid (Db, StrlayerID, StrobjectID, DEMdb, DEMLayerID, DEMs() As DEMType As Snapshot, vbSObjArr(), snVert)
‘ Finds DEM elements inside a grid cell

Function FindGridFlowDirection (db, Gridid, gridObjectID&, DEMdb, DEMID, gAng, delta)
' Delta is the degree distance that the point must be within the line. Finds flow direction by assigning dominant direction of DEM

Function FindGridSlope# (db, GridlayerID, gridObjectID, DEMdb, DEMID, length, elevat#, demTbl As Snapshot, vbObjArr(), nVert)
' Length is the length of the grid cell
' Find Highest DEM elevation
' Find Lowest DEM elevation
‘ Calculates and returns slope value

Sub GetDEMValues (db As Database, DEMs)
‘ Get Elevation value from DEM layer

Sub GetFlowArrow (db As Database, pt As Table, layerID As Integer, objectID As Long, FlowDir As Integer, x1#, y1#, x2#, y2#, plyg() As layerPointType)
‘ Get flow direction according to DEM direction

Sub PointForRecievingCell (x1#, y1#, x2#, y2#, X#, y#) As Integer
‘ Calculate point for receiving cell.

FORTRAN Code for the Water Quality Component

READWQD SUBROUTINE

```
c*************************************************************************
      SUBROUTINE readwqd
C*************************************************************************
C
C THIS SUBROUTINE READS IN THE WATER QUALITY DATA FILE
C LFLV - Oct/98

      include 'area1.for'
      include 'area2.for'
      include 'area3.for'
      include 'area111.for'

C Open and read *.wqd file (*=file name, .wqd=water quality data)
      open(unit=70 ,file='basin/duff.wqd' ,status='old',err=99000)

C SEDIMENT DATA...
      read(70,*)
      read(70,*)gamma,ro,viskin,grav,A,B
      read(70,*)(GC(i),i=1,ntype+1)
      read(70,*)(CF(i),i=1,ntype+1)

C     Particle size - d50
      read(70,*)
      do 10 i=imax,1,-1
        read(70,*)(diam(i,j),j=1,jmax)
   10 continue

C     Specific weight - spg
      read(70,*)
      do 20 i=imax,1,-1
        read(70,*)(spew(i,j),j=1,jmax)
   20 continue

C     Erodibility - Erod
      read(70,*)
      do 30 i=imax,1,-1
        read(70,*)(erodi(i,j),j=1,jmax)
   30 continue

C     Put into vector format
      do 50 n=1,naa
        i=yy(n)
        j=xx(n)
        D50(n)=diam(i,j)
        spg(n)=spew(i,j)
        Erod(n)=erodi(i,j)
   50 continue

C NUTRIENT DATA...
      read(70,*)
      read(70,*)Ncrn,Ndec,Pdec
      read(70,*)Nscn,Ncpw,Nrec,Nlec
      read(70,*)Pscn,Pcpw,Prec,Plec

C     Fertilizer Application
      read(70,*)
```

```
      read(70,*)NoFer
      do 60 l=1,NoFer
        read(70,*)i,j,mNfer(i,j),mPfer(i,j),mNfa(i,j),mPfa(i,j)
   60 continue

C     Put into vector format
      do 70 n=1,naa
        i=yy(n)
        j=xx(n)
        Nfer(n)=mNfer(i,j)
        Pfer(n)=mPfer(i,j)
        Nfa(n)=mNfa(i,j)
        Pfa(n)=mPfa(i,j)
   70 continue

      return

C     Water Quality Data file not found:
99000 write(*,*)' Error opening or reading WQD file - File not found'
      pause 'Abnormal ending in SPL #1 - hit any key to continue'
      stop

      END
```

FORTRAN Code for the Water Quality Component (Cont.)

SEDIMENT SUBROUTINE

```
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE SED(JAN,AL)
C **************************************************************************
C Modified: May/97 - Luis Leon
C          THIS SUBROUTINE CALCULATES THE SEDIMENT CONCENTRATION
C          FOR EACH ELEMENT OF THE WATERSHED,Y(N,II)
      include 'AREA1.FOR'
      include 'AREA3.FOR'
      include 'AREA4.FOR'
      include 'AREA6.FOR'
      include 'AREAsed.FOR'
      common/area111/gamma,ro,viskin,grav,A,B,D50(99),spg(99)
     *Erod(99),Erf(99),Y(99,5),Kf(5),GC(5),CF(5)
C ****** EQUATIONS REFERENCES AND UNITS FROM:
C       HARTLEY, D.M.  1987.  "SIMPLIFIED PROCESS MODEL FOR WATER
C          SEDIMENT YIELD FROM SINGLE STORMS.  PART I - MODEL
C          FORMULATION"  IN TRANSACTIONS OF THE ASAE 30(3):710-717.
C      NAA = NUMBER OF "N" ELEMENTS
C      NTYPE = TOTAL NUMBER OF "II" LAND CLASSES
C      gamma = SPECIFIC WEIGHT OF WATER [kg/m2/s2]
C      ro = DENSITY OF THE FLUID [kg/m3]
C      viskin = KINEMATIVE VISCOSITY [m2/s]
C      grav = STANDARD ACCELERATION [m/s2]
C      A,B = EMPIRICAL CONSTANTS USED IN TRANSPORT CAPACITY FORMULA
C      Kf(II) = OVERLAND FLOW RESISTANCE [-]
C      GC(II) = GROUND COVER IN PERCENT
C      CF(II) = COVER FACTOR IN PERCENT - DETERMINED BY FIG. 5.(HARTLEY)
C      Erod(N) = SOIL ERODIBILITY D [g/J]
C      D50(N) = AVERAGE D50 OF THE DETACHED SEDIMENT [mm] (LOOKUP TABLE)
C      spg(N) = SPECIFIC GRAVITY OF DETACHED SEDIMENT [-] (LOOKUP TABLE)
C      REY = SHIELD'S CRITERION REYNOLD'S NUMBER [-]
C      phi = SHIELD'S PARMETER [-]
C      slope(N) = SQRT OF CHANNEL SLOPE
C      sl1(N) = SQRT OF OVERLAND SLOPE
C      al = LENGTH OF ELEMENT [m]
C      frac(N) = FRACTION OF ELEMENT WITHIN THE BASIN
C      aclass(N,II) = FRACTION OF CLASS II IN N
C      D1 = DEPTH OF WATER [mm]
C      DS = depression storage [mm]
C      HSED(N,II) = d1-ds, RUNOFF DEPTH [mm]
C      QL(N,II) = OVERLAND FLOW [m2/hr]
C      FLOW_SHEAR = FLOW SHEAR STRESS [Units of gamma * mm]
C      CRIT_SHEAR = CRITICAL SHEAR [Units of gamma * mm]
C      C_val = VOLUMETRIC SEDIMENT CONCENTRATION [-]
C      Y_crit = SEDIMENT TRANSPORT CAPACITY [kg/m2]
C      Coef_Cover = COEFFICIENT GROUP FOR EQ.44
C      Erf = RATE OF RAINFALL ENERGY [J/m2/hr]
C      Ero = RATE OF RUNOFF ENERGY [J/m2/hr]
C      Grf = RAINFALL SOIL DETACHMENT [Kg/m2/hr]
C      Gro = RUNOFF SOIL DETACHMENT [Kg/m2/hr]
C      Y_pot = POTENTIAL SEDIMENT SUPPLY/YIELD [kg/m2]
C      Y(N,II) = MININUM OF Y_pot AND Y_crit [kg/m2]
C      Yrot(N) = TOTAL YIELD FROM ALL CLASSES FOR ROUTING [Kg/m3]

C  ****** Initialize Values  ********************
      if(jan.eq.1)then
```

```
      close (unit=41)
        open(unit=41,file='basin\sed.par',status='unknown',err=9998)
        read(41,4100,err=9999)(GC(i),i=1,NTYPE)
        read(41,4100,err=9999)(CF(i),i=1,NTYPE)
        read(41,4100,err=9999)gamma,ro,viskin,grav,A,B
C *Note: Temporal - Assume constants for now and change to (N) later
        read(41,4100,err=9999)d50t,spgt,erodt
        close(unit=41)
        DO 40 II = 1,NTYPE
C                 Equation 16 (for each landclass)
           Kf(II) = 60.0 + 3140.*GC(II)**1.65
 40     CONTINUE
      endif
C ****** Initialize Sediment Yields *************
      DO 50 N = 1,NAA
C *Note: Temporal assignment of constants values - read later
        D50(N) = d50t
        spg(N) = spgt
        Erod(N) = erodt
        Yrot(N) = 0.0
        do 50 II = 1,NTYPE
           Y(N,II) =0.0
  50  CONTINUE
C ****** Calculate Rainffall Energy *************
      DO 60 N = 1,NAA
      I=YY(N)
        J=XX(N)
          IF (P(I,J).LE.0.0) THEN
            Erf(N) = 0.0
          ELSE
C           Eq43 [J/m2/hr] (for time step = 1hr, P=rain intensity
C                If not then divide P by time step)
           Erf(N)=P(I,J)*(11.9+8.7*LOG10(P(I,J)))
          ENDIF
  60  CONTINUE
C ****** FOR EACH ELEMENT ***********************
     DO 120 N = 1,NAA
       IF(slope(N).GT.0.0) THEN
         I=YY(N)
         J=XX(N)
         I3=IBN(N)
         IF(NTYPE.LE.0)THEN
C          if ntype =0 then program runs in the lumped mode
           II1=IBN(N)
           II2=IBN(N)
         ELSE
C           parameters grouped by land cover/use
           II1=1
           II2=NTYPE
         ENDIF
C ****** FOR EACH LAND CLASS TYPE ***************
      DO 110 II=1,NTYPE
C *Note: Probable change of QS [m3/s] to QL [m2/hr]
        IF ((QS(N,II).GT.0.0).and.(HSED(N,II).gt.0.0)) THEN


C * * * *  Calculate the Sediment Tranport Capacity
C            Eq27 [gamma * mm] gamma cancels in Eq21 : 5/8 is beta/beta+1
             FLOW_SHEAR = (5*60/8*Kf(II)) * HSED(N,II) * sl1(N)
C            Eq29 [-] including grav because shear has no gamma
             REY = ((FLOW_SHEAR*grav/1000)**0.5) * D50(N)/viskin
```

```
C            From equation of Shields Diagram
             phi = (0.11/REY) + 0.021*LOG10(REY)
C            Eq20 [gamma * mm] gamma cancels in Eq21
             CRIT_SHEAR = (spg(N)-1) * phi * D50(N)
C            Eq21 [-] for volumetric concentration
             C_val = A * (FLOW_SHEAR/CRIT_SHEAR)**B
C            Eq42 [kg/m2] to compare with potential yield
             Y_crit = 2.65 * ro * C_val * rf(N,II)/1000

C * * * *  Calculate the Sediment Supply
C            Eq44 [kg/m2/hr] divide by 1000 to convert g to kg
             Coef_Cover = (1-GC(II)) * CF(II)
             Grf = (Erf(N) * Coef_Cover * Erod(N)) / 1000
C            Eq45 [J/m2/hr] Warning! the constant 60 has units
             Ero = (60/Kf(II)) * gamma * (QL(N,II)/2) * sl1(N)
C            Eq46 [kg/m2/hr] divide by 1000 to convert g to kg
             Gro = (Ero * Erod(N)) / 1000
C            Eq47 [kg/m2] Only if time step is 1 hr
             Y_pot = (Grf + Gro)

C * * * *  Calculate the Sediment Yield
C            Compare capacity with supply and choose minimum
             Y(N,II) = MIN(Y_crit,Y_pot)
C            Class weighted and converted to [kg/m3] * (A/Qt)
             Y(N,II) = Y(N,II) * frac(N) * aclass(N,II) *
     *        (al*al/(QS(N,II)*3600))
C            Add for all classes to sediment supply per cell
             Yrot(N) = Yrot(N) + Y(N,II)
        ENDIF
110       CONTINUE
        ENDIF
120     CONTINUE

C                 * * * END OF MAIN LOOP * * *
C  Yrot(N)=TOTAL SEDIMENT INFLOW FROM ELEMENT (N) should be in [kg/m3]
C              AND IS ROUTED INTO THE DOWNSTREAM ELEMENT.

4100    format(5f10.6)
4101    format(5f12.6)
       return
9998    write(*,6227)'basin\sed.par'
6227    format(' sed: error on unit=41 fln=basin\sed.par',a30/
     *' probable cause: file not found')
       stop
9999    write(*,6229)'basin\sed.par'
6229    format(' sed: read error on unit=41 fln=basin\sed.par',a30/
     *' probable cause: format error, no data, or end of file')
       stop
      END
```

FORTRAN Code for the Water Quality Component (Cont.)

NUTRIENTS SUBROUTINE

```
C* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      SUBROUTINE NUT(JAN,AL)
C **************************************************************************
C Created Jul/98 - Luis Leon
C          THIS SUBROUTINE CALCULATES THE NUTRIENT CONCENTRATIONS
C          FOR EACH ELEMENT OF THE WATERSHED: Cron(N,II), Crop(N,II)
      include 'AREA1.FOR'
      include 'AREA3.FOR'
      include 'AREA4.FOR'
      include 'AREA6.FOR'
      include 'AREAsed.FOR'
      common/area111/Ncrn,Ndec,Pcec,Nscn,Ncpw,Nrec,Nlec,Pscn,Pcpw,Prec,Plec
      *NoFer,Nfer(99),Nfa(99),Pfer(99),Pfa(99),Cron(99,5),Crop(99,5)
C ***** EQUATIONS, REFERENCES AND UNITS FROM THE CREAMS MODEL:
C      Frere et.al. (1980) The Nutrient Submodel. In: Knisel, W.G.
C           CREAMS: A Field Scale Model for Chemicals, Runoff, and Erosion
C           from Agricultural Management Systems, USDA, Rep No. 26
C      Young e.al. (1986) Agricultural Nonpoint source Pollution Model:
C           A Watershed Analysis Tool, Model Documentation, USDA.
C [General]
C      NAA = Number of "N" elements
C      NTYPE = Total number of "II" land classes
C      Ieff = Effective Infiltration for the Storm [mm] - WatFlood Value F(N,II)??
C      Roff = Total Runoff for the Storm [mm] - WatFlood Value HSED(N,II) = (D1-DS)
C      Fpor = Porosity Factor [-]
C      Peff = Effective Precipitation [mm]
C      Por =  Soil Porosity [-]
C      spg(N) =  Soil Specific Weight [-]
C      P(I,J) = Storm Precipitation for the time step [mm] - WatFlood Value
C      ER = Nuetrient Enrichment Ratio
C      Ysed = Total Sediment Yield [kg/ha]
C      NoFer = Number of Cells with Fertilizer Application
C [Nitrogen]
C      Cron(N,II) = Soluble Nitrogen Concentration in the Runoff [kg/ha]
C      Cron_rot(N) = Nitrogen Concentration all Classes for Routing [kg/ha]
C      Navs = Available Nitrogen in the Surface [kg/ha]
C      Navr = Available Nitrogen due to Rainfall [kg/ha]
C      Ndmv = Rate for Downward Movement of Nitrogen into the Soil [1/mm]
C      Nrmv = Rate for Nitrogen Movement into the Runoff [1/mm]
C      Nrnc = Nitrogen Contribution due to Rain [kg/ha]
C      Soln = Soluble Nitrogen in the Surface CM of the Soil [kg/ha]
C      Nfer(N) = Nitrogen Fertilizer Application [kg/ha] - Input Data
C      Nfa(N)  = Fraction of Nitrogen Availability [%/100] - Input Data
C      Ncpw = Nitrogen Concentration in Pore Water [ppm] - Input Data
C      Ncrn = Nitrogen Concentration in Rainfall [ppm] - Input Data
C      Nlec = Nitrogen Leaching Extraction Coefficient - Input Data
C      Nrec = Nitrogen Runoff Extraction Coefficient - Input Data
C      Ndec = Nitrogen Decay Fraction [%]
C      Nsed = Overland Nitrogen Transported by Sediment [kg/ha]
C      Nscn = Soil Nitrogen Concentration [g N/g soil] - Input Data
C [Phosphorus]
C      Crop(N,II) = Soluble Phosphorus Concentration in the Runoff [kg/ha]
C      Crop_rot(N) = Phosphorus Concentration all Classes for Routing [kg/ha]
C      Pavs = Available Phosphorus in the Surface [kg/ha]
C      Pavr = Available Phosphorus due to Residual in Soil [kg/ha]
C      Pdmv = Rate for Downward Movement of Phosphorus into the Soil [1/mm]
```

```
C      Prmv = Rate for Phosphorus Movement into the Runoff [1/mm]
C      Solp = Soluble Phosphorus in the Surface CM of the Soil [kg/ha]
C      Pfer(N) = Phosphorus Fertilizer Application [kg/ha] - Input Data
C      Pfa(N)  = Fraction of Phosphorus Availability [%/100] - Input Data
C      Pcpw = Phosphorus Concentration in Pore Water [ppm] - Input Data
C      Plec = Phosphorus Leaching Extraction Coefficient - Input Data
C      Prec = Phosphorus Runoff Extraction Coefficient - Input Data
C      Pdec = Phosphorus Decay Fraction [%]
C      Psed = Overland Phosphorus Transported by Sediment [kg/ha]
C      Pscn = Soil Phosphorus Concentration [g P/g soil] - Input Data

C  ****** Initialize Values  **********
       DO 10 N = 1,NAA
         Cron_rot(N) = 0.0
         Crop_rot(N) = 0.0
         do 10 II = 1,NTYPE
            Cron(N,II) = 0.0
            Crop(N,II) = 0.0
   10  CONTINUE
C ****** Calculate Constants First **************
C      Calculate the Available Nitrogen due to Rainfall
       Navr = Ncrn * 0.000001
C ****** FOR EACH ELEMENT ***********************
       DO 20 N = 1,NAA
       I=YY(N)
       J=XX(N)
          IF (P(I,J).GT.0.0) THEN
C * * * *    Calculate Soil Porosity and Porosity Factor:
             Por = 1 - (spg(N)/2.65)
             Fpor = 0.00001/Por
C * * * *    Calculate Nutrients Movement Rates:
             Ndmv = Nlec / (10*Por)
             Nrmv = Nrec / (10*Por)
             Pdmv = Plec / (10*Por)
             Prmv = Prec / (10*Por)
C * * * *    Calculate Soluble Nutrients in Top cm of Soil:
             Soln = 0.10 * Ncpw * Por
             Solp = 0.10 * Pcpw * Por
C * * * *    Calculate Available Phosphorus due to Soil Residual:
             Pavr = Solp * Fpor
C * * * *    Calculate Nitrogen Contribution due to Rainfall:
             Nrnc = Ncrn * P(I,J)
C * * * *    Calculate the Effective Precipitation in the top cm:
             Peff = P(I,J) - (10 * Por)
C * * * *    Calculate the Available Nutrients in the Surface:
             Nfa = Nfa / 100
             Navs = (Soln + (Nfer(N) * Nfa(N))) * Fpor
             Pfa = Pfa / 100
             Pavs = (Solp + (Pfer(N) * Pfa(N))) * Fpor
C ****** FOR EACH LAND CLASS TYPE ***************
       DO 30 II=1,NTYPE
          IF ((HSED(N,II).gt.0.0)) THEN
C * * * *    Calculate the Nitrogen Concentration in Runoff
             NTerm1 = (Navs - Navr) / Fpor
             NTermexp1 = exp(-Ndmv * F(N,II))
             NTermexp2 = exp(-Ndmv * F(N,II) - Nrmv * HSED(N,II))
             NTerm2 = NTermexp1 - NTermexp2
             NTerm3 = Nrnc * HSED(N,II) / Peff
             Cron(N,II) = (NTerm1 * NTerm2) + NTerm3
C * * * *    Calculate the Phosphorus Concentration in Runoff
             PTerm1 = (Pavs - Pavr) / Fpor
```

```
            PTermexp1 = exp(-Pdmv * F(N,II))
            PTermexp2 = exp(-Pdmv * F(N,II) - Prmv * HSED(N,II))
            PTerm2 = PTermexp1 - PTermexp2
            PTerm3 = Pavr * Prmv * HSED(N,II) / Fpor
            Crop(N,II) = (PTerm1 * PTerm2) + PTerm3
C * * * *    Class weighted by fractions of landclass
            Cron(N,II) = Cron(N,II) * frac(N) * aclass(N,II)
            Crop(N,II) = Crop(N,II) * frac(N) * aclass(N,II)
C * * * *    Add for all classes to nutrient concentration per cell
            Cron_rot(N) = Cron_rot(N) + Cron(N,II)
            Crop_rot(N) = Crop_rot(N) + Crop(N,II)

         ENDIF
30    CONTINUE
         ENDIF
20    CONTINUE

C                * * * END OF MAIN LOOP * * *
C  Cron_rot(N) and Crop_rot(N)=TOTAL NUTRIENTS INFLOW FROM ELEMENT (N)
C             AND ARE ROUTED INTO THE DOWNSTREAM ELEMENT.

4100   format(5f10.6)
4101   format(5f12.6)
       return

9998   write(*,6227)'basin\sed.par'
6227   format(' sed: error on unit=41 fln=basin\sed.par',a30/
     *' probable cause: file not found')
       stop

9999   write(*,6229)'basin\sed.par'
6229   format(' sed: read error on unit=41 fln=basin\sed.par',a30/
     *' probable cause: format error, no data, or end of file')
       stop

       END
```