Contract No:

This document was prepared in conjunction with work accomplished under Contract No. DE-AC09-08SR22470 with the U.S. Department of Energy (DOE) Office of Environmental Management (EM).

Disclaimer:

HDTS 2017.0 Testing and Verification Document

Tad S. Whiteside

July 2017

SRNL-STI-2017-00306, Revision 0

DISCLAIMER

Printed in the United States of America
Prepared for
U.S. Department of Energy

Keywords: HDTS
Dose Tracking
Code Verification

Retention: *permanent*

HDTS 2017.0 Testing and Verification Document

Tad S. Whiteside

July 2017

Prepared for the U.S. Department of Energy under contract number DE-AC09-08SR22470.

Savannah River National Laboratory™
OPERATED BY SAVANNAH RIVER NUCLEAR SOLUTIONS

Reviews and Approvals

Authors:

T. S. Whiteside, Environmental Modeling — Date

Technical Review:

K. L. Dixon, Geosciences — Date

Approval:

T. P. Eddy, Manager
Sample Data Management — Date

D. A. Crowley, Manager
Environmental Modeling — Date

Executive Summary

This report is a continuation of the series of Hunter Dose Tracking System (HDTS) Quality Assurance documents including (Foley and Powell, 2010; Dixon, 2012). In this report we have created a suite of automated test cases and a system to analyze the results of those tests as well as documented the methodology to ensure the field system performs within specifications.

The software test cases cover all of the functions and interactions of functions that are practical to test. With the developed framework, if software defects are discovered, it will be easy to create one or more test cases to reproduce the defect and ensure that code changes correct the defect.

These tests confirm HDTS version 2017.0 performs according to its specifications and documentation and that its performance meets the needs of its users at the Savannah River Site.

Contents

List of Abbreviations

DOE Department of Energy

HDTS Hunter Dose Tracking System

QA Quality Assurance

SRNL Savannah River National Laboratory

SRNS Savannah River Nuclear Solutions

SRS Savannah River Site

1 Objective

This document describes how the Software and Hardware that comprise the Hunter Dose Tracking System (HDTS 2017.0) is tested for correctness and the results from those tests. In addition, this document describes how future tests should be performed if changes to the system are made. Following this method will ensure the HDTS performs according to its specifications and documentation and that its performance meets the needs of its users at the Savannah River Site.

The Method and Scope of Software Testing uses all of the functions and interactions of functions that are practical to test. Unfortunately, due to equipment limitations, we do not have a set of development hardware identical to the field systems. Therefore, when any changes are made to the software and prior to use in the field, the changed software should be uploaded to the field systems and tested as described in the Method and Scope of System Testing.

2 Software Description

The HDTS 2017.0 is an integrated system to measure Cs-137 in animals harvested at the Savannah River Site, calculate the activity (pCi/g) in those animals, calculate the DOE ingestion dose (mrem) from that activity (pCi/g), ensure that dose is below limits, and track the annual and lifetime dose received by the consumers of those animals (the hunters). Please see (Whiteside, 2017a,b,c) for a complete description of the software and features.

Because the HDTS is used to comply with regulatory laws, environmental permits, regulations, and/or commitments to compliance it is classified as Level C software, see (Jannik, 2013) for complete details.

3 Method and Scope of Software Testing

Software testing is the process of executing a program with the intent of finding errors (Myers et al., 2004). For the HDTS we developed a framework to test each function of the program. We created test cases for the most important functions and for those functions where it was cost-effective. Each test case had one or more results, and we developed units to check the correctness of these results. For those untested functions, having the framework available allows for future functions, cases, and units to be easily created and added to the testing system.

In the HDTS 2017.0, the results of this testing are:

Table 3.1: Results of testing

Num Functions	205
Num Functions not tested	138
Num Functions tested	67
Num Cases tested	312
Num Units tested	351
Num Units passed	351
Num Units failed	0

In Appendix A there is a complete listing of the testing code. This listing shows the test suite "driver" software and the individual test cases used to test each function. The tested functions are those that perform the calculations or manipulate the data upon receiving it from the detectors or database. The majority of the untested functions are related to the User-Interface - they either layout the User Interface or pass user input on to the actual (tested) functions that do the work of processing that input. The test suite program creates a summary document, Appendix B, showing the results of running each test and a final table showing the overall results, Table 3.1.

The test suite reads the source code of the HDTS and generates the list of functions found in that code. For each function found, the code looks for a test of that function and if found, runs the test cases associated with that function. If a test of the function is not found, the code alerts the user so they can create an appropriate function test.

In creating the test cases for each function, we attempted to design cases that tested valid input, invalid input, and input at the limits of each function.

Along with the software logic that controls the detector hardware and converts measured counts to activity and then to dose, the HDTS has a database where the results from each measurement are stored as well as details about the hunt day and each hunter. The software test suite also tests the functions that modify the database and ensures those actions are correct. During the software and laboratory testing, the database checked and operated on is not the real database but a copy of the database created when the tests are run. This ensures the latest data and table layout are tested but any issues in the software at this phase will not impact real data.

If software defects are discovered during use of the HDTS, it is hoped they will be reported such that it will be easy to create one or more test cases to reproduce the defect and then, after code changes are made, ensure those changes correct the defect and do not introduce additional defects.

4 Method and Scope of System Testing

After the software test suite reports no unit tests have failed, the software should be uploaded onto the field systems. If a change in the database layout is made, it should also be reloaded onto the field laptops, ensuring no data is lost from previous hunts.

Prior to running the test cases and turning the system on, modify the `settings.ini` file:

- Go to the `system_information` section
- Change the `detector_real` setting to `True`
- Change the `database_real` setting to `False`

Then the field system hardware should be connected and operated as during a hunt, as described in (Whiteside, 2017b). After setup, the following cases should be tested. The output values with nothing on the detector should be 0.0 - 1.0, but close to 0.5, depending on the background activity. The Priscilla phantom's activity should be within 16% of the listed values.

Test Case 1

- Put nothing on the detector.
- Input (under the "Process Animal tab)

  Animal Type: DEER

  Cinch #: 1234

  HUNTER ID: 000000HUN

  COMP. #: 1

  STAND #: 1

  WEIGHT (lbs): 100

  Live Weight: X
- Output (under the "Review Animal" tab)

  Cs 137 Meat Conc: 0.50 (0.0 - 1.0)

Test Case 2

- Put the Priscilla phantom on the detector.
- Input (under the "Process Animal tab)

  Animal Type: DEER

  Cinch #: 1234

HUNTER ID: 000000HUN

COMP. #: 1

STAND #: 1

WEIGHT (lbs): 100

Live Weight: X

- Output (under the "Review Animal" tab)

  Cs 137 Meat Conc: 3.56 (2.85 - 4.27) - note, this range will change over time as the Cs-137 decays

After running the test cases and exiting the HDTS software, modify the `settings.ini` file:

- Go to the `system_information` section
- Change the `detector_real` setting to `True`
- Change the `database_real` setting to `True`

After these steps have been completed, we can say the system is performing within its design parameters and is ready for use in the field.

5 References

Dixon, K. L. (2012). Hunter dose tracking system - excel (HDTS-XL) version 2012. Technical Report SRNL-TR-2012-00120, Rev. 0., Savannah River National Laboratory, Savannah River Site, Aiken SC.

Foley, T. Q. and Powell, A. (2010). Hunter dose tracking system verson 1.0. Technical report, Savannah River National Laboratory, Savannah River Site, Aiken SC.

Jannik, G. T. (2013). Software classification document for hunter dose tracking system excel. Technical Report Q-SWCD-A-00034 rev 2, Savannah River National Laboratory, Savannah River Site.

Myers, G. J., Sandler, C., Badgett, T., and Thomas, T. M. (2004). *The art of software testing*. John Wiley & Sons, Hoboken N.J, 2nd edition.

Whiteside, T. S. (2017a). HDTS 2017.0 Software Documentation. Technical Report SRNL-IM-2017-00021, Rev. 0., Savannah River National Laboratory, Savannah River Site, Aiken SC.

Whiteside, T. S. (2017b). HDTS 2017.0 User Guide. Technical Report SRNL-IM-2017-00020, Rev. 0., Savannah River National Laboratory, Savannah River Site, Aiken SC.

Whiteside, T. S. (2017c). Improvements to the Hunter Dose Tracking System. Technical Report SRNL-STI-2017-00091, Rev. 0., Savannah River National Laboratory, Savannah River Site, Aiken SC.

Appendix A Test Suite Files

Listing 1: Code to run test suite

```
#cs
testSuite Case generator / launcher
This program reads all the files of the project (*.au3 files) and reads their
    functions and calls a test case for each function.
Each test case checks for Assert Failures - if none then code passes

This is the master file that calls each sub-suites (HDTS and Det)
#ce

;declare scope of all variables before use
Opt('MustDeclareVars', 1)

#include <Misc.au3>
#include <File.au3>
#include <Array.au3>

;exit if another copy is already running
_Singleton("testSuiteGenerator")

Global $test_suite    = True
Global Const $inifile = _PathFull(@ScriptDir) & "\settings.ini"
Global $unit_counter              = 0
Global $unit_passed_counter       = 0
Global $unit_failed_counter       = 0
Global $case_counter              = 0
Global $function_counter          = 0
Global $untested_function_counter = 0

Global Const $summary_file    = _PathFull(@ScriptDir) & "\qa\summary.txt"
Global Const $table_file      = _PathFull(@ScriptDir) & "\qa\table.txt"
Global Const $sysdoc_settings = _PathFull(@ScriptDir) & "\docs\required_docs\tex\
    settings.tex"
Global Const $sysdoc_code     = _PathFull(@ScriptDir) & "\docs\required_docs\tex\
    code.tex"
Global $sfh = ""

;for getting necessary settings to test stuff
#include "tadinifuncs.au3"

;-------------------------------------------------------------------------------
;Prewritten tests
;-------------------------------------------------------------------------------
#include "testCasesHDTS.au3"
#include "testCasesDet.au3"


;-------------------------------------------------------------------------------
```

```
;Test Runner
;===============================================================================
Func UTAssert(ByRef $testArray, Const $bool, Const $msg = "Assert Failure", Const
    $erl = @ScriptLineNumber)
        Local $test_msg = @TAB & @TAB & @TAB & "Unit "
    If NOT $bool Then ;Unit Test was false, so tell the user
                $test_msg = $test_msg & "Line (" & $erl & ") := " & $msg & @CRLF
                $unit_failed_counter = $unit_failed_counter + 1
        Else
                $test_msg = $test_msg & "OK" & @CRLF
                $unit_passed_counter = $unit_passed_counter + 1
    EndIf
        FileWrite($sfh, $test_msg)
        ConsoleWrite($test_msg)

        _ArrayAdd($testArray, $bool)

        $unit_counter = $unit_counter + 1

    Return $bool
EndFunc

;===============================================================================
;Pre-test is before each individual test case of a function
;===============================================================================
Func PreTest(Const $TestNum)
        Local $msg = @TAB & @TAB & "Running case " & $TestNum & "..." & @CRLF
        FileWrite($sfh, $msg)
        ConsoleWrite($msg)
EndFunc

;===============================================================================
;Post-test is after each individual test case of a function
;===============================================================================
Func PostTest(ByRef $testArray)
        Local $msg = @TAB & @TAB & "...Failed"
        Local $tOK = True
        _ArrayDelete($testArray,0) ;_ArrayAdd makes a 0 based array, so clean that
            up here
        ;_ArrayDisplay($testArray)

        For $t In $testArray
                If ($t = True And $tOK = True) Then
                        $tOK = True
                Else
                        $tOK = False
                EndIf
        Next
        If $tOK = True Then
                $msg = @TAB & @TAB & "...OK"
        EndIf
```

```
        $msg = $msg & @CRLF
        FileWrite($sfh, $msg)
        ConsoleWrite($msg)

        $case_counter = $case_counter + 1

        Dim $testArray[1] ;clear the array
EndFunc


;-------------------------------------------------------------------------------
;PrePre-test is at before all individual test cases of a function
;-------------------------------------------------------------------------------
Func PrePreTest(Const $FuncName)
        Local $msg = @TAB & $FuncName & "..." & @CRLF
        FileWrite($sfh,$msg)
        ConsoleWrite($msg)
EndFunc

;-------------------------------------------------------------------------------
;PostPost-test is after all test cases of a function
;-------------------------------------------------------------------------------
Func PostPostTest()
        Local $msg = @TAB & "...Done" & @CRLF
        FileWrite($sfh,$msg)
        ConsoleWrite($msg)
        $function_counter = $function_counter + 1
EndFunc

;-----------------------------------------------
; Stub used until a test is written
;-----------------------------------------------
Func NoTestWritten($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    _ArrayAdd($tArray, True)
        Local $msg = @TAB & @TAB & @TAB & "no case written" & @CRLF
        FileWrite($sfh, $msg)
    ConsoleWrite($msg)
        ;unit_counter would be increased here
    PostTest($tArray)
        $case_counter = $case_counter - 1
        $untested_function_counter = $untested_function_counter + 1
    PostPostTest()
EndFunc


;-----------------------------------------------
;setup/teardown of database for testing purposes - these two functions should maybe
   be moved up to testSuite.au3
```

```
;-------------------------------------------------
Func SetupTestDB()
    ;$inifile is Global

    ;copy database
    Local $data_dir  = TadIniRead($inifile, "directories", "data_dir")
    Local $data_file = TadIniRead($inifile, "directories", "database")
    Local $data_temp = "HDTStemp.accdb"
    Local $db        = @ScriptDir & "\" & $data_dir & "\" & $data_file
    Local $dbtemp    = @ScriptDir & "\" & $data_dir & "\" & $data_temp
    FileCopy($db, $dbtemp)

    ;copy and update settings
    Local $initemp = TadIniRead($inifile,"system_information","initemp")
    FileCopy($inifile, $initemp)
    TadIniWrite($initemp, "directories", "database", $data_temp)

    Local $adata[2]
    $adata[0] = $dbtemp
    $adata[1] = $initemp
    Return $adata
EndFunc

Func TeardownTestDB($adata)
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]
    ;delete copies
    FileDelete($dbtemp)
    FileDelete($initemp)
EndFunc

;-------------------------------------------------------------------------------
;find out which *.au3 files are included
;-------------------------------------------------------------------------------
Func GetIncludeFiles($a_files)
        Local $ifiles = ""
        For $file in $a_files
                ;ConsoleWrite(@LF)
                Local $include_files = ""
                Local $fh = FileOpen($file)
                While 1
                        Local $line = FileReadLine($fh)
                        If @error = -1 Then
                                ExitLoop
                        Else
                                $line = StringStripWS($line,3)
                                If StringLeft($line, 10) = '#include "' Then
                                        Local $ifile = StringReplace($line,'#include
                                            "','")
                                        $ifile = StringReplace($ifile,'"','")
                                        $include_files = $include_files & $ifile & "
```

```
                                        ,"

                                ;ElseIf StringLeft($line, 4) == "Func" Then
                                ;       ConsoleWrite("x" & @CRLF)
                                EndIf
                        EndIf
                WEnd
                FileClose($fh)
                $include_files = StringTrimRight($include_files,1)
                Local $ainclude_files = StringSplit($include_files,",",2)
                ;_ArrayDisplay($ainclude_files)
                If StringLen($include_files) > 1 Then
                        ;ConsoleWrite("i: " & $include_files & @CRLF)
                        $ifiles = $ifiles & $include_files & ","
                EndIf
        Next
        $ifiles = StringTrimRight($ifiles,1)
        Local $a_ifiles = StringSplit($ifiles,",",2)
        $a_ifiles = _ArrayUnique($a_ifiles) ;return 1-based array
        _ArrayDelete($a_ifiles,0) ;return 0-based array

        ;_ArrayDisplay($a_ifiles)
        ;ConsoleWrite($ifiles & @CRLF)
        Return $a_ifiles
EndFunc

;-------------------------------------------------------------------------------
Func ParseSettingsAux($docfh,$lab,$doc,$sec)
        Local $msg = ""
        $msg = $msg & "\subsection{" & $lab & "}" & @CRLF
        $msg = $msg & $doc & @CRLF
        $msg = $msg & $sec & @CRLF
        FileWrite($docfh,$msg)
EndFunc

Func ParseSettings($inifile)
        Local $docfh  = FileOpen($sysdoc_settings,2)
        Local $fh     = FileOpen($inifile)
        Local $indoc  = False
        Local $insec  = False
        Local $infake = False
        Local $lab    = ""
        Local $doc    = ""
        Local $sec    = ""
        While 1
                Local $line = FileReadLine($fh)
                If @error = -1 Then ;EOF
                        $insec = False
                        $sec = $sec & "\end{lstlisting}" & @CRLF

                        ParseSettingsAux($docfh,$lab,$doc,$sec)
```

```
            ExitLoop
Else
            $line = StringStripWS($line,3)
            If StringLeft($line, 8) == ";/docbeg" Then
                    If $insec = True Then
                            $insec = False
                            $sec = $sec & "\end{lstlisting}" & @CRLF
                            ParseSettingsAux($docfh,$lab,$doc,$sec)
                    EndIf
                    $indoc = True
                    $doc   = ""
            ElseIf StringLeft($line, 8) == ";/docend" Then
                    $indoc = False
                    $doc = $doc & @CRLF
                    ;FileWrite($docfh,$doc)
            ElseIf $indoc = True Then
                    ;ConsoleWrite(StringLen($line) & "␣" & $line & @CRLF
                       )
                    If Stringlen($line) = 1 Then
                            $line = @CRLF & @CRLF
                    Else
                            $line = StringTrimLeft($line,1) ;remove ;
                               from begining of String
                            $line = StringStripWS($line, 3)
                            If StringLeft($line,1) == "\" Then
                                    $line = @CRLF & $line & @CRLF
                            ElseIf StringRight($line,2) == "\\" Then
                                    $line = $line & @CRLF
                            ;ElseIf $line == "" Then
                            ;       $line = $line & @CRLF
                            Else
                                    $line = $line & "␣"
                            EndIf
                    EndIf
                    $doc = $doc & $line
            EndIf

            If StringLeft($line, 1) == "[" Then
                    If $insec = True Then ;in previous section, end it
                       and start new one here
                            $insec = False
                            $sec = $sec & "\end{lstlisting}" & @CRLF
                            ParseSettingsAux($docfh,$lab,$doc,$sec)
                    EndIf
                    $insec = True
                    $lab = StringReplace($line,"_","\_")
                    $sec = "\begin{lstlisting}" & @CRLF
                    $sec = $sec & $line & @CRLF
                    If StringInStr($line,"fake_system") Then
                            $infake = True
```

```
                EndIf
            ElseIf $insec = True Then
                If StringLeft($line,1) == ";" Then ;don't include
                    comments in listing
                        $sec = $sec
                Else
                        If $infake = True Then
                                If StringInStr($line, "=") Then
                                        Local $aline = StringSplit(
                                            $line,"=",2)
                                        Local $key = $aline[0]
                                        Local $val = $aline[1]
                                        Local $spec = StringReplace(
                                            $val,'"','')
                                        Local $aspec = StringSplit(
                                            $spec,",",2)

                                        Local $speclines = ""
                                        Local $c = 0
                                        For $i In $aspec
                                                $speclines =
                                                    $speclines & $i
                                                    & ","
                                                $c = $c + 1
                                                If Mod($c,16) = 0
                                                    Then
                                                        $speclines =

                                                            $speclines
                                                            & @CRLF
                                                EndIf
                                        Next

                                        $speclines = StringTrimRight
                                            ($speclines,1)

                                        $sec = $sec & $key & '="' &
                                            $speclines & '"' & @CRLF
                                Else
                                        $sec = $sec & $line & @CRLF
                                EndIf
                        Else
                                $sec = $sec & $line & @CRLF
                        EndIf

                EndIf
            EndIf

        EndIf
WEnd
FileClose($fh)
```

```
        FileClose($docfh)
EndFunc



Func ParseCode($file)

        Local $docfh    = FileOpen($sysdoc_code,2)
        Local $fh       = FileOpen($file)
        Local $indoc    = False
        Local $insec    = False
        Local $infake   = False
        Local $lab      = ""
        Local $doc      = ""
        Local $sec      = ""
        Local $funcname = ""
        Local $funcargs = ""

        While 1
                Local $line = FileReadLine($fh)
                If @error = -1 Then ;EOF
                        $insec = False
                        ;$sec = $sec & "\end{lstlisting}" & @CRLF

                        ;ParseSettingsAux($docfh,$lab,$doc,$sec)

                        ExitLoop
                Else
                        $line = StringStripWS($line,3)
                        If StringLeft($line, 8) == ";/docbeg" Then
                                #cs
                                If $insec = True Then
                                        $insec = False
                                        $sec = $sec & "\end{lstlisting}" & @CRLF
                                        ParseSettingsAux($docfh,$lab,$doc,$sec)
                                EndIf
                                #ce
                                $indoc = True
                                $doc   = ""
                        ElseIf StringLeft($line, 8) == ";/docend" Then
                                $indoc = False
                                $doc = $doc & @CRLF
                                ConsoleWrite($doc)
                                ;FileWrite($docfh,$doc)
                        ElseIf $indoc = True Then
                                If Stringlen($line) = 1 Then
                                        $line = @CRLF & @CRLF
                                Else
                                        $line = StringTrimLeft($line,1) ;remove ;
                                           from begining of String
                                        $line = StringStripWS($line, 3)
```

```
                If StringLeft($line,1) == "\" Then
                        $line = @CRLF & $line & @CRLF
                ElseIf StringRight($line,2) == "\\" Then
                        $line = $line & @CRLF
                ;ElseIf $line == "" Then
                ;       $line = $line & @CRLF
                Else
                        $line = $line & "␣"
                EndIf
        EndIf
        $doc = $doc & $line
ElseIf StringLeft($line, 4) == "Func" Then
        $funcname = $line
        $funcargs = $line

        $funcname = ""

EndIf

#cs
If StringLeft($line, 1) == "[" Then
        If $insec = True Then ;in previous section, end it
           and start new one here
                $insec = False
                $sec = $sec & "\end{lstlisting}" & @CRLF
                ParseSettingsAux($docfh,$lab,$doc,$sec)
        EndIf
        $insec = True
        $lab = StringReplace($line,"_","\_")
        $sec = "\begin{lstlisting}" & @CRLF
        $sec = $sec & $line & @CRLF
        If StringInStr($line,"fake_system") Then
                $infake = True
        EndIf
ElseIf $insec = True Then
        If StringLeft($line,1) == ";" Then ;don't include
           comments in listing
                $sec = $sec
        Else
                If $infake = True Then
                        If StringInStr($line, "=") Then
                                Local $aline = StringSplit(
                                   $line,"=",2)
                                Local $key = $aline[0]
                                Local $val = $aline[1]
                                Local $spec = StringReplace(
                                   $val,'"','')
                                Local $aspec = StringSplit(
                                   $spec,",",2)

                                Local $speclines = ""
```

```
                                                Local $c = 0
                                                For $i In $aspec
                                                        $speclines =
                                                            $speclines & $i
                                                            & ","
                                                        $c = $c + 1
                                                        If Mod($c,16) = 0
                                                            Then
                                                                $speclines =

                                                                    $speclines
                                                                    & @CRLF
                                                        EndIf
                                                Next

                                                $speclines = StringTrimRight
                                                    ($speclines,1)

                                                $sec = $sec & $key & '="' &
                                                    $speclines & '"' & @CRLF
                                            Else
                                                $sec = $sec & $line & @CRLF
                                            EndIf
                                        Else
                                            $sec = $sec & $line & @CRLF
                                        EndIf

                                    EndIf
                                EndIf
                                #ce

                            EndIf

                    WEnd
                    FileClose($fh)
                    FileClose($docfh)
EndFunc


;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
;MAIN CODE HERE
;-------------------------------------------------------------------------------
Func _MainTest()

        ParseSettings($inifile)
        ;Local $file = _PathFull(@ScriptDir) & "\HDTS.au3"
        ;ParseCode($file)
        ;Return
```

```
    $sfh = FileOpen($summary_file,2)
    Local $msg = ""
    $msg = $msg & ";----------------------------------------" & @CRLF
$msg = $msg & ";change $test_suite = False to show MsgBoxes" & @CRLF
    $msg = $msg & ";test failures are shown in Console Window  " & @CRLF
$msg = $msg & ";----------------------------------------" & @CRLF
$msg = $msg & "------ Start of Tests -------" & @CRLF
    $msg = $msg & @CRLF
    FileWrite($sfh,$msg)
    ConsoleWrite($msg)

    ;TODO programmatically get list of all used files - FOR NOW HARDCODING
    Local $a_files[2]
    $a_files[0] = "HDTS.au3"
    $a_files[1] = "guiDet.au3"

    Local $a_i_1files   = GetIncludeFiles($a_files)
    Local $a_i_2files   = GetIncludeFiles($a_i_1files)
    Local $a_i_3files   = GetIncludeFiles($a_i_2files)
    _ArrayConcatenate($a_files, $a_i_1files)
    _ArrayConcatenate($a_files, $a_i_2files)
    _ArrayConcatenate($a_files, $a_i_3files)
    $a_files = _ArrayUnique($a_files)
    _ArrayDelete($a_files,0)
    ;_ArrayDisplay($a_files)

    ;Now that we have the list of files with functions used in this application
    ;Loop over each and
    ;  1) find the name of the function
    ;  2) execute the test case for that function
    For $file In $a_files
        $msg = ""
        $msg = $msg & "------ Testing " & $file & "  -------" & @CRLF
        FileWrite($sfh,$msg)
        ConsoleWrite($msg)
        ;ConsoleWrite("Getting Funcs from " & $file & @CRLF)



        Local $fh = FileOpen($file)
        While 1
            Local $line = FileReadLine($fh)
            If @error = -1 Then
                ExitLoop
            Else
                $line = StringStripWS($line,3)
                If StringLeft($line, 4) == "Func" Then
                    Local $func = StringTrimLeft($line,4) ;
                       remove the Func string - doing this vs
                       StringReplace b/c sometimes have Func in
```

```
                                        function name ie ADOErrFunc1 in
                                       lib_dbcalls.au3
                                    $func = StringStripWS($func,3)
                                    Local $whole_func = $func
                                    ;ConsoleWrite(@TAB & $whole_func & @CRLF)

                                    Local $a_func     = StringSplit($func,"(",2)
                                    Local $func_name = $a_func[0]
                                    Local $test_func = "Test_" & $func_name & "(
                                       " & '"' & $func_name & '"' & ")" ;
                                       passing name so can run pretest cleanly
                                    ;ConsoleWrite($test_func & @CRLF)
                                    Execute($test_func)
                                    If @error <> 0 Then
                                            MsgBox(0,"Error","Function " &
                                                $test_func & " failed!")
                                            Exit
                                    EndIf
                                    Local $unit_limit = 999 ;5
                                    Local $limit_test = True

                                    If $limit_test = True and $unit_counter >
                                       $unit_limit Then
                                            FileClose($fh)
                                            $msg = @CRLF
                                            FileWrite($sfh,$msg)
                                            ConsoleWrite($msg)
                                            ExitLoop 2 ;get all the way out
                                    EndIf
                                EndIf
                            EndIf
                        WEnd
                        FileClose($fh)
                        $msg = @CRLF
                        FileWrite($sfh,$msg)
                        ConsoleWrite($msg)
Next
;start of table
Local $tmsg = ""
$tmsg = $tmsg & "\begin{center}" & @CRLF
$tmsg = $tmsg & "\begin{longtable}{lr}" & @CRLF
$tmsg = $tmsg & "\caption{Results of testing}\\" & @CRLF
$tmsg = $tmsg & "\label{table:results}" & @CRLF
$tmsg = $tmsg & "\endfirsthead" & @CRLF
$tmsg = $tmsg & "\toprule    " & @CRLF

;start of summary_table
Local $smsg = ""

;function info section
$msg = ""
```

```
$msg = $msg & "Num␣Functions␣␣␣␣␣␣␣␣␣␣␣␣&␣" & $function_counter & "␣\\" &
    @CRLF
$msg = $msg & "Num␣Functions␣not␣tested␣&␣" & $untested_function_counter & "
    ␣\\" & @CRLF
$msg = $msg & "Num␣Functions␣tested␣␣␣␣␣&␣" & $function_counter -
    $untested_function_counter & "␣\\" & @CRLF

;output and divider
$tmsg = $tmsg & $msg
$tmsg = $tmsg & "\midrule" & @CRLF

$smsg = $smsg & $msg
$smsg = $smsg & @CRLF

;case info section
$msg = ""
$msg = $msg & "Num␣Cases␣tested␣␣␣␣␣␣␣␣&␣" & $case_counter & "␣\\" & @CRLF

;output and divider
$tmsg = $tmsg & $msg
$tmsg = $tmsg & "\midrule" & @CRLF

$smsg = $smsg & $msg
$smsg = $smsg & @CRLF

;unit info section
$msg = ""
$msg = $msg & "Num␣Units␣tested␣␣␣␣␣␣␣␣&␣" & $unit_counter & "␣\\" & @CRLF
$msg = $msg & "Num␣Units␣passed␣␣␣␣␣␣␣␣&␣" & $unit_passed_counter & "␣\\" &
    @CRLF
$msg = $msg & "Num␣Units␣failed␣␣␣␣␣␣␣␣&␣" & $unit_failed_counter   & "␣\\"
    & @CRLF

;output and divider
$tmsg = $tmsg & $msg
$smsg = $smsg & $msg

;end of summary_table
$smsg = $smsg & @CRLF

;end of table
$tmsg = $tmsg & "\bottomrule" & @CRLF
$tmsg = $tmsg & "\end{longtable}" & @CRLF
$tmsg = $tmsg & "\end{center}" & @CRLF

FileWrite($sfh,$smsg)
ConsoleWrite($smsg)
FileClose($sfh)

Local $tfh = FileOpen($table_file,2)
FileWrite($tfh,$tmsg)
```

```
        FileClose($tfh)

EndFunc

;launch the code here
_MainTest()
```

Exit

Listing 2: Code to run test suite

```
;-------------------------------------------------------------------------------
;testCases
;these are the testCases run by testSuite
;-------------------------------------------------------------------------------

;declare scope of all variables before use
Opt('MustDeclareVars', 1)

;exit if another copy is already running
_Singleton("testCases")

#include-once

#include "tadinifuncs.au3"
#include "lib_hdts.au3"

Func Test_HDTSSetupTestDB($func_name)
    NoTestWritten($func_name)  ;these are for creating fake db and initemp
EndFunc
Func Test_HDTSTeardownTestDB($func_name)
    NoTestWritten($func_name)  ;this is for removeing fake db and initemp
EndFunc


Func Test_GetHuntDay($func_name)
    NoTestWritten($func_name)
EndFunc

Func Test_GetHuntTypeUI($func_name)
    ;GetHuntTypeUI()
    NoTestWritten($func_name)
EndFunc

Func Test_DetectorButton($func_name)
    ;DetectorButton()
    NoTestWritten($func_name)
 EndFunc

Func Test_EndHuntButton($func_name)
    ;EndHuntButton()
```

```
    NoTestWritten($func_name)
EndFunc

Func Test_QueryButton($func_name)
    ;QueryButton()
    NoTestWritten($func_name)
EndFunc

Func Test_CLOSEClicked($func_name)
    ;CLOSEClicked()
    NoTestWritten($func_name)
EndFunc

Func Test_MainHDTS($func_name)
    NoTestWritten($func_name)
EndFunc

;FROM lib_hdts

Func Test_GetPastHuntDays($func_name)
    ;GetPastHuntDays($inifile)
    NoTestWritten($func_name)
EndFunc
Func Test_ValidHuntDay($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    UTAssert($tArray, ValidHuntDay($inifile, "20170101") = True)   ;real date
    PostTest($tArray)

    PreTest(2)
    UTAssert($tArray, ValidHuntDay($inifile, "20170001") = False) ;wrong month
    PostTest($tArray)

    PreTest(3)
    UTAssert($tArray, ValidHuntDay($inifile, "20171301") = False) ;wrong month
    PostTest($tArray)

    PreTest(4)
    UTAssert($tArray, ValidHuntDay($inifile, "20171100") = False) ;wrong day
    PostTest($tArray)

    PreTest(5)
    UTAssert($tArray, ValidHuntDay($inifile, "20171132") = False) ;wrong day
    PostTest($tArray)

    PreTest(6)
    UTAssert($tArray, ValidHuntDay($inifile, "2017") = False)      ;bad format
    PostTest($tArray)

    PostPostTest()
```

```
EndFunc
Func Test_NotNowHuntOK($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, NotNowHuntOK("20170101") = True ) ;real date in the past
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    Local $dd  = @MDAY
    $dd = StringFormat("%02s",$dd)
    UTAssert($tArray, NotNowHuntOK(@YEAR & @MON & $dd) = True ) ;today real date
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, NotNowHuntOK("21000101") = True ) ;future real date (but not
        too far b/c _DateDiff has limit on size of output
    PostTest($tArray)
    PostPostTest()
EndFunc
Func Test_HuntEndedOK($func_name)
    ;HuntEndedOK($inifile, $answer) = True) ;this is an update query so not testing
        right now
    NoTestWritten($func_name)
EndFunc
Func Test_CreateHDTSTable($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata  = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    PreTest(1)

    Local $yyp1   = @YEAR + 1
    Local $huntday = $yyp1 & "1231"
    CreateHDTSTable($initemp, $huntday)

    Local $prev_dose = 0

    Local $adoCon = OpenDatabase($initemp)
    Local $adoRs  = ObjCreate("ADODB.Recordset")
    $adoRs.CursorType = 3 ;static, forward-only cursor
    $adoRs.LockType   = 1 ;readonly
    Local $qry = "SELECT␣prev_years_dose␣FROM␣hdts␣WHERE␣hunterid␣=␣'999999HUN'"
    $adoRs.Open($qry, $adoCon)
```

```
    While Not $adoRs.EOF
        $prev_dose = $adoRs.Fields("prev_years_dose").Value
        $adoRs.MoveNext
    WEnd
    $adoRs.Close
    $adoCon.Close
    $prev_dose = StringFormat("%.1f",$prev_dose)*1

    UTAssert($tArray, $prev_dose = 108.3)

    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_CreateHuntDayTable($func_name)
    ; - pass through function - see CreateHuntDayTable_DB_Call")
    NoTestWritten($func_name)
EndFunc

Func Test_SetADetInfo($func_name)
    ;SetADetInfo(ByRef $amap, ByRef $max_det, ByRef $a_ctrlid, ByRef $a_detpid)
    NoTestWritten($func_name)
EndFunc
Func Test_GetMasterDetPos($func_name)
    ;GetMasterDetPos($isFake, $dll_h, $nDetMax, $digi_snum)
    NoTestWritten($func_name)
EndFunc
Func Test_MapDefinedToPhysical($func_name)
    ;MapDefinedToPhysical(Const ByRef $dll_h, Const ByRef $inifile, ByRef $amap,
    ByRef $max_det, ByRef $a_ctrlid, ByRef $a_detpid)
    NoTestWritten($func_name)
EndFunc
Func Test_GetHuntStats($func_name)
    ;GetHuntStats($inifile, $huntday);runs the query and displays on screen
    NoTestWritten($func_name)
EndFunc

Func Test_EndHunt($func_name)
    ;EndHunt($inifile, $huntday) ;do query to update ended
    NoTestWritten($func_name)
EndFunc
```

Listing 3: Code to run test suite

```
;-------------------------------------------------------------------------------
;testCases
```

```
;these are the testCases run by testSuite
;-------------------------------------------------------------------------------

;declare scope of all variables before use
Opt('MustDeclareVars', 1)

;exit if another copy is already running
_Singleton("testCasesDet")

#include-once

#include "method.au3"

;-----------------------------------------------
;Getting Funcs from guiDet.au3

Func Test_GUIDetUpdateIni($func_name)
    NoTestWritten($func_name) ;this is for repointing the inifile during office/lab
       debugging
EndFunc


Func Test_ControlChart($func_name)
        ;ControlChart()
        NoTestWritten($func_name)
EndFunc

Func Test_ControlChartDet($func_name)
        ;ControlChartDet()
        NoTestWritten($func_name)
EndFunc

;this is in method.au3
Func Test_GetPhantomExpectedConc($func_name)
        NoTestWritten($func_name)
EndFunc
Func Test_GetControlChart($func_name)
        NoTestWritten($func_name)
EndFunc
Func Test_GetPhantomActivityQuery($func_name)
        NoTestWritten($func_name)
EndFunc



Func Test_OnDet($func_name)
    ;OnDet()
    NoTestWritten($func_name)
EndFunc
Func Test_OffDet($func_name)
    ;OffDet()
```

```
    NoTestWritten($func_name)
EndFunc
Func Test_WarmupDet($func_name)
    ;WarmupDet()
    NoTestWritten($func_name)
EndFunc
Func Test_CalibDet($func_name)
    ;CalibDet()
    NoTestWritten($func_name)
EndFunc
Func Test_BkgDet($func_name)
    ;BkgDet()
    NoTestWritten($func_name)
EndFunc
Func Test_HandleDetError($func_name)
    ;HandleDetError()
    NoTestWritten($func_name)
EndFunc
Func Test_ValidateProcessReprocessInput($func_name)
    ;ValidateProcessReprocessInput($inifile, $a_handles)
    NoTestWritten($func_name)
EndFunc
Func Test_CountSampleTop($func_name)
    ;CountSampleTop($inifile, $a_data, $ddid, $pdid)
    NoTestWritten($func_name)
EndFunc
Func Test_CountPhantomTop($func_name)
    ;CountPhantomTop($inifile, $ddid, $pdid)
    NoTestWritten($func_name)
EndFunc
Func Test_CountSample($func_name)
    ;CountSample($hunter_id, $cinch_num, $animal_type, $animal_wt_live, $hunt_cmpt,
        $hunt_stand, $ddid, $pdid, $animal_id, $huntday)
    NoTestWritten($func_name)
EndFunc
Func Test_CountPhantom($func_name)
    ;CountPhantom($animal_type, $animal_wt, $ddid, $pdid, $animal_id, $huntday)
    NoTestWritten($func_name)
EndFunc
Func Test_CountPhantomOrSample($func_name)
    ;CountPhantomOrSample($animal_type, $animal_wt_live, $ddid, $pdid, $animal_id,
        $huntday)
    NoTestWritten($func_name)
EndFunc
Func Test_AddAnimal($func_name)
    ;AddAnimal($huntday, $HunterId, $CinchNum, $AnimalType, $AnimalWeight,
        $AnimalWeightType, $HuntCompartment, $HuntStand, $ddid)
    NoTestWritten($func_name)
EndFunc
Func Test_StartButton($func_name)
    ;StartButton()
```

```
    NoTestWritten($func_name)
EndFunc
Func Test_StopButton($func_name)
    ;StopButton()
    NoTestWritten($func_name)
EndFunc
Func Test_OnButton($func_name)
    ;OnButton()
    NoTestWritten($func_name)
EndFunc
Func Test_OffButton($func_name)
    ;OffButton()
    NoTestWritten($func_name)
EndFunc
Func Test_CalibButton($func_name)
    ;CalibButton()
    NoTestWritten($func_name)
EndFunc
Func Test_BackgroundButton($func_name)
    ;BackgroundButton()
    NoTestWritten($func_name)
EndFunc
Func Test_PhantomButton($func_name)
    ;PhantomButton()
    NoTestWritten($func_name)
EndFunc
Func Test_ClearForm($func_name)
    ;ClearForm($a_handles)
    NoTestWritten($func_name)
EndFunc
Func Test_DisableInput($func_name)
    ;DisableInput($a_handles)
    NoTestWritten($func_name)
EndFunc
Func Test_EnableInput($func_name)
    ;EnableInput($a_handles)
    NoTestWritten($func_name)
EndFunc
Func Test_ProcessButton($func_name)
    ;ProcessButton()
    NoTestWritten($func_name)
EndFunc
Func Test_ReprocessButton($func_name)
    ;ReprocessButton()
    NoTestWritten($func_name)
EndFunc
Func Test_DeleteAnimalButton($func_name)
    ;DeleteAnimalButton()
    NoTestWritten($func_name)
EndFunc
Func Test_ShowSpectraButton($func_name)
```

```
    ;ShowSpectraButton()
    NoTestWritten($func_name)
EndFunc
Func Test_AddHunterButton($func_name)
    ;AddHunterButton()
    NoTestWritten($func_name)
EndFunc
#cs same function name as in testCasesHDTS - either make common or unique
Func Test_CloseClicked($func_name)
    PrePreTest($func_name)
     ;CloseClicked()
     ConsoleWrite("no test written")
     PostPostTest()
EndFunc
#ce
Func Test_UpdateAnimalIdComboAux($func_name)
    ;UpdateAnimalIdComboAux($h_to_update)
    NoTestWritten($func_name)
EndFunc
Func Test_UpdateAnimalIdCombo($func_name)
    ;UpdateAnimalIdCombo()
    NoTestWritten($func_name)
EndFunc
Func Test_UpdateAnimalIdComboDelete($func_name)
    ;UpdateAnimalIdComboDelete()
    NoTestWritten($func_name)
EndFunc
Func Test_UpdateAnimalIdComboReview($func_name)
    ;UpdateAnimalIdComboReview()
    NoTestWritten($func_name)
EndFunc
Func Test_UpdateAnimalIdComboReprocess($func_name)
    ;UpdateAnimalIdComboReprocess()
    NoTestWritten($func_name)
EndFunc

Func Test_FillInputReprocess($func_name)
    ;FillInputReprocess()
    NoTestWritten($func_name)
EndFunc

Func Test_FillInputReview($func_name)
    ;FillInputReview()
    NoTestWritten($func_name)
EndFunc

Func Test_FillInputAux1($func_name)
    ;FillInputAux1($inifile, $huntday, $a_handles)
    NoTestWritten($func_name)
EndFunc
```

```
Func Test_FillInputAux2($func_name)
    ;FillInputAux2($inifile, $huntday, $a_handles)
    NoTestWritten($func_name)
EndFunc

Func Test_SetAnimalTypes($func_name)
    NoTestWritten($func_name)
EndFunc

Func Test_SetLiveorDressed($func_name)
    NoTestWritten($func_name)
EndFunc

Func Test_MainDet($func_name)
    NoTestWritten($func_name)
EndFunc

;-------------------------------------------------------------------------------
; Getting Funcs from method.au3
;-------------------------------------------------------------------------------
Func Test_ErrorHandler($func_name)
    ;ErrorHandler(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, Const
        ByRef $msg)
    NoTestWritten($func_name)
EndFunc

Func Test_ChecksumOK($func_name)
    ;ChecksumOK($string)
    ;these are from the ORTEC manual
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$C00000087") = 87)
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$C00001088") = 88)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$C00002089") = 89)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$C00003090") = 90)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$C00250094") = 94)
```

```
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$C01023093") = 93)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$C01024094") = 94)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$C00256100") = 100)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$G0000000000075") = 75)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$G0000000001076") = 76)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$G2147483646120") = 120)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$G2147483647121") = 121)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$G4294967294131") = 131)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$G4294967295132") = 132)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$D0000000000072") = 72)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$D0100000050078") = 78)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, ChecksumOK("$D0000001024079") = 79)
PostTest($tArray)
$t = $t + 1
PreTest($t)
```

```
    UTAssert($tArray, ChecksumOK("$D0000000512080") = 80)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$D0215000150086") = 86)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ChecksumOK("$D0051200512088") = 88)
    PostTest($tArray)
    ;UTAssert(ChecksumOK("$D0100000007082") = 82)  ;82 <- 80 is actual - typo in
        manual
    ;UTAssert(ChecksumOK("$D0100000002076") = 76)  ;76 <- 75 is actual - typo in
        manual
    ;UTAssert(ChecksumOK("$D0100000003077") = 77)  ;77 <- 76 is actual - typo in
        manual

    PostPostTest()
EndFunc
Func Test_ParseDollarResponse($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, ParseDollarResponse("$G4294967295132") = 4294967295)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ParseDollarResponse("$G0000000001076") = 0000000001)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ParseDollarResponse("$FTSWISGREAT")    = "TSWISGREAT")
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_methodMIOComm($func_name)
    ;methodMIOComm(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, Const
        $cmd, $i, Const ByRef $ti, Const ByRef $verbose)
    NoTestWritten($func_name)
EndFunc
Func Test_GetDetectorInformation($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    UTAssert($tArray, GetDetectorInformation(1) = "detector_1_information")
    PostTest($tArray)
    PostPostTest()
EndFunc
Func Test_GetSpeHeader($func_name)
```

```
    ;GetSpeHeader($huntdate_id, $live_time, $real_time, $wStartChan, $wNumChans)
    NoTestWritten($func_name)
EndFunc
Func Test_GetSpeFooter($func_name)
    ;GetSpeFooter($cs_roi_start, $roi_numchan, $ene_calib, $ene_units)
    NoTestWritten($func_name)
EndFunc
Func Test_SpectrumSetup($func_name)
    ;SpectrumSetup(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef
        $cs_roi_start, Const ByRef $roi_numchan, Const ByRef $hDet, ByRef $i, Const
        ByRef $ti, Const ByRef $verbose)
    NoTestWritten($func_name)
EndFunc
Func Test_DetectorReady($func_name)
    ;DetectorReady($isReal, Const ByRef $dll_h, $inifile, $ddid, $hDet, $i, $ti,
        $verbose)  ;i gets increased by 2
    NoTestWritten($func_name)
EndFunc
Func Test_GetROICounts($func_name)
    ;GetROICounts($chn, $counts, $aChnInfo, ByRef $leftCounts, ByRef $peakCounts,
        ByRef $rightCounts)
    NoTestWritten($func_name)
EndFunc
Func Test_GetPeakInfo($func_name)
    ;GetPeakInfo($inifile, $detector_information, $element, ByRef
        $element_roi_startchn, ByRef $element_roi_numchn, ByRef $peak_chn)
    NoTestWritten($func_name)
EndFunc
Func Test_Count($func_name)
    ;Count($isReal, $dll_h, $inifile, $ddid, $hDet, $huntdate_id, $ticks, $CsCounts,
         $CsCountsLeft, $CsCountsRight, $EuCPM, $DetOK)
    NoTestWritten($func_name)
EndFunc
Func Test_CalcConc($func_name)
    ;CalcConc($inifile, $in_animal_type, $animal_wt_live, $totalCounts, $verbose)

    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)

    Local $aConc[2] ;aConc[0] = concWhole aConc[1] = concMeat
    Local $concWhole = 0.0
    Local $concMeat  = 0.0
    ;                              animal,     wt,   cpm, verbose
    ;---------------- DEER ------------------------------
    $aConc = CalcConc($inifile, "DEER",  100, 400, True)
    $concWhole = StringFormat("%.3f",$aConc[0] )*1
    $concMeat  = StringFormat("%.3f",$aConc[1] )*1
    UTAssert($tArray, $concWhole = 1.029)     ;pci/g
    UTAssert($tArray, $concMeat  = 1.303)     ;pCi/g
```

```
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DRESSED_DEER", 100, 400, True)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 1.191)    ;pci/g
UTAssert($tArray, $concMeat  = 1.678)    ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DEER", 40, 400, True) ;wt below min - does linear
    extrapolation
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 1.848)    ;pci/g
UTAssert($tArray, $concMeat  = 2.339)    ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "COYOTE", 40, 400, True) ;count small deer like
    coyote (maybe should combine this test w/ above)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
Local $concWholeC = StringFormat("%.3f",$aConc[0])*1
Local $concMeatC  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = $concWholeC )   ;pci/g
;UTAssert($tArray, $concMeat  = $concMeatC)    ;pCi/g          ;cannot compare
    concMeats b/c Deer and Coyote have different BCF
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DEER", 400, 400, True) ;wt above max  - does
    linear extrapolation
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 0.706)    ;pci/g
UTAssert($tArray, $concMeat  = 0.894)    ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DEER", 100, 0, True) ;what happens with 0 counts
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
```

```
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 0)     ;pci/g
UTAssert($tArray, $concMeat  = 0)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DEER",  100, -100, True) ;what happens with neg
   counts - throw error msg and set to 0
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 0)     ;pci/g
UTAssert($tArray, $concMeat  = 0)     ;pCi/g
PostTest($tArray)

;--------------- HOG ------------------------------
$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "HOG",  100, 400, True)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 1.057)     ;pci/g
UTAssert($tArray, $concMeat  = 1.854)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DRESSED_HOG",  100, 400, True)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 1.232)     ;pci/g
UTAssert($tArray, $concMeat  = 2.161)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "HOG",  40, 400, True) ;wt below min - counts as a
   turkey, calculates activity (concMeat like a HOG)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 0.989)     ;pci/g
UTAssert($tArray, $concMeat  = 1.735)     ;pCi/g If counted as TURKEY, this would
    be 1.978
PostTest($tArray)

$t = $t + 1
PreTest($t)
```

```
$aConc = CalcConc($inifile, "TURKEY", 40, 400, True) ;counting a turkey to show
    that above is same
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
Local $concWholeT = StringFormat("%.3f",$aConc[0])*1
Local $concMeatT  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = $concWholeT )    ;pci/g
;UTAssert($tArray, $concMeat  = 1.978 )    ;pCi/g          ;should not compare
   concMeats b/c HOG and TURKEY have different BCF
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "HOG", 400, 400, True) ;wt above max - does linear
    extrapolation
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 0.660)    ;pci/g
UTAssert($tArray, $concMeat  = 1.158)    ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "HOG", 100, 0, True) ;what happens with 0 counts
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 0)    ;pci/g
UTAssert($tArray, $concMeat  = 0)    ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "HOG", 100, -100, True) ;what happens with neg
   counts - throw error msg and set to 0
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 0)    ;pci/g
UTAssert($tArray, $concMeat  = 0)    ;pCi/g
PostTest($tArray)

;---------------- TURKEY ------------------------------
$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "TURKEY", 30, 400, True)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 1.076)    ;pci/g
UTAssert($tArray, $concMeat  = 2.153)    ;pCi/g
```

```
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DRESSED_TURKEY", 30, 400, True); this isn't
    allowed from the UI - what is going to happen - it works
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 1.209)     ;pci/g
UTAssert($tArray, $concMeat  = 2.418)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "TURKEY", 10, 400, True) ;wt below min
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 1.952)     ;pci/g
UTAssert($tArray, $concMeat  = 3.904)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "TURKEY", 50, 400, True) ;wt above max - does
    linear extrapolation
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 0.946)     ;pci/g
UTAssert($tArray, $concMeat  = 1.892)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "TURKEY", 30, 0, True) ;what happens with 0 counts
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
UTAssert($tArray, $concWhole = 0)     ;pci/g
UTAssert($tArray, $concMeat  = 0)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "TURKEY", 30, -100, True) ;what happens with neg
    counts - throw error msg and set to 0
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0])*1
$concMeat  = StringFormat("%.3f",$aConc[1])*1
```

```
UTAssert($tArray, $concWhole = 0)     ;pci/g
UTAssert($tArray, $concMeat  = 0)     ;pCi/g
PostTest($tArray)

;---------------- COYOTE ----------------------------------
$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "COYOTE",  50, 400, True)
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 1.589)     ;pci/g
UTAssert($tArray, $concMeat  = 3.178)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "DRESSED_COYOTE",  50, 400, True); this isn't
    allowed from the UI - but it works
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 1.589)     ;pci/g
UTAssert($tArray, $concMeat  = 3.178)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "COYOTE",  10, 400, True) ;wt below min
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 3.696)     ;pci/g
UTAssert($tArray, $concMeat  = 7.391)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "COYOTE",  150, 400, True) ;wt above max - does
    linear extrapolation
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
$concWhole = StringFormat("%.3f",$aConc[0] )*1
$concMeat  = StringFormat("%.3f",$aConc[1] )*1
UTAssert($tArray, $concWhole = 0.854)     ;pci/g
UTAssert($tArray, $concMeat  = 1.709)     ;pCi/g
PostTest($tArray)

$t = $t + 1
PreTest($t)
$aConc = CalcConc($inifile, "COYOTE",  50, 0, True) ;what happens with 0 counts
;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
```

```
    $concWhole = StringFormat("%.3f",$aConc[0] )*1
    $concMeat  = StringFormat("%.3f",$aConc[1] )*1
    UTAssert($tArray, $concWhole = 0)     ;pci/g
    UTAssert($tArray, $concMeat  = 0)     ;pCi/g
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    $aConc = CalcConc($inifile, "COYOTE",  50, -100, True) ;what happens with neg
       counts - throw error msg and set to 0
    ;ConsoleWrite($aConc[0] & "␣" & $aConc[1] & @CRLF)
    $concWhole = StringFormat("%.3f",$aConc[0] )*1
    $concMeat  = StringFormat("%.3f",$aConc[1] )*1
    UTAssert($tArray, $concWhole = 0)     ;pci/g
    UTAssert($tArray, $concMeat  = 0)     ;pCi/g
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_VoltageFlux($func_name)
    ;VoltageFlux(ByRef $DetHV, Const ByRef $voltage_variable)
    NoTestWritten($func_name)
EndFunc
Func Test_GetVoltage($func_name)
    ;doesn't check checksum
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, GetVoltage("$D01000000050078") = 1000)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, GetVoltage("$D00900000050077") = 900)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test__binary($func_name)
    ;_binary($num)
    ;least significant bit first
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, _binary(0) = "0")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(1) = "1")
```

```
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(2) = "01")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(3) = "11")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(4) = "001")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(5) = "101")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(6) = "011")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(7) = "111")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, _binary(8) = "0001")
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_GetHVState($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, GetHVState("$D0100000003077") = "disabled")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, GetHVState("$D0100000002076") = "disabled")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, GetHVState("$D0100000007076") = "enabled")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, GetHVState("$D0100000000077") = "disabled")
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, GetHVState("$D0100000004077") = "enabled")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, GetHVState("$D0100000001077") = "disabled")
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_ConvertTime($func_name)
    ;ConvertTime(Const $in_time, Const $in_unit, Const $out_unit)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest(1)
    UTAssert($tArray, ConvertTime(1, "min", "sec")     = 60)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ConvertTime(1, "min", "msec")    = 60000)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ConvertTime(1, "min", "ticks")   = 3000)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ConvertTime(3000, "ticks","min") = 1)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ConvertTime(1, "ticks", "msec")  = 20)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ConvertTime(1, "hr", "min")      = 60)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ConvertTime(1, "hr", "sec")      = 3600)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_GetChnInfo($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
```

```
    PreTest($t)
    Local $aChnInfo = GetChnInfo(100, 25)
    UTAssert($tArray, $aChnInfo[0] = 100)        ;0 = peak_start
    UTAssert($tArray, $aChnInfo[1] = 125)        ;1 = peak_stop
    UTAssert($tArray, $aChnInfo[2] = 86)         ;2 = left_start (25/2 = 12.5; 99-13
       = 86)
    UTAssert($tArray, $aChnInfo[3] = 99)         ;3 = left_stop
    UTAssert($tArray, $aChnInfo[4] = 126)        ;4 = right_start
    UTAssert($tArray, $aChnInfo[5] = 139)        ;5 = right_stop  (25/2 = 12.5; 126 +
        13 = 139)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_CalcCPM($func_name)
    ;CalcCPM($roiCounts, $leftCounts, $rightCounts, $time)
    ;                roi, left, right, time(min)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcCPM(10,     0,     0, 1) == 10.0)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcCPM(10,     1,     1, 1) == 8.0)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcCPM(3,      1,     1, 1) == 1)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcCPM(2,      1,     1, 1) == 0)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcCPM(2,      1,     1, 0) == 1e99)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcCPM(0,      10,    5, 1) == -15)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_CalcMDA($func_name)
    ;CalcMDA($bkg_gross, $left_counts, $right_counts, $time)
    ;                bkg, left, right, time(min)
    PrePreTest($func_name)
    Local $tArray[1]
```

```
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcMDA(1,      1,      1, 0) == 1e99)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcMDA(0,      0,      0, 1) == 2.71)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcMDA(-1,     1,      0, 1) == 2.71)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcMDA(-1,     1,      1, 1) == 7.36)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcMDA(0,      2,      2, 1) == 12.01)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcMDA(2,      1,      1, 1) == 12.01)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_DetectorError($func_name)
    ;DetectorError(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, $msg,
        $i, Const ByRef $verbose, Const $errtype)
    NoTestWritten($func_name)
EndFunc
Func Test_CodeClash($func_name)
    ;CodeClash()
    NoTestWritten($func_name)
EndFunc
Func Test_SetupDetector($func_name)
    ;SetupDetector(Const ByRef $inifile, Const ByRef $verbose, Const ByRef $ddid,
       Const ByRef $pdid)
    NoTestWritten($func_name)
EndFunc
Func Test_TeardownDetector($func_name)
    ;TeardownDetector(Const ByRef $inifile, Const ByRef $verbose, Const ByRef $ddid,
        Const ByRef $pdid)
    NoTestWritten($func_name)
EndFunc
Func Test_AutoCalibrate($func_name)
    ;AutoCalibrate($inifile, $verbose, $ddid, $pdid, $huntdate)
    NoTestWritten($func_name)
EndFunc
Func Test_GetBackground($func_name)
```

```
    ;GetBackground($inifile, $verbose, $ddid, $pdid, $huntdate)
    NoTestWritten($func_name)
EndFunc

Func Test_CalcUncertaintyCountingStats($func_name)
        ;                   CalcUncertaintyCountingStats($csROI, $bkgROI, $csNET,
            $bkgNET)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
        ;count napoleon %0.050 err
        UTAssert($tArray, CalcUncertaintyCountingStats(73073, 938, 59388, 0) ==
            0.0050)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count wilbur %0.069
        UTAssert($tArray, CalcUncertaintyCountingStats(40096, 938, 32397, 0) ==
            0.0069)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count arnold jr
        UTAssert($tArray, CalcUncertaintyCountingStats(20097, 938, 15858, 0) ==
            0.0102)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count porky jr
        UTAssert($tArray, CalcUncertaintyCountingStats(9988, 938, 7438, 0) ==
            0.0161)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count miss piggy
        UTAssert($tArray, CalcUncertaintyCountingStats(6258, 938, 4373, 0) ==
            0.0229)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count priscilla
        UTAssert($tArray, CalcUncertaintyCountingStats(2720, 938, 1468, 0) ==
            0.0521)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count average fall 2016 deer %17.20
        UTAssert($tArray, CalcUncertaintyCountingStats(1399, 938, 381, 0) == 0.1720)
    PostTest($tArray)
        $t = $t + 1
```

```
    PreTest($t)
        ;count median fall 2016 deer $23.24
        UTAssert($tArray, CalcUncertaintyCountingStats(1273, 938, 277, 0) == 0.2324)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count mda deer %30.73
        UTAssert($tArray, CalcUncertaintyCountingStats(1189, 938, 207, 0) == 0.3073)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
        ;count background deer %6126.17 err
        UTAssert($tArray, CalcUncertaintyCountingStats(938, 938, 0, 0) == 61.2617)
    PostTest($tArray)

        $t = $t + 1
        PreTest($t)
        ;for a typical phantom count (3.56pCi/g)

    UTAssert($tArray, CalcUncertaintyCountingStats(2685,      897,    1440,      -71)
         == 0.0504)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
        ;count like median deer
        UTAssert($tArray, CalcUncertaintyCountingStats(1273, 897, 277, -71) ==
            0.1848)
    PostTest($tArray)

        $t = $t + 1


    PostPostTest()
EndFunc

Func Test_CalcUncertainty($func_name)
        ; RefEuCPM = 40000 - defined in GetRefEuCPM for test suite
        ;         CalcUncertainty($inifile, $ddid, $EuCPM, $csROI, $bkgROI, $csNET,
            $bkgNET)
    ;                          ($inifile, $ddid, $EuCPM, $CsroiCounts, $bkgCPM,
        $bkg_gross)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
        ;0% drift, 5.04% counting stats, 2.50% placement, 3.0% model bias. Total
            uncertainty = 6.38% (typical beginning phantom count)
    UTAssert($tArray, CalcUncertainty($inifile, 1, 40000, 2685, 897, 1440, -71) =
        0.0638)
    PostTest($tArray)
    $t = $t + 1
```

```
    PreTest($t)
        ;7% Eu drift=11.92% Cs drift, 5% detector, 3.96% counting stats, 2.95%
            placement, 3.04% model bias. Total uncertainty = 14.51% (typical worst
            case ending phantom count)
        UTAssert($tArray, CalcUncertainty($inifile, 1, 42800, 2685, 897, 1440, -71)
            = 0.1352)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
        ;5% Eu drift=7.45% Cs drift, 5% detector, 3.96% counting stats, 2.95%
            placement, 3.04% model bias. Total uncertainty = 10.68% (typical ending
            phantom count)
        UTAssert($tArray, CalcUncertainty($inifile, 1, 42000, 2685, 897, 1440, -71)
            = 0.0981)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
        ;5% Eu drift=7.45% Cs drift, 5% detector, 15.02% counting stats, 2.95%
            placement, 3.04% model bias. Total uncertainty = 10.68% (typical deer
            count)
        UTAssert($tArray, CalcUncertainty($inifile, 1, 42000, 1500, 897, 255, -71) =
             0.2246)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_GetSample($func_name)
    ;GetSample($inifile, $verbose, $ddid, $pdid, $huntdate, $animal_id) ;returns
        array
    NoTestWritten($func_name)
EndFunc
Func Test_Warmup($func_name)
    ;Warmup($warmup_min, $verbose, $ddid)
    NoTestWritten($func_name)
EndFunc
Func Test_GetCorrectedActivityAux($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, GetCorrectedActivityAux(100, 0, 100) == 100)
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.3f",GetCorrectedActivityAux(100, 1, 0.69314))
        *1.0 == 36.788)
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
```

```
    UTAssert($tArray, StringFormat("%.3f",GetCorrectedActivityAux(100, 60.14, 30.07)
        )*1.0 == 25) ;2 half lives
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_GetCorrectedActivity($func_name)
    ;GetCorrectedActivity($inifile, $ActivityDPS, $VolumeL, $DensityG_ML, $YearsPast
        , $HalfLife)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.2f",GetCorrectedActivity($inifile, 3893, 20.0,
        1.15, 0, 30.07))*1     = 4.57) ;no past
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.2f",GetCorrectedActivity($inifile, 3893, 20.0,
        1.15, 30.07, 30.07))*1 = 2.29) ;1 half-life past
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.2f",GetCorrectedActivity($inifile, 3893, 20.0,
        1.15, 10.38, 30.07))*1 = 3.60) ;11/30/2016 (10.38 years) - see
       get_years_between
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_validate_cinch_num($func_name)
    ;validate_cinch_num($cinch_num)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_cinch_num("99999") = True)  ;ok
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_cinch_num("00001") = True)  ;ok
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_cinch_num("00000") = False) ;too small
    PostTest($tArray)
    $t = $t + 1
```

```
    PreTest($t)
    UTAssert($tArray, validate_cinch_num("o9999") = False) ;not a number
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_cinch_num("")      = False) ;not a number
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_cinch_num("1.1")   = False) ;not an integer
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_animal_type($func_name)
    ;validate_animal_type($animal_type)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_animal_type("") = True)     ;always return true b/c
       animal_type is automatically generated from ini file
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_type("DEER") = True) ;always return true b/c
       animal_type is automatically generated from ini file
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_animal_wt($func_name)
    ;validate_animal_wt($animal_wt)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("")      = False) ;not a number
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("a")     = False) ;not a number
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("-1")    = False) ;really too small
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("0")     = False) ;too small
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("9999")  = False) ;too big
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("19999") = False) ;really too big
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_animal_wt("123")   = True)  ;good weight
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_hunt_cmpt($func_name)
    ;validate_hunt_cmpt($hunt_cmpt)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("")   = False)  ; not a number (function
       multiplies this string by 1, so making it a number 0, fails b/c 0 is not a
       hunt cmpt
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("A")  = False)  ; not a number (function
       multiplies this string by 1, so making it a number 0, fails b/c 0 is not a
       hunt cmpt
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("1.1") = False) ; not an integer
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("-1") = False)  ; really too small
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("0")  = False)  ; too small
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("1")  = True)   ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("50") = True)   ; good
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("51") = False)  ; too big
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_cmpt("52") = False)  ; really too big
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_hunt_stand($func_name)
    ;validate_hunt_stand($hunt_stand)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("")    = False) ; not a number (function
       multiplies this string by 1, so making it a number 0, fails b/c 0 is not a
       hunt cmpt
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("A")   = False) ; not a number (function
       multiplies this string by 1, so making it a number 0, fails b/c 0 is not a
       hunt cmpt
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("1.1") = False) ; not an integer
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("-1")  = False) ; really too small
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("0")   = False) ; too small
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("1")   = True)  ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("100") = True)  ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("101") = False) ; too big
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunt_stand("102") = False) ; really too big
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_last($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_last("AAA") = True)  ; more than 2 letters
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_last("AA_") = True)  ; 2 letters and underscore
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_last("AA")  = False) ; 2 letters
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_last_hunter_id($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_last_hunter_id("AAA", "000000AAA") = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_last_hunter_id("AA_", "000000AA_") = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_last_hunter_id("AAA", "000000AAB") = False)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_validate_hunter_id($func_name)
    ;validate_hunter_id($hunter_id)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("111111AAA")  = True)    ;good
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("1111111AAA") = False)  ;too long of a
        string
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("11111AAA")   = False)  ;too short of a
        string
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("1111111AA")  = False)  ;too many nums  (
        sss has nonalpha)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("11111AAAX")  = False)  ;too many letts (
        sss not match and nnnnnn is too short)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("111111HO")   = False)  ;last name too
        short
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, validate_hunter_id("111111HO_")  = True)   ;last name fixed
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_is_hunter_in_db($func_name)
    ;is_hunter_in_db($hunter_id, $inifile)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, is_hunter_in_db("000000HUN", $inifile) = True)   ;hunter in
        database
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, is_hunter_in_db("000000QQQ", $inifile) = False)  ;hunter not
        in database
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_convert_mm($func_name)
```

```
    ;convert_mm($mon)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, convert_mm("102")     = "00") ; bad input
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, convert_mm("january") = "01") ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, convert_mm("jan")     = "01") ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, convert_mm("JAN")     = "01") ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, convert_mm("janx")    = "00") ; bad input
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_format_date($func_name)
    ;format_date($date)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, format_date("20161130")      = "2016/11/30") ; good format
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, format_date("July␣14,␣2006") = "2006/07/14") ; good format
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, format_date("July␣4,␣2006")  = "2006/07/04") ; good format
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, format_date("7/14/2006")     = "2006/07/14") ; good format
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_get_years_between($func_name)
    ;get_years_between($date1, $date2)
```

```
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, get_years_between("20161130", "20060714")      = 10.38) ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, get_years_between("20161130", "July␣14,␣2006") = 10.38) ; good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, get_years_between("July␣14,␣2006", "20161130") = 10.38) ; good
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_GetRefEuCPM($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    UTAssert($tArray, GetRefEuCPM($inifile, 1) = 40000) ; good
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_GetDetectorUncertainty($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    UTAssert($tArray, GetDetectorUncertainty($inifile, 1) = 0.05) ; good
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_EuCPMok($func_name)
    ;EuCPMok($EuCPM, $ddid, $inifile)
    ;RefEuCPM = 40000
    ;EuPrecision = 7.00%
    ;Eu Range = 37200 - 42800
    ;                  $EuCPM, $detid
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(     0, 1, $inifile) = False) ; EuCPM out
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(37199, 1, $inifile) = False) ; EuCPM out (just outside
        border)
```

```
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(37200, 1, $inifile) = False)  ; EuCPM OK (on border)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(37201, 1, $inifile) = True)  ; EuCPM OK (just inside)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(40000, 1, $inifile) = True)  ; EuCPM OK (40000 defined
        in method.au3 when GetRefEuCPM called)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(42799, 1, $inifile) = True)  ; EuCPM OK (just inside
       border)
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(42800, 1, $inifile) = False)  ; EuCPM OK (on border)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(42801, 1, $inifile) = False) ; EuCPM out (just outside
        border)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, EuCPMok(50000, 1, $inifile) = False) ; EuCPM high
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_CalcIngestionDose($func_name)
    ;these values are dependant on Ingestion Dose, which is derived from Effective
       Dose Eqivalent
    ;                                   conc, live_wt_lbs, whole_animal_type
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 0, 0.6304, "DEER")   == 0)       ;
       at the limits
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 100, 0.6304, "DEER") == 2.08)  ;at
        the limits
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 100, 100, "DEER")    == 103.192)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 100, 200, "DEER")    == 204.944)
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 100, 100, "HOG")     == 130.101)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 100, 30, "TURKEY")   == 34.981)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcIngestionDose($inifile, 100, 50, "COYOTE")   == 53.638)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_GetNonDOEBackground($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.2f",GetNonDOEBackground("DEER", "2013/1/1"))
        *1.0 == 2.59)  ;measured date (Gaines report 137Cs Bodyburdens in
        Whitetailed deer...)
    PostTest($tArray)
    ;$t = $t + 1
    ;PreTest($t)
    ;UTAssert($tArray, StringFormat("%.2f",GetNonDOEBackground("DEER", "2036/3/17"))
        *1.0 == 1.30) ;almost 1 half-life (1.29709)
    ;UTAssert($tArray, StringFormat("%.2f",GetNonDOEBackground("DEER", "2006/1/1"))
        *1.0 == 0.76) ;measured date (recommended value)
    ;UTAssert($tArray, StringFormat("%.2f",GetNonDOEBackground("DEER", "2036/1/1"))
        *1.0 == 0.38) ;1 half-life
    ;PostTest($tArray)


    PostPostTest()
EndFunc

Func Test_CalcSRSDose($func_name)
    ;CalcSRSDose($CsConcMeat, $CsCPM_uncert, $animal_live_wt, $animal_type, $inifile
        , $huntday) ;mrem
    ;                  pCi/g, uncert_pCi/g, animal_live_wt, animal_type, $inifile,
```

```
        $huntday
    ; 2013/01/01 is midpoint date in Karen Gaines report
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(0.0,  42.3794,  100, "DEER", $inifile, "20130101")
        == 0)      ;below background
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(2.59, 0.1113,  100, "DEER", $inifile, "20130101")
       == 0)       ;at background
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(3.00, 0.1094,  100, "DEER", $inifile, "20130101")
       == 0.423)    ;deer
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(3.63,  0.1068,  100, "DEER", $inifile, "20130101")
        == 1.073)   ;phantom
    PostTest($tArray)

        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(0.00, 42.3794,  100, "DEER", $inifile, "20170101")
        == 0)      ;below background
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(2.59, 0.1113,  100, "DEER", $inifile, "20170101")
       == 0.235)   ;at background
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(3.00, 0.1094,  100, "DEER", $inifile, "20170101")
       == 0.658)       ;deer
    PostTest($tArray)
        $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcSRSDose(3.63,  0.1068,  100, "DEER", $inifile, "20170101")
        == 1.309)      ;phantom
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_GetHuntType($func_name)
    PrePreTest($func_name)
```

```
    Local $tArray[1]
    PreTest(1)
    UTAssert($tArray, GetHuntType("20161017", $inifile) = "forestry")
    PostTest($tArray)
    PreTest(2)
    UTAssert($tArray, GetHuntType("20161112", $inifile) = "regular")
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_CalcAnimalReleaseStatusAux($func_name)
    ;CalcAnimalReleaseStatusAux($hunt_type, $csConcMeat, $csSRSdose, $cinch_num,
        $inifile, $animal_type, $huntday)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("forestry" ,          0,
        0, 123456, $inifile,"DEER",20170101    ) == "release")    ;release if conc
        below bkg
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("forestry" ,          3,
        0, 123456, $inifile,"DEER",20170101    ) == "forestry")   ;release to
        forestry if bkg < conc and dose < limit
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("forestry" ,          5,
        23, 123456, $inifile,"DEER",20170101    ) == "confiscate") ;confiscate if
        limit < dose (and bkg < conc
    PostTest($tArray)

    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("regular"  ,          0,
        0, 123456, $inifile,"DEER",20170101    )=="release")      ;regular hunts
        don't care about CsConc
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("regular"  ,          0,
        21, 123456, $inifile,"DEER",20170101    )=="release")      ;below limit
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("regular"  ,          0,
        22, 123456, $inifile,"DEER",20170101    )=="release")      ;release if
        CsSRSDose <= annual limit (22)
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcAnimalReleaseStatusAux("regular"   ,           0,
        23, 123456, $inifile,"DEER",20170101     )=="confiscate")     ;above limit
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_CalcAnimalReleaseStatus($func_name)
    ;CalcAnimalReleaseStatus($csConcMeat, $csSRSdose, $cinch_num, $huntday,
        $animal_type, $inifile)
    NoTestWritten($func_name)
EndFunc

Func Test_CalcHunterRecieveStatus($func_name)
    ;CalcHunterRecieveStatus($ReleaseStatus, $AnimalType, $csSRSdose, $ConsumerId,
        $huntday, $inifile)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcHunterRecieveStatus("confiscate", "DEER", 23, "000000HUN",
         "20161112", $inifile) = True) ;confiscating
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcHunterRecieveStatus("release", "PHANTOM", 23, "000000HUN",
         "20161112", $inifile) = True) ;it's a phantom
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcHunterRecieveStatus("release", "DEER", 30, "000000HUN", "
        20161112", $inifile) = True)     ;above limit but its PhantomHunter so doesn'
        t matter
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcHunterRecieveStatus("release", "DEER", 17, "000000HUN", "
        20161112", $inifile) = True)     ;see nothing
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_GetConsumerId($func_name)
    ;GetConsumerId($huntday, $inifile, $hunter_id, $release_status, $animal_type,
        $csSRSdose, $cinch_num)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
```

```
    UTAssert($tArray, GetConsumerId("20161112", $inifile, "000000HUN", "release", "
        DEER", 0, 123456) = "000000HUN")
    PostTest($tArray)
    PreTest(2)
    UTAssert($tArray, GetConsumerId("20161112", $inifile, "123456QQQ", "release", "
        DEER", 23, 123456) = "000000HUN")
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_GetTotalHuntQuery($func_name)
    ;GetTotalHuntQuery($adoCon, $all_or_specific_id)
    ;database query should probably be moved
    NoTestWritten($func_name)
EndFunc

Func Test_getLIMSdb($func_name)
    ;getLIMSdb($inifile, $huntday, $animal_id)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $aLIMS[2]

    PreTest(1)
    $aLIMS = getLIMSdb($inifile, "20161109", 33)
    UTAssert($tArray, $aLIMS[0] = "156")
    UTAssert($tArray, $aLIMS[1] = "126")
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_getLIMS($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    Local $aLIMS = getLIMS()
    UTAssert($tArray, $aLIMS[0] == 1)
    UTAssert($tArray, $aLIMS[1] == 2)
    PostTest($tArray)
    PostPostTest()
EndFunc

 Func Test_needLIMSaux($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, needLIMSaux(-1) = False) ;should never happen, but I like to
        test
    PostTest($tArray)
```

```
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(0) = False)  ;1-20 animals, need lims every 5; 21-
   end every 10
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(1) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(2) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(3) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(4) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(5) = True)  ;lims here
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(6) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(9) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(10) = True) ;lims here
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(11) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(15) = True)
PostTest($tArray)
$t = $t + 1
PreTest($t)
UTAssert($tArray, needLIMSaux(19) = False)
PostTest($tArray)
$t = $t + 1
PreTest($t)
```

```
    UTAssert($tArray, needLIMSaux(20) = True) ;lims here
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, needLIMSaux(21) = False)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, needLIMSaux(25) = False)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, needLIMSaux(30) = True) ;lims here
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_needLIMS($func_name)
    ;wrapper function could/should be moved to lib_dbcalls
    ;needLIMS($inifile, $huntday)
    NoTestWritten($func_name)
EndFunc

Func Test_LIMSanimal($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "TURKEY", 40)        = False) ;dont lims
        turkeys
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "COYOTE", 100)       = False) ;dont lims
        coyotes
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "DRESSED_DEER", 100) = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "DEER", 100)         = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "DEER", 60)          = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
```

```
    UTAssert($tArray, LIMSanimal($inifile, "DEER", 59)          = False) ;too little
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "HOG", 100)          = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "HOG", 60)           = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "HOG", 59)           = False) ;too little
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, LIMSanimal($inifile, "DRESSED_HOG", 100)  = True)
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_doLIMS($func_name)
    ;doLIMS($inifile, $huntday, $csConcMeat, $animal_type, $animal_wt_live) ;return
        $lims_meat and $lims_bone
    Local $aLIMS[2]

    PrePreTest($func_name)
    Local $tArray[1]

    Local $t = 1
    PreTest($t)
    $aLIMS = doLIMS($inifile, "test_huntday", 4.00, "DEER", 100)    ;above 60 lbs
        and LIMSable
    UTAssert($tArray, $aLIMS[0] = "1")
    UTAssert($tArray, $aLIMS[1] = "2")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    $aLIMS = doLIMS($inifile, "test_huntday", 4.00, "DEER", 100)    ;same
    UTAssert($tArray, $aLIMS[0] = "1")
    UTAssert($tArray, $aLIMS[1] = "2")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    $aLIMS = doLIMS($inifile, "test_huntday", 4.00, "DEER", 40)    ;not LIMSable (
        low weight)
    UTAssert($tArray, $aLIMS[0] = "")
    UTAssert($tArray, $aLIMS[1] = "")
    PostTest($tArray)
    $t = $t + 1
```

```
    PreTest($t)
    $aLIMS = doLIMS($inifile, "test_huntday", 4.00, "TURKEY", 40)     ;not LIMSable
        animal
    UTAssert($tArray, $aLIMS[0] = "")
    UTAssert($tArray, $aLIMS[1] = "")
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_GetAnimalIds($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    UTAssert($tArray, GetAnimalIds($inifile, "20161104") = "3|4")
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_InvalidateAnimal($func_name)
    ;InvalidateAnimal($inifile, $huntday, $animal_id)
    NoTestWritten($func_name)
EndFunc
Func Test_ShowSpectra($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, ShowSpectra($inifile, "20161102", 1)  <> 0) ;good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ShowSpectra($inifile, "20161102", 15) <> 0) ;good
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ShowSpectra($inifile, "", "")          = 0) ;doesn't exist
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_GetNotIncludeAnimals($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    Local $a_tmp[2]
    $a_tmp = GetNotIncludeAnimals($inifile)
    UTAssert($tArray,  $a_tmp[0] = 1)
    UTAssert($tArray,  $a_tmp[1] = "phantom")
    PostTest($tArray)
```

```
    PostPostTest()
EndFunc

Func Test_GetAnimalTypes($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    Local $aNotInclude[2] = [1, 'phantom']
    UTAssert($tArray, GetAnimalTypes($inifile, $aNotInclude,"")           = "COYOTE|
        DEER|HOG|TURKEY")
    UTAssert($tArray, GetAnimalTypes($inifile, $aNotInclude,"dressed")    = "COYOTE|
        DEER|DRESSED_DEER|HOG|DRESSED_HOG|TURKEY")
    PostTest($tArray)

    PreTest(2)
    Local $aNotInclude[4] = [3, 'COYOTE', 'PHANTOM', 'TURKEY']
    UTAssert($tArray, GetAnimalTypes($inifile, $aNotInclude,"dressed") = "DEER|
        DRESSED_DEER|HOG|DRESSED_HOG")
    UTAssert($tArray, GetAnimalTypes($inifile, $aNotInclude,"")        = "DEER|HOG")
    PostTest($tArray)

    PostPostTest()
EndFunc

Func Test_AddHunter($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata   = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    Local $t = 1
    PreTest($t)
    Local $a_data[6]
    $a_data[0] = "" ;hunterid  ;nothing input so fail
    $a_data[1] = "" ;first
    $a_data[2] = "" ;middle
    $a_data[3] = "" ;last
    $a_data[4] = "" ;suffix
    $a_data[5] = "" ;comment

    UTAssert($tArray, AddHunter($initemp, $a_data) = False)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    $a_data[0] = "654321ZZZ"
    $a_data[1] = "JACK"
    $a_data[2] = "M"
    $a_data[3] = "ZZZOOTOPIA"
```

```
    $a_data[4] = "Jr"
    $a_data[5] = "no comment"

    UTAssert($tArray, AddHunter($initemp, $a_data) = True)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    $a_data[0] = "987654321" ;all numbers - fail
    $a_data[1] = ""
    $a_data[2] = ""
    $a_data[3] = "ZZZOOTOPIA"
    $a_data[4] = ""
    $a_data[5] = ""

    UTAssert($tArray, AddHunter($initemp, $a_data) = False)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    $a_data[0] = "777321ZZZ" ;hunterid doesn't match last name
    $a_data[1] = ""
    $a_data[2] = ""
    $a_data[3] = "AAAOOTOPIA"
    $a_data[4] = ""
    $a_data[5] = ""

    UTAssert($tArray, AddHunter($initemp, $a_data) = False)
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_GetLiveFromDressed($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    ;                                                   animal_type,  dressed_wt
    PreTest(1)
    UTAssert($tArray, StringFormat("%.3f",GetLiveFromDressed("DRESSED_DEER", 100))
        *1.0 == 127.706)
    PostTest($tArray)

    PreTest(2)
    UTAssert($tArray, StringFormat("%.3f",GetLiveFromDressed("DEER", 100))*1.0 ==
        127.706)
    PostTest($tArray)

    PostPostTest()
EndFunc
```

```
Func Test_Reprocess($func_name)
    ;Reprocess($inifile, $huntday, $a_data)
    NoTestWritten($func_name)
EndFunc

Func Test_ProcessAnimal($func_name)
    ;ProcessAnimal($inifile, $a_process)
    NoTestWritten($func_name)
EndFunc

Func Test_UpdateAnimalInfo($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata   = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    Local $huntday   = 20161119
    Local $animal_id = 2   ;HOG

    Local $old_atype = "DEER"

    Local $a_data[10]
    $a_data[0] = $animal_id
    $a_data[1] = $old_atype
    $a_data[2] = 123456
    $a_data[3] = "000000HUN"
    $a_data[4] = 1
    $a_data[5] = 1
    $a_data[6] = 100
    $a_data[7] = "live"
    $a_data[8] = "77"
    $a_data[9] = "88"

    PreTest(1)
    UpdateAnimalInfo($initemp, $huntday, $animal_id, $a_data)
    Local $what       = "animaltype"
    Local $SQL        = "SELECT␣animaltype␣AS␣" & $what & "␣FROM␣" & $huntday & "␣
       WHERE␣ID␣=␣" & $animal_id & "␣;"
    Local $animaltype = SelectDataQuery($initemp, $SQL, $what)
    UTAssert($tArray, $old_atype == $animaltype)
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
```

```
EndFunc

Func Test_UpdateAnimalActivity($func_name)
    ;UpdateAnimalActivity($huntday, $animal_id, $EuCPMref, $EuCPM, $bkgGross,
        $bkgCPM, $CsCPM, $CsCPM_uncert, $CsCPM_type, $CsConcAnimal, $CsConcMeat)
    NoTestWritten($func_name)
EndFunc
Func Test_UpdateAnimalDose($func_name)
    ;UpdateAnimalDose($huntday, $animal_id, $csSRSdose, $release_status)
    NoTestWritten($func_name)
EndFunc
Func Test_UpdateAnimalConsumer($func_name)
    ;UpdateAnimalConsumer($huntday, $animal_id, $ConsumerId, $meatLIMS, $boneLIMS)
    NoTestWritten($func_name)
EndFunc


;Getting Funcs from lib_maestro.au3
Func Test_DebugDll($func_name)
    ;DebugDll(Const ByRef $result)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOCleanup($func_name)
    ;MIOCleanup(Const ByRef $isReal, Const ByRef $dll_h)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOCloseDetector($func_name)
    ;MIOCloseDetector(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOComm($func_name)
    ;MIOComm(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, Const ByRef
        $lpszCmd)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetAppData($func_name)
    ;MIOGetAppData(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, Const
        $lpszDataName)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetData($func_name)
    ;MIOGetData(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, Const
        ByRef $wStartChan, Const ByRef $wNumChans)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetDetLength($func_name)
    ;MIOGetDetLength(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetDetectorInfo($func_name)
    ;MIOGetDetectorInfo(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet)
```

```
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetLastError($func_name)
    ;MIOGetLastError(Const ByRef $isReal, Const ByRef $dll_h)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOIsActive($func_name)
    ;MIOIsActive(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOIsDetector($func_name)
    ;MIOIsDetector(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOOpenDetector($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $isReal = "False"
    Local $dll_h  = ""
    Local $detid  = 1
    PreTest(1)
    UTAssert($tArray, MIOOpenDetector($isReal, $dll_h, $detid, "") = $detid)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_MIOOpenDetID($func_name)
    ;MIOOpenDetID(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $dwDetID)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOSetAppData($func_name)
    ;MIOSetAppData(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $hDet, Const
        $lpszDataName, Const $lpszDataString)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOStartup($func_name)
    ;MIOStartup(Const ByRef $isReal, ByRef $dll_h)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOCreateConfigList($func_name)
    ;MIOCreateConfigList(Const ByRef $isReal, Const ByRef $dll_h,  Const
       $lpszListName)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOAddConfigItem($func_name)
    ;MIOAddConfigItem(Const ByRef $isReal, Const ByRef $dll_h, $nPickListIndex,
       Const $lpszListName, Const $nMasterDet )
    NoTestWritten($func_name)
EndFunc
Func Test_MIOCompressConfigList($func_name)
```

```
    ;MIOCompressConfigList(Const ByRef $isReal, Const ByRef $dll_h, Const
        $lpszListName)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetConfigItem($func_name)
    ;MIOGetConfigItem(Const ByRef $isReal, Const ByRef $dll_h, Const $lpszListName)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOGetConfigName($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $isReal       = "False"
    Local $dll_h        = ""
    Local $lpszListName = ""

    PreTest(1)
    Local $aConfigInfo = MIOGetConfigName($isReal, $dll_h, 1, $lpszListName)
    ;_ArrayDisplay($aConfigInfo)
    UTAssert($tArray, $aConfigInfo[0] = 2)
    PostTest($tArray)

    PreTest(2)
    Local $aConfigInfo = MIOGetConfigName($isReal, $dll_h, 2, $lpszListName)
    ;_ArrayDisplay($aConfigInfo)
    UTAssert($tArray, $aConfigInfo[0] = 3)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_MIOGetConfigMax($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $isReal = "False"
    Local $dll_h  = ""

    PreTest(1)
    UTAssert($tArray, MIOGetConfigMax($isReal, $dll_h, "") = 2)
    PostTest($tArray)
    PostPostTest()
EndFunc
Func Test_MIODebug($func_name)
    ;MIODebug(Const ByRef $level, Const ByRef $dll_h)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOCreateConfigItem($func_name)
    ;MIOCreateConfigItem(Const ByRef $isReal, Const ByRef $dll_h)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOResetMasterList($func_name)
```

```
    ;MIOResetMasterList(Const ByRef $isReal, Const ByRef $dll_h, Const $hdet)
    NoTestWritten($func_name)
EndFunc
Func Test_MIOWhoAreYou($func_name)
    ;MIOWhoAreYou(Const ByRef $isReal, Const ByRef $dll_h, Const ByRef $test)
    NoTestWritten($func_name)
EndFunc

;Getting Funcs from lib_ui.au3
Func Test_WriteMsgStart($func_name)
    ;WriteMsgStart(Const ByRef $title, Const ByRef $msg, Const ByRef $verbose)
    NoTestWritten($func_name)
EndFunc
Func Test_WriteMsg($func_name)
    ;WriteMsg(Const ByRef $i, Const ByRef $ti, $msg, Const ByRef $verbose, $main = "
        ")
    NoTestWritten($func_name)
EndFunc
Func Test_WriteMsgStop($func_name)
    ;WriteMsgStop(Const ByRef $verbose)
    NoTestWritten($func_name)
EndFunc


;Getting Funcs from lib_dbcalls.au3
Func Test_SelectDataQuery($func_name)
    ;SelectDataQuery($inifile, $SQL, $what)
    NoTestWritten($func_name)
EndFunc
Func Test_GetListFromDB($func_name)
    ;GetListFromDB($inifile, $SQL, $thing)
    NoTestWritten($func_name)
EndFunc
Func Test_GetAnimalIds_DB_Call($func_name)
    ;GetAnimalIds_DB_Call($inifile, $huntday)
    NoTestWritten($func_name)
EndFunc
Func Test_GetPastHuntDays_DB_Call($func_name)
    ;GetPastHuntDays_DB_Call($inifile)
    NoTestWritten($func_name)
EndFunc
Func Test_ValidHuntDay_DB_Call($func_name)
    ;ValidHuntDay_DB_Call($inifile, $proposedhuntday)
    NoTestWritten($func_name)
EndFunc


Func Test_DropTable($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
```

```
    Local $adata = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    PreTest(1)

    Local $table = "hunts"
    DropTable($initemp, $table)

    UTAssert($tArray, DoesTableExist($initemp, $table) = False)

    PostTest($tArray)
    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_RenameTable($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    Local $adoCon = OpenDatabase($initemp)

    PreTest(1)

    RenameTableRaw($adoCon, "hunter","frank")

    UTAssert($tArray, DoesTableExistRaw($adoCon,"frank")  = True)
    UTAssert($tArray, DoesTableExistRaw($adoCon,"hunter") = False)

    PostTest($tArray)

    $adoCon.Close
    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    ConsoleWrite(@TAB & "...Done" & @CRLF)
EndFunc

Func Test_DoesTableExist($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
```

```
    PreTest(1)
    UTAssert($tArray, DoesTableExist($inifile, "hunter") = True)
    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_CreateTableFromTemplate($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    Local $to_table   = "hdts"
    Local $from_table = "hdts_template"

    DropTable($initemp, $to_table)

    PreTest(1)
    CreateTableFromTemplate($initemp, $to_table, $from_table)

    UTAssert($tArray, DoesTableExist($initemp, $to_table) = True)
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_CreateHunterTable_DB_Call_3($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;make a past hunt
    Local $pasthunt = "20161230"
    Local $pasthunttype = "regular"
    CreateTableFromTemplate($initemp, $pasthunt,"huntdate_template")

    Local $adoCon = OpenDatabase($initemp)
    Local $qry = "INSERT␣INTO␣hunts␣(huntday,␣type,␣ended)␣VALUES␣('" & $pasthunt &
        "','" & $pasthunttype & "',␣yes)␣;"
    $adoCon.Execute($qry)
    $adoCon.Close
```

```
    ;--------- TEST IN HERE ---------
    Local $hdts_table    = "hdts"

    PreTest(1)
    Local $huntday       = "20161231"
    DropTable($initemp, $hdts_table)
    CreateTableFromTemplate($initemp, $hdts_table, "hdts_template")
    CreateHunterTable_DB_Call_3($initemp, $huntday, $hdts_table)
    UTAssert($tArray, DoesTableExist($initemp, $hdts_table) = True)
    PostTest($tArray)

    PreTest(2)
    Local $huntday       = "20171231"
    DropTable($initemp, $hdts_table)
    CreateTableFromTemplate($initemp, $hdts_table, "hdts_template")
    CreateHunterTable_DB_Call_3($initemp, $huntday, $hdts_table)
    UTAssert($tArray, DoesTableExist($initemp, $hdts_table) = True)
    PostTest($tArray)

    PreTest(3)
    Local $huntday       = "20181231"
    DropTable($initemp, $hdts_table)
    CreateTableFromTemplate($initemp, $hdts_table, "hdts_template")
    CreateHunterTable_DB_Call_3($initemp, $huntday, $hdts_table)
    UTAssert($tArray, DoesTableExist($initemp, $hdts_table) = True)
    PostTest($tArray)

    PreTest(4)
    Local $yyp1          = @YEAR + 1
    Local $huntday       = $yyp1 & "1231"
    DropTable($initemp, $hdts_table)
    CreateTableFromTemplate($initemp, $hdts_table, "hdts_template")
    CreateHunterTable_DB_Call_3($initemp, $huntday, $hdts_table)
    UTAssert($tArray, DoesTableExist($initemp, $hdts_table) = True)
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_ADOErrFunc1($func_name)
    ;ADOErrFunc1() ;this just writes a line to the console or msgbox in real mode
    NoTestWritten($func_name)
EndFunc

Func Test_ADOErrFunc2($func_name)
    ;ADOErrFunc2() ;this just writes a line to the console in test mode or msgbox in
```

```
        real mode
    NoTestWritten($func_name)
EndFunc

Func Test_CreateHunterTable_DB_Call_6($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;make a past hunt
    Local $pasthunt = "20161230"
    Local $pasthunttype = "regular"
    CreateTableFromTemplate($initemp, $pasthunt,"huntdate_template")

    Local $adoCon = OpenDatabase($initemp)
    Local $qry = "INSERT␣INTO␣hunts␣(huntday,␣type,␣ended)␣VALUES␣('" & $pasthunt &
       "','" & $pasthunttype & "',␣yes)␣;"
    $adoCon.Execute($qry)
    $adoCon.Close

    ;--------- TEST IN HERE ---------
    Local $ytd_temp = "hdts_ytd_temp"
    Local $huntday  = "11111111"

    PreTest(1)
    $huntday      = "30161231" ;definately not in the database
    DropTable($initemp, $ytd_temp)
    CreateTableFromTemplate($initemp, $ytd_temp, "hdts_template")

    CreateHunterTable_DB_Call_6($initemp, $huntday, $ytd_temp)
    UTAssert($tArray, DoesTableExist($initemp, $ytd_temp) = True) ;a better test for
        this case is exists and has 0 records
    PostTest($tArray)

    PreTest(2)
    $huntday      = "20161231"
    DropTable($initemp, $ytd_temp)
    CreateTableFromTemplate($initemp, $ytd_temp, "hdts_template")

    CreateHunterTable_DB_Call_6($initemp, $huntday, $ytd_temp)
    UTAssert($tArray, DoesTableExist($initemp, $ytd_temp) = True) ;a better test for
        this case is exists and has 0 records
    PostTest($tArray)

    PreTest(3)
    $huntday      = "20181231"
    DropTable($initemp, $ytd_temp)
    CreateTableFromTemplate($initemp, $ytd_temp, "hdts_template")
```

```
    CreateHunterTable_DB_Call_6($initemp, $huntday, $ytd_temp)
    UTAssert($tArray, DoesTableExist($initemp, $ytd_temp) = True) ;a better test for
        this case is exists and has 0 records
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    ConsoleWrite(@TAB & "...Done" & @CRLF)
EndFunc

Func Test_CreateHunterTable_DB_Call_7($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $adata   = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------
    ;why was this test bothered to be written

    PreTest(1)
    _ArrayAdd($tArray, True)
    ConsoleWrite("no test written")
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_CreateHuntDayTable_DB_Call($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata   = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------

    Local $yyp1    = @YEAR + 1
    Local $huntday = $yyp1 & "1231"
    Local $hunttype= "regular"
    CreateHuntDayTable_DB_Call($initemp, $huntday, $hunttype)

    PreTest(1)
```

```
    UTAssert($tArray, DoesTableExist($initemp, $huntday) = True)
    PostTest($tArray)

    ;should also check that huntday in hunts table
    PreTest(2)
    Local $what        = "xhuntday"
    Local $SQL         = "SELECT␣huntday␣AS␣" & $what & "␣FROM␣hunts␣WHERE␣huntday␣=
       ␣'" & $huntday & "'␣;"
    Local $huntdayinDB = SelectDataQuery($initemp, $SQL, $what)
    UTAssert($tArray, $huntday = $huntdayinDB)
    PostTest($tArray)

    ;--------- TEST IN HERE ---------

    TeardownTestDB($adata)

    PostPostTest()
EndFunc

Func Test_GetHuntStats_DB_Call($func_name)
    ;GetHuntStats_DB_Call($inifile, $huntday)
    NoTestWritten($func_name)
EndFunc
Func Test_GetHuntYearQuery($func_name)
    ;GetHuntYearQuery($huntday, $adoCon, $all_or_specific_id)
    NoTestWritten($func_name)
EndFunc
Func Test_EndHunt_DB_Call($func_name)
    ;EndHunt_DB_Call($inifile, $huntday)
    NoTestWritten($func_name)
EndFunc
Func Test_GetZeroNotNull($func_name)
    ;GetZeroNotNull($value)
    NoTestWritten($func_name)
EndFunc
Func Test_GetHuntDose_DB_Call($func_name)
    ;GetHuntDose_DB_Call($ConsumerId, $huntday, $inifile)
    NoTestWritten($func_name)
EndFunc
Func Test_GetPastDose_DB_Call($func_name)
    ;GetPastDose_DB_Call($ConsumerId, $huntday, $inifile)
    NoTestWritten($func_name)
EndFunc


;Getting Funcs from taddbfuncs.au3
Func Test_OpenDatabase($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    PreTest(1)
    Local $adoCon = OpenDatabase($inifile)
```

```
        UTAssert($tArray, $adoCon <> "")

    $adoCon.Close

    PostTest($tArray)
    PostPostTest()
EndFunc

Func Test_DoesTableExistRaw($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $adoCon = OpenDatabase($inifile)

    PreTest(1)
    UTAssert($tArray, DoesTableExistRaw($adoCon, "hunter") = True)
    PostTest($tArray)
    PreTest(2)
    UTAssert($tArray, DoesTableExistRaw($adoCon, "qqq")    = False)
    PostTest($tArray)

    $adoCon.Close

    PostPostTest()
 EndFunc

 Func Test_RenameTableRaw($func_name)
    PrePreTest($func_name)
    Local $tArray[1]

    Local $adata   = SetupTestDB()
    Local $dbtemp  = $adata[0]
    Local $initemp = $adata[1]

    ;--------- TEST IN HERE ---------

    Local $adoCon = OpenDatabase($initemp)

    Local $t = 1
    PreTest($t)

    RenameTableRaw($adoCon, "hunter","frank")

    UTAssert($tArray, DoesTableExistRaw($adoCon,"frank")  = True)
    UTAssert($tArray, DoesTableExistRaw($adoCon,"hunter") = False)

    PostTest($tArray)

    $adoCon.Close
    ;--------- TEST IN HERE ---------
```

```
    TeardownTestDB($adata)

    PostPostTest()
EndFunc

;Getting Funcs from tadinifuncs.au3
Func Test_IniErrorHandler($func_name)
    ;IniErrorHandler(Const ByRef $msg)
    NoTestWritten($func_name)
EndFunc
Func Test_FatalError($func_name)
    ;FatalError(Const ByRef $msg)
    NoTestWritten($func_name)
EndFunc
Func Test_AlertMsg($func_name)
    ;AlertMsg(Const ByRef $msg)
    NoTestWritten($func_name)
EndFunc
Func Test_TadIniRead($func_name)
    ;TadIniRead(Const $inifile, Const $section, Const $key)
    NoTestWritten($func_name)
EndFunc
Func Test_TadIniWrite($func_name)
    ;TadIniWrite(Const $inifile, Const $section, Const $key, Const $value)
    NoTestWritten($func_name)
EndFunc
Func Test_TadIniWriteSection($func_name)
    ;TadIniWriteSection(Const $inifile, Const $section, Const $data)
    NoTestWritten($func_name)
EndFunc
Func Test_TadIniReadSectionNames($func_name)
    ;TadIniReadSectionNames(Const $inifile)
    NoTestWritten($func_name)
EndFunc
Func Test_TadIniReadSection($func_name)
    ;TadIniReadSection(Const $inifile, Const $section)
    NoTestWritten($func_name)
EndFunc
Func Test_TadIniDelete($func_name)
    ;TadIniDelete(Const $inifile, Const $section)
    NoTestWritten($func_name)
EndFunc


;Getting Funcs from tadmathfuncs.au3
Func Test_CalcValPoly($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("1,1,1", 1) = 3)
```

```
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("0,0,0", 0) = 0)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("2,0,0", 1) = 2)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("2,0,0", 2) = 8)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("0,0,2", 2) = 2)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("3", 2)     = 3)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("0,0,3", 2) = 3)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcValPoly("0,3,3", 2) = 9)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.3f",CalcValPoly("26.024,0.0968,0", .047)) == "
        0.062")
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, StringFormat("%.3f",CalcValPoly("26.024,0.0968,0", 0.000)) ==
        "0.000")
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_ExtrapolateValues($func_name)
    ;ExtrapolateValues($y2, $y1, $x2, $x1, $x)
    ;                          y2, y1, x2, x1, x    = y
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, ExtrapolateValues( 1,  0,  1,  0,  2) = 2)
    PostTest($tArray)
```

```
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ExtrapolateValues(-1,  0, -1,  0, -2)  = -2)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ExtrapolateValues(-1,  1, -1,  1, 0)   = 0)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ExtrapolateValues(-1,  3, -1,  3, 2)   = 2)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ExtrapolateValues( 2,  1, 20, 10, 30)  = 3)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, ExtrapolateValues( 1,  2,  5, 10, 2.5) = 0.5)
    PostTest($tArray)

    PostPostTest()
EndFunc
Func Test_CalcXLinear($func_name)
    PrePreTest($func_name)
    Local $tArray[1]
    Local $t = 1
    PreTest($t)
    UTAssert($tArray, CalcXLinear("1, 0", 0) = 0)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcXLinear("0, 0", 0) = 0)    ;when m=0, should probably
       throw an error (corrected in function to make m= really small)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcXLinear("0, 0", 1) = 1e99) ;when m=0, should probably
       throw an error (corrected in function to make m= really small)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcXLinear("1, 1", 0) = -1)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcXLinear("2, 0", 1) = 0.5)
    PostTest($tArray)
    $t = $t + 1
    PreTest($t)
    UTAssert($tArray, CalcXLinear("2, 1", 1) = 0)
```

```
    PostTest($tArray)

    PostPostTest()
EndFunc


;-------------------------------------------------------------------------------
; Tests for method
;-------------------------------------------------------------------------------

Func Test_WriteMsgAll()
    Local $verbose = True
    WriteMsgStart("Test title", "Test msg", $verbose)
    Local $max = 10
    Local $ti  = $max / 100
    For $i = 1 To $max
        WriteMsg($i, $ti, "Step " & $i, $verbose)
    Next
    WriteMsgStop($verbose)
EndFunc
```

Appendix B Test Suite Summary Example

Listing 4: Example of test suite output

```
;-----------------------------------------------
;change $test_suite = False to show MsgBoxes
;test failures are shown in Console Window
;-----------------------------------------------
------- Start of Tests -------

------- Testing HDTS.au3 -------
        GetHuntDay...
                Running case 1...
                        no case written
                ...OK
        ...Done
        GetHuntTypeUI...
                Running case 1...
                        no case written
                ...OK
        ...Done
        DetectorButton...
                Running case 1...
                        no case written
                ...OK
        ...Done
        EndHuntButton...
                Running case 1...
                        no case written
```

```
            ...OK
        ...Done
        QueryButton...
            Running case 1...
                no case written
            ...OK
        ...Done
        CLOSEClicked...
            Running case 1...
                no case written
            ...OK
        ...Done
        HDTSSetupTestDB...
            Running case 1...
                no case written
            ...OK
        ...Done
        HDTSTeardownTestDB...
            Running case 1...
                no case written
            ...OK
        ...Done
        MainHDTS...
            Running case 1...
                no case written
            ...OK
        ...Done

------ Testing guiDet.au3 ------
        OnDet...
            Running case 1...
                no case written
            ...OK
        ...Done
        OffDet...
            Running case 1...
                no case written
            ...OK
        ...Done
        WarmupDet...
            Running case 1...
                no case written
            ...OK
        ...Done
        CalibDet...
            Running case 1...
                no case written
            ...OK
        ...Done
        BkgDet...
            Running case 1...
```

```
                no case written
        ...OK
...Done
HandleDetError...
        Running case 1...
                no case written
        ...OK
...Done
ControlChartDet...
        Running case 1...
                no case written
        ...OK
...Done
ValidateProcessReprocessInput...
        Running case 1...
                no case written
        ...OK
...Done
CountSampleTop...
        Running case 1...
                no case written
        ...OK
...Done
CountPhantomTop...
        Running case 1...
                no case written
        ...OK
...Done
CountSample...
        Running case 1...
                no case written
        ...OK
...Done
CountPhantom...
        Running case 1...
                no case written
        ...OK
...Done
CountPhantomOrSample...
        Running case 1...
                no case written
        ...OK
...Done
AddAnimal...
        Running case 1...
                no case written
        ...OK
...Done
StartButton...
        Running case 1...
                no case written
```

```
        ...OK
...Done
StopButton...
        Running case 1...
                no case written
        ...OK
...Done
OnButton...
        Running case 1...
                no case written
        ...OK
...Done
OffButton...
        Running case 1...
                no case written
        ...OK
...Done
ControlChart...
        Running case 1...
                no case written
        ...OK
...Done
CalibButton...
        Running case 1...
                no case written
        ...OK
...Done
BackgroundButton...
        Running case 1...
                no case written
        ...OK
...Done
PhantomButton...
        Running case 1...
                no case written
        ...OK
...Done
ClearForm...
        Running case 1...
                no case written
        ...OK
...Done
DisableInput...
        Running case 1...
                no case written
        ...OK
...Done
EnableInput...
        Running case 1...
                no case written
        ...OK
```

```
...Done
ProcessButton...
        Running case 1...
                no case written
        ...OK
...Done
ReprocessButton...
        Running case 1...
                no case written
        ...OK
...Done
DeleteAnimalButton...
        Running case 1...
                no case written
        ...OK
...Done
ShowSpectraButton...
        Running case 1...
                no case written
        ...OK
...Done
AddHunterButton...
        Running case 1...
                no case written
        ...OK
...Done
CloseClicked...
        Running case 1...
                no case written
        ...OK
...Done
UpdateAnimalIdComboAux...
        Running case 1...
                no case written
        ...OK
...Done
UpdateAnimalIdCombo...
        Running case 1...
                no case written
        ...OK
...Done
UpdateAnimalIdComboDelete...
        Running case 1...
                no case written
        ...OK
...Done
UpdateAnimalIdComboReview...
        Running case 1...
                no case written
        ...OK
...Done
```

```
    UpdateAnimalIdComboReprocess...
            Running case 1...
                    no case written
            ...OK
    ...Done
    FillInputReprocess...
            Running case 1...
                    no case written
            ...OK
    ...Done
    FillInputReview...
            Running case 1...
                    no case written
            ...OK
    ...Done
    FillInputAux1...
            Running case 1...
                    no case written
            ...OK
    ...Done
    FillInputAux2...
            Running case 1...
                    no case written
            ...OK
    ...Done
    SetAnimalTypes...
            Running case 1...
                    no case written
            ...OK
    ...Done
    SetLiveorDressed...
            Running case 1...
                    no case written
            ...OK
    ...Done
    GUIDetUpdateIni...
            Running case 1...
                    no case written
            ...OK
    ...Done
    MainDet...
            Running case 1...
                    no case written
            ...OK
    ...Done

------ Testing tadinifuncs.au3 ------
    IniErrorHandler...
            Running case 1...
                    no case written
            ...OK
```

```
...Done
FatalError...
        Running case 1...
                no case written
        ...OK
...Done
AlertMsg...
        Running case 1...
                no case written
        ...OK
...Done
TadIniRead...
        Running case 1...
                no case written
        ...OK
...Done
TadIniWrite...
        Running case 1...
                no case written
        ...OK
...Done
TadIniWriteSection...
        Running case 1...
                no case written
        ...OK
...Done
TadIniReadSectionNames...
        Running case 1...
                no case written
        ...OK
...Done
TadIniReadSection...
        Running case 1...
                no case written
        ...OK
...Done
TadIniDelete...
        Running case 1...
                no case written
        ...OK
...Done

------ Testing lib_hdts.au3 ------
GetPastHuntDays...
        Running case 1...
                no case written
        ...OK
...Done
ValidHuntDay...
        Running case 1...
                Unit OK
```

```
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
...Done
NotNowHuntOK...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
HuntEndedOK...
        Running case 1...
                no case written
        ...OK
...Done
CreateHDTSTable...
        Running case 1...
                Unit OK
        ...OK
...Done
CreateHuntDayTable...
        Running case 1...
                no case written
        ...OK
...Done
SetADetInfo...
        Running case 1...
                no case written
        ...OK
...Done
GetMasterDetPos...
        Running case 1...
                no case written
```

```
            ...OK
        ...Done
        MapDefinedToPhysical...
            Running case 1...
                no case written
            ...OK
        ...Done
        GetHuntStats...
            Running case 1...
                no case written
            ...OK
        ...Done
        EndHunt...
            Running case 1...
                no case written
            ...OK
        ...Done

------ Testing method.au3 ------
        ErrorHandler...
            Running case 1...
                no case written
            ...OK
        ...Done
        ChecksumOK...
            Running case 1...
                Unit OK
            ...OK
            Running case 2...
                Unit OK
            ...OK
            Running case 3...
                Unit OK
            ...OK
            Running case 4...
                Unit OK
            ...OK
            Running case 5...
                Unit OK
            ...OK
            Running case 6...
                Unit OK
            ...OK
            Running case 7...
                Unit OK
            ...OK
            Running case 8...
                Unit OK
            ...OK
            Running case 9...
                Unit OK
```

```
        ...OK
        Running case 10...
                Unit OK
        ...OK
        Running case 11...
                Unit OK
        ...OK
        Running case 12...
                Unit OK
        ...OK
        Running case 13...
                Unit OK
        ...OK
        Running case 14...
                Unit OK
        ...OK
        Running case 15...
                Unit OK
        ...OK
        Running case 16...
                Unit OK
        ...OK
        Running case 17...
                Unit OK
        ...OK
        Running case 18...
                Unit OK
        ...OK
        Running case 19...
                Unit OK
        ...OK
        Running case 20...
                Unit OK
        ...OK
...Done
ParseDollarResponse...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
methodMIOComm...
        Running case 1...
                no case written
        ...OK
...Done
```

```
GetDetectorInformation...
        Running case 1...
                Unit OK
        ...OK
...Done
GetSpeHeader...
        Running case 1...
                no case written
        ...OK
...Done
GetSpeFooter...
        Running case 1...
                no case written
        ...OK
...Done
SpectrumSetup...
        Running case 1...
                no case written
        ...OK
...Done
DetectorReady...
        Running case 1...
                no case written
        ...OK
...Done
GetROICounts...
        Running case 1...
                no case written
        ...OK
...Done
GetPeakInfo...
        Running case 1...
                no case written
        ...OK
...Done
Count...
        Running case 1...
                no case written
        ...OK
...Done
CalcConc...
        Running case 1...
                Unit OK
                Unit OK
        ...OK
        Running case 2...
                Unit OK
                Unit OK
        ...OK
        Running case 3...
                Unit OK
```

```
        Unit OK
...OK
Running case 4...
        Unit OK
...OK
Running case 5...
        Unit OK
        Unit OK
...OK
Running case 6...
        Unit OK
        Unit OK
...OK
Running case 7...
        Unit OK
        Unit OK
...OK
Running case 8...
        Unit OK
        Unit OK
...OK
Running case 9...
        Unit OK
        Unit OK
...OK
Running case 10...
        Unit OK
        Unit OK
...OK
Running case 11...
        Unit OK
...OK
Running case 12...
        Unit OK
        Unit OK
...OK
Running case 13...
        Unit OK
        Unit OK
...OK
Running case 14...
        Unit OK
        Unit OK
...OK
Running case 15...
        Unit OK
        Unit OK
...OK
Running case 16...
        Unit OK
        Unit OK
```

```
        ...OK
        Running case 17...
                Unit OK
                Unit OK
        ...OK
        Running case 18...
                Unit OK
                Unit OK
        ...OK
        Running case 19...
                Unit OK
                Unit OK
        ...OK
        Running case 20...
                Unit OK
                Unit OK
        ...OK
        Running case 21...
                Unit OK
                Unit OK
        ...OK
        Running case 22...
                Unit OK
                Unit OK
        ...OK
        Running case 23...
                Unit OK
                Unit OK
        ...OK
        Running case 24...
                Unit OK
                Unit OK
        ...OK
        Running case 25...
                Unit OK
                Unit OK
        ...OK
        Running case 26...
                Unit OK
                Unit OK
        ...OK
...Done
VoltageFlux...
        Running case 1...
                no case written
        ...OK
...Done
GetVoltage...
        Running case 1...
                Unit OK
        ...OK
```

```
        Running case 2...
                Unit OK
        ...OK
...Done
_binary...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
...Done
GetHVState...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
```

```
        ...OK
... Done
ConvertTime ...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
... Done
GetChnInfo ...
        Running case 1...
                Unit OK
                Unit OK
                Unit OK
                Unit OK
                Unit OK
                Unit OK
        ...OK
... Done
CalcCPM ...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
```

```
        Running case 6...
                Unit OK
        ...OK
...Done
CalcMDA...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
...Done
CodeClash...
        Running case 1...
                no case written
        ...OK
...Done
DetectorError...
        Running case 1...
                no case written
        ...OK
...Done
SetupDetector...
        Running case 1...
                no case written
        ...OK
...Done
TeardownDetector...
        Running case 1...
                no case written
        ...OK
...Done
AutoCalibrate...
        Running case 1...
                no case written
        ...OK
...Done
GetBackground...
        Running case 1...
```

```
                no case written
        ...OK
...Done
GetPhantomExpectedConc...
        Running case 1...
                no case written
        ...OK
...Done
GetControlChart...
        Running case 1...
                no case written
        ...OK
...Done
GetDetectorUncertainty...
        Running case 1...
                Unit OK
        ...OK
...Done
CalcUncertaintyCountingStats...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
        Running case 10...
                Unit OK
        ...OK
        Running case 11...
                Unit OK
```

```
        ...OK
        Running case 12...
                Unit OK
        ...OK
...Done
CalcUncertainty...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
...Done
GetSample...
        Running case 1...
                no case written
        ...OK
...Done
Warmup...
        Running case 1...
                no case written
        ...OK
...Done
GetCorrectedActivityAux...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
GetCorrectedActivity...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
```

```
validate_cinch_num ...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
...Done
validate_animal_type ...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
validate_animal_wt ...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
...Done
```

```
validate_hunt_cmpt...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
...Done
validate_hunt_stand...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
```

```
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
...Done
validate_last...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
validate_last_hunter_id...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
validate_hunter_id...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
```

```
...Done
is_hunter_in_db...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
convert_mm...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
...Done
format_date...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
...Done
get_years_between...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
```

```
GetRefEuCPM...
        Running case 1...
                Unit OK
        ...OK
...Done
EuCPMok...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
...Done
CalcIngestionDose...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
```

```
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
...Done
GetNonDOEBackground...
        Running case 1...
                Unit OK
        ...OK
...Done
CalcSRSDose...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
...Done
GetHuntType...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
CalcAnimalReleaseStatusAux...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
```

```
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
...Done
CalcAnimalReleaseStatus...
        Running case 1...
                no case written
        ...OK
...Done
CalcHunterRecieveStatus...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
...Done
GetConsumerId...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
GetTotalHuntQuery...
        Running case 1...
                no case written
        ...OK
...Done
GetPhantomActivityQuery...
        Running case 1...
```

```
                no case written
        ...OK
...Done
getLIMSdb...
        Running case 1...
                Unit OK
                Unit OK
        ...OK
...Done
getLIMS...
        Running case 1...
                Unit OK
                Unit OK
        ...OK
...Done
needLIMSaux...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
        Running case 10...
                Unit OK
        ...OK
        Running case 11...
                Unit OK
        ...OK
        Running case 12...
                Unit OK
```

```
        ...OK
        Running case 13...
                Unit OK
        ...OK
        Running case 14...
                Unit OK
        ...OK
        Running case 15...
                Unit OK
        ...OK
        Running case 16...
                Unit OK
        ...OK
        Running case 17...
                Unit OK
        ...OK
...Done
needLIMS...
        Running case 1...
                no case written
        ...OK
...Done
LIMSanimal...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
        Running case 10...
```

```
                Unit OK
        ...OK
...Done
doLIMS...
        Running case 1...
                Unit OK
                Unit OK
        ...OK
        Running case 2...
                Unit OK
                Unit OK
        ...OK
        Running case 3...
                Unit OK
                Unit OK
        ...OK
        Running case 4...
                Unit OK
                Unit OK
        ...OK
...Done
GetAnimalIds...
        Running case 1...
                Unit OK
        ...OK
...Done
InvalidateAnimal...
        Running case 1...
                no case written
        ...OK
...Done
ShowSpectra...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
...Done
GetNotIncludeAnimals...
        Running case 1...
                Unit OK
                Unit OK
        ...OK
...Done
GetAnimalTypes...
        Running case 1...
                Unit OK
```

```
                Unit OK
        ...OK
        Running case 2...
                Unit OK
                Unit OK
        ...OK
...Done
AddHunter...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
...Done
GetLiveFromDressed...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
Reprocess...
        Running case 1...
                no case written
        ...OK
...Done
ProcessAnimal...
        Running case 1...
                no case written
        ...OK
...Done
UpdateAnimalInfo...
        Running case 1...
                Unit OK
        ...OK
...Done
UpdateAnimalActivity...
        Running case 1...
                no case written
        ...OK
...Done
UpdateAnimalDose...
        Running case 1...
```

```
                no case written
        ...OK
...Done
UpdateAnimalConsumer...
        Running case 1...
                no case written
        ...OK
...Done

------ Testing lib_dbcalls.au3 ------
SelectDataQuery...
        Running case 1...
                no case written
        ...OK
...Done
GetListFromDB...
        Running case 1...
                no case written
        ...OK
...Done
GetAnimalIds_DB_Call...
        Running case 1...
                no case written
        ...OK
...Done
GetPastHuntDays_DB_Call...
        Running case 1...
                no case written
        ...OK
...Done
ValidHuntDay_DB_Call...
        Running case 1...
                no case written
        ...OK
...Done
DoesTableExist...
        Running case 1...
                Unit OK
        ...OK
...Done
DropTable...
        Running case 1...
                Unit OK
        ...OK
...Done
RenameTable...
        Running case 1...
                Unit OK
                Unit OK
        ...OK
CreateTableFromTemplate...
```

```
        Running case 1...
                Unit OK
        ...OK
...Done
CreateHunterTable_DB_Call_3...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
...Done
ADOErrFunc1...
        Running case 1...
                no case written
        ...OK
...Done
ADOErrFunc2...
        Running case 1...
                no case written
        ...OK
...Done
CreateHunterTable_DB_Call_6...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
CreateHunterTable_DB_Call_7...
        Running case 1...
        ...OK
...Done
CreateHuntDayTable_DB_Call...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
GetHuntStats_DB_Call...
```

```
            Running case 1...
                    no case written
            ...OK
    ...Done
    GetHuntYearQuery...
            Running case 1...
                    no case written
            ...OK
    ...Done
    EndHunt_DB_Call...
            Running case 1...
                    no case written
            ...OK
    ...Done
    GetZeroNotNull...
            Running case 1...
                    no case written
            ...OK
    ...Done
    GetHuntDose_DB_Call...
            Running case 1...
                    no case written
            ...OK
    ...Done
    GetPastDose_DB_Call...
            Running case 1...
                    no case written
            ...OK
    ...Done

------ Testing lib_ui.au3 ------
    WriteMsgStart...
            Running case 1...
                    no case written
            ...OK
    ...Done
    WriteMsg...
            Running case 1...
                    no case written
            ...OK
    ...Done
    WriteMsgStop...
            Running case 1...
                    no case written
            ...OK
    ...Done

------ Testing lib_maestro.au3 ------
    DebugDll...
            Running case 1...
                    no case written
```

```
        ...OK
...Done
MIOCleanup...
        Running case 1...
                no case written
        ...OK
...Done
MIOCloseDetector...
        Running case 1...
                no case written
        ...OK
...Done
MIOComm...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetAppData...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetData...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetDetLength...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetDetectorInfo...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetLastError...
        Running case 1...
                no case written
        ...OK
...Done
MIOIsActive...
        Running case 1...
                no case written
        ...OK
...Done
MIOIsDetector...
        Running case 1...
                no case written
        ...OK
```

```
...Done
MIOOpenDetector...
        Running case 1...
                Unit OK
        ...OK
...Done
MIOOpenDetID...
        Running case 1...
                no case written
        ...OK
...Done
MIOSetAppData...
        Running case 1...
                no case written
        ...OK
...Done
MIOStartup...
        Running case 1...
                no case written
        ...OK
...Done
MIOCreateConfigList...
        Running case 1...
                no case written
        ...OK
...Done
MIOAddConfigItem...
        Running case 1...
                no case written
        ...OK
...Done
MIOCompressConfigList...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetConfigItem...
        Running case 1...
                no case written
        ...OK
...Done
MIOGetConfigName...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
...Done
MIOGetConfigMax...
        Running case 1...
```

```
                Unit OK
            ...OK
        ...Done
        MIODebug...
            Running case 1...
                no case written
            ...OK
        ...Done
        MIOCreateConfigItem...
            Running case 1...
                no case written
            ...OK
        ...Done
        MIOResetMasterList...
            Running case 1...
                no case written
            ...OK
        ...Done
        MIOWhoAreYou...
            Running case 1...
                no case written
            ...OK
        ...Done

------ Testing taddbfuncs.au3 ------
        OpenDatabase...
            Running case 1...
                Unit OK
            ...OK
        ...Done
        DoesTableExistRaw...
            Running case 1...
                Unit OK
            ...OK
            Running case 2...
                Unit OK
            ...OK
        ...Done
        RenameTableRaw...
            Running case 1...
                Unit OK
                Unit OK
            ...OK
        ...Done

------ Testing tadmathfuncs.au3 ------
        CalcValPoly...
            Running case 1...
                Unit OK
            ...OK
            Running case 2...
```

```
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
        Running case 7...
                Unit OK
        ...OK
        Running case 8...
                Unit OK
        ...OK
        Running case 9...
                Unit OK
        ...OK
        Running case 10...
                Unit OK
        ...OK
...Done
ExtrapolateValues...
        Running case 1...
                Unit OK
        ...OK
        Running case 2...
                Unit OK
        ...OK
        Running case 3...
                Unit OK
        ...OK
        Running case 4...
                Unit OK
        ...OK
        Running case 5...
                Unit OK
        ...OK
        Running case 6...
                Unit OK
        ...OK
...Done
CalcXLinear...
        Running case 1...
                Unit OK
        ...OK
```

```
                Running case 2...
                        Unit OK
                ...OK
                Running case 3...
                        Unit OK
                ...OK
                Running case 4...
                        Unit OK
                ...OK
                Running case 5...
                        Unit OK
                ...OK
                Running case 6...
                        Unit OK
                ...OK
        ...Done

Num Functions             & 205 \\
Num Functions not tested  & 138 \\
Num Functions tested      & 67 \\

Num Cases tested          & 312 \\

Num Units tested          & 351 \\
Num Units passed          & 351 \\
Num Units failed          & 0 \\
```

Distribution:

T. J. Aucott, 773-41A
A. D. Brand, 773-41A
D. A. Crowley, 773-42A
D. P. DiPrete, 773-41A
K. L. Dixon, 773-42A
T. P. Eddy, 730-4B
G. T. Jannik, 999-W
K. M. Kostelnik, 773-42A
K. M. Vangelas, 730-4B