#1$2+3K4RB_Text_Format - Format Text

{bml BLUEDOT.BMP}Function

This function will return a string variable passed as parameter 1 formatted to print with a line length of parameter 2. It will break each line at the end of a word

This routine is useful for printing a memo, or other text field, on a report.

{bml BLUEDOT.BMP}Calling Procedure

```
newstring = RB_Text_Format(oldstring, 65)
Printer.Print newstring
```

This will print the contents of oldstring as 65 character lines.

{bml BLUEDOT.BMP}Notes

This routine will work with printer ScaleMode set to character or inches. The second parameter must correspond with the Printer ScaleMode currently set.

Example Code

Files To Include To Use RDBLIB

1# RB_Text_Format
2$ RB_Text_Format Function
3+ 00001:006
4K RB_Text_Format;Formatting Text;

#[5]

5# RBErrFrm

#[6]

6# RBProbRpt

#7$8+9K10RB_Text_Format Sample Code

```
oldstring = SampleTable!Comments
newstring = RB_Text_Format(oldstring, 65)
Printer.Print newstring
```

7# RB_Text_FormatCode
8$ RB_Format_Text Sample Code
9+ 00013:007
10K RB_Format_Text;Sample Code;

#[11]

11# RBLogOptions

#12$13+14K15RB_Validate_Date Function

{bml BLUEDOT.BMP}Function

This function validates the input (date) contained in control passed as parameter 1. It will return True if the input is a valid date, the string "__/__/__" or null. It will display a msgbox with an "Enter a valid date" msg and return False if the input date is invalid.

{bml BLUEDOT.BMP}Calling Procedure

TxtDate_LostFocus

```
IF Not RB_Validate_Date(TxtDate) then
    TxtDate.setfocus
End If
```

{bml BLUEDOT.BMP}Notes

The input date can be entered in 1 or 2 digit mm/dd/yy format or as a 6 digit number. All of the following are valid formats for April 1, 1995: 4/1/95, 04/01/95, 04-01-95, 4-1-95, and 040195.

Sample Code
Files To Include To Use RDBLIB

12# RB_Validate_Date
13$ RB_Validate_Date Function
14+ 00001:007
15K RB_Validate_Date;Date Validation;

#[16]$[17]+[18]K[19]RB_Validate_Date Sample Code

```
IF Not RB_Validate_Date(TxtDate) then
    TxtDate.setfocus
End If
```

16# RB_Validate_DateCode
17$ RB_Validate_Date Sample Code
18+ 00015:007
19K RB_Validate_Date;Sample Code;

#[20]$[21]K[22]RDBLIB Routines

RDBLIB is a set of common subroutines and functions for use in Visual Basic Programs. The followings functions are available:

20# Contents
21$ Table Of Contents
22K Contents;Index;Table Of Contents;

#[23]$[24]+[25]K[26]RB_ErrorHandler

{bml BLUEDOT.BMP}Function

This sub-routine provides standardized error handling for a Visual Basic program. It will display a message box indicating the error that occurred and provide options to create a problem report, display help on the error, ignore the error, retry the function, or abort the program. The program will also log the error to the logging database (RDBLOG.RDB). Queries are available in it to display the errors.

The function returns a code indicating the user's choice to retry or ignore the error.

{bml BLUEDOT.BMP}Calling Procedure

```
erraction = RB_ErrorHandler("FormName", "Routine")
Select Case erraction
Case 1
      Resume 0        ' Retry option selected
Case 2
      Resume Next     ' Ignore option selected
End Select
```

FormName is the name of the form where the error occurs

Routine is the name of the routine where the error occurs. You can provide as much information in the EventName parameter as is needed to help you identify where the error occurs. As a minimum it should be the event name.

The value returned (erraction above) indicates the users choice for handling the error. A value of 0 indicates the Retry choice, 1 indicates the Ignore choice. The Abort option terminates the program before returning to the calling routine.

Prior to calling the error routine, normally in the main form load event, you can set the following two variables to automatically fill in on the problem report.

RB_systemname (As String) - the name of your system (program)
RB_version (As String) - the version number of your system (program)

{bml BLUEDOT.BMP}Notes

These routines provide a means for the user to create a problem report that contains information that can assist the developer in determining the cause of the error condition. This report can be printed.

Sample Code
Error Message Box
Problem Report Form
Files To Include To Use RDBLIB

23# RBErrorHandler
24$ RB_ErrorHandler
25+ 00001:002
26K Error Handler;RB_ErrorHandler;

#[27]$[28]+[29]K[30]Files To Include To Use RDBLIB

RDBLIB.BAS
RBERRFRM.FRM
RBLOGOPT.FRM
RBPROBRP.FRM
RBSCRN.FRM

27# Includes
28$ Includes Required To Use RDBLIB
29+ 00001:010
30K Includes;

#31$32+33K34RB_Center - Centering Text

{bml BLUEDOT.BMP}Function

This sub-routine will center a string passed as parameter 1 on the printer line passed as parameter 2 or current line if parameter 2 = 0. It Will skip to next line if parameter 3 = true

{bml BLUEDOT.BMP}Calling Procedure

RB_Center (str_to_print As String, line_no, skip_line As Integer)

str_to_print = the string that is to print centered and printed

line_no = the line number to print the string on, if this value is zero the text will be printed on the current line

skip_line = true causes the routine to skip to the next line after printing the text, other values leave the current print line on the line specified by parameter 2

Sample Code
Files To Include To Use RDBLIB

31# RBCenter
32$ RB_Center
33+ 00001:003
34K Centering Text;RB_Center;

```
#[35]$[36]+[37]K[38]     RB_systemname = "RDBLIB Demo"
    RB_version = "1.1"

    on error goto errorroutine
....
....

errorroutine:
     erraction = RB_ErrorHandler("MainForm", "Sample Code")
     Select Case erraction
     Case 1
          Resume 0         ' Retry option selected
     Case 2
          Resume Next     ' Ignore option selected
     End Select
```

35# RBErrorCode
36$ RB_ErrorHandler Sample Code
37+ 00002:007
38K Sample Code;RB_ErrorHandler;

RB_Center Sample Code

```
sString = "This is a string to centered by itself on line 6 of a printout"

RB_Center sString, 6, True
```

39# RB_CenterCode
40$ RB_Center Sample Code
41+ 00007:005
42K Sample Code;RB_Center;

#[43]$[44]+[45]K[46]RB_RJustify - Right Justify Number To Specified Position

{bml BLUEDOT.BMP}Function

This function will print a number passed as parameter 1 according to the format passed as parameter 2 right justified on the column passed as parameter 3. It returns the leftmost column position where printing started

{bml BLUEDOT.BMP}Calling Procedure

leftcol = RB_Rjustify(200, "###,###.##", 40)

This will print " 200.00" with the rightmost 0 at column 40

{bml BLUEDOT.BMP}Notes

This routine will work with the printer scalemode set to inches or characters. The third parameter must be appropriate for the current printer scale mode.

Sample Code

Files To Include to Use RDBLIB

43# RB_RJustify
44$ RB_RJustify
45+ 00001:004
46K RB_RJustify;Justifying numbers;Right justify numbers;

RB_Rjustify Sample Code

```
leftcol = RB_Rjustify(200, "###,###.##", 40)
```

47# RB_RJustifyCode
48$ RB_RJustify Sample Code
49+ 00009:007
50K Sample Code;RB_RJustify;

#[51]$[52]+[53]K[54]RB_StatusMsg - Return Status Bar Message

{bml BLUEDOT.BMP}Function

This function can be used with VSVBX or other controls that use the first part of the tag property for some purpose and you want to use the second part of the tag property for your purposes. For example VSVBX can use the first part of the tag property for a label. I use the second part for a status bar message (hence the name StatusMsg). The separator that is used to indicate the separation of the parts is the pipe symbol (|).

{bml BLUEDOT.BMP}Calling Procedure

Set the tag property of a control, e.g. 'Name|Enter the persons name'

StatusBar.Caption = RB_StatusMsg((TxtName.Tag))

This will display "Enter the persons name" in the status bar.

{bml BLUEDOT.BMP}Notes

If you are passing the tag property directly, as in the example above, you must enclose it in parenthesis because the parameter is specified As Text.

Sample Code
Files To Include To Use RDBLIB

51# RB_StatusMsg
52$ RB_StatusMsg Function
53+ 00001:005
54K RB_StatusMsg;Status message;

RB_StatusMsg Sample Code

```
StatusBar.Caption = RB_StatusMsg((TxtName.Tag))
```

55# RB_StatusMsgCode
56$ RB_StatusMsg Sample Code
57+ 00011:007
58K Sample Code;RB_StatusMsg;

#[59]$[60]+[61]K[62]ShellAndWait Subroutine

{bml BLUEDOT.BMP}Function

This subroutine will start (via the Shell Function) the command passed as parameter 1 and wait until the command has completed and the window closed

This routine is useful if you do not want to continue execution until the program you started finishes.

{bml BLUEDOT.BMP}Calling Procedure

```
ShellAndWait("COPY A.TXT B.TXT")
' The B.TXT file will be available now
```

{bml BLUEDOT.BMP}Notes

If you use the normal VB Shell function your program continues as soon as the other program is initiated.

The program you start can be a Windows or DOS program.

Sample Code
Files To Include To Use RDBLIB

59# ShellAndWait
60$ ShellAndWait Subroutine
61+ 00001:008
62K ShellAndWait;Shelling another program;Waiting for another program;

ShellAndWait Sample Code

```
ShellAndWait("COPY A.TXT B.TXT")
```

63# ShellAndWaitCode
64$ ShellAndWait Sample Code
65+ 00017:007
66K ShellAndWait;Sample Code;

#[67]$[68]+[69]K[70]Performance Logging Functions

RDBLIB provides a facility for logging the performance (response times) of components of your application. The basic steps required are:

There are a number of Queries defined in the logging database (RDBLOG.MDB) that can be used to display and analyze the performance data.

This facility can also be handy for troubleshooting at times. You can use the log records to determine where you were in your program when an error occurred.

67# PerfLogFunctions
68$ Performance Logging Functions
69+ 00001:009
70K Performance Logging;Logging, Performance;

#71$72+73K74RB_SetLogOptions

{bml BLUEDOT.BMP}Function

This sub-routine can be called to set the performance and error logging options. It displays a form that allows you to indicate if you want performance data logged, where the logging database is located and the users name.

{bml BLUEDOT.BMP}Calling Procedures

RB_SetLogOptions

{bml BLUEDOT.BMP}Form

{bmc LOGOPTN.SHG}

71# RB_SetLogOptions
72$ RB_SetLogOptions Subroutine
73+ 00019:002
74K RB_SetLogOptions;Setting Logging Options;Logging Options;Options, Logging;Performance Logging;

#[75]$[76]Check this box if you want to log performance data. If this box is not checked then performance data will not be logged.

75# ChkLogPerfData
76$ Log Performance Data Checkbox

#[77]Enter the full path for where the logging database (RDBLOG.MDB) is located. If you don't know the location use the browse button to the right of this field to locate the database.

77# TxtLogFileLoc

#[78]Click on this button to obtain a Windows File browse dialog for locating the logging database.

78# BtnFindFile

#[79]Enter your name here. This name will appear in the error and performance log data to identify where the data came from.

79# TxtUserName

#[80]Click this button to save the changes you have made into the RDBLIB.INI file.

#[81]Click this button to cancel the changes you have made.

81# BtnCancel

#[82]$[83]+[84]K[85]RB_OpenLog Routine

{bml BLUEDOT.BMP}Function

This routine is called to open the logging database. You should do this at the beginning of your program. Along with opening the logging database this routine will determine from the RDBLIB.INI file if you are logging performance data or not.

{bml BLUEDOT.BMP}Calling Procedure

RB_OpenLog

{bml BLUEDOT.BMP}Notes

If the routine can not open the logging database for some reason it will set the flag for logging performance data off.

82# RB_OpenLog
83$ RB_OpenLog Routine
84+ 00019:003
85K RB_OpenLog;Open Logging Database;Logging Database, opening;Performance Logging;

#86$87+88K89RB_LogTask Function

{bml BLUEDOT.BMP}Function

This routine is called to log the start or end of a task. The difference between the start and end determine the duration (response time) for a task.

{bml BLUEDOT.BMP}Calling Procedure

RB_LogTask sTask, sForm, iFunction, sComments

Where:
sTask = a string identifying the task being timed.
sForm = a string identifying the form where this task is being performed
iFunction = RB_STARTTASK to indicate the start of a task or RB_ENDTASK to indicate the end of a task
sComments = any additional information regarding the task

{bml BLUEDOT.BMP}Notes

RB_STARTTASK and RB_ENDTASK are defined as Global Constants in RDBLIB.BAS.

The RB_STARTTASK will log the ending time for the previous task if a RB_ENDTASK was not done.

The comments for a task can be set with either the time of the STARTTASK or ENDTASK function. The comments are useful for saving information that may indicate the detail of the function being done and why a function may be taking a long time. For example, if you are timing the execution of an SQL statement, comments can contain the SQL statement.

Sample Code

86# RB_LogTask
87$ RB_LogTask Routine
88+ 00019:004
89K Logging Performance Data;RB_LogTask;Performance Data, Logging;

#[90]$[91]+[92]K[93]
```
RB_LogTask ("Opening DataBase", "MainForm", RB_STARTTASK, sDataBaseName)
Set dbSampleDB = OpenDatabase(sDataBaseName)
sSQL = "SELECT * FROM 'Sample Table' WHERE key = '12345'"
RB_LogTask("Selecting Data", "MainForm", RB_STARTTASK, sSQL)
Set tbSampleDS = dbSampleDB.OpenDynaset(sSQL)
RB_LogTask("Selecting Data", "MainForm", RB_ENDTASK, "")
```

90# RB_LogTaskCode
91$ RB_LogTask Sample Code
92+ 00022:007
93K RB_LogTask;Sample Code;

#[94]$[95]+[96]K[97]RB_CloseLog Routine

{bml BLUEDOT.BMP}Function

This routine is called to close the logging database when your program is ending.

{bml BLUEDOT.BMP}Calling Procedure

RB_CloseLog

94# RB_CloseLog
95$ RB_CloseLog Routine
96+ 00019:005
97K RB_CloseLog;Closing Logging Database;

#[98]+[99]{bmc ERRMSG.SHG}

98# ErrMsgBox
99+ 00002:008

#[100]+[101]{bml PROBRPT.BMP}

100# BtnProbRpt
101+ 00002:009

#[102]This message indicates the error that occurred.

#[103]Click this button to abort (end) the program now.

103# BtnAbort

#[104]Click this button to retry the function that originally created the error.

104# BtnRetry

#[105]Click this button to ignore the error and continue on to the next statement.

105# BtnIgnore

#[106]Click this button to obtain additional information about the cause of the error and potential resolutions to it.

106# BtnHelp