

NRC Publications Archive Archives des publications du CNRC

The EER database and its Web interface Goosney, D.

For the publisher's version, please access the DOI link below./ Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

Publisher's version / Version de l'éditeur:

<https://doi.org/10.4224/8895009>

Student Report (National Research Council of Canada. Institute for Ocean Technology); no. SR-2005-17, 2005

NRC Publications Archive Record / Notice des Archives des publications du CNRC :

<https://nrc-publications.canada.ca/eng/view/object/?id=d6abe1c3-7ab5-4af4-8482-8cb9a7c78146>

<https://publications-cnrc.canada.ca/fra/voir/objet/?id=d6abe1c3-7ab5-4af4-8482-8cb9a7c78146>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

DOCUMENTATION PAGE

REPORT NUMBER	NRC REPORT NUMBER	DATE	
SR-2005-17		August 2005	
REPORT SECURITY CLASSIFICATION		DISTRIBUTION	
Unclassified		Unlimited	
TITLE			
THE EER DATABASE AND ITS WEB INTERFACE			
AUTHOR(S)			
David Goosney			
CORPORATE AUTHOR(S)/PERFORMING AGENCY(S)			
Institute for Ocean Technology, National Research Council, St. John's, NL			
PUBLICATION			
SPONSORING AGENCY(S)			
Institute for Ocean Technology, National Research Council, St. John's, NL			
IOT PROJECT NUMBER		NRC FILE NUMBER	
874			
KEY WORDS	PAGES	FIGS.	TABLES
EER Database	5, App. A-E		
SUMMARY			
<p>The EER database was created to store the data from the PJ00874 series of tests and to present this data in a concise and easy to access format. The data is organized in phases which are organized into series containing multiple runs. All of the runs in each series are done under the same conditions. The database is set up in the same way with one table for the runs and one for the series. The database is made up of three main components: the SQL Server Database the DLL file and the web page. The database stores all of the data from the rows and series. The DLL file transforms data from a format recognizable by the webpage and a format recognizable by the database and vice versa. A diagram of the entire system can be seen in Appendix A.</p>			
ADDRESS	National Research Council Institute for Ocean Technology Arctic Avenue, P. O. Box 12093 St. John's, Newfoundland, Canada A1B 3T5 Tel.: (709) 772-5185, Fax: (709) 772-2462		



National Research Council
Canada

Conseil national de recherches
Canada

Institute for Ocean
Technology

Institut des technologies
océaniques

THE EER DATABASE AND ITS WEB INTERFACE

SR-2005-17

David Goosney

August 2005

Table of Contents

Table of Contents.....	1
1. Introduction.....	2
2. Background.....	2
3. Database.....	2
4. DLL File.....	3
5. Web Interface.....	3
6. COM+ Package.....	4
7. DTS and Importing Data.....	4
7.1 DTS Routines.....	4
7.2 Inputting Data	5
Appendix A.....	6
Appendix B.....	7
Appendix C.....	22
Appendix D.....	24
Appendix E.....	25

1. Introduction

The EER database was created to store the data from the PJ00874 series of tests and to present this data in a concise and easy to access format. The data is organized in phases which are organized into series containing multiple runs. All of the runs in each series are done under the same conditions. The database is set up in the same way with one table for the runs and one for the series. The database is made up of three main components: the SQL Server Database the DLL file and the web page. The database stores all of the data from the rows and series. The DLL file transforms data from a format recognizable by the webpage and a format recognizable by the database and vice versa. A diagram of the entire system can be seen in Appendix A.

2. Background

The database is intended to store the data from a series of experiments done on lifeboats done between 2000 and 2003. The experiments involved deploying the lifeboat models under different conditions such as: height, deployment system, weather system, etc.

There are plans to expand this database to include other experiments and eventually make it accessible to people outside of NRC. This would require the implementation of security features to ensure that data is not altered.

3. Database

The database was created under SQL Server and is stored on the ODIN server. It stores all of the information that is displayed on the web page. The database is organized with two tables per phase: one holds the data for all the series and the other holds the data for

all the runs. Each entry in both tables is identified by a unique sequential ID number assigned at the time the entry was created. This ID number is automatically assigned by SQL server. The database also keeps track of when the entry was created and last modified.

4. DLL File

The DLL file is written and compiled in Visual Basic 6.0. It can be found along with its source code in the shared directory [\\odin\wwweer\\$\internal\component](\\odin\wwweer$\internal\component). There is one class for each phase. Each of these classes have identical code except for the names of the variables. The function of the DLL is to convert the information from the webpage into a format that the database can understand and vice versa. The webpage passes variables directly to the DLL file which converts them to SQL strings that are sent to the database. These SQL strings query the database. The DLL converts the data returned from the queries into ADO objects which can be passed back to the webpage. In order to communicate with the database it must use the COM+ interface which will be discussed in the next section. In order to use the DLL file the it must first be registered with the local computer and COM+. An example of the code from one phase of the DLL file can be seen in appendix B.

5. Web Interface

The web interface is accessible on the local intranet at “http:\\odin\eer”. The functions of this interface are: adding new runs/series, modifying the data in existing rows, Searching the database and displaying statistics. All of the information displayed on the website comes from the database. The web interface accepts an input from the user. This input is

then sent to the DLL file which queries the database and returns the data in the form of a ADO object. The data from this object is then displayed on the web page.

6. COM+ Package

The COM+ package is built into windows 2000 and facilitates communications between programs and databases. It is a part of the IIS(Internet Information Systems) package that can be installed from the control panel. It allows the programmer to specify the location of the database, the login information and the type of database. This means that in the future the maintenance of the database will be easier. If the network is changed the programmer will not have to recompile the DLL file to update the system, they will just need to change the connection string in COM+. COM+ also simplifies the connection process. The COM+ object automatically connects upon construction so the DLL only needs to call the constructor function and the rest is done automatically. A disadvantage to this component is that the programmer has to reregister the DLL file every time it is changed. This increases the development time but the time saved in maintenance greatly outweighs the increased development time.

7. DTS and Importing Data

7.1 DTS Routines

Although data can be inserted into the database through the web interface it would be very tedious to input all of the data using this method. In order to input the large amounts of data in the database we need another tool called Data Transformation Services(DTS).

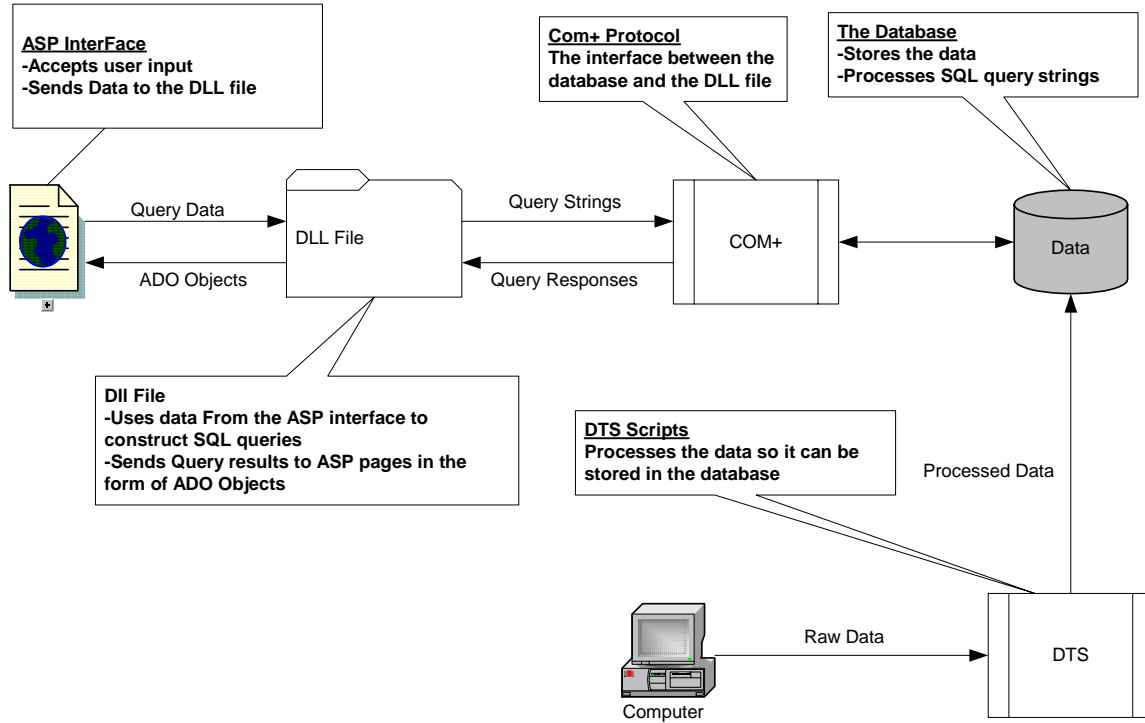
DTS is a package built into SQL server 2000 that allows the manipulation of large amounts of data. It is a user-friendly graphical language that uses SQL script to

manipulate data. The data is stored in a comma separated format on [\\odin\eer\\$](#). The comma separated format is a native format of both DTS and Visual Basic so it is a natural choice. A sample of the SQL code used to transform the data can be seen in Appendix C.

7.2 Inputting Data

The data to be converted by DTS originally came from Excel spreadsheets or IGOR .pxp files that were compiled during the testing phase. The data in the IGOR files is the raw data from the experiments and needs heavy processing before entry into the database. The data in the excel spreadsheets is already processed and only needs to be reformatted to fit in the database. This formatting was done using VBA (Visual Basic for Applications) a package built into Excel. An example of the formatting needed can be seen in appendix D. An example of the code used to process the data can be seen in appendix E.

Appendix A



Appendix B

Option Explicit

Implements IObjectConstruct 'from COM+ services type library

```
Private Sub IObjectConstruct_Construct(ByVal pCtorObj As Object)
```

```
    '*** Get the constructor string from COM+
```

```
    gsConnString = pCtorObj.ConstructString
```

```
End Sub
```

```
*****
```

```
' METHOD: getSeriesInfo
```

```
*****
```

```
Public Function getSeriesInfo(ByVal strSeriesID As String) As ADODB.Recordset
```

```
On Error GoTo ErrorHandler
```

```
    Dim rsSeries
```

```
    Set rsSeries = New ADODB.Recordset
```

```
    lngRetCode = initializeDB()
```

```
    strSQL = "select * from Phase0Series where SeriesID = " & strSeriesID
```

```
    rsSeries.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly
```

```
    Set getSeriesInfo = rsSeries
```

```
    Exit Function
```

```
ErrorHandler:
```

```
    Set getSeriesInfo = rsSeries
```

```
    strDesc = Err.Description
```

```
    If Err.Number <> 0 Then
```

```
        Open "C:\Temp\NRCLog.dat" For Append Shared As #1
```

```
        Write #1, CStr(Now), "nrcPh0.getSeriesInfo", getSeriesInfo, strDesc, strSQL
```

```
        Close #1
```

```
    Else
```

```
        Set getSeriesInfo = rsSeries
```

```
    End If
```

```
End Function
```

```
*****  
' METHOD: getRunInfo  
*****
```

```
Public Function getRunInfo(ByVal strRunID As String) As ADODB.Recordset
```

```
On Error GoTo ErrorHandler
```

```
Dim rsRun  
Set rsRun = New ADODB.Recordset
```

```
lngRetCode = initializeDB()
```

```
strSQL = "select * from Phase0Run where RunID = " & strRunID
```

```
rsRun.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly
```

```
Set getRunInfo = rsRun
```

```
Exit Function
```

```
ErrorHandler:
```

```
Set getRunInfo = rsRun  
strDesc = Err.Description  
If Err.Number <> 0 Then  
    Open "C:\Temp\NRCLog.dat" For Append Shared As #1  
    Write #1, CStr(Now), "nrcPh0.getRunInfo", getRunInfo, strDesc, strSQL  
    Close #1  
Else  
    Set getRunInfo = rsRun  
End If
```

```
End Function
```

```
*****  
' METHOD: listRunsForSeries  
*****
```

```
Public Function listRunsForSeries(ByVal strSeriesID As String) As ADODB.Recordset
```

```
Dim rsRunsForSeries
```

```
On Error GoTo ErrorHandler
```

```
Set rsRunsForSeries = New ADODB.Recordset
```

```

    lngRetCode = initializeDB()

    strSQL = "select * from Phase0Run where SeriesID = " & strSeriesID & " order by
RunName"

    rsRunsForSeries.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly

    Set listRunsForSeries = rsRunsForSeries

    Exit Function

ErrorHandler:
    Set listRunsForSeries = rsRunsForSeries
    strDesc = Err.Description
    If Err.Number <> 0 Then
        Open "C:\Temp\NRCLog.dat" For Append Shared As #1
        Write #1, CStr(Now), "nrcPh0.listRunsForSeries", listRunsForSeries, strDesc,
strSQL
        Close #1
    Else
        Set listRunsForSeries = rsRunsForSeries
    End If

End Function

'*****
' METHOD: listAllSeries
'*****

Public Function listAllSeries() As ADODB.Recordset

Dim rsSeries
Dim strSeriesName As String

On Error GoTo ErrorHandler

    Set rsSeries = New ADODB.Recordset

    lngRetCode = initializeDB()

    strSeriesName = "%"

    strSQL = "select Phase0Series.SeriesID,Phase0Series.SeriesName from Phase0Series "
    strSQL = strSQL & "where SeriesName like '" & strSeriesName & "' "

```

```
strSQL = strSQL & "order by SeriesName"  
rsSeries.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly
```

```
Set listAllSeries = rsSeries
```

```
Exit Function
```

```
ErrorHandler:
```

```
Set listAllSeries = rsSeries  
strDesc = Err.Description  
If Err.Number <> 0 Then  
    Open "C:\Temp\NRCLog.dat" For Append Shared As #1  
    Write #1, CStr(Now), "nrcPh0.listAllSeries", listAllSeries, strDesc, strSQL  
    Close #1  
Else  
    Set listAllSeries = rsSeries  
End If
```

```
End Function
```

```
*****  
' METHOD: listSeries  
*****
```

```
Public Function listSeries(ByVal strSeriesName As String, ByVal strWeatherType As  
String, _  
ByVal strLaunchConfiguration As String) As ADODB.Recordset
```

```
Dim rsSeries
```

```
On Error GoTo ErrorHandler
```

```
Set rsSeries = New ADODB.Recordset
```

```
lngRetCode = initializeDB()
```

```
strSeriesName = Replace(strSeriesName, "*", "%")
```

```
If strSeriesName = "" Then  
    strSeriesName = "%"  
End If
```

```

    strSQL = "select Phase0Series.SeriesID,Phase0Series.SeriesName, Phase0Run.RunID,
Phase0Run.RunName from Phase0Run INNER JOIN Phase0Series ON
Phase0Run.SeriesID = Phase0Series.SeriesID "
    strSQL = strSQL & "where SeriesName like '" & strSeriesName & "' "
    If strWeatherType <> 0 Then
        strSQL = strSQL & "and WeatherType = '" & CInt(strWeatherType) & "' "
    End If
    If strLaunchConfiguration <> 0 Then
        strSQL = strSQL & "and LaunchConfiguration = '" & CInt(strLaunchConfiguration)
& "' "
    End If

```

```

strSQL = strSQL & "order by SeriesName"
rsSeries.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly

```

```

Set listSeries = rsSeries

```

```

Exit Function

```

```

ErrorHandler:

```

```

    Set listSeries = rsSeries
    strDesc = Err.Description
    If Err.Number <> 0 Then
        Open "C:\Temp\NRCLog.dat" For Append Shared As #1
        Write #1, CStr(Now), "nrcPh0.listSeries", listSeries, strDesc, strSQL
        Close #1
    Else
        Set listSeries = rsSeries
    End If

```

```

End Function

```

```

*****
' METHOD: updateSeries
*****

```

```

Public Function updateSeries(ByVal strSeriesID As String, _
    ByVal strSeriesName As String, ByVal strWeatherType As String, _
    ByVal strLaunchConfiguration As String, ByVal strComments As String)
As Long

```

```

On Error GoTo ErrorHandler

```

```
lngRetCode = initializeDB()
```

```
strSQL = "update Phase0Series set "  
strSQL = strSQL & "SeriesName = " & strSeriesName & ""  
strSQL = strSQL & ", WeatherType = " & strWeatherType  
strSQL = strSQL & ", LaunchConfiguration = " & strLaunchConfiguration  
strSQL = strSQL & ", Comments = " & strComments & ""  
strSQL = strSQL & " where SeriesID = " & strSeriesID
```

```
dbNRC.Execute strSQL
```

Exit Function

ErrorHandler:

```
updateSeries = Err.Number  
strDesc = Err.Description  
If Err.Number <> 0 Then  
    Open "C:\Temp\NRCLog.dat" For Append Shared As #1  
    Write #1, CStr(Now), "nrcPh0.updateSeries", updateSeries, strDesc, strSQL  
    Close #1  
Else  
    updateSeries = -1  
End If
```

End Function

```
*****  
' METHOD: newSeries  
*****
```

```
Public Function newSeries(ByRef strSeriesID As Variant, _  
    ByVal strSeriesName As String, ByVal strWeatherType As String, _  
    ByVal strLaunchConfiguration As String, ByVal strComments As String)  
As Long
```

On Error GoTo ErrorHandler

```
lngRetCode = initializeDB()
```

```
strSQL = "insert into  
Phase0Series(SeriesName,WeatherType,LaunchConfiguration,Comments) "
```

```
strSQL = strSQL & "values('" & strSeriesName & "','" & strWeatherType & "','" &
strLaunchConfiguration
strSQL = strSQL & "','" & strComments & "''")"
```

```
dbNRC.Execute strSQL
```

```
strSQL = "SELECT SeriesID FROM Phase0Series WHERE SeriesName = '" &
strSeriesName & "' Order by SeriesID DESC"
```

```
Dim rsSeries
Set rsSeries = New ADODB.Recordset
rsSeries.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly
strSeriesID = rsSeries!SeriesID
rsSeries.Close
Set rsSeries = Nothing
```

Exit Function

ErrorHandler:

```
newSeries = Err.Number
strDesc = Err.Description
If Err.Number <> 0 Then
    Open "C:\Temp\NRCLog.dat" For Append Shared As #1
    Write #1, CStr(Now), "nrcPh0.newSeries", newSeries, strDesc, strSQL
    Close #1
Else
    newSeries = -1
End If
```

End Function

```
*****
' METHOD: deleteSeries
*****
```

Public Function deleteSeries(ByVal strSeriesID As String) As Long

On Error GoTo ErrorHandler

```
lngRetCode = initializeDB()
```

```
strSQL = "DELETE FROM Phase0Run WHERE SeriesID = " & strSeriesID
dbNRC.Execute strSQL
```

```
strSQL = "DELETE FROM Phase0Series WHERE SeriesID = " & strSeriesID
```



```
dbNRC.Execute strSQL
```

```
Exit Function
```

```
ErrorHandler:
```

```
deleteSeries = Err.Number
```

```
strDesc = Err.Description
```

```
If Err.Number <> 0 Then
```

```
Open "C:\Temp\NRCLog.dat" For Append Shared As #1
```

```
Write #1, CStr(Now), "nrcPh0.deleteSeries", deleteSeries, strDesc, strSQL
```

```
Close #1
```

```
Else
```

```
deleteSeries = -1
```

```
End If
```

```
End Function
```

```
*****  
' METHOD: deleteRun  
*****
```

```
Public Function deleteRun(ByVal strRunID As String) As Long
```

```
On Error GoTo ErrorHandler
```

```
lngRetCode = initializeDB()
```

```
strSQL = "DELETE FROM Phase0Run WHERE RunID = " & strRunID
```

```
dbNRC.Execute strSQL
```

```
Exit Function
```

```
ErrorHandler:
```

```
deleteRun = Err.Number
```

```
strDesc = Err.Description
```

```
If Err.Number <> 0 Then
```

```
Open "C:\Temp\NRCLog.dat" For Append Shared As #1
```

```
Write #1, CStr(Now), "nrcPh0.deleteRun", deleteRun, strDesc, strSQL
```

```
Close #1
```

```
Else
```

```
deleteRun = -1
```

```
End If
```

```
End Function
```

```
*****
' METHOD: getStats
*****
```

```
Public Function getStats(ByVal strSeriesName As String, ByVal strWeatherType As
String, _
                ByVal strLaunchConfiguration As String) As ADODB.Recordset
```

```
Dim rsStats
```

```
On Error GoTo ErrorHandler
```

```
Set rsStats = New ADODB.Recordset
```

```
lngRetCode = initializeDB()
```

```
'star * is the wildcard character used on the frontend. replace it with a % so sql
recognizes it.
```

```
strSeriesName = Replace(strSeriesName, "*", "%")
```

```
'if nothing is entered in the series name search criteria, use a % by itself so it won't
limit the results to
```

```
'series with out a name (wich should never occur because name is mandatory)
```

```
If strSeriesName = "" Then
```

```
strSeriesName = "%"
```

```
End If
```

```
'sample for testing in sql query analyzer:
```

```
'SELECT AVG(r.SplashDownData-0) AS LaunchToSplashAvg,
MAX(r.SplashDownData-0) AS LaunchToSplashMax, MIN(r.SplashDownData-0) AS
LaunchToSplashMin
```

```
'FROM Phase0Run r, Phase0Series s
```

```
'Where r.SeriesID = s.SeriesID
```

```
'and s.seriesname like '%'
```

```
'and (s.WaveSteepness = 2)
```

```
strSQL = "select AVG(r.SplashDownData-0) AS LaunchToSplashAvg,
MAX(r.SplashDownData-0) AS LaunchToSplashMax, MIN(r.SplashDownData-0) AS
LaunchToSplashMin, STDEV(r.SplashDownData-0) AS LaunchToSplashStdev "
```

```
strSQL = strSQL & "AVG(r.DavitFallsReleaseData - r.SplashDownVMS) AS
SplashtoDavitAvg, MAX(r.DavitFallsReleaseData - r.SplashDownVMS) AS
SplashtoDavitMax, MIN(r.DavitFallsReleaseData - r.SplashDownVMS) AS
SplashtoDavitMin, STDEV(r.DavitFallsReleaseData - r.SplashDownVMS) AS
SplashtoDavitStdev, "
```

```

    strSQL = strSQL & "AVG(r.DavitFallsReleaseData-r.LaunchStartVMS) AS
LaunchtoDavitAvg, MAX(r.DavitFallsReleaseData-r.LaunchStartVMS) AS
LaunchtoDavitMax, MIN(r.DavitFallsReleaseData-r.LaunchStartVMS) AS
LaunchtoDavitMin, STDEV(r.DavitFallsReleaseData-r.LaunchStartVMS) AS
LaunchtoDavitStdev, "
    strSQL = strSQL & "from Phase0Run r, Phase0Series s "
    strSQL = strSQL & "where r.SeriesID = s.SeriesID "
    strSQL = strSQL & "and s.SeriesName like '" & strSeriesName & "' "
    'on the front end, if no wavesteeppness is selected from the drop down on the search
page
    'then a value of zero 0 is passed into this function
    'so if it's anything but zero 0 include it as a condition of the sql query
    If strWeatherType <> 0 Then
        strSQL = strSQL & "and s.WeatherType = '" & CInt(strWeatherType) & "' "
    End If
    If strLaunchConfiguration <> 0 Then
        strSQL = strSQL & "and s.LaunchConfiguration = '" &
CInt(strLaunchConfiguration) & "' "
    End If

    'strSQL = strSQL & "order by s.SeriesName"
    rsStats.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly

```

```

Set getStats = rsStats

```

```

Exit Function

```

```

ErrorHandler:

```

```

Set getStats = rsStats
strDesc = Err.Description
If Err.Number <> 0 Then
    Open "C:\Temp\NRCLog.dat" For Append Shared As #1
    Write #1, CStr(Now), "nrcPh0.getStats", getStats, strDesc, strSQL
    Close #1
Else
    Set getStats = rsStats
End If

```

```

End Function

```

```

*****
' METHOD: newRun
*****

```

```

Public Function newRun(ByRef RunID As Variant, ByVal SeriesID As String, ByVal
RunName As String, ByVal WaveFile As String, ByVal TimeToComplete As String,
ByVal RunComments As String, ByVal FrontEndDeletesIGORPoints As String, _
'
ByVal SplashDownData As String, ByVal DavitFallsReleaseData As
String, ByVal TagLineReleaseData As String, ByVal PayoutRate As String, ByVal
DavitLoadsInboard As String, ByVal DavitLoadsOutboard As String, _
'
ByVal TagLineatSplash As String, ByVal BoomBaseMaxLoadData As
String, ByVal BoomBaseMaxLoadDataIGORPoints As String, ByVal BoomAngleStart
As String, ByVal BoomAngleMin As String, ByVal BoomAngleRelease As String, _
'
ByVal SplashOnWave As String, ByVal Impact As String, ByVal
BoundaryCrossingDangerIGORPoints As String, ByVal
BoundaryCrossingSplashIGORPoints As String, ByVal
BoundaryCrossingSafeIGORPoints As String, _
'
ByVal TEMPSCTravelDistanceSpashDanger As String, ByVal
TEMPSCTravelDistanceSpashtoSplash As String, ByVal
TEMPSCTravelDistanceSpashtoRescue As String, ByVal WaveHeight As String, ByVal
WavePeriod As String, _
'
ByVal WindSpeedB1 As String, ByVal WindSpeed2 As String, ByVal
WindSpeed3 As String, ByVal WindSpeed4T As String, ByVal WavePhaseStartT1 As
String, ByVal WavePhaseEndT2 As String, ByVal StartofLaunchCoordinatesX0 As
String, _
'
ByVal StartofLaunchCoordinatesY0 As String, ByVal
StartofLaunchCoordinatesZ0 As String, ByVal StartofLaunchCoordinatesD0 As String,
ByVal SplashDownCoordinatesX1 As String, _
'
ByVal SplashDownCoordinatesY1 As String, ByVal
SplashDownCoordinatesZ1 As String, ByVal SplashDownCoordinatesD1 As String,
ByVal TaglineReleaseCoordinatesX2 As String, _
'
ByVal TaglineReleaseCoordinatesY2 As String, ByVal
TaglineReleaseCoordinatesZ2 As String, ByVal TaglineReleaseCoordinatesD2 As
String, ByVal TaglineReleaseCoordinatesPitch As String, _
'
ByVal TaglineReleaseCoordinatesRoll As String, ByVal
SetbackCoordinatesX3 As String, ByVal SetbackCoordinatesY3 As String, ByVal
SetbackCoordinatesZ3 As String, ByVal SetbackCoordinatesD3 As String, _
'
ByVal SetbackCoordinatesPitch As String, ByVal
SetbackCoordinatesRoll As String, ByVal ProgressiveSetbackCoordinatesX4 As String,
ByVal ProgressiveSetbackCoordinatesY4 As String, _
'
ByVal ProgressiveSetbackCoordinatesZ4 As String, ByVal
ProgressiveSetbackCoordinatesD4 As String, ByVal
ProgressiveSetbackCoordinatesPitch As String, ByVal
ProgressiveSetbackCoordinatesRoll As String, _
'
ByVal LoweringPhaseX As String, ByVal LoweringPhaseY As String,
ByVal LoweringPhaseZ As String, ByVal LoweringPhaseYAWrms As String, ByVal
LoweringPhasePITCHrms As String, _
'
ByVal LoweringPhaseROLLrms As String, ByVal LoweringPhaseAccX
As String, ByVal LoweringPhaseAccY As String, ByVal LoweringPhaseAccZ As String,
ByVal LoweringPhaseT1 As String, ByVal LoweringPhaseT2 As String, _

```

```

'           ByVal SplashDownAccX As String, ByVal SplashDownAccY As String,
ByVal SplashDownAccZ As String, ByVal SplashDownTz1 As String, ByVal
SplashDownTz2 As String, ByVal SplashDownPhaseYAW As String, _
'           ByVal SplashDownPhasePITCH As String, ByVal
SplashDownPhaseROLL As String, ByVal AccelerationsMotionsDuringSailawayAccX
As String, ByVal AccelerationsMotionsDuringSailawayAccY As String, _
'           ByVal AccelerationsMotionsDuringSailawayAccZ As String, ByVal
AccelerationsMotionsDuringSailawayTZ1 As String, ByVal
AccelerationsMotionsDuringSailawayTZ2 As String, ByVal
AccelerationsMotionsDuringSailawayYAW As String, _
'           ByVal AccelerationsMotionsDuringSailawayPITCH As String, ByVal
AccelerationsMotionsDuringSailawayROLL As String) As Long

```

```

Public Function newRun(ByRef strRunID As Variant, ByVal strRunInfo As String) As
Long

```

```

On Error GoTo ErrorHandler

```

```

    lngRetCode = initializeDB()

```

```

    strSQL = "insert into
Phase0Run(SeriesID,RunName,Comments,TimeToComplete,WaveFile,LaunchStartVM
S,"
    strSQL = strSQL &
"LaunchStartIGORPoint,SplashDownVMS,SplashDownIGORPoint,DavitFallsReleaseDa
ta,DavitFallsReleaseDataIGORPoint,TagLineReleaseIGORPoint,"
    strSQL = strSQL &
"SplashonaWave,VideoSplash,BoundaryCrossingDangerIGORPoints,BoundaryCrossing
SafeIGORPoints,TEMPSCTravelDistanceSplashtoDanger,TEMPSCTravelDistanceDang
ertoSafe,AccelerationsDuringLoweringMin,"
    strSQL = strSQL &
"AccelerationsDuringLoweringMean,AccelerationsDuringLoweringMax,AccelerationsD
uringLoweringStdDev,AccelerationsDuringSailAwayMin,AccelerationsDuringSailAway
Mean,AccelerationsDuringSailAwayMax,AccelerationsDuringSailAwayStdDev,"
    strSQL = strSQL &
"StartofLaunchCoordinatesX0,StartofLaunchCoordinatesY0,StartofLaunchCoordinatesZ
0,SplashDownCoordinatesX1,SplashDownCoordinatesY1,SplashDownCoordinatesZ1,Se
tbackCoordinatesX2,"
    strSQL = strSQL &
"SetbackCoordinatesY2,SetbackCoordinatesZ2,WaveHeightUpstream,WaveHeightBeam
,WindSpeed) "
    strSQL = strSQL & "values(" & strRunInfo & ")"

```

```

'   strSQL = "insert into
Phase0Run(SeriesID,RunName,WaveFile,TimeToComplete,Comments,FrontEndDeletes
IGORPoints,SplashDownData,"

```

```

' strSQL = strSQL &
"DavitFallsReleaseData,TagLineReleaseData,PayoutRate,DavitLoadsInboard,DavitLoad
sOutboard,"
' strSQL = strSQL &
"TagLineatSplash,BoomBaseMaxLoadData,BoomBaseMaxLoadDataIGORPoints,Boom
AngleStart,BoomAngleMin,"
' strSQL = strSQL &
"BoomAngleRelease,SplashOnaWave,Impact,BoundaryCrossingDangerIGORPoints,Bou
ndaryCrossingSplashIGORPoints,"
' strSQL = strSQL &
"BoundaryCrossingSafeIGORPoints,TEMPSCTravelDistanceSpashDanger,TEMPSCTra
velDistanceSpashtoSplash,"
' strSQL = strSQL &
"TEMPSCTravelDistanceSpashtoRescue,WaveHeight,WavePeriod,WindSpeedB1,Wind
Speed2,WindSpeed3,WindSpeed4T,"
' strSQL = strSQL &
"WavePhaseStartT1,WavePhaseEndT2,StartofLaunchCoordinatesX0,StartofLaunchCoor
dinatesY0,StartofLaunchCoordinatesZ0,"
' strSQL = strSQL &
"StartofLaunchCoordinatesD0,SplashDownCoordinatesX1,SplashDownCoordinatesY1,S
plashDownCoordinatesZ1,"
' strSQL = strSQL &
"SplashDownCoordinatesD1,TaglineReleaseCoordinatesX2,TaglineReleaseCoordinates
Y2,TaglineReleaseCoordinatesZ2,"
' strSQL = strSQL &
"TaglineReleaseCoordinatesD2,TaglineReleaseCoordinatesPitch,TaglineReleaseCoordin
atesRoll,SetbackCoordinatesX3,"
' strSQL = strSQL &
"SetbackCoordinatesY3,SetbackCoordinatesZ3,SetbackCoordinatesD3,SetbackCoordinat
esPitch,SetbackCoordinatesRoll,"
' strSQL = strSQL &
"ProgressiveSetbackCoordinatesX4,ProgressiveSetbackCoordinatesY4,ProgressiveSetba
ckCoordinatesZ4,"
' strSQL = strSQL &
"ProgressiveSetbackCoordinatesD4,ProgressiveSetbackCoordinatesPitch,ProgressiveSetb
ackCoordinatesRoll,"
' strSQL = strSQL &
"LoweringPhaseX,LoweringPhaseY,LoweringPhaseZ,LoweringPhaseYAWrms,Lowerin
gPhasePITCHrms,LoweringPhaseROLLrms,"
' strSQL = strSQL &
"LoweringPhaseAccX,LoweringPhaseAccY,LoweringPhaseAccZ,LoweringPhaseT1,Lo
weringPhaseT2,SplashDownAccX,"
' strSQL = strSQL &
"SplashDownAccY,SplashDownAccZ,SplashDownTz1,SplashDownTz2,SplashDownPh
aseYAW,SplashDownPhasePITCH,"

```

```
' strSQL = strSQL &
"SplashDownPhaseROLL,AccelerationsMotionsDuringSailawayAccX,AccelerationsMot
ionsDuringSailawayAccY,"
' strSQL = strSQL &
"AccelerationsMotionsDuringSailawayAccZ,AccelerationsMotionsDuringSailawayTZ1,
AccelerationsMotionsDuringSailawayTZ2,"
' strSQL = strSQL &
"AccelerationsMotionsDuringSailawayYAW,AccelerationsMotionsDuringSailawayPITC
H,AccelerationsMotionsDuringSailawayROLL)"
```

```
dbNRC.Execute strSQL
```

```
Dim RunInfoArray, strRunName
RunInfoArray = Split(strRunInfo, ",")
strRunName = RunInfoArray(1)
'Counter = UBound(RunInfoArray)
```

```
'For x = 0 To Counter
' strValue = RunInfoArray(x)
'Next
```

```
strSQL = "SELECT RunID FROM Phase0Run WHERE RunName = " & strRunName
& " Order by RunID DESC"
```

```
Dim rsRun
Set rsRun = New ADODB.Recordset
rsRun.Open strSQL, dbNRC, adOpenStatic, adLockReadOnly
strRunID = rsRun!RunID
rsRun.Close
Set rsRun = Nothing
```

Exit Function

ErrorHandler:

```
newRun = Err.Number
strDesc = Err.Description
If Err.Number <> 0 Then
    Open "C:\Temp\NRCLog.dat" For Append Shared As #1
    Write #1, CStr(Now), "nrcPh0.newRun", newRun, strDesc, strSQL
    Close #1
Else
    newRun = -1
End If
```

End Function

```
*****  
' METHOD: updateRun  
*****
```

```
Public Function updateRun(ByVal strRunID As String, ByVal strRunInfo As String) As  
Long
```

```
On Error GoTo ErrorHandler
```

```
    lngRetCode = initializeDB()
```

```
    strSQL = "update Phase0Run set " & strRunInfo & " where RunID = " & strRunID
```

```
    dbNRC.Execute strSQL
```

Exit Function

```
ErrorHandler:
```

```
    updateRun = Err.Number
```

```
    strDesc = Err.Description
```

```
    If Err.Number <> 0 Then
```

```
        Open "C:\Temp\NRCLog.dat" For Append Shared As #1
```

```
        Write #1, CStr(Now), "nrcPh0.updateRun", updateRun, strDesc, strSQL
```

```
        Close #1
```

```
    Else
```

```
        updateRun = -1
```

```
    End If
```

End Function

Appendix C

```
DECLARE @SeriesName varchar(10)
DECLARE @RunNumber varchar(3)
DECLARE @PlotName varchar(50)
DECLARE @PlotValue nvarchar(20)
DECLARE @P0SeriesIDCheck int
DECLARE @P0RunIDCheck int

DECLARE @P0SeriesID nvarchar(5)

DECLARE @MySQL nvarchar(200)
DECLARE @TableName nvarchar(20)

SET @TableName = 'Phase0Run'

DECLARE Import_Cursor CURSOR FOR
SELECT SeriesName, RunNumber, PlotName, PlotValue FROM dbo.ImportData
OPEN Import_Cursor
FETCH NEXT FROM Import_Cursor INTO @SeriesName, @RunNumber, @PlotName,
@PlotValue

    WHILE @@FETCH_STATUS = 0
    BEGIN

        SET @P0SeriesIDCheck = (select SeriesID from Phase0Series where SeriesName =
@SeriesName)

        IF isnull(@P0SeriesIDCheck,0) = 0
        BEGIN
            insert into Phase0Series
                (
                    SeriesName,
                    WeatherType,
                    LaunchConfiguration,
                    Orientation,
                    Clearance,
                    Height,
                    Payload
                )
            values
                (
                    @SeriesName,
                    right(@SeriesName,1),
                    0,
                    1,
                    1,
                    1,
                    1
                )
        END
    END
```

```

        )
    END

    SET @P0SeriesID = (select SeriesID from Phase0Series where SeriesName =
@SeriesName)
    SET @P0RunIDCheck = (select RunID from Phase0Run where SeriesID = @P0SeriesID
and RunName = @RunNumber)

    IF isnull(@P0RunIDCheck,0) = 0
    BEGIN
        SET @MySQL = 'INSERT INTO dbo.' + @TableName + '(' + 'SeriesID,
RunName, ' + rtrim(@PlotName) + ')' +
        space(1) + 'values' + space(1) + '(' + @P0SeriesID + ', ' + "" + @RunNumber + ""
+ ', ' + @PlotValue + ')'

        EXEC (@MySQL)
        UPDATE dbo.Phase0Run SET DateTimeStampCreate = getdate() WHERE
SeriesID = @P0SeriesID and RunName = @RunNumber
    END

    ELSE
    BEGIN
        SET @MySQL = 'UPDATE dbo.' + @TableName + ' SET ' + @PlotName + ' = '
+ @PlotValue + ' FROM ' +
        @TableName + ' WHERE ' + ' SeriesID ' + ' = ' + @P0SeriesID + ' AND ' +
'RunName' + ' = ' + @RunNumber

        EXEC (@MySQL)
        UPDATE dbo.Phase0Run SET DateTimeStampModify = getdate() WHERE
SeriesID = @P0SeriesID and RunName = @RunNumber
    END

    FETCH NEXT FROM Import_Cursor INTO @SeriesName, @RunNumber,
@PlotName, @PlotValue
    END

CLOSE Import_Cursor
DEALLOCATE Import_Cursor

```

Appendix D

Series Name	Run Number	WindSpeed	Impact	SplashDownVMS
700	001	8.13245	Y	82.58
900	003	1.76349	N	45.87

Becomes

“SeriesName”, “RunNumber”, “PlotName”, “PlotValue”

“700”, “001”, “WindSpeed”, 8.13245

“700”, “001”, “Impact”, 1

“700”, “001”, “SplashDownVMS”, 82.58

“900”, “003”, “WindSpeed”, 1.76349

“900”, “003”, “Impact”, 0

“900”, “003”, “SplashDownVMS”, 45.87

Appendix E

Sub ExportFile()
,

' ExportFile Macro

' Macro recorded 6/28/2005 by David Goosney

,

' Keyboard Shortcut: Ctrl+Shift+G

,

Dim SeriesName As String

Dim RunName As String

Dim ValueName As Variant

Dim tempStr As String

Dim tempValue As Variant

Dim i As Integer

```
ValueName = Array("LaunchStartVMS", "", "LaunchStartIGORPoint", "SplashDownVMS", _  
    "SplashDownIGORPoint", "DavitFallsReleaseData", "DavitFallsReleaseDataIGORPoint", _  
"ReleaseVideo", "", _  
    "", "", "", "SplashonaWave", "VideoSplash", _  
    "PayoutRate", "DeltarandDeltarSumDeletes", "", "BoundaryCrossingDangerIGORPoints",  
"", _  
    "BoundaryCrossingSplashIGORPoints", "", "BoundaryCrossingSafeIGORPoints",  
"TEMPSCTravelDistanceSplashtoDanger", "TEMPSCTravelDistanceDangertoSplash", _  
    "TEMPSCTravelDistanceDangertoSafe", "", "", "", "Impact", _  
    "WavePhaseStartT1", "WavePhaseEndT2", "", "", "WaveAmplitude", _  
    "WavePeriod", "WaveSteepness", "WindSpeed", "StartofLaunchCoordinatesX0",  
"StartofLaunchCoordinatesY0", _  
    "StartofLaunchCoordinatesZ0", "StartofLaunchCoordinatesD0",  
"SplashDownCoordinatesX1", "SplashDownCoordinatesY1", "SplashDownCoordinatesZ1", _  
    "SplashDownCoordinatesD1", "SetbackCoordinatesX2", "SetbackCoordinatesY2",  
"SetbackCoordinatesZ2", "SetbackCoordinatesD2", _  
    "SetbackCoordinatesIGORPoints", "", "", "", "", _  
    "", "", "", "", "", _  
    "SplashDownAccX", "SplashDownAccY", "SplashDownAccZ", "", "SetBackAccX", _  
    "SetBackAccY", "SetBackAccZ", "", "ImpactAccX", "ImpactAccY", _  
    "ImpactAccZ", "", "SailAwayAccX", "SailAwayAccY", "SailAwayAccZ", _  
    "", "MiscAccX", "MiscAccY", "MiscAccZ", "", _  
    "AccelerationsMotionsDuringSailawayLoweringPhaseAccX",  
"AccelerationsMotionsDuringSailawayLoweringPhaseAccY",  
"AccelerationsMotionsDuringSailawayLoweringPhaseAccZ",  
"AccelerationsMotionsDuringSailawayLoweringPhaseX",  
"AccelerationsMotionsDuringSailawayLoweringPhaseY", _  
    "AccelerationsMotionsDuringSailawayLoweringPhaseYAW",  
"AccelerationsMotionsDuringSailawayAccX", "AccelerationsMotionsDuringSailawayAccY",  
"AccelerationsMotionsDuringSailawayAccZ", "AccelerationsMotionsDuringSailawayX", _
```

```

    "AccelerationsMotionsDuringSailawayY", "AccelerationsMotionsDuringSailawayZ",
    "AccelerationsMotionsDuringSailawayYAW", "AccelerationsMotionsDuringSailawayPITCH",
    "AccelerationsMotionsDuringSailawayROLL", _
    "DavitLoadsInboard", "DavitLoadsOutboard")

```

```

Open "c:\PhaseOneData.dat" For Append As #1

```

```

SeriesName = ActiveCell.Value
ActiveCell.Offset(1, 0).Select
Do While (ActiveCell.Value <> Empty)
    RunName = ActiveCell.Value
    For i = LBound(ValueName) To UBound(ValueName)
        ActiveCell.Offset(0, 1).Select
        tempStr = ActiveCell.Value
        Select Case tempStr
            Case "N", "calm"
                tempValue = 0
            Case "Y", "C"
                tempValue = 1
            Case "T"
                tempValue = 2
            Case "U"
                tempValue = 3
            Case "D"
                tempValue = 4
            Case Else
                tempValue = ActiveCell.Value
        End Select

        If ValueName(i) <> "" And tempStr <> "" And tempStr <> "-" Then
            Write #1, SeriesName, RunName, ValueName(i), tempValue
        End If
    Next
    ActiveCell.Offset(1, -UBound(ValueName) - 1).Select
Loop

Close #1
End Sub

```