VREXX
VISUAL REXX FOR PRESENTATION MANAGER
VERSION 1.0

September 9, 1992

Richard B. Lam

IBM T.J. Watson Research Center
Route 134 POB 218
Yorktown Heights, NY 10598

CONTENTS

INTRODUCTION 1
System Requirements 2
Installation 2

USING VREXX 3

COMMAND LIST 7

COMMAND REFERENCE 11
VArc 12
VBackColor 13
VCheckBox 13
VClearWindow 14
VCloseWindow 15
VColorBox 15
VDialogPos 16
VDraw 16
VDrawParms 18
VExit 19
VFileBox 20
VFontBox 21
VForeColor 22
VGetVersion 22

VInit 23
VInputBox 24
VListBox 25
VMsgBox 26
VMultBox 26
VOpenWindow 28
VRadioBox 29
VResize 30
VSay 31
VSetFont 32
VSetTitle 33
VTableBox 33

EXAMPLE VREXX PROCEDURES 37
TESTWIN.CMD 37
TESTDLGS.CMD 40
TESTDRAW.CMD 44

TECHNICAL DATA 49

RELEASE NOTES AND COMMENTS 51

INTRODUCTION

VREXX is Visual REXX - a new way for OS/2 users to create their own Presentation Manager (PM) programs using REXX! VREXX provides users with a set of functions that can be called from REXX procedures. These functions open and close standard PM windows, providing programmable control over the appearance and positioning of the windows. Dialog box functions allow file selection, display of messages, entering numbers or text strings, and making single or multiple selections through radiobutton, checkbox or listbox controls. Table, Color and font selection dialogs are also available. And, graphics functions for setting pixels, drawing markers, lines, polygons, splines, arcs, circles and text (in multiple fonts) are included.

With VREXX, OS/2 REXX procedures can use all of the standard features of REXX under OS/2, except that the old text window input and output procedures are replaced with PM windows and dialogs. No prior experience with PM programming is necessary. The OS/2 Programming toolkit is NOT required. All you need to do is write a REXX program that makes function calls to the VREXX functions.

VREXX features:

o   Creation and manipulation of standard PM windows

o   Powerful dialog functions, including:

    -   Positioning control over the dialogs

    -   Dialog button selections

    -   Standard filename selection dialog

    -   Data Table, Color selection and Font selection dialogs

    -   10 line message box

    -   Input boxes for entering text or numbers

    -   Radiobutton, checkbox and listbox controls for selecting item(s) from a list

o   Graphics support, with functions for:

    -   Setting window foreground and background colors

    -   Setting individual pixels

    -   Drawing markers, with 10 different marker types

    - Polylines, with 7 different line types

    - Filled polygons, with 6 different fill types

    - Splines

    - Arcs and circles

- o On-line help facility

SYSTEM REQUIREMENTS

VREXX runs under OS/2 PM version 2.0 on IBM PS/2 or PC-compatible systems.

INSTALLATION

Copy VREXX.INF to a BOOKSHELF help file directory specified in your CONFIG.SYS file. Copy VREXX.EXE and the sample command files to a utility directory included in your PATH statement. Copy VREXX.DLL and DEVBASE.DLL to a directory specified in your LIBPATH in CONFIG.SYS.

USING VREXX

REXX procedures that call VREXX functions are started normally, by either typing the CMD filename on an OS/2 command line, or by using the OS/2 START command.

To run a REXX procedure named EXAMPLE.CMD which calls VREXX functions, simply type:

```
example
```

or

```
start example.cmd
```

from an OS/2 command line prompt. The EXAMPLE.CMD procedure will then execute normally, in addition to providing access to the VREXX functions. To access on-line help for VREXX, use the OS/2 VIEW command by typing:

```
view vrexx.inf
```

or

```
vrexx
```

from an OS/2 command line prompt.

Before calling VREXX functions in your REXX procedures, you must load and initialize the external functions by calling VInit. Also, the VExit function must be called at the end of your REXX procedure to clean up the system resources allocated in the initialization. The recommended approach for this is to structure your REXX procedure as follows:

```
/* EXAMPLE.CMD - structure for initializing and */
/*              terminating VREXX procedures   */
/* initialize VREXX */
'@echo off'
call RxFuncAdd 'VInit', 'VREXX', 'VINIT'
initcode = VInit()
if initcode = 'ERROR' then signal CLEANUP
signal on failure name CLEANUP
signal on halt name CLEANUP
signal on syntax name CLEANUP

/* REXX statements and VREXX function calls go here */
/* ...                                           */
/* end of REXX statements                         */

/* terminate VREXX - add any other clean-up */
/* for your REXX procedure here also        */

CLEANUP:
   call VExit
 exit
```

The SIGNAL statements ensure that VExit is called if your REXX procedure contains an error. You may optionally add a SIGNAL ON ERROR NAME CLEANUP statement also, depending on whether you provide another error handler for non-fatal ERROR return codes.

After initialization, VREXX lets you create multiple windows, with each window returning a specific id that you use to refer to the window for later operations. Note for PM programmers: the REXX command files are procedural, not event-driven. Therefore, your REXX procedure executes from top to bottom as a normal REXX program. But, at any time, especially when dialogs are displayed, the windows created with VREXX calls can be manipulated just like other PM windows - they may be iconized, resized, moved, etc. Also, the contents of the window are maintained internally - you don't need to redraw the window every time it is moved or sized. The windows are destroyed by calling a window close function, passing it the id of the window to close.

Although multiple windows may be created, only 1 dialog box at a time may be processed by the running REXX procedure.

Graphics coordinates for the windows are always set from 0 to 1000 in both the x and y directions, with the origin

at the lower left corner of the window. The current color and line type apply to all graphics operations.

The window and dialog positioning functions always operate with numbers representing a percentage of the screen. Thus, to center a window on the screen with the window filling half of the screen area, the left and bottom corners of the window are set to 25, while the right and top corners of the window are set to 75. See the command reference section for more examples on graphics and window positioning.

There are three sample REXX programs that come with the package, called TESTWIN.CMD, TESTDLGS.CMD, and TESTDRAW.CMD, which demonstrate the syntax of the VREXX functions. The next two sections are a summary of these functions and a function reference, including notes on the syntax and arguments for VREXX functions.

COMMAND LIST

This section provides a summary of the functions which can be called from a REXX procedure running under VREXX. See the EXAMPLES section for some REXX procedures which implement the VREXX commands.

The following functions are provided:

- o   Startup, Termination and Version Functions

VEXIT	Cleans up the current VREXX system resources
VGETVERSION	Returns the current VREXX program version number
VINIT	Initializes the VREXX functions and system resources

o Window Functions

VBACKCOLOR	Sets the background color of a window
VCLEARWINDOW	Clears the contents of a window
VCLOSEWINDOW	Closes a window
VFORECOLOR	Sets the foreground color of a window
VOPENWINDOW	Opens a new window
VRESIZE	Resizes and repositions a window on the screen
VSETTITLE	Sets the titlebar of a window to a specified string

o Dialog Functions

VCHECKBOX	Creates a checkbox dialog for selecting multiple items from a list
VCOLORBOX	Allows selection of foreground and background colors from a dialog

VDIALOGPOS	Controls the positioning of dialog windows on the screen

VFILEBOX	Allows selection of a full pathname of a file from a dialog
VFONTBOX	Allows selection of the typeface and point size to use for text output
VINPUTBOX	Creates an entryfield dialog with prompt strings for entering numbers or strings
VLISTBOX	Creates a listbox dialog for selecting 1 item from a list
VMSGBOX	Creates a message box for dis-playing from 1 to 10 message strings
VMULTBOX	Creates a multiple entryfield dialog, with 1 to 10 entryfields and a prompt string for each field, with optional echoing of input characters (e.g. for entering passwords).
VRADIOBOX	Creates a radiobox dialog for selecting 1 item from a large list
VTABLEBOX	Constructs a table dialog as a listbox, with programmable column widths

o Graphics Functions

VARC	Draws an arc or complete circle, optionally filled with the current fill style
VDRAW	Draws pixels, markers, lines, polygons or splines using the current marker type, line attri-bute and fill style

VDRAWPARMS	Sets the current marker type, line attribute and fill style to use for subsequent graphics oper- ations
VSAY	Draws a text string in the current font on a window
VSETFONT	Sets the current font to use for drawing text

COMMAND LIST 9

COMMAND REFERENCE

This is an alphabetical list of the VREXX functions. The calling arguments are described, and implementation limits and notes on each function are given.

For the dialog functions, several of them take a [stem] variable name as an argument. For example, the VMsgBox function is called as follows:

```
  msg.0 = 2
  msg.1 = 'This is the first line'
  msg.2 = 'This is the second line'

  buttons = 1

  call VMsgBox 'Dialog title', msg, buttons
```

where msg is the variable name of a stem variable. This variable uses the same format for all dialog functions, where the stem.0 variable holds the number of items, and stem.1 through stem.n hold the actual items. In the example above, there are 2 message lines to be displayed, so msg.0 is set to 2, and msg.1 and msg.2 hold the actual lines that will be displayed by the function.

The dialogs also take a standard [buttons] argument, which is defined as a number between 1 and 6, denoting that the following buttons be created on the dialog:

[buttons] value	Buttons created	Return value
1	OK	'OK'
2	Cancel	'CANCEL'
3	OK and Cancel	'OK' or 'CANCEL'
4	Yes	'YES'
5	No	'NO'
6	Yes and No	'YES' or 'NO'

In the example above, the [buttons] argument to the VMsgBox function was 1, so the message box dialog would be created with a single pushbutton labelled "OK". The VMsgBox function could also be called with the syntax:

```
  return_button = VMsgBox('Dialog title', msg, buttons)
```

where the return_button variable would be set to the return value corresponding to the pushbutton selected by the user (return_value = 'OK' in this example).

Those dialogs which need to return a selected string will place the selected string in a [stem].vstring variable.
For example, to access the string typed into an entryfield with the VInputBox function, use the following code:

```
  str.0 = 1
  str.1 = 'Type a string'
  call VInputBox 'Example', str, 1

  answer = str.vstring

  /* answer now contains the user input */
```

PURPOSE Draws an arc or complete circle, optionally filled with the current fill style

DEFINITION VARC [ID] [X] [Y] [RADIUS] [ANGLE1] [ANGLE2]

PARAMETERS [id] is the window id. [x] and [y] are the center point of the arc, and [radius] is the radius of the arc, in units of 0 to 1000. [angle1] and [angle2] are the angles to draw the arc between, starting with angle 0 at 3 o'clock, increasing in a counter-clockwise direction.

COMMENTS [angle1] should be less than [angle2] and both angles should be between 0 and 360 degrees. The angles may be specified in floating point format.

FUNCTION RESULT none

Example:

```
/* draw an arc in the center of a window, going
   from 12 o'clock to 6 o'clock, with a radius of
   100 */

a1 = 90
a1 = 270.0
call VArc id, 500, 500, 100, a1, a2
```

VBACKCOLOR

PURPOSE Sets the background color of a window

DEFINITION VBACKCOLOR [ID] [COLOR]

PARAMETERS [id] is the window id and [color] is the new background color to use for the window.

COMMENTS [color] must be specified as a string, in either upper, lower or mixed case, and must equal one of 'BLACK', 'WHITE', 'RED', 'GREEN', 'BLUE', 'CYAN', 'YELLOW' or 'PINK'.

FUNCTION RESULT none

Example:
```
   /* change a window background color to 'RED' */
  call VBackColor id, 'RED'
```

VCHECKBOX

PURPOSE Creates a checkbox dialog for selecting multiple items from a list

DEFINITION VCHECKBOX [TITLE] [STEM] [OUTPUT] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar, and [stem] is the variable name of the stem variable containing the items that will be used in constructing the dialog. [output] is the variable name of the stem variable where the selected items will be placed, and [buttons] denotes the desired button types to be placed on the dialog.

COMMENTS A maximum of 10 items may be passed to this function. The [output] stem variable need not exist when this function is called. The number of items selected is given by the [output].0 variable name (e.g. if [output] = user_selection, then the

REXX variable user_selection.0 holds the number of items checked in the dialog, and user_selection.1 through user_selection.n hold the actual selections). The [output] variable can be initialized before calling this function with the default strings to be checked when the dialog is created.

FUNCTION RESULT 'OK', 'CANCEL', 'YES' or 'NO', depending on the [buttons] argument

Example:

```
/* let user select movies */

movie.0 = 5
movie.1 = 'Silence of the Lambs'
movie.2 = 'Dr. Strangelove'
movie.3 = 'Terminator 2'
movie.4 = 'Goldfinger'
movie.5 = 'Basic Instinct'

button = VCheckBox('Select movies', movie, selection, 3)
if button = 'OK' then do
call VMsgBox('Your selections', selection, 1)
end
```

VCLEARWINDOW

PURPOSE Clears the contents of a window

DEFINITION VCLEARWINDOW [ID]

PARAMETERS [id] is the id of the window to clear.

COMMENTS This function erases all graphics from a window, enabling you to start over with a new set of graphics commands.

FUNCTION RESULT none

Example:

```
/* clear a window of all graphics */
call VClearWindow id
```

VCLOSEWINDOW

PURPOSE Closes a window

DEFINITION VCLOSEWINDOW [ID]

PARAMETERS [id] is the id of the window you wish to close.

COMMENTS The window must have been opened with a call to VOpenWindow.

FUNCTION RESULT none

Example:

```
/* close a window */
call VCloseWindow id
```

VCOLORBOX

PURPOSE Allows selection of foreground and background colors from a dialog

DEFINITION VCOLORBOX [STEM]

PARAMETERS [stem] is the name of a stem variable which holds the .fore and .back color values for the foreground and back-ground colors.

COMMENTS The colors should be specified as one of 'BLACK', 'WHITE', 'RED', 'GREEN', 'BLUE', 'CYAN', 'YELLOW' or 'PINK'.

FUNCTION RESULT 'OK' or 'CANCEL'

Example:

```
/* get new foreground and background colors for a
   window and set them */

color.fore = 'BLACK'
color.back = 'WHITE'
button = VColorBox color

if button = 'OK' then do
  call VForeColor color.fore
  call VBackColor color.back
end
```

VDIALOGPOS

PURPOSE	Controls the positioning of dialog windows on the screen
DEFINITION	VDIALOGPOS [X] [Y]
PARAMETERS	[x] and [y] are the center position to use for positioning subsequent dialog boxes on the screen.
COMMENTS	[x] and [y] should be integers between 0 and 100, specified in percentage of the screen.
FUNCTION RESULT	none

Example:

```
  /* position a message box in the center of the screen */
     call VDialogPos 50, 50

  msg.0 = 1
  msg.1 = 'This box is in the center of the screen'
  call VMsgBox 'TEST', msg, 1
```

VDRAW

PURPOSE Draws pixels, markers, lines, polygons or splines using the current marker type, line attribute and fill style

DEFINITION VDRAW [ID] [DRAWTYPE] [XSTEM] [YSTEM] [NUM]

PARAMETERS [id] is the id of the window to use for drawing the graphics. [drawtype] is a string, which must be one of 'PIXEL', 'MARKER', 'LINE', 'POLYGON' or 'SPLINE', depending on the graphic to be drawn. [xstem] and [ystem] are variable names for stem variables, which contain the coordinates to be used for drawing the graphics (ranging from .1 to .n). [num] is the number of data points specified in the [xstem] and [ystem] variables.

COMMENTS The coordinates should range between 0 and 1000. The drawtypes and their effects are:

o 'PIXEL' sets a pixel in the fore-ground color for each point

o 'MARKER' draws a marker at each

point using the current marker type

- 'LINE' draws a polyline connecting all of the points using the current line attribute

- 'POLYGON' draws a closed figure using the coordinates as vertices, filling the figure with the current fill type

- 'SPLINE' requires 4 data points, and draws a Bezier cubic spline that passes through points 1 and 4, using points 2 and 3 as control points.

FUNCTION RESULT none

Example:

```
/* see the TESTDRAW.CMD procedure for examples of
   using this function */
```

VDRAWPARMS

PURPOSE Sets the current marker type, line attribute and fill style to use for subsequent graphics operations

DEFINITION VDRAWPARMS [ID] [MARKERTYPE] [LINETYPE] [FILLTYPE]

PARAMETERS [id] is the window id. [markertype] is the marker type to draw, [linetype] is the line attribute to use, and [filltype] is the fill style to use in subsequent VDraw operations.

COMMENTS 0 is the default for all 3 attributes, equal to a cross marker, a solid line, or an empty fill style. The other values and their corresponding meanings are shown in the example.

FUNCTION RESULT none

Example:

```
/* VDrawParms marker, line and fill values */

default = 0

/* marker types */
```

```
cross        = 1     /* X */
plus         = 2     /* + */
diamond      = 3     /* &diamond. */
square       = 4     /* [_] */
star6        = 5     /* 6 point star */
star8        = 6     /* 8 point star */
soliddiamond = 7     /* &DIAMOND. */
solidsquare  = 8     /* &sqbul. */
soliddot     = 9     /* . */
circle       = 10    /* O */

/* line types */

solid      = 0       /* ____  */
dot        = 1       /* ....  */
dash       = 2       /* ----  */
dashdot    = 3       /* -.-.  */
dotdot     = 4       /* .. .. */
longdash   = 5       /* __ __ */
dashdotdot = 6       /* -..-  */

/* set up fill types */

nofill    = 0        /*       */
solidfill = 1        /* &BOX. */
horz      = 2        /* ===== */
vert      = 3        /* ||||| */
leftdiag  = 4        /* \\\\\ */
rightdiag = 5        /* ///// */

/* sample function call */

call VDrawParms diamond, dotdot, leftdiag
```

VEXIT

PURPOSE Cleans up the current VREXX system resources

DEFINITION VEXIT

PARAMETERS none

COMMENTS This function should be called after all VREXX function calls in the current REXX procedure are made.

FUNCTION RESULT none

Example:

```
  /* terminate VREXX */

  call VExit
```

VFILEBOX

PURPOSE Allows selection of a full pathname of a file from a dialog

DEFINITION VFILEBOX [TITLE] [TEMPLATE] [STEM]

PARAMETERS [title] is the string to use for the dialog titlebar. [template] is the pathname template that specifies the file types to display. [stem] is the name of a stem variable that contains the full pathname of the selected file.

COMMENTS If the name of the [stem] variable is fname, the full pathname is returned in the REXX variable fname.vstring.

FUNCTION RESULT 'OK' or 'CANCEL'

Example:

```
/* get a filename */

button = VFileBox('Pick a file', '*.dat', name)
  if button = 'OK' then do
     filename = name.vstring

/* get size of file */
```

```
        bytes = stream(filename, C, 'query size')
        end
```

VFONTBOX

PURPOSE Allows selection of the typeface and point size to use for text output

DEFINITION VFONTBOX [STEM]

PARAMETERS [stem] is the name of a stem variable, with [stem].type and [stem].size containing the selected font type and font point size returned from the dialog box.

COMMENTS The point size must be a positive integer greater than zero. The font type must be one of the following strings:

- 'SYSTEM' - standard system font
- 'SYMBOL' - greek/math symbols
- 'COUR' - Courier, Courier Bold, Courier Italic, Courier Bold Italic
- 'COURB'
- 'COURI'
- 'COURBI'
- 'HELV' - Helvetica, Helvetica Bold, Helvetica Italic, Helvetica Bold Italic

- 'HELVB'
- 'HELVI'
- 'HELVBI'
- 'TIME' - Times Roman, TR Bold, TR Italic, TR Bold Italic
- 'TIMEB'
- 'TIMEI'
- 'TIMEBI'

FUNCTION RESULT 'OK' or 'CANCEL'

Example:

```
  /* let user pick a new font */

  cur_font.type = 'SYSTEM'
  cur_font.size = 10

  button = VFontBox(cur_font)

  if button = 'OK' then do
    call VSetFont id, cur_font.type, cur_font.size
   end
```

VFORECOLOR

PURPOSE Sets the foreground color of a window

DEFINITION VFORECOLOR [ID] [COLOR]

PARAMETERS [id] is the window id and [color] is the new foreground color to use for the window.

COMMENTS [color] must be specified as a string, in either upper, lower or mixed case, and must equal one of 'BLACK', 'WHITE', 'RED', 'GREEN', 'BLUE', 'CYAN', 'YELLOW' or 'PINK'.

FUNCTION RESULT   none

Example:

```
  /* change a window foreground color to 'PINK' */
   call VForeColor id, 'PINK'
```

VGETVERSION

PURPOSE   Returns the current VREXX program version number

DEFINITION   VGETVERSION

PARAMETERS   none

COMMENTS   none

FUNCTION RESULT   Returns the version number as major.minor

Example:

```
  /* test version of VREXX */

  ver = VGetVersion()

  if ver <> '2.1' then do
     msg.0 = 1
     msg.1 = 'Wrong version of VREXX'

     call VMsgBox('Initialization Error', msg, 2)
   exit
   end
```

VINIT

PURPOSE Initializes the VREXX functions and system resources

DEFINITION VINIT

PARAMETERS none

COMMENTS This function must be called before calling any other VREXX functions. The VExit routine should be called at the end of the REXX procedure if VInit was called.

FUNCTION RESULT Returns 'ERROR' if initialization was unsuccessful.

Example:

```
  /* load and initialize VREXX (use the */
  /* RxUtils function RxFuncAdd)        */

  call RxFuncAdd 'VInit', 'VREXX', 'VINIT'
  call VInit

  /* or */

  call RxFuncAdd 'VInit', 'VREXX', 'VINIT'
  initcode = VInit()
```

VINPUTBOX

PURPOSE Creates an entryfield dialog with prompt strings for entering numbers or strings

DEFINITION VINPUTBOX [TITLE] [STEM] [WIDTH] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar. [stem] is the name

of the stem variable containing the
prompt strings to display in the
dialog, and [width] if the width (in
character units) of the entryfield.
[buttons] is a number between 1 and 6
denoting the pushbuttons to display
on the dialog.

COMMENTS Up to 10 strings can be specified for
a prompt, and all strings should be
80 characters or less in length. The
[stem].vstring field may contain a
default value for the entryfield on
input, and holds the contents of the
entryfield when the dialog is fin-
ished.

FUNCTION RESULT 'OK', 'CANCEL', 'YES' or 'NO',
depending on the value of [buttons]

Example:

```
  /* get the user's name */

  prompt.0 = 4
  prompt.1 = 'Please enter your name'
  prompt.2 = 'Enter it first name last, last name first'
  prompt.3 = ''
  prompt.4 = 'Leave out your middle initial'

  prompt.vstring = 'Doe John'

  button = VInputBox('Verify info', prompt, 25, 2)

  if button = 'OK' then do
    name = prompt.vstring
  end
```

VLISTBOX

PURPOSE Creates a listbox dialog for

selecting 1 item from a large list

DEFINITION VLISTBOX [TITLE] [STEM] [WIDTH] [HEIGHT] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar. [stem] is the name of the stem variable which contains the number of items and text of each item to be placed in the listbox. [width] and [height] are the dimensions of the listbox in character units, and [buttons] is a number between 1 and 6 denoting the type of pushbuttons to display on the dialog.

COMMENTS Any number of strings may be passed to this function. On input, [stem].vstring may contain the default list item to be selected when the dialog is created. If a default is not specified, the first item becomes the default selection. On output, [stem].vstring contains the listbox item selected by the user.

FUNCTION RESULT 'OK', 'CANCEL', 'YES' or 'NO', depending on the value of [buttons]

Example:

```
/* select 1 item from a listbox */

clone.0 = 8
clone.1 = 'Northgate'
clone.2 = 'Everex'
clone.3 = 'Gateway'
clone.4 = 'PC Brand'
clone.5 = 'AST Research'
clone.6 = 'Tandy'
clone.7 = 'Swan'
clone.8 = 'Commodore'

call VListBox 'Pick an IBM PC clone', clone, 10, 5, 1
selection = clone.vstring
```

VMSGBOX

PURPOSE Creates a message box for displaying from 1 to 10 message strings

DEFINITION VMSGBOX [TITLE] [STEM] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar. [stem] is the name of the stem variable which contains the number of message lines text of each line to be displayed. [buttons] is a number between 1 and 6 denoting the type of pushbuttons to display on the dialog.

COMMENTS Up to 10 lines of 80 characters each may be displayed.

FUNCTION RESULT 'OK', 'CANCEL', 'YES' or 'NO', depending on the value of [buttons]

Example:

```
  /* display a message box */

  mbox.0 = 4
  mbox.1 = 'VREXX Version 1.0'
  mbox.2 = ''
  mbox.3 = 'Written by R.B. Lam'
  mbox.4 = '(C) Copyright IBM Corp.  1992'

  call VMsgBox 'VREXX Info', mbox, 1
```

VMULTBOX

PURPOSE Creates a multiple entryfield dialog, with 1 to 10 entryfields and a prompt string for each field, with optional echoing of input characters (e.g. for entering passwords).

DEFINITION VMULTBOX [TITLE] [PROMPT] [WIDTH] [HIDE] [RETURN] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar. [prompt] is the name of the stem variable containing the prompt strings to display (1 for each entryfield) in the dialog, where

[prompt].0 is the number of entryfields. [width] is an array of widths (in character units) to use, one for each entryfield. [hide] is an array where the elements are 0 or 1, depending on whether or not you wish to echo (0) or not echo (1) the characters as they are typed into the entryfield. This is useful for entering passwords (see the example below). [return] is the array of return strings, which represent what was typed into each entryfield. [return] may be initialized with default strings for each entryfield before this function is called. [buttons] is a number between 1 and 6 denoting the pushbuttons to display on the dialog.

COMMENTS Up to 10 strings can be specified for a prompt, and all strings should be 80 characters or less in length.

FUNCTION RESULT 'OK', 'CANCEL', 'YES' or 'NO', depending on the value of [buttons]

Example:

```
/* get system, userid, and password */
/* here are the prompts */

p.0 = 3
p.1 = 'System name'
p.2 = 'User ID'
p.3 = 'Password'

/* here are the widths for each entryfield */

w.0 = p.0
w.1 = 20
w.2 = 10
w.3 = 8

/* don't echo the password field */

h.0 = p.0
h.1 = 0
h.2 = 0
h.3 = 1

/* default strings */

r.0 = p.0
r.1 = 'IBMVM'
```

```
r.2 = 'johndoe'
r.3 = ''

button = VMultBox('Logon Panel', p, w, h, r, 2)

if button = 'OK' then do
  call VMsgBox 'Logon Info', r, 1
end
```

VOPENWINDOW

PURPOSE Opens a new window

DEFINITION VOPENWINDOW [TITLE] [COLOR] [STEM]

PARAMETERS [title] is the string to use for the window titlebar. [color] is the background color to use for the window. [stem] is the name of a stem variable which contains the position and size of the window when it it created. There are 4 fields, [stem].left, [stem].right,

[stem].bottom and [stem].top, which must be specified.

COMMENTS The left, right, top and bottom fields should be integer numbers representing a percentage of the screen, ranging between 0 and 100.

FUNCTION RESULT Returns an integer id number used to refer to the new window in subsequent function calls

Example:

```
/* put up a new window in the upper left quadrant of
   the screen, with a background of white */

pos.left   = 0
```

```
  pos.bottom = 0
  pos.right  = 50
  pos.top    = 100
  color = 'WHITE'
  new_id = VOpenWindow('An example window', color, pos)
```

VRADIOBOX

PURPOSE Creates a radiobox dialog for selecting 1 item from a list

DEFINITION VRADIOBOX [TITLE] [STEM] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar. [stem] is the name of the stem variable which contains the number of items and text of each item to be placed in the dialog. [buttons] is a number between 1 and 6 denoting the type of pushbuttons to display on the dialog.

COMMENTS A maximum of 10 items may be passed to this function. On input, [stem].vstring can specify the default radio button to be pressed when the dialog is created. If none is specified, the first item becomes the default. On output, [stem].vstring contains the item selected by the user.

FUNCTION RESULT 'OK', 'CANCEL', 'YES' or 'NO', depending on the value of [buttons]

Example:

```
  /* have user select a font by pushing a radiobutton */

  font.0 = 7
  font.1 = 'Garamond'
```

```
font.2 = 'Helvetica'
font.3 = 'Times Italic'
font.4 = 'Weather'
font.5 = 'Math'
font.6 = 'Orator'
font.7 = 'Default'

call VRadioBox 'Select a font', font, 1

msg.0 = 1
msg.1 = 'You selected' font.vstring

call VMsgBox 'Selection', msg, 1
```

VRESIZE

PURPOSE Resizes and repositions a window on the screen

DEFINITION VRESIZE [ID] [STEM]

PARAMETERS [id] is the id of the window to move and size. [stem] is the name of a stem variable containing the new coordinates of the window in percentage of screen units. The size and position are given in [stem].left, [stem].right, [stem].bottom and [stem].top.

COMMENTS The left, bottom, right and top coordinates should be integers between 0 and 100.

FUNCTION RESULT none

Example:

```
/* create a small window in the center of the screen,
   then move it to the lower right quadrant and
```

```
    increase its size */

  pos.left   = 40
  pos.bottom = 40
  pos.right  = 60
  pos.top    = 60
  id = VOpenWindow('Small window', 'WHITE', pos)

  pos.left   = 50
  pos.bottom = 0
  pos.right  = 100
  pos.top    = 50
  call VResize id, pos
```

VSAY

PURPOSE Draws a text string in the current font on a window

DEFINITION VSAY [ID] [X] [Y] [TEXT]

PARAMETERS [id] is the id of the window where the text will be drawn. [x] and [y] are the starting coordinates for the text, expressed as integers between 0 and 1000. [text] is the text string to draw.

COMMENTS The text will be drawn in the current font and the current color.

FUNCTION RESULT none

Example:

```
  /* draw a set of text strings */

  str.1 = 'You will need:
  str.2 = '*  C Compiler'
  str.3 = '*  OS/2 Programmer's Reference'

  x = 50
  y = 900
  do i = 1 to 3
    call VSay id, x, y, str.i
  y = y - 100
 end
```

VSETFONT

PURPOSE Sets the current font to use for drawing text

DEFINITION VSETFONT [ID] [TYPE] [SIZE]

PARAMETERS [id] is the id of the window. [type] is a string representing the typeface requested, and [size] is the point size for the requested font.

COMMENTS The point size must be a positive integer greater than zero. The typeface must be one of the following strings:

- 'SYSTEM' - standard system font
- 'SYMBOL' - greek/math symbols
- 'COUR' - Courier, Courier Bold, Courier Italic, Courier Bold Italic
- 'COURB'
- 'COURI'
- 'COURBI'
- 'HELV' - Helvetica, Helvetica Bold, Helvetica Italic, Helvetica Bold Italic
- 'HELVB'
- 'HELVI'
- 'HELVBI'

o 'TIME' - Times Roman, TR Bold, TR Italic, TR Bold Italic

o 'TIMEB'

o 'TIMEI'

o 'TIMEBI'

FUNCTION RESULT   none

Example:

```
  /* set the font to 20 point Helvetica Bold */
   call VSetFont id, 'HELVB', 20
```

VSETTITLE

PURPOSE   Sets the titlebar of a window to a specified string

DEFINITION   VSETTITLE [ID] [TITLE]

PARAMETERS   [id] is the id of the window, and [title] is the new string to use for the window's titlebar.

COMMENTS

FUNCTION RESULT   none

Example:

```
  /* open a window with one title, then change it */

  pos.left   = 25
  pos.bottom = 25
  pos.right  = 75
  pos.top    = 75

  id = VOpenWindow('Old Window Title', 'WHITE', pos)
```

```
call VSetTitle id, 'New Window Title'
```

VTABLEBOX

PURPOSE Constructs a table dialog as a listbox, with programmable column widths

DEFINITION VTABLEBOX [TITLE] [STEM] [SELECTION] [WIDTH] [HEIGHT] [BUTTONS]

PARAMETERS [title] is the string to use for the dialog titlebar. [stem] is the name of the stem variable which contains the number of rows and columns, column widths, column labels and text of each item to be placed in a table-

style listbox. [selection] contains the number of the table row to be selected when the dialog is created. [width] and [height] are the dimensions of the table in character units, and [buttons] is a number between 1 and 6 denoting the type of pushbuttons to display on the dialog.

COMMENTS Any number of strings may be passed to this function, all with a maximum length of 80. The number of columns in the table is limited to 10. The number of rows and columns in the table are specified with the [stem].rows and [stem].cols variables. The column widths are specified in [stem].width.1, [stem].width.2, etc. The column labels are specified in [stem].label.1, [stem].label.2, etc. Finally, the entries for the table are stored in row-column order, with [stem].1.1 being the entry for row 1, column 1, [stem].1.2 being the entry for row 1, column 2, etc. On output, [stem].vstring contains the table row number selected by the user.

FUNCTION RESULT   'OK', 'CANCEL', 'YES' or 'NO', depending on the value of [buttons]

Example:

```
/* display a table of data */

table.rows = 50
table.cols = 3

table.label.1 = 'Name'
table.label.2 = 'Division'
table.label.3 = 'Serial Number'

table.width.1 = 20
table.width.2 = 10
table.width.3 = 15

table.1.1 = 'John Doe'
table.1.2 = 10
table.1.3 = 'CR1034'

table.2.1 = 'Mary Jane'
table.2.2 = 44
table.2.3 = 'TX1143'

/* etc. */

table.50.1 = 'Joe Programmer'
table.50.2 = 11
table.50.3 = '001101'

call VTableBox 'Pick a row from the table', table, 1,             50, 15, 1
selection_number = table.vstring
```

EXAMPLE VREXX PROCEDURES

This section provides several example REXX procedures which give you some ideas about how to incorporate the VREXX functions in your own REXX programs. The following examples are presented:

TESTWIN.CMD	Shows creation and manipulation of PM windows, text display, etc.
TESTDLGS.CMD	Demonstrates the use of the standard dialog functions, including filename and list item selections.
TESTDRAW.CMD	Draws some arbitrary graphics to PM windows.

TESTWIN.CMD

```
        /* TESTWIN.CMD */
        '@echo off'
        call RxFuncAdd 'VInit', 'VREXX', 'VINIT'
        initcode = VInit()
        if initcode = 'ERROR' then signal CLEANUP
        signal on failure name CLEANUP
        signal on halt name CLEANUP
        signal on syntax name CLEANUP

        /* display the version number of VREXX */
        ver = VGetVersion()
        msg.0 = 1
        msg.1 = 'VREXX version # ' ver
        call VMsgBox 'TESTWIN.CMD', msg, 1

        /* open a window and draw some text */
        win.left   = 20
        win.right  = 70
        win.top    = 80
        win.bottom = 40
        id = VOpenWindow('My VREXX Window', 'RED', win)
        text.1 = 'This is a VREXX window, created with a call to
VOpenWindow.'
        text.2 = 'The window currently has a title = My VREXX
Window, and it'
        text.3 = 'has a red background, which can be changed by            a
call to the'
        text.4 = 'VBackColor function.  The font is 12 point               Times
Roman.'

        call VForeColor id, 'WHITE'
        call VSetFont id, 'TIME', 12

        x = 10
        y = 900
        do i = 1 to 4
          call VSay id, x, y, text.i
          y = y - 50
        end
        /* now display a message box */

        msg.0 = 2
        msg.1 = 'Press OK to change the window title, the'
        msg.2 = 'window background color, and the font...'
        call VMsgBox 'TESTWIN.CMD', msg, 1
        /* change the title and background color */
        call VSetTitle id, 'A New Title!'
        text.2 = 'The new window title = A New Title!, and it'
        call VClearWindow id
```

```
        call VBackColor id, 'BLUE'
        text.3 = 'has a blue background, which can be changed by           a
call to the'
        call VForeColor id, 'WHITE'

        /* change the font */
        call VSetFont id, 'HELVB', 15
        text.4 = 'VBackColor function.  The font is now 15 point
Helvetica Bold.'

        /* redraw the text in the window */
        x = 10
        y = 900
        do i = 1 to 4
          call VSay id, x, y, text.i
          y = y - 60
        end

        /* now move and resize the window */
        msg.0 = 3
        msg.1 = 'Now the window will be cleared and moved
around'
        msg.2 = 'and resized using the VResize function.  Press'
        msg.3 = 'OK to continue...'
        call VMsgBox 'TESTWIN.CMD', msg, 1
        call VClearWindow id
        win.left   = 5
        win.right  = 15
        win.bottom = 80
        win.top    = 95
        call VResize id, win
```

```
do 8
  win.left   = win.left   + 5
  win.right  = win.right  + 10
  win.top    = win.top    - 5
  win.bottom = win.bottom - 10
  call VResize id, win
end

/* put up a message box */
msg.0 = 1
msg.1 = 'Press Cancel to end...'
call VMsgBox 'TESTWIN.CMD', msg, 2
call VCloseWindow id

/* end of CMD file */
CLEANUP:
call VExit
exit
```

TESTDLGS.CMD

```
/* TESTDLGS.CMD */

'@echo off'
call RxFuncAdd 'VInit', 'VREXX', 'VINIT'
initcode = VInit()
if initcode = 'ERROR' then signal CLEANUP

signal on failure name CLEANUP
signal on halt name CLEANUP
signal on syntax name CLEANUP

/* example VMsgBox call */
msg.0 = 4
msg.1 = 'This is a 4 line message box dialog.'
msg.2 = 'This is the line 2.  Line 3 is blank.'
msg.3 = ''
msg.4 = 'Press YES or NO to continue...'

call VDialogPos 50, 50
rb = VMsgBox('TESTDLGS.CMD', msg, 6)
if rb = 'YES' then do
  msg.0 = 1
  msg.1 = 'You pressed YES'
end
else do
  msg.0 = 1
  msg.1 = 'You pressed NO'
end
call VMsgBox 'VMsgBox Result', msg, 1

/* VInputBox example */
prompt.0 = 2
prompt.1 = 'Enter your name'
prompt.2 = '(Last name first, First name last)'
prompt.vstring = 'Doe John'
button = VInputBox('VInputBox example', prompt, 20, 3)
if button = 'OK' then do
  msg.0 = 3
  msg.1 = 'You entered the name'
  msg.2 = prompt.vstring
  msg.3 = 'and you pressed OK'
end
else do
  msg.0 = 1
  msg.1 = 'You pressed CANCEL'
end
call VMsgBox 'VInputBox Result', msg, 1

/* VMultBox example */
prompt.0 = 2   /* 2 prompt lines */
```

```
prompt.1 = 'User ID'
prompt.2 = 'Password'
width.0 = 2
width.1 = 10   /* widths in character units */
width.2 = 8    /* for both entryfields */
```

```
hide.0 = 2
hide.1 = 0     /* echo the User ID input */
hide.2 = 1     /* don't echo the Password */
answer.0 = 2
answer.1 = ''  /* these are the default strings */
answer.2 = ''  /* which will contain the input */
button = VMultBox('VMultBox example', prompt, width,                hide,
answer, 3)
if button = 'OK' then do
  call VMsgBox 'VMultBox Result', answer, 1
end
else do
  msg.0 = 1
  msg.1 = 'You pressed CANCEL'
  call VMsgBox 'VMultBox Result', msg, 1
end

/* VListBox example */

list.0 = 17
list.1  = 'OS/2 2.0 Standard Edition'
list.2  = 'OS/2 2.0 Extended Edition'
list.3  = 'MMPM/2 Multimedia Extensions'
list.4  = 'Windows 3.0 Multimedia Extensions'
list.5  = 'Adobe Type Manager'
list.6  = 'C-Set/2 Compiler'
list.7  = 'OS/2 2.0 Programmer Toolkit'
list.8  = 'WorkFrame/2'
list.9  = 'Lan Server'
list.10 = 'Lan Requester'
list.11 = 'TCP/IP'
list.12 = 'PMGlobe Demo Program'
list.13 = 'ASYNC Terminal Emulator'
list.14 = 'IPFC Preprocessor'
list.15 = 'VREXX'
list.16 = 'OS/2 2.0 Corrective Service'
list.17 = 'IBM SAA CUA Controls Library'
list.vstring = list.15          /* default selection */
call VDialogPos 25, 25
call VListBox 'Select a Product and Press YES', list,              35, 8, 4
msg.0 = 1
msg.1 = list.vstring
call VMsgBox 'VListBox Selection', msg, 1
```

```
/* test of VTableBox */

table.rows = 5
table.cols = 3
table.label.1 = 'Name'
table.label.2 = 'Division'
table.label.3 = 'Serial Number'
table.width.1 = 25
table.width.2 = 10
table.width.3 = 15
table.1.1 = 'Mary Jacobs'
table.1.2 = 20
table.1.3 = '243611'
table.2.1 = 'Joe Johnson'
table.2.2 = 19
table.2.3 = '837462'
table.3.1 = 'Henry Hill'
table.3.2 = 79
table.3.3 = '832628'
table.4.1 = 'Ruby Potts'
table.4.2 = 11
table.4.3 = '937567'
table.5.1 = 'Gary Williams'
table.5.2 = 22
table.5.3 = '086203'
button = VTableBox('Employee List', table, 1, 40, 10, 1)
msg.0 = 2
msg.1 = 'Button pressed was' button
msg.2 = 'Selection number =' table.vstring
call VMsgBox 'VTableBox Result', msg, 1

/* VRadioBox example */

list.0 = 10
call VRadioBox 'Select 1 item', list, 1
msg.0 = 1
msg.1 = list.vstring
call VMsgBox 'Selected item', msg, 1

/* test of VCheckBox */

list.0 = 10
sel.0 = 2
sel.1 = list.2
sel.2 = list.3
```

```
call VCheckBox 'Select items', list, sel, 1
if sel.0 > 0 then do
  call VMsgBox 'Selected items', sel, 1
end

/* VColorBox example */

call VDialogPos 75, 75
color.fore = 'YELLOW'
color.back = 'BLUE'
call VColorBox color
msg.0 = 2
msg.1 = 'Foreground color is' color.fore
msg.2 = 'Background color is' color.back
call VMsgBox 'Color selections', msg, 1

/* VFontBox example */

font.type = 'HELVB'
font.size = 25
call VFontBox font
msg.0 = 2
msg.1 = 'Font type is' font.type
msg.2 = 'Font size is' font.size
call VMsgBox 'Font selection', msg, 1

/* test of VFileBox */

call VDialogPos 10, 50
button = VFileBox('Pick a file...', 'c:\os2\*.exe',                'file')
msg.0 = 3
msg.1 = 'File name picked was'
msg.2 = file.vstring
msg.3 = 'Button pressed was' button
call VMsgBox 'VFileBox Result', msg, 1

/* end of CMD file */
```

```
CLEANUP:
call VExit

exit
```

TESTDRAW.CMD

```
/* TESTDRAW.CMD */

'@echo off'
call RxFuncAdd 'VInit', 'VREXX', 'VINIT'
initcode = VInit()
if initcode = 'ERROR' then signal CLEANUP
signal on failure name CLEANUP
signal on halt name CLEANUP
signal on syntax name CLEANUP

/* set up marker types */
default      = 0
cross        = 1
plus         = 2
diamond      = 3
square       = 4
star6        = 5
star8        = 6
soliddiamond = 7
solidsquare  = 8
soliddot     = 9
circle       = 10

/* set up line types */
solid      = 0
dot        = 1
dash       = 2
dashdot    = 3
dotdot     = 4
```

```
        longdash   = 5
        dashdotdot = 6

        /* set up fill types */
        nofill    = 0
        solidfill = 1
        horz      = 2
        vert      = 3
        leftdiag  = 4
        rightdiag = 5

        /* create 2 windows for drawing some graphics */
        win1.left   = 15
        win1.bottom = 30
        win1.right  = 55
        win1.top    = 70
        id1 = VOpenWindow('TESTDRAW.CMD Graphics Window 1',
'WHITE', win1)
        win2.left   = 60
        win2.bottom = 10
```

```
        win2.right  = 95
        win2.top    = 40
        id2 = VOpenWindow('TESTDRAW.CMD Graphics Window 2',
'BLACK', win2)

        /* draw a line graph in window 1 */
        call VForeColor id1, 'BLACK'
        x.1 = 100
        y.1 = 600
        x.2 = 400
        y.2 = 600
        call VDraw id1, 'LINE', x, y, 2         /* x axis */
        x.1 = 100
        y.1 = 600
        x.2 = 100
        y.2 = 900
        call VDraw id1, 'LINE', x, y, 2         /* y axis */

        a = -0.000222   /* construct a quadratic polynomial */
        b = 0.861       /* Y = a*X*X + b*X + c */
        c = 566
```

```
x.1 = 100
y.1 = a*100*100 + b*100 + c
do i = 2 to 5
  j = i - 1
  x.i = x.j + 75
  y.i = a * x.i * x.i + b * x.i + c
end

call VDrawParms id1, soliddiamond, dashdot, default
call VDraw id1, 'MARKER', x, y, 5
call VDraw id1, 'LINE', x, y, 5

/* draw a set of arcs in window 2 */
call VForeColor id2, 'YELLOW'
cx = 100
cy = 200
radius = 20
angle1 = 0
angle2 = 60

do i = 1 to 6
  call VArc id2, cx, cy, radius, angle1, angle2
  radius = radius + 20
  cx = cx + 150
  angle2 = angle2 + 60
end

/* draw a bar graph in window 1 */
```

```
call VDrawParms id1, default, default, default
x.1 = 550
y.1 = 600
x.2 = 950
y.2 = 600
call VDraw id1, 'LINE', x, y, 2         /* x axis */
x.1 = 550
y.1 = 600
x.2 = 550
y.2 = 900
call VDraw id1, 'LINE', x, y, 2         /* y axis */
px.1 = 600
py.1 = 600
px.2 = 600
```

```
py.2 = 650
px.3 = 650
py.3 = 650
px.4 = 650
py.4 = 600
call VForeColor id1, 'RED'
do i = 1 to 6
  /* draw bar with a new fill type */
  call VDrawParms id1, default, solid, i-1
  call VDraw id1, 'POLYGON', px, py, 4
  call VDraw id1, 'LINE', px, py, 4
  px.1 = px.1 + 50
  px.2 = px.1
  px.3 = px.3 + 50
  px.4 = px.3
  py.2 = py.2 + 45
  py.3 = py.2
end

/* draw some lines of different types in window 2 */

color.1 = 'WHITE'              /* set up color array */
color.2 = 'RED'
color.3 = 'GREEN'
color.4 = 'BLUE'
color.5 = 'CYAN'
color.6 = 'YELLOW'
color.7 = 'PINK'

x.1 = 200
y.1 = 950
x.2 = 800
y.2 = 950

do i = 1 to 7
```

```
  call VForeColor id2, color.i
  call VDrawParms id2, default, i-1, default
  call VDraw id2, 'LINE', x, y, 2

  y.1 = y.1 - 100
  y.2 = y.1
```

```
        end

        /* set up a spline in window 1, drawing the control              points of
the spline as markers, and labelling them              with text  */

        sx.1 = 350
        sy.1 = 450
        sx.2 = 700
        sy.2 = 200
        sx.3 = 200
        sy.3 = 125
        sx.4 = 650
        sy.4 = 425

        call VForeColor id1, 'BLUE'
        call VDrawParms id1, soliddot, default, default
        call VDraw id1, 'MARKER', sx, sy, 4
        call VDraw id1, 'SPLINE', sx, sy, 4

        call VForeColor id1, 'GREEN'
        call VSetFont id1, 'HELVB', 12
        call VSay id1, 300, 75, 'Spline Control Points'

        /* put up a message box */

        msg.0 = 1
        msg.1 = 'Press OK to close the windows'
        call VMsgBox 'TESTDRAW.CMD', msg, 1

        call VCloseWindow id1
        call VCloseWindow id2

        /* end of CMD file */

        CLEANUP:
        call VExit

        exit
```

TECHNICAL DATA

VREXX packages its external functions in the dynamic link library VREXX.DLL. Thus, REXX procedures can load and call the VInit function, which sets up system resources and initializes the other VREXX external functions for access by REXX. The VExit function then frees up these system resources before the REXX procedure exits.

When VInit is called, it starts a copy of the new VREXX.EXE program and sets up a shared memory block to pass variables between the DLL and the program. The program creates an invisible control window, and waits for the VREXX external functions to post messages to the window. The control window then creates the windows, draws graphics, processes dialogs, etc.

Variables are shared between the DLL and the REXX environment through the shared variable pool. Stem variables are used to facilitate the use of REXX arrays by the user. Shared memory blocks and semaphores are used to pass data and synchronize between the DLL and the control window.

50 VREXX: Visual REXX for Presentation Manager

RELEASE NOTES AND COMMENTS

Version 1.0 (9/9/92) is the initial release.