R:BASE eXtreme 9.0

Function Index

The Programmer's Guide to Building
R:BASE Databases and Custom Applications

R:BASE Technologies, Inc.

R:BASE eXtreme 9.0

Function Index

by R:BASE Technologies, Inc.

Welcome to R:BASE eXtreme 9.0

R:BASE eXtreme 9.0 for Windows is a completely new relational database development environment that has been amplified to include the latest menu-driven features to database professionals. With the R:BASE engine established as the foundation of stability, the program interface and designers have been enhanced to fully exhibit a higher quality of development usability.

The "eXtreme" in the product's name signifies the cutting-edge development interface incorporated into the R:BASE program making database and application development more productive.

R:BASE eXtreme 9.0 Function Index

Printed: December 2009 in Murrysville, PA

First Edition

Table of Contents

Part I Function Index 2

1 Function Categories 2

2 A 3

ABS 3
ACOS 3
ADDDAY 3
ADDFRC 3
ADDHR 3
ADDMIN 3
ADDMON 3
ADDSEC 4
ADDYR 4
AINT 4
ANINT 4
ASIN 4
ATAN 4
ATAN2 4

3 B 5

BRND 5

4 C 5

CHAR 5
CHKCUR 5
CHKFILE 5
CHKFUNC 5
CHKKEY 6
CHKVAR 6
COS 6
COSH 6
CTR 6
CTXT 7
CVAL 7

AND 9
ANSI 9
AUTOCOMMIT 9
AUTODROP 9
AUTOSKIP 10
BELL 10
BLANK 10
BUILD 10
CASE 10
CLEAR 10
CLIPBOARDTEXT 10
COLOR 10
COMPUTER 11
CONNECTIONS 11
CURRDIR 11

CURRDRV 11
CURRENCY 11
CURRENTPRINTER 11
DATABASE 11
DATE 12
DATECENTURY 12
DATEFORMAT 12
DATESEQUENCE 12
DATEYEAR 12
DBPATH 12
DEBUG 12
DELIMIT 13
DRIVES 13
ECHO 13
EDITOR 13
EOFCHAR 13
EQNULL 13
ERROR 14
ERRORDETAIL 14
ERRORVARIABLE 14
ESCAPE 14
EXPLODE 14
FASTFK 14
FASTLOCK 14
FEEDBACK 14
FILES 15
FIXED 15
HEADINGS 15
IDQUOTES 15
INSERT 15
INTENSITY 15
INTERVAL 15
LASTBLOCKTABLE 15
LASTERROR 15
LAYOUT 16
LINEEND 16
LINES 16
LOOKUP 16
MANOPT 16
MANY 16
MAXTRANS 16
MDI 16
MESSAGES 17
MIRROR 17
MULTI 17
NAME 17
NETUSER 17
NOTE_PAD 17
NULL 17
OFFMESS 17
OLDLINE 17
ONELINE 18
PAGEMODE 18
PASSTHROUGH 18

PLATFORM 18
PLUS 18
POSFIXED 18
PORTS 18
PRINTERS 18
PRN_STATUS 19
PRN_ORIENTATION 19
PRN_SIZE 19
PRN_SOURCE 19
PRN_QUALITY 19
PRN_COPIES 19
PRN_COLORMODE 19
PRN_DUPLEXMODE 19
PRN_COLLATION 19
QUALCOLS 19
QUOTES 20
REFRESH 20
REVERSE 20
ROWLOCKS 20
RULES 20
SCRATCH 20
SCREENSIZE 20
SELMARGIN 20
SEMI 20
SERVER 20
SINGLE 20
SORT 21
SORTMENU 21
STATICDB 21
TIME 21
TIMEFORMAT 21
TIMESEQUENCE 21
TIMEOUT 21
TOLERANCE 21
TRACE 21
TRANSACT 21
USER 21
USERAPP 22
USERDOMAIN 22
USERID 22
VERIFY 22
VERSION 22
VERSIONBITS 22
VERSIONBUILD 22
VERSIONSYSTEM 23
WAIT 23
WALKMENU 23
WHILEOPT 23
WIDTH 23
WINBEEP 23
WINDOWSPRINTER 23
WRAP 23
WRITECHK 23
ZERO 23

ZOOMEDIT ... 24
CVTYPE ... 24
5 D ... 24
DATETIME ... 24
DELFUNC ... 25
DEXTRACT ... 25
DIM ... 25
DLCALL ... 25
DLFREE ... 30
DLLOAD ... 31
DNW ... 31
DWE ... 31
DWRD ... 31
6 E ... 32
ENVVAL ... 32
EXP ... 32
7 F ... 32
FILENAME ... 32
FINDFILE ... 32
FLOAT ... 32
FORMAT ... 33
Aligning Decimals ... 34
Capturing date and time using system variables ... 34
Formatting Currency ... 35
Formatting Text ... 35
Punctuating Long Numbers ... 35
FV1 ... 36
FV2 ... 36
8 G ... 36
GETDATE ... 36
GETKEY ... 36
GETVAL ... 37
CheckMessageStatus ... 37
GetDriveReady ... 37
GetIPAddress ... 37
GetLock ... 38
GetMACAddr ... 39
GetVolumeID ... 40
PlayAndExit ... 40
PlayAndWait ... 41
9 H ... 41
HTML ... 41
10 I ... 41
ICAP ... 41
ICAP1 ... 41
ICAP2 ... 41
ICHAR ... 41
IDAY ... 42
IDIM ... 42
IDOY ... 42
IDWK ... 42
IFEQ ... 42

IFEXISTS **43**
IFGT **43**
IFLT **43**
IFNULL **43**
IFRC **43**
IFWINDOW **43**
IHASH **44**
IHR **45**
ILY **45**
IMIN **45**
IMON **45**
INT **45**
ISALPHA **46**
ISDIGIT **46**
ISEC **46**
ISLOWER **46**
ISSPACE **46**
ISTAT **46**
CURRNUMALLOC 47
CURSORCOL 47
CURSORROW 47
DBSIZE 47
DISKSPACE 47
FORM_CONTROL_TYPE 48
FORM_DIRTY_FLAG 48
ISRUNTIME 48
LIMITNUMALLOC 48
MAXFREE 48
MAXNUMALLOC 49
MEMORY 49
MOUSECOL 49
MOUSEROW 49
PAGECOL 49
PAGEROW 49
RB1SIZE 49
RB2SIZE 50
RB3SIZE 50
RB4SIZE 50
RX1SIZE 50
RX2SIZE 50
RX3SIZE 51
RX4SIZE 51
TOTALALLOC 51
TOTALFREE 51
TOTALLOCKS 51
TOTALREADS 52
TOTALWRITES 52
ISTR **52**
ISUPPER **52**
ITEMCNT **52**
IWOY **53**
IYR **53**
IYR4 ..... **53**

11 J **54**

JDATE 54
12 L 54
LASTKEY 54
LAVG 55
LJS 55
LMAX 55
LMIN 55
LOG 56
LOG10 56
LTRIM 56
LUC 56
13 M 56
MOD 56
14 N 57
NEXT 57
NINT 57
15 P 57
PMT1 57
PMT2 57
PV1 58
PV2 58
16 R 58
RANDOM 58
RATE1 58
RATE2 58
RATE3 59
RDATE 59
RJS 59
ROUND 59
RTIME 60
RTRIM 60
RWP 60
17 S 61
SFIL 61
SGET 61
SIGN 61
SIN 61
SINH 61
SKEEP 61
SKEEPI 62
SLEN 62
SLOC 62
SLOCP 63
SMOVE 63
SOUNDEX 63
SPUT 63
SQRT 64
SRPL 64
SSTRIP 64
SSTRIPI 65
SSUB 65
STRIM 66

18 T 66

TAN 66
TANH 66
TDWK 66
TERM1 66
TERM2 66
TERM3 67
TEXTRACT 67
TMON 67
TRIM 67

19 U 67

UDF (User-Defined Functions) 67
ERRORLEVEL 69
Memory Issues 69
Returning Multiple Values 69
Sample UDF 69
ULC 78

20 W 78

WINUDF 78

Index 80

Part I

1 Function Index

1.1 Function Categories

Arithmetic and Mathematical Functions

ABS	BRND	DIM	EXP
LAVG	LMAX	LMIN	LOG
LOG10	MOD	RANDOM	ROUND
SIGN	SQRT		

Conversion Functions

AINT	ANINT	CHAR	CTXT
DWRD	FLOAT	HTML	ICHAR
IHASH	INT	NINT	SOUNDEX

Date and Time Functions

ADDDAY	ADDFRC	ADDHR	ADDMIN
ADDMON	ADDSEC	ADDYR	DATETIME
DEXTRACT	DNW	DWE	IDAY
IDIM	IDOY	IDWK	IFRC
IHR	ILY	IMIN	IMON
ISEC	IWOY	IYR	IYR4
JDATE	RDATE	RTIME	TDWK
TEXTRACT	TMON		

Financial Functions

FV1	FV2	PMT1	PMT2
PV1	PV2	RATE1	RATE2
RATE3	TERM1	TERM2	TERM3

Keyboard and Operating System Functions

ENVVAL	CVAL	CHKCUR	CHKKEY
CHKFILE	CHKVAR	CVTYPE	DELFUNC
DLCALL	DLFREE	DLLOAD	FILENAME
FINDFILE	IFWINDOW	ISTAT	GETKEY
LASTKEY			

Logical Functions

IFEQ	IFEXISTS	IFGT	IFLT
IFNULL			

String Manipulation Functions

CTR	FORMAT	ICAP	ICAP1
ICAP2	ISALPHA	ISDIGIT	ISLOWER
ISSPACE	ISTR	ISUPPER	ITEMCNT
LJS	LTRIM	LUC	RJS
RTRIM	SFIL	SGET	SKEEP
SKEEPI	SLEN	SLOC	SLOCP
SMOVE	SPUT	SRPL	SSTRIP
SSTRIPI	SSUB	STRIM	TRIM
ULC			

Trigonometric Functions

ACOS	ASIN	ATAN	ATAN2
COS	COSH	SIN	SINH
TAN	TANH		

Custom Functions

UDF	WINUDF

1.2 A

1.2.1 ABS

(ABS(*arg*))

Returns the absolute or positive value of *arg* (a value with a DOUBLE, REAL, NUMERIC, or INTEGER data type).

In the following example, the value of *vabs* is *2*.

```
SET VAR vnum = -2
SET VAR vabs = (ABS(.vnum))
```

1.2.2 ACOS

(ACOS(*arg*))

Computes the arccosine of *arg* where *arg* is in the range -1 to 1. The result is an angle in radians between 0 and pi (where pi = 3.14159265358979).

In the following example, the value of *vacos* is *2.094395*, the arccosine of -0.5.

```
SET VAR vacos = (ACOS(-0.5))
```

1.2.3 ADDDAY

(ADDDAY(*date,int*))

Adds the specified number of days to a date or datetime value. Functions that result in an invalid date, for example, February 30, will return NULL.

1.2.4 ADDFRC

(ADDFRC(*time,int*))

Adds the specified number of milliseconds to a time or datetime value.

1.2.5 ADDHR

(ADDHR(*time,int*))

Adds the specified number of hours to a time or datetime value.

1.2.6 ADDMIN

(ADDMIN(*time,int*))

Adds the specified number of minutes to a time or datetime value.

1.2.7 ADDMON

(ADDMON(*date,int*))

Adds the specified number of months to a date or datetime value.

1.2.8 ADDSEC

(ADDSEC(*time,int*))

Adds the specified number of seconds to a time or datetime value.

1.2.9 ADDYR

(ADDYR(*date,int*))

Adds the specified number of years to a date or datetime value.

1.2.10 AINT

(AINT(*arg*))

Truncates the decimal fraction, returning a whole number in the original REAL, NUMERIC, or DOUBLE data type.

In the following example, the value of *vaint* is *1*.

```
SET VAR vaint = (AINT(1.8))
```

1.2.11 ANINT

(ANINT(*arg*))

Rounds the decimal fraction to the nearest integer, returning a whole number in the original REAL, NUMERIC, or DOUBLE data type.

In the following example, the value of *vanint1* is *3.0* and the value of *vanint2* is *4.0*.

```
SET VAR vanint1 = (ANINT(2.6))
SET VAR vanint2 = (ANINT(4.45))
```

1.2.12 ASIN

(ASIN(*arg*))

Computes the arcsine of *arg* where *arg* is in the range -1 to 1. The result is an angle in radians between -pi/2 and pi/2.

In the following example, the value of *vasin* is *-0.5236* (-pi/6 radians)

```
SET VAR vasin = (ASIN(-0.5))
```

1.2.13 ATAN

(ATAN(*arg*))

Computes the arctangent of *arg* in radians where *arg* is any amount. The result is an angle in radians between -pi/2 and pi/2.

In the following example, the value of *vatan* is *0.7854* (pi/4 radians).

```
SET VAR vatan = (ATAN(1))
```

1.2.14 ATAN2

(ATAN2(*x,y*))

Computes the arctangent of *x/y*. The result is the angle in radians between -pi/2 and pi/2.

In the following example, the value of *vatan2* is *0.7854*.

```
SET VAR vatan2 = (ATAN2(1,1))
```

1.3 B

1.3.1 BRND

(BRND(*arg1,arg2,arg3*))

Rounds REAL, DOUBLE, or CURRENCY data to a specific number of decimal places and allows specification of the number of significant digits to return. *Arg1* is the value to be rounded. *Arg2* is the number of significant digits to return, and *arg3* is the precision The precision is specified as a decimal number, for example, .01 rounds to two decimal places.

In the following example, the value of *vresult* is 1234.57.

```
SET VAR vresult = (BRND(1234.5678342,8,.01))
```

1.4 C

1.4.1 CHAR

(CHAR(*integer*))

Converts an ASCII integer value to its corresponding character. This is not the same as the CHAR data type.

In the following example, the value of *vchar1* is *A* and the value of *vchar2* is *a*.

```
SET VAR vchar1 = (CHAR(65))
SET VAR vchar2 = (CHAR(97))
```

See also:

ASCII Character Chart

1.4.2 CHKCUR

(CHKCUR('*cursorname*'))

Checks to see if a cursor is declared and is not dropped. The function returns an integer value of 1 if the name exists and is not dropped, and 0 if it is not declared or dropped.

1.4.3 CHKFILE

(CHKFILE('*filespec*'))

Checks to see if a file or folder name exists. If no path is specified, the function checks for the file or folder name in the current directory. Otherwise, the function checks for the file or folder name in the specified location.

The function returns a 1 if the file or folder name is found, and 0 if it is not found. Wildcards in the filename will produce unpredicatable results.

1.4.4 CHKFUNC

(CHKFUNC('function_name'))

Checks to see if a DLL function exists or not. If the DLL function exists, a 1 is returned. If the DLL

function does not exist, a 0 is returned.

Example:

```
SET VAR v1 = (CHKFUNC('FunctionName'))
```

See Also:

DELFUNC
DLLCALL
DLLOAD
DLFREE
LIST FUNCTIONS

1.4.5 CHKKEY

(CHKKEY(0))

Returns an integer value of *1* if there are keystrokes available in the type-ahead buffer. Returns *0* if no keystrokes are available. Use CHKKEY before GETKEY to determine if a key is available.

CHKKEY does nothing with the zero that you enter in parentheses; CHKKEY returns a value without receiving one.

1.4.6 CHKVAR

(CHKVAR('var*name*'))

Checks to see if a variable name exists. The function returns an integer value of 1 if the variable name exists or declared, and 0 if it is not found or declared.

1.4.7 COS

(COS(*angle*))

Returns the trigonometric cosine of *angle*. The result is between -1 and 1.

In the following example, the value of *vcos* is *0.5002*.

```
SET VAR vcos = (COS(1.047))
```

1.4.8 COSH

(COSH(*angle*))

Returns the hyperbolic cosine of *angle*.

In the following example, the value of *vcosh* is *1.6000*.

```
SET VAR vcosh = (COSH(1.047))
```

1.4.9 CTR

(CTR(*text*,*width*))

Centers *text* in *width* characters, returning a text string.

In the following example, the value of *vctr* is *ABCD* .

The text string is centered in a 10-character field.

```
SET VAR vctr = (CTR('ABCD',10))
```

1.4.10 CTXT

(CTXT(*arg*))

Converts an internal value, returning a text string.

In the following example, the value of *vctxt1* is the value *37.6* and is a REAL data type. The value of *vctxt2* is the value *37.6* and is a TEXT data type. If you attempt to use *vctxt2* in a mathematical function, it will fail.

```
SET VAR vctxt2 = (CTXT(.vctxt1))
```

1.4.11 CVAL

(CVAL('*showkeyword*'))

Returns the current value or setting of *'showkeyword'*. You must either enclose the SHOW keyword in quotation marks or use a dot variable that has a TEXT data type to which you have assigned the SHOW keyword. You can use all SHOW keywords with CVAL, as well as DATABASE, DBPATH, CURRDIR.

The following keywords can be used for (CVAL('keyword')):

- AND
- ANSI
- AUTOCOMMIT
- AUTODROP
- AUTOSKIP
- BELL
- BLANK
- BUILD
- CASE
- CLEAR
- CLIPBOARDTEXT
- COLOR
- COMPUTER
- CONNECTIONS
- CURRDIR
- CURRDRV
- CURRENCY
- CURRENTPRINTER
- DATABASE
- DATE
- DATE_CENTURY
- DATE_FORMAT
- DATE_SEQUENCE
- DATE_YEAR
- DBPATH
- DEBUG
- DELIMIT
- DRIVES
- ECHO
- EDITOR
- EOFCHAR
- EQNULL
- ERROR
- ERROR_DETAIL
- ERROR_VARIABLE
- ESCAPE
- EXPLODE
- FASTFK
- FASTLOCK

- FEEDBACK
- FILES
- FIXED
- HEADINGS
- IDQUOTES
- INSERT
- INTENSITY
- INTERVAL
- LAST_BLOCK_TABLE
- LAST_ERROR
- LAYOUT
- LINEEND
- LINES
- LOOKUP
- MANOPT
- MANY
- MAXTRANS
- MDI
- MESSAGES
- MIRROR
- MULTI
- NAME
- NETUSER
- NOTE_PAD
- NULL
- OLDLINE
- ONELINE
- PAGEMODE
- PASSTHROUGH
- PLATFORM
- PLUS
- POSFIXED
- PORTS
- PRINTERS
- PRN_Status
- PRN_Orientation
- PRN_Size
- PRN_Source
- PRN_Quality
- PRN_Copies
- PRN_ColorMode
- PRN_DuplexMode
- PRN_Collation
- QUALCOLS
- QUOTES
- REFRESH
- REVERSE
- ROWLOCKS
- RULES
- SCRATCH
- SCREENSIZE
- SELMARGIN
- SEMI
- SERVER
- SINGLE
- SORT
- SORTMENU
- STATICDB
- TIME
- TIME_FORMAT
- TIME_SEQUENCE
- TIMEOUT
- TOLERANCE
- TRACE
- TRANSACT

- USER
- USERAPP
- USERDOMAIN
- USERID
- VERIFY
- VERSION
- VERSION_BITS
- VERSION_BUILD
- VERSION_SYSTEM
- WAIT
- WALKMENU
- WHILEOPT
- WIDTH
- WINBEEP
- WINDOWSPRINTER
- WRAP
- WRITECHK
- ZERO
- ZOOMEDIT

Examples:

In the following example, the value of *vcval* is *OFF* if the value of MULTI is set to off.

```
SET VAR vcval = (CVAL('MULTI'))
```

In the following example, the user keyword is loaded into a variable and then the variable is used in the CVAL function. It returns the current user identifier.

```
SET VAR vword text = 'USER'
SET VAR vuser = (CVAL(.vword))
```

For more information about SHOW keywords, see SHOW.

1.4.11.1 AND

(CVAL('AND'))

Returns the status of the AND command parameter (ON/OFF).

1.4.11.2 ANSI

(CVAL('ANSI'))

Returns the status of the ANSI command parameter (ON/OFF).

1.4.11.3 AUTOCOMMIT

(CVAL('AUTOCOMMIT'))

Returns the status of the AUTOCOMMIT command parameter (ON/OFF).

1.4.11.4 AUTODROP

(CVAL('AUTODROP'))

Returns the status of the AUTODROP display control (ON/OFF).

1.4.11.5 AUTOSKIP

(CVAL('AUTOSKIP'))

Returns the status of the AUTOSKIP command parameter (ON/OFF).

1.4.11.6 BELL

(CVAL('BELL'))

Returns the status of the BELL command parameter (ON/OFF).

1.4.11.7 BLANK

(CVAL('BLANK'))

Returns an empty variable value.

BLANK is similar to a null value, but is not based on your database NULL setting.

1.4.11.8 BUILD

(CVAL('BUILD'))

The BUILD parameter of the SHOW command can be used to determine the exact build number of the R:BASE Front-End GUI. This can also be used with CVAL to store the build number information in a variable.

See also:
(CVAL('VERSION')) to determine the version as well as build number of R:BASE Engine
(CVAL('VERSION_BUILD')) to determine only the build number of the R:BASE Engine

1.4.11.9 CASE

(CVAL('CASE'))

Returns the status of the CASE command parameter (ON/OFF).

1.4.11.10 CLEAR

(CVAL('CLEAR'))

Returns the status of the CLEAR command parameter (ON/OFF).

1.4.11.11 CLIPBOARDTEXT

(CVAL('CLIPBOARDTEXT'))

Returns the contents of the Windows Clipboard.

1.4.11.12 COLOR

(CVAL('COLOR'))

Returns the value for the foreground and background COLOR of data displays.

This CVAL parameter is specific to R:BASE for DOS.

1.4.11.13 COMPUTER

(CVAL('COMPUTER'))

Returns the name of your computer.

Example:

```
SET VAR vComp = (CVAL('COMPUTER'))
```

1.4.11.14 CONNECTIONS

(CVAL('CONNECTIONS'))

Returns the number of users currently connected to the current database or 0 if not connected. In the event of a non-graceful disconnect R:BASE will not be able to decrement the connections count. This can occur if the network session terminates unexpectedly, or if the users operating system crashes. If this happens the count will be innacurate until all users disconnect from the database. This works because the last session of R:BASE will be able to tell that NO users are connected and will reset the count to zero. Unfortunatly, due to file system limitations, R:BASE is only able to tell if a database is open by any users or not at all, and is not able to tell, except by this count, how many users are connected.

1.4.11.15 CURRDIR

(CVAL('CURRDIR'))

Returns the current directory. This is the same information returned by using the CD command at the R> Prompt.

1.4.11.16 CURRDRV

(CVAL('CURRDRV'))

Returns the current drive. The drive is in X: format with the drive letter followed by a colon.

1.4.11.17 CURRENCY

(CVAL('CURRENCY'))

The CURRENCY data type holds monetary values of up to 23 digits represented in the currency format, established using SET CURRENCY. Amounts are in the range ±$99,999,999,999,999.99. Commas or the current delimiter can be used. If no decimal point is included, .00 is assumed.

Data is stored as two long integer values, reserving four bytes of internal storage.

1.4.11.18 CURRENTPRINTER

(CVAL('CURRENTPRINTER'))

This parameter returns the current printer for the R:BASE session.

1.4.11.19 DATABASE

(CVAL('DATABASE'))

Returns a text string containing the current connected database or NULL if the user is not connected to a database. This can be used to ensure that the user is connected before trying to execute some code. The example below shows how this might work.

```
SET VAR vDB = (CVAL('database'))
```

```
IF vDB IS NULL THEN
    CONN MyDB
ENDIF

SET VAR vDB = (CVAL('database'))
IF vDB IS NULL THEN
    PAUSE 2 USING 'MyDB is currently unavailable'
ENDIF
```

The check is repeated to ensure that the attempt to connect to the database was successful before continuing with the command file.

1.4.11.20 DATE

(CVAL('DATE'))

Returns the DATE format of the currently connected database.

1.4.11.21 DATE CENTURY

(CVAL('DATE CENTURY'))

Returns the default DATE CENTURY of the currently connected database.

1.4.11.22 DATE FORMAT

(CVAL('DATE FORMAT'))

Returns the default DATE FORMAT of the currently connected database.

1.4.11.23 DATE SEQUENCE

(CVAL('DATE SEQUENCE'))

Returns the default DATE SEQUENCE of the currently connected database.

1.4.11.24 DATE YEAR

(CVAL('DATE YEAR'))

Returns the default DATE YEAR of the currently connected database.

1.4.11.25 DBPATH

(CVAL('DBPATH'))

DBPATH returns the full path to the current connected database.

1.4.11.26 DEBUG

(CVAL('DEBUG'))

Returns the status of the DEBUG command parameter(ON/OFF).

1.4.11.27 DELIMIT

(CVAL('DELIMIT'))

Returns the value of the DELIMIT character setting.

1.4.11.28 DRIVES

(CVAL('DRIVES'))

Returns the list of all currently available drives

Example:

```
SET VAR vDrives = (CVAL('Drives'))
```

vDrives will include the list of ALL drives!

Result: aCDeF

In that list of drives, drives with CAPITAL letters, such as C,D or F would be the hard disk drive, and drives with lower case letters would be removable drives, such as a or e.

a = floppy disk drive
C = Hard Disk
D = Hard Disk/CD-ROM/DVD
e = zip disk
F = Hard Disk/CD-ROM/DVD or even mapped network drive

All hard drives, including CD-ROM/DVD and network mapped drives would be CAPITAL letters.

All removable drives, including floppy disk drives and zip drives would be lower case letters.

1.4.11.29 ECHO

(CVAL('ECHO'))

Returns the status of the ECHO command parameter (ON/OFF).

1.4.11.30 EDITOR

(CVAL('EDITOR'))

Returns the current text editor used by R:BASE.

The default editor is RBEDIT.

1.4.11.31 EOFCHAR

(CVAL('EOFCHAR'))

Returns the status of the EOFCHAR command parameter (ON/OFF).

1.4.11.32 EQNULL

(CVAL('EQNULL'))

Returns the status of the EQNULL command parameter (ON/OFF).

1.4.11.33 ERROR

(CVAL('ERROR'))

Returns the status of the ERROR MESSAGES display control (ON/OFF).

1.4.11.34 ERROR DETAIL

(CVAL('ERROR DETAIL'))

When an -ERROR- occurs now you can track additional information including the name of the file being run and the byte offset within the file. If the thing being run is a procedure it saves information on the select used to fetch the procedure too. We have implemented a new method of simple "stack" to keep the last three errors. To see the information tracked use this new CVAL option.

```
SET VAR vError = (CVAL('ERR0R DETAIL'))
```

Each time you call this particular CVAL function the stack pointer decrements so successive calls allow you to see all three errors. Call it a fourth time and you will see the first one again, etc.

1.4.11.35 ERROR VARIABLE

(CVAL('ERROR VARIABLE'))

Returns the value of the current ERROR VARIABLE.

1.4.11.36 ESCAPE

(CVAL('ESCAPE'))

Returns the status of the ESCAPE command parameter (ON/OFF).

1.4.11.37 EXPLODE

(CVAL('EXPLODE'))

Returns the status of the EXPLODE command parameter (ON/OFF).

Used with R:BASE for DOS only. When EXPLODE is set on, dialog boxes are displayed in full size instantly. When EXPLODE is set off, dialog boxes are displayed in an expanding fashion from the center.

1.4.11.38 FASTFK

(CVAL('FASTFK'))

Returns the status of the FASTFK command parameter (ON/OFF).

1.4.11.39 FASTLOCK

(CVAL('FASTLOCK'))

Returns the status of the FASTLOCK command parameter (ON/OFF).

1.4.11.40 FEEDBACK

(CVAL('FEEDBACK'))

Returns the status of the FEEDBACK command parameter (ON/OFF).

1.4.11.41 FILES

(CVAL('FILES'))

Returns the value of the FILES operating condition.

1.4.11.42 FIXED

(CVAL('FIXED'))

Returns the status of the FIXED command parameter (ON/OFF).

1.4.11.43 HEADINGS

(CVAL('HEADINGS'))

Returns the status of the HEADINGS display control (ON/OFF).

1.4.11.44 IDQUOTES

(CVAL('IDQUOTES'))

Returns the value of the IDQUOTES character setting.

1.4.11.45 INSERT

(CVAL('INSERT'))

Returns the status of INSERT display control (ON/OFF).

1.4.11.46 INTENSITY

(CVAL('INTENSITY'))

Returns the status of INTENSITY display control (ON/OFF).

Used with R:BASE 6.5++ for Windows only.

1.4.11.47 INTERVAL

(CVAL('INTERVAL'))

Returns value of the INTERVAL command parameter (ON/OFF).

1.4.11.48 LAST BLOCK TABLE

(CVAL('LastBlockTable'))

Tells you which table holds the last block in File 2.

If the last block table contains significant dead space, packing it should free up space at the end of File 2.

1.4.11.49 LAST ERROR

(CVAL('LAST ERROR'))

This is an alternative to the current ERROR VARIABLE system of -ERROR- trapping. Something more along the lines of an error variable which must be EXPLICITLY CLEARED.

Why? The current system does not allow error trapping in nested code segments or large blocks of code. Furthermore, since the -ERROR variable is automatically cleared after each command, it requires error trapping logic immediately after each command.

1.4.11.50 LAYOUT

(CVAL('LAYOUT'))

Returns the status of the LAYOUT display control (ON/OFF).

1.4.11.51 LINEEND

(CVAL('LINEEND'))

Returns the value of the LINEEND character setting.

1.4.11.52 LINES

(CVAL('LINES'))

Returns value of the LINES display control.

1.4.11.53 LOOKUP

(CVAL('LOOKUP'))

Returns the value of the LOOKUP command parameter (ON/OFF).

1.4.11.54 MANOPT

(CVAL('MANOPT'))

Returns value of the MANOPT command parameter (ON/OFF).

1.4.11.55 MANY

(CVAL('MANY'))

Returns the value of the MANY character setting.

1.4.11.56 MAXTRANS

(CVAL('MAXTRANS'))

Returns value of the MAXTRANS operating condition.

1.4.11.57 MDI

(CVAL('MDI'))

Returns the value of the MDI operating condition (ON/OFF).

1.4.11.58 MESSAGES

(CVAL('MESSAGES'))

Returns the status of the MESSAGES display control (ON/OFF).

1.4.11.59 MIRROR

(CVAL('MIRROR'))

Returns the value of the MIRROR operating condition.

1.4.11.60 MULTI

(CVAL('MULTI'))

Returns the value of the MULTI operating condition (ON/OFF).

1.4.11.61 NAME

(CVAL('NAME'))

Returns the currently logged-in R:BASE user NAME.

1.4.11.62 NETUSER

(CVAL('NETUSER'))

Returns the currently logged-in network user name.

1.4.11.63 NOTE_PAD

(CVAL('NOTE_PAD'))

Returns value of the NOTE_PAD operating condition.

1.4.11.64 NULL

(CVAL('NULL'))

Returns the value of the NULL character setting.

1.4.11.65 OFFMESS

(CVAL('OFFMESS'))

Return a text string, separated by a comma, with a list of all turned off -ERROR- message numbers in a current R:BASE session.

See SET ERROR MESSAGE.

1.4.11.66 OLDLINE

(CVAL('OLDLINE'))

Returns the value of the OLDLINE operating condition (ON/OFF).

Used with R:BASE 6.5++ for Windows only. Allows the ability to have presentation objects, such as lines, cross multiple sections in Reports.

1.4.11.67 ONELINE

(CVAL('ONELINE'))

Returns the value of the ONELINE operating condition (ON/OFF).

Used with R:BASE for DOS only. When set to ON NOTE and TEXT fields will never wrap to the next line in Reports and SELECTS. Instead they will be truncated at the end of the column.

1.4.11.68 PAGEMODE

(CVAL('PAGEMODE'))

Returns the value of the PAGEMODE operating condition (ON/OFF).

1.4.11.69 PASSTHROUGH

(CVAL('PASSTHROUGH'))

Returns the value of the PASSTHROUGH operating condition (ON/OFF).

1.4.11.70 PLATFORM

(CVAL('PLATFORM'))

Returns 16 or 32 and WIN or DOS based on what R:BASE can determine about the Operating System. This may be complicated by using the Windows options which hide Windows from DOS programs.

1.4.11.71 PLUS

(CVAL('PLUS'))

Returns the value of the PLUS character setting.

1.4.11.72 POSFIXED

(CVAL('POSFIXED'))

Returns the value of the POSFIXED operating condition.

1.4.11.73 PORTS

(CVAL('PORTS'))

Returns the list of all available printer ports, separated by comma, on that workstation.

Example:

```
SET VAR vAvailablePorts = (CVAL('PORTS'))
SHOW VARIABLE vAvailablePorts
```

Will return the text string with a list of all printer ports that are available on that workstation. Each item in the list will be separated by comma (or the database character settings for comma).

1.4.11.74 PRINTERS

(CVAL('PRINTERS'))

Returns the list of all installed printers.

1.4.11.75 PRN_STATUS

(CVAL('PRN_STATUS'))

Captures the printer status.

1.4.11.76 PRN_ORIENTATION

(CVAL('PRN_ORIENTATION'))

Captures the printer orientation.

1.4.11.77 PRN_SIZE

(CVAL('PRN_SIZE'))

Captures the printer paper size.

1.4.11.78 PRN_SOURCE

(CVAL('PRN_SOURCE'))

Captures the printer paper source.

1.4.11.79 PRN_QUALITY

(CVAL('PRN_QUALITY'))

Captures the printer print quality (DPI).

1.4.11.80 PRN_COPIES

(CVAL('PRN_COPIES'))

Captures the printer copy count.

1.4.11.81 PRN_COLORMODE

(CVAL('PRN_COLORMODE'))

Captures the printer color mode.

1.4.11.82 PRN_DUPLEXMODE

(CVAL('PRN_DUPLEXMODE'))

Captures the printer duplex mode.

1.4.11.83 PRN_COLLATION

(CVAL('PRN_COLLATION'))

Captures the printer collation.

1.4.11.84 QUALCOLS

(CVAL('QUALCOLS'))

Returns the value of the QUALCOLS operating condition.

1.4.11.85 QUOTES

(CVAL('QUOTES'))

Returns the value of the QUOTES character setting.

1.4.11.86 REFRESH

(CVAL('REFRESH'))

Returns the value of the REFRESH operating condition.

1.4.11.87 REVERSE

(CVAL('REVERSE'))

Returns the value of the REVERSE operating condition.

1.4.11.88 ROWLOCKS

(CVAL('ROWLOCKS'))

Returns the value of the ROWLOCKS operating condition.

1.4.11.89 RULES

(CVAL('RULES'))

Returns the value of the RULES operating condition.

1.4.11.90 SCRATCH

(CVAL('SCRATCH'))

Returns the value of the SCRATCH operating condition.

1.4.11.91 SCREENSIZE

(CVAL('SCREENSIZE'))

Returns the Screen Area in Pixels

1.4.11.92 SELMARGIN

(CVAL('SELMARGIN'))

Returns the value of the SELMARGIN operating condition.

1.4.11.93 SEMI

(CVAL('SEMI'))

Returns the value of the SEMI operating condition.

1.4.11.94 SERVER

(CVAL('SERVER'))

Returns the value of the SERVER operating condition.

1.4.11.95 SINGLE

(CVAL('SINGLE'))

Returns the value of the SINGLE operating condition.

1.4.11.96 SORT

(CVAL('SORT'))

Returns the value of the SORT operating condition.

1.4.11.97 SORTMENU

(CVAL('SORTMENU'))

Returns the value of the SORTMENU operating condition.

1.4.11.98 STATICDB

(CVAL('STATICDB'))

Returns the value of the STATICDB operating condition.

1.4.11.99 TIME

(CVAL('TIME'))

Returns the TIME format of the currently connected database.

1.4.11.100 TIME FORMAT

(CVAL('TIME FORMAT'))

Returns the default TIME FORMAT of the currently connected database.

1.4.11.101 TIME SEQUENCE

(CVAL('TIME SEQUENCE'))

Returns the default TIME SEQUENCE of the currently connected database.

1.4.11.102 TIMEOUT

(CVAL('TIMEOUT'))

Returns the value of the TIMEOUT operating condition.

1.4.11.103 TOLERANCE

(CVAL('TOLERANCE'))

Returns the value of the TOLERANCE operating condition.

1.4.11.104 TRACE

(CVAL('TRACE'))

Returns the status of the TRACE command parameter (ON/OFF).

1.4.11.105 TRANSACT

(CVAL('TRANSACT'))

Returns the value of the TRANSACT operating condition.

1.4.11.106 USER

(CVAL('USER'))

Returns the R:BASE USER identifier.

1.4.11.107 USERAPP

(CVAL('USERAPP'))

Returns the file extensions to be displayed in the Object Manager.

Used with R:BASE 6.5++ for Windows only. You can specify which files display in the Object Manager after you click the Apps tab. Your choices are saved in the RBASE.CFG file. You can specify a maximum of three extensions.

1.4.11.108 USERDOMAIN

(CVAL('USERDOMAIN'))

Returns the Name of Logged-In Network Domain.

1.4.11.109 USERID

(CVAL('USERID'))

Returns the Windows User ID.

1.4.11.110 VERIFY

(CVAL('VERIFY'))

Returns the value of the VERIFY operating condition.

1.4.11.111 VERSION

(CVAL('VERSION'))

The VERSION parameter of the SHOW command can be used to determine the exact version as well as the build number of the R:BASE Engine. This can also be used with CVAL to store the version as well as build number information in a variable.

See also:
(CVAL('BUILD')) to determine the exact build number of the R:BASE Front-End GUI
(CVAL('VERSION BUILD')) to determine only the build number of the R:BASE Engine

1.4.11.112 VERSION BITS

(CVAL('VERSION BITS')

Returns 16 or 32. At this time there are no versions of R:BASE that we can recommend running on 16 bit systems and as such this will always return 32.

1.4.11.113 VERSION BUILD

(CVAL('VERSION BUILD'))

Returns only the current build number of the R:BASE Engine. This can be useful in determining which features are available to you. This can also be used with CVAL to store the build number build number of the R:BASE Engine in a variable.

See also:
(CVAL('BUILD')) to determine the exact build number of the R:BASE Front-End GUI
(CVAL('VERSION')) to determine the version as well as build number of R:BASE Engine

1.4.11.114 VERSION SYSTEM

(CVAL('VERSION SYSTEM'))

Returns the value WIN or DOS.

1.4.11.115 WAIT

(CVAL('WAIT'))

Returns the value of the WAIT operating condition.

1.4.11.116 WALKMENU

(CVAL('WALKMENU'))

Returns the value of the WALKMENU operating condition.

1.4.11.117 WHILEOPT

(CVAL('WHILEOPT'))

Returns the value of the WHILEOPT operating condition (ON/OFF).

1.4.11.118 WIDTH

(CVAL('WIDTH'))

Returns the value of the WIDTH operating condition.

1.4.11.119 WINBEEP

(CVAL('WINBEEP'))

Returns the value of the WINBEEP operating condition.

1.4.11.120 WINDOWSPRINTER

(CVAL('WINDOWSPRINTER'))

This parameter returns the windows default printer.

1.4.11.121 WRAP

(CVAL('WRAP'))

Returns the value of the WRAP operating condition (ON/OFF).

1.4.11.122 WRITECHK

(CVAL('WRITECHK'))

Returns the value of the WRITECHK operating condition (ON/OFF).

1.4.11.123 ZERO

(CVAL('ZERO'))

Returns the value of the ZERO operating condition (ON/OFF).

1.4.11.12 ZOOMEDIT

(CVAL('ZOOMEDIT'))

Returns the value of the ZOOMEDIT operating condition (ON/OFF).

1.4.12 CVTYPE

(CVTYPE('colvarname',flag))

Returns the data type for a given column or variable name.

To return the data type for a given column, a zero "0" flag must be used. To return the data type for a given variable, the one "1" flag must be used.

After connecting to ConComp or RRBYW14, try the following two examples at the R> Prompt:

Example 01.

```
SET VAR vCustIDType TEXT = (CVTYPE('CustID',0))
SET VAR vEmpCity TEXT = (CVTYPE('EmpCity',0))

SHOW VARIABLES

vCustIDType = INTEGER TEXT
vEmpCity = TEXT,16 TEXT
```

Example 02.

```
SET VAR vCustIDType TEXT = (CVTYPE('CustID',0))
SET VAR vVarInquiry TEXT = (CVTYPE('vCustIDType',1))

SHOW VARIABLES

vCustIDType = INTEGER TEXT
vVarInquiry = TEXT,7 TEXT
```

NOTES:

- When using the zero flag to return column data types, you must be connected to a database.
- When returning a TEXT data type, the length is included and is separated with a comma. When returning a NUMERIC data type, the precision and scale are separated with commas.

1.5 D

1.5.1 DATETIME

(DATETIME(*date,time*))

Concatenates date and time variables or constants.

The following expression concatenates the #DATE and #TIME values into a variable (*vdatetime*) that has a DATETIME data type.

```
SET VAR vdatetime=(DATETIME(.#DATE,.#TIME))
```

1.5.2 DELFUNC

(DELFUNC('function_name'))

Deletes a declared DLL function. If the DLL function is successfully deleted, a 1 is returned. If the DLL function is not deleted, a 0 is returned.

Example:

```
SET VAR v1 = (DELFUNC('FunctionName'))
```

See Also:

CHKFUNC
DLLCALL
DLLOAD
DLFREE
LIST FUNCTIONS

1.5.3 DEXTRACT

(DEXTRACT(*datetime*))

Returns the date portion of a value that has a DATETIME data type.

In the following example, the value of *vdextract* is *06/12/93*.

```
SET VAR vdextract = (DEXTRACT('06/12/93 12:15:30.123'))
```

1.5.4 DIM

(DIM(*arg1*,*arg2*))

Returns the positive difference between *arg1* and *arg2*. *Arg1* must be a value with a DOUBLE, REAL, NUMERIC, or INTEGER data type. *Arg2* must be any value with a DOUBLE, REAL, NUMERIC, or INTEGER data type other than 0. The result is always positive or zero. If the result is less than or equal to zero, DIM returns *0*.

In the following example, the value of *vdim1* is *0*; *vdim2* is *2*; *vdim3* is *2*; *vdim4* is *0*.

```
SET VAR vdim1 = (DIM(2,4))
SET VAR vdim2 = (DIM(4,2))
SET VAR vdim3 = (DIM(-2,-4))
SET VAR vdim4 = (DIM(-4,-2))
```

1.5.5 DLCALL

The DLCALL Function calls any Windows dynamic link library (DLL) and loads it into memory for use with R:BASE.

Syntax for External DLL:

(DLCALL('libraryname.ext', 'FunctionOrProcedureName', [arg],[arg],[.....]))

Syntax for Windows API:

(DLCALL('libraryname', 'FunctionOrProcedureName', [arg],[arg],[.....]))

External DLLs must have the file name listed with the extension, such as 'MyLibrary.dll'.

Windows APIs require only the name of the Library, without the extension, such as: 'Kernel32' or 'User32'.

The DLLs must be created as Standard Windows 32-bit DLLs. Any number of functions or procedures can be used in a single DLL. Functions or Procedures to be used with DLCALL must be Exported in the DLL. No special code is necessary in the DLL for it to be used by R:BASE.

DLL Location:

DLLs can be located in the Legal Windows Search Path, and If elsewhere, then specify the FullPathName in DLLoad.

Search Path Used by Windows to Locate a DLL

With both implicit and explicit linking, Windows first searches for "known DLLs", such as Kernel32.dll and User32.dll. Windows then searches for the DLLs in the following sequence:

1. The directory where the executable module for the current process is located.
2. The current directory.
3. The Windows system directory. The GetSystemDirectory function retrieves the path of this directory.
4. The Windows directory. The GetWindowsDirectory function retrieves the path of this directory.
5. The directories listed in the PATH environment variable.

When or If DLLOAD is Used:

DLLOAD may be called "anytime" during the Session to Load the Library into Memory OR as a subsequent call to determine if the Library is Loaded.

In any event, DLLOAD is called internally when the Library is first referenced by DLCALL, and THEN the Library remains in memory until either the R:BASE session is ended OR DLFREE is called.

DataType Rules:

DataTypes in the functions must be of the Same Storage Size as the corresponding RBase DataTypes.

Example: 32-bit Win32 Integer = 4 bytes of storage vs 32-bit R:BASE Integer = 4 bytes of storage

Declaration Logic:

Calls to a function OR procedure from ANY DLL must be DECLARED at Least ONCE in the Session in which they will be referenced.

Two Calling Conventions are supported, using the STDCALL and CDECL Keyword in the Declaration.

STDCALL

Function Declaration using STDCALL:

```
STDCALL FUNCTION 'functionName' (PTR VARCHAR (SIZE), ..... ) : VARCHAR (SIZE)
STDCALL FUNCTION 'functionName' ALIAS 'functionAliasName' (PTR TEXT (SIZE), ..... ) : TEXT (SIZE)
```

Procedure Declaration using STDCALL:

```
STDCALL VOID FunctionOrProcedureThatHasNoReturnValue (PTR TEXT (SIZE))
```

Windows API Declaration using STDCALL:

STDCALL FUNCTION 'GetCurrentDirectoryA' ALIAS 'GetCurrentDir' (PTR TEXT, INTEGER) : INTEGER

CDECL

Function Declaration using CDECL:

CDECL FUNCTION 'functionName' (INTEGER) : INTEGER
CDECL FUNCTION 'functionName' ALIAS 'functionAliasName' (PTR DOUBLE) : DOUBLE

Procedure Declaration using CDECL

CDECL VOID FunctionOrProcedureThatHasNoReturnValue (PTR TEXT (SIZE))

Parameters

'SIZE' applies to TEXT and VARCHAR DataTypes.

Important Notes:

A. Parameters in the Declaration are in the REVERSE order from the Actual Function or Procedure in the DLL.
Parameters can be 0 to n Parameters of any legal R:BASE Datatype.

B. Function Names ARE CASE SENSITIVE in the DECLARATION ONLY.
The Case must match the casing used in the DLL.
Case is INSENSITIVE when used in DLCALL.

Remarks:

- For best results, TEXT and VARCHAR data should be passed with the PTR (Pointer) Attribute and ANY VARCHAR data type Larger than 32K as a Parameter, MUST be passed as PTR.
- VARCHAR datatype as a Return Value restricted to 32K, but Any SIZE up to 256MB can be passed as a Pointer to the RBase Variable. Modification of the Data Passed as Pointer must be on the data pointed to.
- It is important that the SIZE parameter on TEXT and VARCHAR be Specified to avoid creating excess buffer space that has to be created from the Declaration.

For Example:

If the Function is Declared like this:
```
STDCALL function 'somefunction' ( ptr varchar (nn)) : integer
```
Then *nn* <= 256Mb.

If the Function is Declared like this:
STDCALL function 'somefunction' (ptr varchar) : integer
Then because SIZE is omitted, the buffer for VARCHAR will have default SIZE = 256MB.
*** AVOID THIS UNLESS THAT IS THE ACTUAL SIZE OF THE DATA TO BE PASSED!**

If the Function is Declared like this:
STDCALL function 'somefunction' (integer) : varchar(nn)
Then because SIZE is Specified, nn bytes <= 32K will be returned.

If the Function is Declared like this:
STDCALL function 'somefunction' (integer) : varchar

Then because SIZE is omitted, the buffer for VARCHAR will have default SIZE = 32K.

When SIZE is Specified, the data passed as parameter or as return value, if Greater than the SIZE, will be truncated to SIZE.

Example of a DLL created in Delphi exporting three functions:

```
// Begin Dll Code
library DemoLib;

uses
  SysUtils, Classes;

{$R *.res}

function MultInt (NumIN : Integer) : Integer; stdcall;
begin
  Result := (NumIN * 2);
end;

function MultDbl (NumDbl : Double) : Double; stdcall;
begin
  Result := (NumDbl * 2);
end;

procedure LCaseByREF(DataIN : PChar); stdcall;
begin
 ansiStrLower(DataIN);
end;

function LCaseByVAL (DataIN : PChar) : PChar; Stdcall;
begin
  Result := ansiStrLower(DataIN);
end;

Exports MultInt, LCaseByREF, LCaseByVAL;

begin

end.
// End Dll Code
```

Example Usage From Within R:BASE:

```
-- BEGIN Demo.rmd
-- Declare the functions to be used from the DLL
STDCALL function 'MultInt' ( Integer ) : Integer
STDCALL VOID 'LCaseByREF' (ptr TEXT (30))
STDCALL function 'LCaseByVAL (ptr TEXT (60)) : TEXT (60)

--Set somme variables for use
Set VAR vTEXT TEXT = 'RBASE TECHNOLOGIES'
SET VAR vINT INTEGER = 128
```

```
SET VAR v1 INTEGER = 0

-- OPTIONALLY CALL DLLOAD
SET VAR v1 = (DLLOAD('DemoLib.dll'))
IF v1 = 0 THEN
  PAUSE 2 USING 'DemoLib.dll NOT LOADED.. EXITING'
  RETURN
ENDIF

SET VAR V1 = (dlcall('demolib.dll', 'changecase', vtext))

{ The Value for v1 will be null because ChangeCase is a procedure and
doesn't
  RETURN A RESULT, but the value of vTEXT which is passed as a POINTER
has been
  changed to 'rbase technologies'}

SET VAR v1 = (DLCALL('demolib.dll', 'MultInt', vINT))

--The value for v1 will be 256 the value returned from the function.

-- running the following against RRBYW14
SELECT (DLCALL('demolib.dll','lcasebyval', Company))=60 FROM +
Customer WHERE LIMIT = 2

{Yields the following output:
 (DLCALL('demolib.dll','lcasebyval', Company)
  ------------------------------------------------------------
 computer warehouse - ii
 microtech university - i
}

SELECT ((ICAP2((DLCALL('demolib.dll','lcasebyval', Company))))) = 60 +
FROM Customer WHERE LIMIT = 2

{Yields the following otput:
 ((ICAP (DLCALL('demolib.dll','lcasebyval',
  ------------------------------------------------------------
 Computer Warehouse - Ii
 Microtech University - I
}

-- Optionally CALL DLFREE
SET VAR v1 = (DLFREE('DemoLib.dll'))

-- END Demo.rmd
```

Example of a DLL created in C++ Exporting three functions:

```
// BEGIN C++ DLL
// loaddll.cpp : Defines the entry point for the DLL application.
//
```

```
#include <windows.h>
#include <stdio.h>

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                                    )
{
   return TRUE;
}
#ifdef __cplusplus    // If used by C++ code,
extern "C" {          // we need to export the C interface
#endif

__declspec(dllexport) int cfunc1(int i){
       return i;
}
__declspec(dllexport) double cfunc2(double *inp){
       double rtn = *inp;
       rtn++;
       return rtn;
}
__declspec(dllexport) char * cfunc3(char *inp){
       strcat(inp," + ");
       return inp;
}

#ifdef __cplusplus
}
#endif

// END C++ DLL
```

See Also:

CHKFUNC
DELFUNC
DLLOAD
DLFREE
LIST FUNCTIONS

Special thanks to:
Mike Byerley (Fort Wayne, Indiana), an Authorized R:BASE Developer, for his contribution to the introduction, implementation and testing of the DLCALL Function in R:BASE.

1.5.6 DLFREE

(DLFREE('libraryname.ext'))

Checks to see if a given library file can be freed.

The function returns an integer value of 1 if the library is freed and 0 if a given library is not freed.

Example:

```
SET VARIABLE vFreePlugin = (DLFREE('RAudioPlayer.RBL'))
```

See Also:

CHKFUNC
DELFUNC
DLLCALL
DLLOAD
LIST FUNCTIONS

1.5.7 DLLOAD

(DLLOAD('libraryname.ext'))

Checks to see if a given library file is loaded.

The function returns an integer value of 1 if the library is loaded and 0 if a given library is not loaded.

Example:

```
SET VARIABLE vLoadPlugin = (DLLOAD('RAudioPlayer.RBL'))
```

See Also:

CHKFUNC
DELFUNC
DLLCALL
DLFREE
LIST FUNCTIONS

1.5.8 DNW

(DNW(*date*))

Returns the date value for the next working (business) day.

This fuction recognizes Monday through Friday as business days and does not recognize Saturday or Sunday. Holidays are not considered.

1.5.9 DWE

(DWE(*date*))

Returns the date value for the next weekend day.

This fuction recognizes Saturday or Sunday as weekend days and does not recognize Monday through Friday. Holidays are not considered.

1.5.10 DWRD

(DWRD(*value*))

Converts a currency value to its word representation. For example, (DWRD($100.50)) returns "one hundred dollars and fifty cents."

1.6 E

1.6.1 ENVVAL

(ENVVAL('*environmentvar*'))

Returns the current value of the specified DOS environment variable. You must either enclose the name of the environment variable in quotation marks or use a text variable to which you have assigned the environment variable.

First, assume you have the command below in your AUTOEXEC.BAT file.

```
SET workstat=10
```

Now, in R:BASE, you can use ENVVAL to find the value of that environment variable, as shown in the example below. The value of *vworkstat* will be the text value *10*.

```
SET VAR vworkstat = (ENVVAL('workstat'))
```

1.6.2 EXP

(EXP(*arg*))

Raises *e* to the *arg* power (where *e* = 2.71828182845905 and *arg* is any value with a DOUBLE, REAL, NUMERIC, or INTEGER data type). By raising *e* to an exponent, EXP performs the inverse operation of LOG.

In the following example, the value of *vexp* is *2.71828182845905*.

```
SET VAR vexp = (EXP(1))
```

1.7 F

1.7.1 FILENAME

(FILENAME(0))

Generates a unique filename with a .$$$ extension, and creates the file in the current directory.

1.7.2 FINDFILE

(FINDFILE('*filename*'))

Returns the location of a file. The function looks first in the current directory and then searches the DOS path for the file. If the file is found, the full path name is returned, if it isn't found, a NULL is returned. Wildcards in the filename will produce unpredictable results.

1.7.3 FLOAT

(FLOAT(*arg*))

Converts a number with a TEXT, INTEGER, or CURRENCY data type to a value with a REAL or DOUBLE data type. This is not the same as the FLOAT data type.

In the following example, the value of *vfloat1* is *2.*, a number that has a REAL data type, and the value of *vfloat2* is also *2.*, a number that has a DOUBLE data type. If a variable is not assigned a data type, it becomes DOUBLE.

```
SET VAR vfloat1 REAL = (FLOAT(2))
SET VAR vfloat2 = (FLOAT(2))
```

1.7.4 FORMAT

(FORMAT (*value*,'*picture-format*'))

Prints picture formats to a variable, rather than only to the screen. You can use FORMAT anywhere that you can use a function. The result of the FORMAT function is always text.

In the syntax for this function, *value* is the value you want to be displayed in a particular format; it can be a column, variable, or a constant value. *'Picture-format'* is the picture format you establish.

The FORMAT function can be useful in several ways:

- Aligning decimals
- Capturing date and time using system variables
- Formatting currency
- Formatting text
- Punctuating long numbers

The characters you can use to format your data are listed below.

For All Data

[<]	Data is left justified.
[>]	Data is right justified.
[^]	Data is centered.

For Numbers

[-]	Places a minus sign to the right of a negative number.
[DB]	Places DB to the right of a negative number.
[( )]	Encloses a negative number in parentheses.
[CR]	Places CR to the right of a positive number.
9	Fills unused space with blanks.
0	Fills unused space with zeros.
*	Fills unused space with asterisks.

For Text

_	Letters are uppercase; other characters are blank.
|	Letters are lowercase; other characters are blank.
%	Letters are uppercase; other characters are unchanged.
?	Letters are lowercase; other characters are unchanged.

For Dates

MMDDYYYY	Displays the month, day, and year
MM	Displays the month
DD	Displays the day
YYYY	Displays the year
WWW	Displays the day name, with a 3-letter abbreviation
WWW+	Displays the full name for the day of the week
MMM	Displays the month name, with a 3-letter abbreviation
MMM+	Displays the full name for the month
CC	Displays the century, AD or BC
any combination	Any combination of the month, day and year can be used.

For Times

HHMMSS	Displays the hour, minute, and second
HH	Displays the hour
MM	Displays the minute
SS	Displays the second
.SSS	Displays thousandths of a second
AP	Displays AM or PM when using a 12-hour format
any combination	Any combination of the hours, minutes, and seconds can be used.

NN	Displays minutes (when capturing date and time using #NOW)

1.7.4.1 Aligning Decimals

The following example shows how you can use the FORMAT function to align decimal points in a column:

```
SELECT (FORMAT(bonuspct,'99.000')) FROM salesbonus
```

The following example shows the effect of the FORMAT function on the above SELECT statement:

Using FORMAT	Without FORMAT
0.003	0.003
0.002	0.002
0.000	0
0.001	0.001

1.7.4.2 Capturing date and time using system variables

Here's a simple routine to create a unique file name by capturing the date and time using R:BASE system variables:

```
-- Example 01: (Using .#DATE and .#TIME as two separate variables)
  CLS
  CLEAR VARIABLES vDateTxt,vTimeTxt,vFileName
  SET VAR vDateTxt TEXT = NULL
  SET VAR vTimeTxt TEXT = NULL
  SET VAR vFileName TEXT = NULL
  SET VAR vDateTxt = (FORMAT(.#DATE,'MMDDYYYY'))
  SET VAR vTimeTxt = (FORMAT(.#TIME,'HHMM'))
  SET VAR vFileName = +
  ((CVAL('DATABASE'))+'_'+.vDateTxt+'_'+.vTimeTxt+'.BKP')
  CLEAR VARIABLES vDateTxt,vTimeTxt
  RETURN
-- vFileName will return the text variable as:
--  RRBYW16_12302008_1630.BKP
-- Database Name, Date and Time will vary on your end


-- Example 02: (Using .#DATE and .#TIME as one variable in expression)
  CLS
  CLEAR VARIABLES vFileName
  SET VAR vFileName TEXT = NULL
  SET VAR vFileName = +
  ((CVAL('DATABASE'))+'_'+(FORMAT(.#DATE,'MMDDYYYY'))+   +
  '_'+(FORMAT(.#TIME,'HHMM'))+'.BKP')
  RETURN
-- vFileName will return the text variable as:
--  RRBYW16_12302008_1630.BKP
-- Database Name, Date and Time will vary on your end


-- Example 03: (Using .#NOW) - Short and Swift
  CLS
  CLEAR VARIABLES vFileName
  SET VAR vFileName TEXT = NULL
  SET VAR vFileName = +
```

```
  ((CVAL('DATABASE'))+''-'+(FORMAT(.#NOW,'MMDDYYYY_HHNN'))+'.BKP')
  RETURN
-- Notice the "NN" for minutes when using .#NOW (not a typo)
-- vFileName will return the text variable as:
--  RRBYW16_12302008_1630.BKP
-- Database Name, Date and Time will vary on your end
```

Now that you have successfully created a unique file name, you may use the following routine to make a backup of your live database, on demand! Here's how:

```
OUTPUT  .vFileName
UNLOAD  ALL
OUTPUT  SCREEN
```

The resulting two files will be created in the same folder, unless a different folder name is specified:

- RRBYW16_12302008_1630.BKP
- RRBYW16_12302008_1630.LOB

1.7.4.3 Formatting Currency

The following example shows how you can use the FORMAT function to only display whole dollars:

```
SELECT (FORMAT(netamount,'[>]$999,999')) FROM salesbonus
```

This SELECT statement displays data as right justified whole dollars, as shown below:

Using FORMAT	Without FORMAT
$176,000	$176,000.00
$87,500	$87,000.00

1.7.4.4 Formatting Text

You must include a format character for each text character. The following example shows how you can use the FORMAT function to display text in uppercase:

```
SELECT (FORMAT(empfname,'________')) FROM employee
```

Using FORMAT	Without FORMAT
JUNE	June
ERNEST	Ernest
PETER	Peter

1.7.4.5 Punctuating Long Numbers

The following example shows how you can use the FORMAT function to include a comma after the thousand's place:

```
SELECT (FORMAT(transid,'999,999')) FROM transmaster
```

The following shows the effect of the FORMAT function on the above SELECT statement:

Using FORMAT	Without FORMAT
104	104
2,002	2002
39,765	39765

1.7.5 FV1

(FV1(*pmt*,*int*,*per*))

Returns the future value of a series of equal payments of the amount, *pmt*, periodic interest rate, *int*, and compounding periods, *per*.

In the following example, FV1 returns the ending balance in your savings account if you deposit $300 each month for four years with an annual interest rate of 6.25%. The value of *vfv1* (the balance) is *$16,311.90*.

```
SET VAR vfv1 = (FV1(300,(.0625/12),(4 *12)))
```

1.7.6 FV2

(FV2(*pv*,*int*,*per*))

Returns the future value of an amount, *pv*, invested at a periodic interest rate, *int*, for compounding periods, *per*.

In the following example, FV2 returns your ending balance for a savings account where you have deposited $1,800 with an annual interest rate of 5.5% compounded monthly (simple interest divided by compounding periods) for seven years (12 periods times seven years). The value of *vfv2* (the balance) is *$2,642.98*.

```
SET VAR vfv2 = (FV2(1800,(.055/12),(7 *12)))
```

1.8 G

1.8.1 GETDATE

(GETDATE('Calendar Caption'))

Example:

In a command file, EEP or Custom Button

```
SET VAR vGetDate = (GETDATE('Select Date'))
```

Will bring up the Windows GUI Calendar with today's date circled in red. Either you could click on OK to accept the circled date or click on any other date on the calendar using the current month or scrolling months!

The GETDATE function will return a valid date in column or variable.

1.8.2 GETKEY

(GETKEY(0))

Gets the text value, in brackets, of the first key available in the type-ahead buffer. If a key is not available, GETKEY waits for the next keystroke. Since R:BASE normally checks the buffer for the [Ctrl] + [Break] keys, you must set ESCAPE to off for GETKEY to work properly. Use CHKKEY before GETKEY to determine if a key is available. GETKEY does nothing with the zero that you enter in parentheses; GETKEY returns a value without receiving one.

If you are executing a process that takes several minutes to complete, you can use the CHKKEY and GETKEY functions to tell R:BASE what to do next, even while the process is executing.

1.8.3 GETVAL

(GETVAL('arg1', 'arg2'))

Gets a value based on the argument data provided. The followng GETVAL arguments are available:

- CheckMessageStatus
- GetDriveReady
- GetIPAddress
- GetLock
- GetMACAddr
- GetVolumeID
- PlayAndExit
- PlayAndWait

1.8.3.1 CheckMessageStatus

(GETVAL('CheckMessageStatus','####'))

Example:

```
SET VAR vStatus TEXT = NULL
SET VAR vStatus = (GETVAL('CheckMessageStatus','2038'))
```

Variable vStatus will return the text value of the current message status (ON/OFF) for error message 2038.

1.8.3.2 GetDriveReady

(GETVAL('GetDriveReady','driveletter'))

Example:

```
SET VAR vReady = (GETVAL('GetDriveReady','A'))
```

Result: False if not ready, True if ready.

1.8.3.3 GetIPAddress

(GETVAL('GetIPAddress','n'))

Where 'n' is the parameter to either retrieve the number of active network adapters or to retrieve the IP address of a given active network adapter. Use '0' to retrieve the number of active network adapters and 1-9 to retrieve the IP address(es) of active network adapter(s) of a given network workstation/ server.

Example:

```
-- Start
-- GetIPAddress.RMD
CLS
CLEAR VARIABLE vActiveAdapters,vIPAddress1,vIPAddress2, +
vIPAddress3,vIPAddress4,vPauseMessage,vCaption
SET VAR vActiveAdapters TEXT = NULL
```

```
SET VAR vIPAddress1 TEXT = NULL
SET VAR vIPAddress2 TEXT = NULL
SET VAR vIPAddress3 TEXT = NULL
SET VAR vIPAddress4 TEXT = NULL
SET VAR vPauseMessage TEXT = NULL
SET VAR vCaption TEXT = 'Understanding New GETVAL Function'

-- To retrieve the number of active network adapters
SET VAR vActiveAdapters = (GETVAL('GetIPAddress','0'))

-- To retrieve the IP address of first active network adapter
SET VAR vIPAddress1 = (GETVAL('GetIPAddress','1'))

-- To retrieve the IP address of second active network adapter
SET VAR vIPAddress2 = (GETVAL('GetIPAddress','2'))

-- To retrieve the IP address of third active network adapter
SET VAR vIPAddress3 = (GETVAL('GetIPAddress','3'))

-- To retrieve the IP address of fourth active network adapter
SET VAR vIPAddress4 = (GETVAL('GetIPAddress','4'))

SET VAR vPauseMessage = +
('Number of Active Adapter(s):'+(CHAR(009))&.vActiveAdapters+ +
 (CHAR(009))+(CHAR(013))+ +
 'IP Address of Active Adapter 1:'+(CHAR(009))&.vIPAddress1+ +
 (CHAR(009))+(CHAR(013))+ +
 'IP Address of Active Adapter 2:'+(CHAR(009))&.vIPAddress2+ +
 (CHAR(009))+(CHAR(013))+ +
 'IP Address of Active Adapter 3:'+(CHAR(009))&.vIPAddress3+ +
 (CHAR(009))+(CHAR(013))+ +
 'IP Address of Active Adapter 4:'+(CHAR(009))&.vIPAddress4+ +
 (CHAR(009))+(CHAR(013)))

CLS
PAUSE 2 USING .vPauseMessage CAPTION .vCaption +
ICON APP +
Button 'Yes, this is the R:BASE you have always wanted!' +
OPTION BACK_COLOR WHITE +
|MESSAGE_COLOR WHITE +
|MESSAGE_FONT_COLOR GREEN +
|BUTTON_COLOR WHITE
CLS
CLEAR VARIABLE vActiveAdapters,vIPAddress1,vIPAddress2, +
vIPAddress3,vIPAddress4,vPauseMessage,vCaption
RETURN
-- end
```

1.8.3.4 GetLock

```
(GETVAL('GetLock','tableviewname'))
```

GetLock is the first required parameter and the table/view name is the name of the table or view. Use this function to programmatically find the LOCK status of a table or view. The returning value is ON or OFF, depending on whether a lock is in place upon the table or view.

Example:

```
SET VAR vCheclLock = (GETVAL('GetLock','Customer'))
```

vCheckLock will return the value of ON or OFF for the *Customer* table

1.8.3.5 GetMACAddr

Use (GETVAL('GetMACAddr','n')) to retrieve the number of active network adapter(s) on workstation/ server as well as the physical MAC (Media Access Control) address(es) of active network adapter(s).

Media Access Control address is a physical hardware address that uniquely identifies each node of a network. In IEEE 802 networks, the Data Link Control (DLC) layer of the OSI Reference Model is divided into two sub-layers: the Logical Link Control (LLC) layer and the Media Access Control (MAC) layer. The MAC layer interfaces directly with the network media. Consequently, each different type of network media requires a different MAC layer.

```
<GETVAL('GetMACAddr','n')>
```

Where 'n' is the parameter to either retrieve the number of active network adapters or to retrieve the MAC address of given active network adapter. Use '0' to retrieve the number of active network adapters and 1-9 to retrieve the MAC address of active network adapter(s) on given network station/server.

Examples:

Example 01: To get the number of active network adapter(s)

```
SET VAR vActiveAdapters = (GETVAL('GetMACAddr','0'))
```

The variable vActiveAdapters will return the number of active network adapters on that work station/ server.

Example 02: To retrieve the MAC address of first active adapter

```
SET VAR vMACAddress = (GETVAL('GetMACAddr','1'))
```

The variable vMACAddress will return the value of first active network adapter on that workstation.

Example 03: To retrieve the MAC address of second active adapter

```
SET VAR vMACAddress = (GETVAL('GetMACAddr','2'))
```

The variable vMACAddress will return the value of second active network adapter on that workstation.

Example 04: To retrieve the MAC address of third active adapter

```
SET VAR vMACAddress = (GETVAL('GetMACAddr','3'))
```

The variable vMACAddress will return the value of third active network adapter on that workstation.

Practically, this new (GETVAL('GetMACAddr','n')) function can be used to customize access to your R: BASE for Windows Applications. Consequently, you can use this function to customize the properties of any control and/or settings of form using the PROPERTY command based on unique MAC address of workstation.

Example 05: Disabling/Hiding Application Menus based on MAC Address

Assuming your application includes 6 main menu options, namely Customers, Contacts, Employees,

Products, Sales, Quarterly Sales Reports, General Inquiry (Read Only) with Component IDs MM_Customers, MM_Contacts, MM_Employees, MM_Products, MM_Sales, MMQuarterlyReports, MM_GeneralInquiry accordingly.

In a secure work environment, suppose only one workstation or notebook can have access to all menus with MAC address 00-0D-43-2B-D6-B4 and everyone else can only access to General Inquiry (Read Only) menu.

If you would like to disable Customers, Contacts, Employees, Products, Sales and Quarterly Sales Report menus to all except the workstation or notebook with the MAC address 00-0D-43-2B-D6-B4, then here's an example to use as embedded Custom EEP in "On After Start EEP" option of form properties:

```
IF (GETVAL('GetMACAddr','1')) <> '00-0D-43-2B-D6-B4' THEN
   PROPERTY MM_Customers ENABLED 'FALSE'
   PROPERTY MM_Contacts ENABLED 'FALSE'
   PROPERTY MM_Employees ENABLED 'FALSE'
   PROPERTY MM_Products ENABLED 'FALSE'
   PROPERTY MM_Sales ENABLED 'FALSE'
   PROPERTY MM_QuarterlyReports ENABLED 'FALSE'
ENDIF
RETURN
```

If you would like to hide Customers, Contacts, Employees, Products, Sales and Quarterly Sales Report menus to all except the workstation or notebook with the MAC address 00-0D-43-2B-D6-B4, then here's an example to use as embedded Custom EEP in "On After Start EEP" option of form properties:

```
IF (GETVAL('GetMACAddr','1')) <> '00-0D-43-2B-D6-B4' THEN
   PROPERTY MM_Customers VISIBLE 'FALSE'
   PROPERTY MM_Contacts VISIBLE 'FALSE'
   PROPERTY MM_Employees VISIBLE 'FALSE'
   PROPERTY MM_Products VISIBLE 'FALSE'
   PROPERTY MM_Sales VISIBLE 'FALSE'
   PROPERTY MM_QuarterlyReports VISIBLE 'FALSE'
ENDIF
RETURN
```

1.8.3.6 GetVolumeID

(GETVAL('GetVolumeID','driveletter'))

Example:

```
SET VAR vVolumeID = (GETVAL('GetVolumeID','C'))
```

Will return the label name of drive C.

1.8.3.7 PlayAndExit

(GETVAL('PlayAndExit','filepath'))

Example:

```
SET VAR vPlay = (GETVAL('PlayAndExit','c:\windows\media\tada.wav'))
```

Will play sound/wave file and continue the next command in a command file or EEP.

1.8.3.8 PlayAndWait

(GETVAL('PlayAndWait','filepath'))

Example:

```
SET VAR vPlay = (GETVAL('PlayAndWait','c:\windows\media\tada.wav'))
```

Will play sound/wave file and waits for the sound to finish before continue to a next command. This could be a long time if the sound file is long. You would want this if, for instance, you're writing an answering machine aplication in R:BASE for Windows.

1.9 H

1.9.1 HTML

(HTML(*string*))

Converts a text value to valid HTML code.

1.10 I

1.10.1 ICAP

(ICAP(*arg*))

Converts *arg* to a string with an initial capital letter on only the first word.

In the following example, the value of *vicap* is *Mary is going to Murrysville*.

```
SET VAR vicap = (ICAP('mary is going to Murrysville'))
```

1.10.2 ICAP1

(ICAP1(*arg*))

Converts *arg* to a string with an initial upper case letter on the first word, and lower case on an initial capital letter on any following words. This is also knwon as Sentence Casing.

In the following example, the value of *vicap1* is *Mary went down the street.*

```
SET VAR vicap1 = (ICAP1('mary went down The Street.'))
```

1.10.3 ICAP2

(ICAP2(*arg*))

Converts *arg* to a string with an initial capital letter on each word. This is also known as Word Case.

In the following example, the value of *vicap2* is *John Smith*.

```
SET VAR vicap2 = (ICAP2('john smith'))
```

1.10.4 ICHAR

(ICHAR(*arg*))

Converts a single character, returning its corresponding ASCII integer value.

In the following example, the integer value of *vichar* is *65*.

```
SET VAR vichar = (ICHAR('A'))
```

1.10.5 IDAY

(IDAY(*arg*))

Where *arg* is a value that has either a DATE or DATETIME data type, IDAY returns the integer day of the month for a particular date.

In the following example, the value of *viday* is *12*.

```
SET VAR viday = (IDAY('06/12/93'))
```

1.10.6 IDIM

(IDIM(*arg*))

Where *arg* is a value that has a DATE data type, IDIM returns the number of days within that month.

In the following example, the value of *vidim* is *31*.

```
SET VAR vidim = (IDIM('03/15/2006'))
```

1.10.7 IDOY

(IDOY(*arg*))

Where *arg* is a value that has a DATE data type, IDOY returns the number day of the year.

In the following example, the value of *vidoy* is *74*.

```
SET VAR vidoy = (IDOY('03/15/2006'))
```

1.10.8 IDWK

(IDWK(*arg*))

Where *arg* is a value that has either a DATE or DATETIME data type, IDWK returns the day of the week where Monday is 1.

In the following example, the value of *vidwk* is *3*.

```
SET VAR vidwk = (IDWK('06/16/93'))
```

1.10.9 IFEQ

(IFEQ(*arg1*,*arg2*,*arg3*,*arg4*))

If *arg1* and *arg2* are equal, IFEQ returns the value of *arg3*. If *arg1* and *arg2* are not equal, IFEQ returns the value of *arg4*. The data types of *arg1* and *arg2* must match and the data types of *arg3* and *arg4* must match.

The following command changes the value of the *thiscol* column to *100* if the values of the variables *vaval* and *vbval* are equal. If *vaval* and *vbval* are not equal, the value of *thiscol* becomes *200* in rows where *thiscol* is greater than or equal to *100*.

```
UPDATE thistab SET thiscol=(IFEQ(.vaval, .vbval, 100, 200))+
WHERE thiscol >= 100
```

1.10.10 IFEXISTS

(IFEXISTS(*arg1,arg2,arg3*))

If *arg1* contains a value, IFEXISTS returns the value of *arg2*. If *arg1* is null, then the value of *arg3* is returned.

1.10.11 IFGT

(IFGT(*arg1,arg2,arg3,arg4*))

If *arg1* is greater than *arg2*, IFGT returns the value of *arg3*. If *arg1* is less than *arg2*, IFGT returns the value of *arg4*. The data types of *arg1* and *arg2* must match and the data types of *arg3* and *arg4* must match.

In the following example, the value of *vifgt* is *4*, because the date 1/1/96 is greater than the date 1/1/95.

```
SET VAR vifgt = (IFGT('1/1/96','1/1/95',4,5))
```

1.10.12 IFLT

(IFLT(*arg1,arg2,arg3,arg4*))

If *arg1* is less than *arg2*, IFLT returns the value of *arg3*. If *arg1* is greater than *arg2*, IFLT returns the value of *arg4*. The data types of *arg1* and *arg2* must match and the data types of *arg3* and *arg4* must match.

In the following example, the value of *viflt* is *4*, since *A* is less than *B*.

```
SET VAR A = 37
SET VAR B = 48
SET VAR viflt = (IFLT(.A,.B,4,5))
```

1.10.13 IFNULL

(IFNULL(*arg1,arg2,arg3*))

If *arg1* is null, then the value of *arg2* is returned. If *arg1* is not null, then the value of *arg3* is returned.

1.10.14 IFRC

(IFRC(*arg*))

Where *arg* is a value that has either a TIME or DATETIME data type, IFRC returns the current thousandth of a second. For this function to work correctly, the TIME format must be set to include thousandths of a second.

In the following example, the value of *vifrc* is *123*.

```
SET TIME FOR HH:MM:SS.SSS
SET VAR vifrc = (IFRC('12:15:30.123'))
```

1.10.15 IFWINDOW

(IFWINDOW('*windowname*'))

Returns 1 if a form with the *windowname* is open, 0 if not. *Windowname* is the name given to the instance of an MDI form started with the "AS *alias*" option when using the ENTER, EDIT USING, or BROWSE USING commands.

1.10.16 IHASH

(IHASH(*arg,n*))

R:BASE includes a function that can be used to create an integer value from a text value. The function was designed to create effective integer keys from long text columns. The function, IHASH, converts the entire text value, or just a specified number of characters.

Using this method is more complex that just indexing the LASTNAME column. First you need to add a computed column to your table using the IHASH function on the LASTNAME column to convert its text to integer values. You can modify your table through the Data Designer or the ALTER TABLE command:

```
ALTER TABLE employee ADD Hash_Lname=(IHASH(lastname,0)) INTEGER
```

The IHASH function converts the entire name to integer when used with the parameter 0. A different parameter converts the specified number of characters from the name, starting at the first character. For example, the parameter 7 will convert the first 7 characters of the lastname to an integer value. Deciding on the number of characters to convert can be one of the hardest things about using this method. Consider the relationships expressed in the following chart:

	Convert FEW characters	**Convert MORE characters**
PROS	Less input required	Less duplicate values
CONS	Greater duplicate values	More input required

After adding the computed column to your table, you need to use some programming commands as shown below to query that column. Using the IHASH function directly in a WHERE clause won't use indexes. First set a variable equal to IHASH of the value you're searching for, then use the variable in the WHERE clause. When using an IHASH column for searching, you won't be able to do ad hoc queries from the R:BASE main menu .

```
SET VAR vname = (IHASH('Smith',0))
SEL * FROM employee WHERE Hash_Lname = .vname
```

This method does provide greater flexibility in that you can have users enter anywhere from 1 character to the entire name based on the number of characters you specify in the IHASH function. For example, add a computed column to the table that will IHASH the first four characters of the name. Then, in your program, check the length that the user enters and if it's greater than four characters use an extra condition on your WHERE clause.

```
FILLIN vname USING 'Enter lastname (at least 4 characters): '
SET VAR vlen1=(SLEN(.vname)),vname1=(SGET(.VNAME,4,1)),+
       vhash=(IHASH(.vname1,4))
  IF vlen1 > 4 THEN
    CHOOSE vchoice FROM #VALUES FOR (firstname & lastname) +
           FROM employee WHERE Hash_Lname=.vhash AND +
                (SGET(lastname,.vlen1,1))=.vname
  ELSE
    CHOOSE vchoice FROM #VALUES FOR (firstname & lastname) +
           FROM employee WHERE Hash_Lname=.vhash
  ENDIF
```

A user can enter any number of letters for use with an IHASH computation, but must enter at least as many characters as specified in the IHASH column definition or enter the full name. If the entry less than the specified number of characters and less than the full length of the name, the correct data is not found. For example, with a column defined as (IHASH(lastname,7)), entering "WILL" will not find "WILLIAMS", it will only find "WILL".

The advantages of using a computed column with the IHASH function are that you can turn an inefficient TEXT index into an efficient INTEGER index and you can provide flexibility in searching. Users will have a larger selection of names to choose from and can select the appropriate person from the list. For example, entering WILLIAM will find WILLIAMSON, WILLIAM, and WILLIAMS if you use IHASH

(lastname,7).

A disadvantage of this method is that you need to add columns to your database. An extra computed column can slow down data entry. If you are tight on disk space this may not be an option. To determine how much additional disk space you'll need for an IHASH column, take the number of rows in the table and multiply by 4. The answer is the number of bytes of disk space you'll need for the additional column.

1.10.17 IHR

(IHR(*arg*))

Returns the integer hour of time, where *arg* is a value that has a TIME or DATETIME data type.

In the following example, the value of *vihr* is *12*.

```
SET VAR vihr = (IHR(12:15:30))
```

1.10.18 ILY

(ILY(*arg*))

Checks to see if the current year is a leap year.

The function returns a 1 if the year is a leap year, and 0 if it is not a leap year.

In the following example, the value of *vily* is *0*.

```
SET VAR vily = (ILY('03/15/2006'))
```

1.10.19 IMIN

(IMIN(*arg*))

Returns the integer minutes of time, where *arg* is a value that has a TIME or DATETIME data type.

In the following example, the value of *vmin* is *15*.

```
SET VAR vmin TO (IMIN(12:15:30))
```

1.10.20 IMON

(IMON(*arg*))

Returns the integer month for a particular date, where *arg* is a value that has a DATE or DATETIME data type.

In the following example, the value of *vimon* is *5*.

```
SET VAR vimon = (IMON('05/20/95'))
```

1.10.21 INT

(INT(*arg*))

Truncates a number that has a REAL, DOUBLE, or CURRENCY data type, returning a value that has an INTEGER data type.

In the following example, the value of *vint* is *1*.

```
SET VAR vint = (INT(1.6))
```

1.10.22 ISALPHA

(ISALPHA(*value*))

Checks the first character of a TEXT string. The function returns a 1 if true and a 0 if false. For example, (ISALPHA('abc')) returns 1 because the first character is a letter.

1.10.23 ISDIGIT

(ISDIGIT(*value*))

Checks the first character of a TEXT string. The function returns a 1 if true and a 0 if false. For example, (ISDIGIT('abc')) returns 0 because the first character is not a number.

1.10.24 ISEC

(ISEC(*arg*))

Returns the integer seconds of time, where *arg* is a value that has a TIME or DATETIME data type.

In the following example, the value of *visec* is *30*.

```
SET VAR visec = (ISEC(12:15:30))
```

1.10.25 ISLOWER

(ISLOWER(*value*))

Checks the first character of a TEXT string. The function returns a 1 if true and a 0 if false. For example, (ISLOWER('abc')) would return 1 because the first character is lower case.

1.10.26 ISSPACE

(ISSPACE(*value*))

Checks the first character of a TEXT string. The function returns a 1 if true and a 0 if false. The ISSPACE function evaluates as true when the first character is a space or one of the following: LF (char (10)), CR (char (13)), VT (char (11)), TAB (char (9)), FF (char (12)), EOL (char (254)).

1.10.27 ISTAT

(ISTAT('keyword'))

Returns the current value or setting of 'keyword'.

Using ISTAT parameters, you can determine database activity, and the current mouse and cursor column or row coordinates.

Use ISTAT to check the efficiency of settings to adjust locking scheme. You should be able to see differences in the results returned by the ISTAT keywords TOTALREADS, TOTALWRITES, and TOTALLOCKS depending on the locking scheme you have set.

The ISTAT function has a number of options that report on available memory in the dynamic data area R:BASE uses for processing in the DOS version. The amount of available extended memory in the DOS version of R:BASE and how that memory is allocated is not checked or displayed by any parameters of the ISTAT function.

You must either enclose the keyword in quotation marks or use a dot variable that has a TEXT data type to which you have assigned the SHOW keyword.

You can use majority of SHOW keywords with CVAL.

Following keywords can be used for (ISTAT('keyword')):

- CURRNUMALLOC
- CURSORCOL
- CURSORROW
- DBSIZE
- DISKSPACE
- FORM_CONTROL_TYPE
- FORM_DIRTY_FLAG
- ISRUNTIME
- LIMITNUMALLOC
- MAXFREE
- MAXNUMALLOC
- MEMORY
- MOUSECOL
- MOUSEROW
- PAGECOL
- PAGEROW
- RX1SIZE
- RX2SIZE
- RX3SIZE
- RX4SIZE
- TOTALALLOC
- TOTALFREE
- TOTALLOCKS
- TOTALREADS
- TOTALWRITES

1.10.27.1 CURRNUMALLOC

(ISTAT('CURRNUMALLOC'))

The CURRNUMALLOC parameter allows you to check the number of memory handles R:BASE for DOS has open. This returns 0 when used with R:BASE for Windows.

1.10.27.2 CURSORCOL

(ISTAT('CURSORCOL'))

(ISTAT('CursorCol')) will return an integer value indicating the current column that contains the cursor.

1.10.27.3 CURSORROW

(ISTAT('CURSORROW'))

(ISTAT('CursorRow')) will return an integer value indicating the current row that contains the cursor.

1.10.27.4 DBSIZE

(ISTAT('DBSIZE'))

DBSIZE returns the size of the currently open database (total of the four database files). If no database is connected, ISTAT returns 0.

1.10.27.5 DISKSPACE

(ISTAT('DISKSPACE'))

DISKSPACE returns the amount of free disk space on the current drive. If you wish to check for the free space on drives with over 2 gigs of free you must use syntax similar to this:

```
SET VAR vSpace DOUBLE = (ISTAT('DISKSPACE'))
```

This is necessary because the default datatype, INTEGER, cannot hold the required number of digits to report over 2 gigs of free space.

When depending on this routine it is best to test it on a platform as close to your target platform as possible.

1.10.27.6 FORM_CONTROL_TYPE

(ISTAT('FORM_CONTROL_TYPE'))

FORM_CONTROL_TYPE returns a value based on which object has focus when the entry/exit procedure containing the ISTAT function is run. Returns the following.

0	No current control
1	Edit Field
2	Combo Box
3	Check Box
5	Radio Button
7	Push Button

1.10.27.7 FORM_DIRTY_FLAG

(ISTAT('FORM_DIRTY_FLAG'))

FORM_DIRTY_FLAG Returns 1 if a change has been made to the data, 0 if no change has been made.

1.10.27.8 ISRUNTIME

(ISTAT('ISRUNTIME'))

Use the ISRUNTIME parameter to determine if the end user is accessing R:BASE via a full version or a Runtime version.

If the end user is using Runtime the value returned will be 1. If the end user is using a Full Version then the value returned will be 0.

1.10.27.9 LIMITNUMALLOC

(ISTAT('LIMITNUMALLOC'))

The LIMITNUMALLOC parameter allows you to check the maximum number of memory handles R:BASE for DOS can open. This returns 0 when used with R:BASE for Windows.

Unless you use the -h startup option this will be 300.

1.10.27.10 MAXFREE

(ISTAT('MAXFREE'))

The MAXFREE parameter allows you to determine the largest free memory block within the area allocated to R:BASE. The TOTALALLOC parameter can be used to determine the total amount of dynamic memory allocated to R:BASE. MAXFREE will always be smaller than TOTALALLOC.

This will not return valid information under R:BASE for Windows.

1.10.27.11 MAXNUMALLOC

(ISTAT('MAXNUMALLOC'))

The MAXNUMALLOC parameter allows you to check the highest number of memory handles R:BASE for DOS has held open at any one point during this session. This returns 0 when used with R:BASE for Windows.

1.10.27.12 MEMORY

(ISTAT('MEMORY'))

The MEMORY parameter allows you to retrieve the amount of memory, in bytes, remaining for R:BASE for DOS to use.

This will not return a valid amount when running R:BASE for Windows.

1.10.27.13 MOUSECOL

(ISTAT('MOUSECOL'))

MOUSECOL returns the column where the mouse pointer is.

This is only supported in DOS, and will not return a valid amount when running R:BASE for Windows.

1.10.27.14 MOUSEROW

(ISTAT('MOUSEROW'))

MOUSEROW returns the row where the mouse pointer is.

This is only supported in DOS, and will not return a valid amount when running R:BASE for Windows.

1.10.27.15 PAGECOL

(ISTAT('PAGECOL'))

PAGECOL returns the column location of the cursor on a virtual page when you are using the SET PAGEMODE command. PAGECOL can only be used with the SHOW VARIABLE command; it does not work with the WRITE command.

1.10.27.16 PAGEROW

(ISTAT('PAGEROW'))

PAGEROW returns the row location of the cursor on a virtual page when you are using the SET PAGEMODE command. PAGEROW can only be used with the SHOW VARIABLE command; it does not work with the WRITE command.

1.10.27.17 RB1SIZE

(ISTAT('RB1SIZE'))

The RB1SIZE parameter returns the size, in bytes, of the RB1 file of the currently connected R:BASE eXtreme 9.0 (32) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RB1SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.18 RB2SIZE

(ISTAT('RB2SIZE'))

The RB2SIZE parameter returns the size, in bytes, of the RB2 file of the currently connected R:BASE eXtreme 9.0 (32) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RB2SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.19 RB3SIZE

(ISTAT('RB3SIZE'))

The RB3SIZE parameter returns the size, in bytes, of the RB3 file of the currently connected R:BASE eXtreme 9.0 (32) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RB3SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.20 RB4SIZE

(ISTAT('RB4SIZE'))

The RB4SIZE parameter returns the size, in bytes, of the RB4 file of the currently connected R:BASE eXtreme 9.0 (32) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RB4SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.21 RX1SIZE

(ISTAT('RX1SIZE'))

The RX1SIZE parameter returns the size, in bytes, of the RX1 file of the currently connected R:BASE eXtreme 9.0 (64) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RX1SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.22 RX2SIZE

(ISTAT('RX2SIZE'))

The RX2SIZE parameter returns the size, in bytes, of the RX2 file of the currently connected R:BASE eXtreme 9.0 (64) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RX2SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.23 RX3SIZE

(ISTAT('RX3SIZE'))

The RX3SIZE parameter returns the size, in bytes, of the RX3 file of the currently connected R:BASE eXtreme 9.0 (64) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RX3SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.24 RX4SIZE

(ISTAT('RX4SIZE'))

The RX4SIZE parameter returns the size, in bytes, of the RX4 file of the currently connected R:BASE eXtreme 9.0 (64) database. You may need to use the following syntax for large databases:

```
SET VAR vSize DOUBLE = (ISTAT('RX4SIZE'))
```

This is simply because the default datatype, INTEGER, may not be able to hold values as large as you need.

For a combined size use the DBSIZE parameter.

1.10.27.25 TOTALALLOC

(ISTAT('TOTALALLOC'))

The TOTALALLOC parameter allows you to determine the amount of dynamic memory allocated to R:BASE. Unless you ZIP out to another program this number should not change during an R:BASE session.

This will not return valid information under R:BASE for Windows.

1.10.27.26 TOTALFREE

(ISTAT('TOTALFREE'))

The TOTALFREE parameter allows you to determine the total amount of free memory within the area allocated to R:BASE. The TOTALALLOC parameter can be used to determine the total amount of dynamic memory allocated to R:BASE. TOTALFREE will always be smaller than TOTALALLOC.

This will not return valid information under R:BASE for Windows.

1.10.27.27 TOTALLOCKS

(ISTAT('TOTALLOCKS'))

TOTALLOCKS is used to determine the total number of locks placed on any assortment of tables, rows,

and indexes.

1.10.27.2 TOTALREADS

(ISTAT('TOTALREADS'))

TOTALREADS returns the total number of disk reads since R:BASE began or the number of reads since the statement was last executed.

1.10.27.2 TOTALWRITES

(ISTAT('TOTALWRITES'))

TOTALWRITES returns the total disk writes since opening or since the last time the command was executed.

1.10.28 ISTR

(ISTR('*string*',*position*))

Returns the corresponding integer value.

This function converts a single character, which you specify within a string by position, returning its corresponding ASCII Character Chart Decimal value.

In the following example, the INTEGER value of *vDecimalValue* is 65 for the capital letter A.

Example 01:

Start R:BASE for Windows

At the R> Prompt:

```
SET VARIABLE vDecimalValue = (ISTR('R:BASE Rocks!',4))

SHOW VARIABLE

vDecimalValue = 65 INTEGER
```

1.10.29 ISUPPER

(ISUPPER(*value*))

Checks the first character of a TEXT string. The function returns a 1 if true and a 0 if false. For example, (ISUPPER('abc')) returns 0 because the first character is not upper case.

1.10.30 ITEMCNT

(ITEMCNT('*Text String*'))

Use to count the number of items in a text string separated by current comma delimiter.

In the following example, the value of *vItems* is *3*.

```
SET VAR vItems = (ITEMCNT('a,b,c'))
```

Here is an example of using this function in a command block to format a CHOOSE box:

```
SET VAR vModels TEXT = NULL
```

```
SET VAR vLines INTEGER = NULL
SET VAR vModel TEXT = NULL
SELECT (LISTOF(Model)) INTO vModels INDIC IModel FROM Product
SET VAR vLines = (ITEMCNT(.vModels))
IF vLines > 18 THEN
SET LINES = 18
ENDIF
CLS
CHOOSE vModel FROM #LIST .vModels AT 4 30 TITLE 'Choose Model' +
CAPTION 'Available Models' Lines .vLines FORMATTED
IF vModel IS NULL OR vModel = '[Esc]' THEN
GOTO Done
ENDIF

-- Do what you have to do here ...

LABEL Done
CLEAR ALL VAR
QUIT TO MainMenu.RMD
RETURN
```

1.10.31 IWOY

(IWOY(*arg*))

Where *arg* is a value that has a DATE data type, IWOY returns the number week of the year.

In the following example, the value of *viwoy* is *11*.

```
SET VAR viwoy = (IWOY('03/15/2006'))
```

1.10.32 IYR

(IYR(*arg*))

Where *arg* is a value that has a DATE or DATETIME data type, IYR returns a two or four-digit integer year of date, depending on how the DATE format is set.

In the following example, the value of *viyr* is *1995* if the year in the DATE format is set to YYYY. The value of *viyr* is *95* if the year in the DATE format is set to YY.

```
SET VAR viyr = (IYR('09/13/95'))
```

1.10.33 IYR4

(IYR4(date)) or (IYR4(datetime))

Where *arg* is a value that has a DATE or DATETIME data type, IYR4 always returns a four-digit integer year of date, This is different than the IYR function with depends on how the DATE format is set.

This function will allow users to capture four digit year, regardless of the DATE FORMAT or DATE SEQUENCE settings. Results will be based on the DATE YEAR and CENTURY settings. See DATE format.

Example 01 (Database with settings):

```
DATE FORMAT: MM/DD/YYYY
DATE SEQ: MMDDYY
DATE YEAR: 30
DATE CENT: 19
SET VAR v1 = (IYR4(.#DATE))
```

Variable v1 will return the four digit INTEGER value of 2000

Example 02:

```
SET VAR v2 DATETIME = ('06/26/2000 08:00')
SET VAR v3 = (IYR4(.v2))
```

Variable v3 will return the four digit INTEGER value of 2000

Example 03 (Database with settings):

```
DATE FORMAT: MM/DD/YY
DATE SEQ: MMDDYY
DATE YEAR: 30
DATE CENT: 19
SET VAR v1 = (IYR4(.#DATE))
```

Variable v1 will ALSO return the four digit INTEGER value of 2000

Example 04:

```
SET VAR v2 DATETIME = ('06/26/00 08:00')
SET VAR v3 = (IYR4(.v2))
```

Variable v3 will ALSO return the four digit INTEGER value of 2000

1.11 J

1.11.1 JDATE

(JDATE(*arg*))

Where *arg* is a value that has a DATE or DATETIME data type, JDATE returns the Julian date of the date in the form YYYYDDD. This is a change from versions prior to R:BASE 6.1a which returned a value in the format YYDDD. This two digit year format was incompatible with the year 2000 and may require altering database structure or program code.

In the following example, the value of *vjdate* is *1995163*. The year is *1995*, and *163* means that the date is the 163rd day of 1995.

```
SET VAR vjdate = (JDATE('06/12/95'))
```

1.12 L

1.12.1 LASTKEY

(LASTKEY(*arg*))

Where *arg* is *0* or *1*. When *arg* is *1*, LASTKEY returns the original mapping for a key that has been remapped (the key that was pressed). When *arg* is *0*, LASTKEY returns the current mapping (the key that was executed).

In the following example, FILLIN displays the message and the current date, which the user can edit. LASTKEY captures the last key pressed by the user, enabling the user to press [Esc] to avoid entering a date.

```
SET VARIABLE vdate TEXT
SET VARIABLE vdate = (CTXT(.#DATE))
FILLIN vdate USING 'Enter the invoice date to print: ' EDIT
```

```
SET VARIABLE vlast = (LASTKEY(0))
IF vlast = '[Esc]' THEN
   GOTO skipprnt
ENDIF
```

1.12.2 LAVG

(LAVG(*list*))

Returns the average of the values in *list*. Values in a list can be a CURRENCY, DOUBLE, REAL, INTEGER, NUMERIC, DATE, or TIME data type. The function LAVG is not the same as the function AVG, which is used only with the SELECT command.

In the following example, the value of *vlavg* is *5*. The total of the list is 20, which is divided by 4, the number of values in the list.

```
SET VAR vlavg = (LAVG(2,4,6,8))
```

1.12.3 LJS

(LJS(*text*,*width*))

Left justifies *text* in *width* characters, returning a text string.

In the following example, the value of *vljs* is *ABCD*. The text string is left justified in the field. In this case, trailing blanks are removed.

```
SET VAR vljs = (LJS('ABCD',10))
```

The value of *vljs2* in the following example is *COLUMN ONE COLUMN TWO*. You can use LJS to embed spaces and concatenate strings, for example, to create a row of column headings for a screen or variable form. When you concatenate strings before assigning the result to a variable, as in this example, trailing blanks are retained. These spaces are preserved even if the first value is null.

```
SET VAR vljs2 = (LJS('COLUMN ONE',20) + 'COLUMN TWO')
```

The following example returns *COLUMN TWO* if *vcolname* is null.

```
SET VAR vcol2 TEXT = 'COLUMN TWO'
SET VAR vljs3 TEXT = (LJS(.vcolname,20) + .vcol2))
```

1.12.4 LMAX

(LMAX(*list*))

Returns the maximum value in *list*. Values in a list can be a CURRENCY, DOUBLE, REAL, INTEGER, NUMERIC, DATE, or TIME data type. The function LMAX is not the same as the function MAX, which is used only with the SELECT command.

In the following example, the value of *vlmax* is *80*, the highest number in the list.

```
SET VAR vlmax = (LMAX(2,80,14,22))
```

1.12.5 LMIN

(LMIN(*list*))

Returns the minimum value in *list*. Values in a list can be a CURRENCY, DOUBLE, REAL, INTEGER, NUMERIC, DATE, or TIME data type. The function LMIN is not the same as the function MIN, which is used only with the SELECT command.

In the following example, the value of *vlmin* is *2*, the lowest number in the list.

```
SET VAR vlmin = (LMIN(2,80,14,22))
```

1.12.6 LOG

(LOG(*arg*))

Returns log base *e* of *arg* (where *e* = 2.71828182845905). *Arg* must be a positive value and have a DOUBLE, REAL, NUMERIC, or INTEGER data type. LOG performs the inverse operation of EXP.

In the following example, the value of *vlog* is *0.6931*.

```
SET VAR vlog = (LOG(2))
```

1.12.7 LOG10

(LOG10(*arg*))

Returns log base 10 of *arg*. *Arg* must be a positive value and have a DOUBLE, REAL, NUMERIC, or INTEGER data type.

In the following example, the value of *vlog2* is *2*.

```
SET VAR vlog2 = (LOG10(100))
```

1.12.8 LTRIM

(LTRIM(*text*))

Trims leading blanks from *text*, returning a text string.

In the following example, the value of *vltrim* is the text string *ABCDE* without the leading blanks.

```
SET VAR vltrim = (LTRIM('   ABCDE'))
```

1.12.9 LUC

(LUC(*arg*))

Converts *arg* from lowercase to uppercase, returning a text string.

In the following example, the value of *vluc* is an uppercase *A*.

```
SET VAR vluc = (LUC('a'))
```

1.13 M

1.13.1 MOD

(MOD(*arg1*,*arg2*))

Computes a modulus or remainder of *arg1* divided by *arg2*. *Arg1* and *arg2* must be values that have DOUBLE, REAL, NUMERIC, or INTEGER data types and *arg2* cannot be 0.

In the following example, the value of *vmod* is *1*, the remainder when 3 is divided by 2.

```
SET VAR vmod = (MOD(3,2))
```

1.14 N

1.14.1 NEXT

(NEXT(tblname,autonumcol))

Returns the next value of an autonumbered column.

Where colname is an autonumbered column in tblname NEXT returns, and increments, the value of the next available autonumber. You cannot use this function with INSERT, but you can use it with LOAD; NONUM. For example: Assume that you have autonumbered the column EmployeeID in the table Employees. The highest number currently used in the database is 134. In this case, in the following example, the value of vNextOne will be 135 and the value of vNextTwo will be 136.

Notice that the value has incremented even though no other commands or functions were issued.

```
SET VAR vNextOne = (NEXT(Employees,EmployeeID))
SET VAR vNextTwo = (NEXT(Employees,EmployeeID))
```

1.14.2 NINT

(NINT(*arg*))

Rounds a number that has a TEXT, REAL, DOUBLE, NUMERIC, or CURRENCY data type to the nearest integer, returning a value that has an INTEGER data type.

In the following example, the value of *vnint1* is *3* and the value of *vnint2* is *4*.

```
SET VAR vnint1 = (NINT(2.6))
SET VAR vnint2 = (NINT(4.4))
```

1.15 P

1.15.1 PMT1

(PMT1(*int*,*per*,*pv*))

Returns the amount of the periodic payment needed to pay off the present value, *pv*, based on the periodic interest rate, *int*, for the number of compounding periods, *per*.

In the following example, PMT1 returns the monthly payment amount for a loan of $12,000 with a 12% annual interest rate and five years to pay it off. The value of *vpmt1* (the monthly payment) is *$266.93*.

```
SET VAR vpmt1 = (PMT1(.01,60,12000))
```

1.15.2 PMT2

(PMT2(*int*,*per*,*fv*))

Returns the amount of the periodic payment to accrue the future value, *fv*, based on the periodic interest rate, *int*, for the number of compounding periods, *per*.

In the following example, PMT2 returns the monthly payment you must make if you want to have $25,000 in 10 years and your annual interest rate is 7%, compounded monthly. The value of *vpmt2* (the payment) is *$144.44*.

```
SET VAR vpmt2 = (PMT2((.07/12),120,25000))
```

1.15.3 PV1

(PV1(*pmt*,*int*,*per*))

Returns the present value of a series of equal payments of the amount, *pmt*, periodic interest rate, *int*, and compounding periods, *per*.

In the following example, PV1 returns the present value of an annuity with an annual interest rate of 9% and for which you want to pay $500 each month for 20 years. The value of *vpv1* (the present value) is *$55,572.48*.

```
SET VAR vpv1 = (PV1(500,.0075,240))
```

1.15.4 PV2

(PV2(*fv*,*int*,*per*))

Returns the present value based on the future value, *fv*, interest rate, *int*, and the number of compounding periods, *per*.

In the following example, PV2 returns the amount you must invest for a period of one year to return a future value of $3,500 where the annual rate is 7.6% (compounded daily). The value of *vpv2* (the amount to invest) is *$3,247.63*.

```
SET VAR vpv2 = (PV2(3500,(.075/365),365))
```

1.16 R

1.16.1 RANDOM

(RANDOM(*value*))

Generates a random number between 0 and the value entered.

1.16.2 RATE1

(RATE1(*fv*,*pv*,*per*))

Returns the periodic interest rate required to return the future value, *fv*, based on the present value, *pv*, over the number of compounding periods, *per*.

In the following example, RATE1 returns the interest rate you must have if your initial investment is $8,500 and you want a future yield of $10,000 after 24 months. The value of *vrate1* (the interest rate) is *.0068*, a monthly rate of .68 percent, or 8.16% annually.

```
SET VAR vrate1 = (RATE1(10000,8500,24))
```

1.16.3 RATE2

(RATE2(*fv*,*pmt*,*per*))

Returns the periodic interest rate on a series of regular payments, *pmt*, whose future value is *fv* over the number of compounding periods, *per*.

In the following example, RATE2 returns the interest rate you must have if your goal is $37,000 and you deposit $500 each month for five years. The value of *vrate2* is *.0069*, a monthly rate of .69 percent, or 8.28% annually.

```
SET VAR vrate2 = (RATE2(37000,500,60))
```

1.16.4 RATE3

(RATE3(*pv*,*pmt*,*per*))

Returns the periodic interest rate required for an annuity of value *pv*, to return a series of equal payments, *pmt*, over a number of compounding periods, *per*.

In the following example, RATE3 returns the interest rate of an annuity, purchased at $50,000, which will pay $570 monthly for 10 years. The value of *vrate3* is *.0055*, a monthly rate of .55 percent, or 6.6% annually.

```
SET VAR vrate3 = (RATE3(50000,570,120))
```

1.16.5 RDATE

(RDATE(*mon*,*day*,*yr*))

Converts integers *mon*, *day*, and *yr* to a DATE data type. *Yr* must be a four-digit year. The result returned by RDATE will vary, depending on the DATE format.

The following command assigns to the *transdate* column in the *transmaster* table the date derived from the values of *vmon* and *vday* (integers) as the month and day, and *1995* as the year in rows where the *transdate* column has a value.

```
UPDATE transmaster SET transdate = (RDATE(.vmon, .vday, 1995)) +
WHERE transdate IS NOT NULL
```

1.16.6 RJS

(RJS(*text*,*width*))

Right justifies *text* in *width* characters returning a text string.

In the following example, the value of *vrjs* is *ABCD*. The text string is right justified in a 10-character field.

```
SET VAR vrjs = (RJS('ABCD',10))
```

1.16.7 ROUND

(ROUND(arg1, arg2))

Where: arg1 is the value to be rounded
arg2 is the position to be rounded after the decimal point

Example 01:

```
SET VAR vNumber DOUBLE = 1.4567
SET VAR vRound = (ROUND(.vNumber,1))
```

Resulting vRound will be equal to 1.5

Example 02:

```
SET VAR vNumber DOUBLE = 1.4567
SET VAR vRound = (ROUND(.vNumber,2))
```

Resulting vRound will be equal to 1.46

Example 03:

```
SET VAR vNumber DOUBLE = 1.4567
SET VAR vRound = (ROUND(.vNumber,3))
```

Resulting vRound will be equal to 1.457

1.16.8 RTIME

(RTIME(*hrs*,*min*,*sec*,*frc*))

Converts integers *hrs*, *min*, *sec*, and *frc* to a TIME data type. *Hrs* is on a 24-hour scale. RTIME can be specified for up to thousandths of a second. The *frc* argument is optional.

In the following example, the value of *vrtime* is *12:15:30*, stored in internal R:BASE time format.

The time value will be displayed according to how you have set the TIME format. When you use time data in expressions, the result is given in seconds. You can use RTIME to convert this result to hours, minutes, and seconds. The value of *velapsed1* is *1170* seconds; the value of *velapsed2* is *0:19:30*.

```
SET VAR vrtime = (RTIME(12,15,30))
SET VAR vstart TIME = '1:10:40'
SET VAR vend TIME = '1:30:10'
SET VAR velapsed1 = (.vend - .vstart)
SET VAR velapsed2 = (RTIME(0,0,.velapsed1))
```

1.16.9 RTRIM

(RTRIM(*text*))

Trims trailing blanks from *text*, returning a text string.

In the following example, the value of *vrtrim* is the text string *ABCDE* without the trailing blanks.

```
SET VAR vrtrim = (RTRIM('ABCDE   '))
```

1.16.10 RWP

(RWP('*string*','*actualword*'))

Returns the position of a specified word in a text string. RWP = Relative Word Position

Example 01:

```
SET VAR vRWP = (RWP('Imagine The Possibilities','Possibilities'))
```

Variable vRWP will return the value of 3.

Example 02:

```
SET VAR vRWP = (RWP('This is the R:BASE you have always
wanted!','always'))
```

Variable vRWP will return the value of 7.

Example 03:

```
SET VAR vRWP = (RWP('Keep in mind that nothing is impossible','nothing'))
```

Variable vRWP will return the value of 5.

1.17 S

1.17.1 SFIL

(SFIL(*chr*,*nchar*))

Fills a text string with a specified character *chr*, for *nchar* characters up to 500. You cannot use a number as a character. Instead, assign the number to a variable that has a TEXT data type using the variable name in SFIL.

In the following example, the value of *vsfil* is the text string ==========. R:BASE programs often include SFIL to draw lines.

```
SET VAR vsfil = (SFIL('=',10))
```

1.17.2 SGET

(SGET(*text*,*nchar*,*pos*))

Gets *nchar* characters from *text* starting at *pos*, returning a text string.

In the following example, the value of *vsget* is *BCD*, the three characters starting in the second position of the text string.

```
SET VAR vsget = (SGET('ABCDE',3,2))
```

1.17.3 SIGN

(SIGN(*arg1*,*arg2*))

Transfers the sign of *arg2* to *arg1*. *Arg1* and *arg2* must be values that have DOUBLE, REAL, NUMERIC, or INTEGER data types.

In the following example, the value of *vsign* is *-15*, changing the sign of the first argument to the sign of the second argument.

```
SET VAR vsign = (SIGN(15,-20))
```

1.17.4 SIN

(SIN(*angle*))

Returns the trigonometric sine of *angle*.

In the following example, the value of *vsin* is *0.8659*.

```
SET VAR vsin = (SIN(1.047))
```

1.17.5 SINH

(SINH(*angle*))

Returns the hyperbolic sine of *angle*.

In the following example, the value of *vsinh* is *1.2491*.

```
SET VAR vsinh = (SINH(1.047))
```

1.17.6 SKEEP

(SKEEP(*source*, *chars*))

Keeps characters within the *source* string, using *chars* as a comparison.

This function is CASE SENSITIVE.

In the following example, the value of *vskeep* is *ldilliamennighway.*

```
SET VAR vskeep = (SKEEP('3935 Old William Penn
Highway','abcdefghijklmnopqrstuvwxyz'))
```

Spaces are also recognized.

In the following example, the value of *vskeep2* is *ld illiam enn ighway.*

```
SET VAR vskeep2 = (SKEEP('3935 Old William Penn Highway','
abcdefghijklmnopqrstuvwxyz'))
```

1.17.7 SKEEPI

(SKEEPI(*source*, *chars*))

Keeps characters within the *source* string, using *chars* as a comparison.

This function is NOT CASE SENSITIVE.

In the following example, the value of *vskeepi* is *OldWilliamPennHighway.*

```
SET VAR vskeepi = (SKEEPI('3935 Old William Penn
Highway','abcdefghijklmnopqrstuvwxyz'))
```

Spaces are also recognized.

In the following example, the value of *vskeepi2* is *Old William Penn Highway.*

```
SET VAR vskeepi2 = (SKEEPI('3935 Old William Penn Highway','
abcdefghijklmnopqrstuvwxyz'))
```

1.17.8 SLEN

(SLEN(*text*))

Returns the length of *text*. SLEN is often used to ensure that a string does not exceed the space allowed for it on a form, variable form, or report. When strings are concatenated or passed as parameters to a procedure file, the length of a string might be unknown.

In the following example, the value of *vslen* is *5*, the number of characters in the text string.

```
SET VAR vslen = (SLEN('ABCDE'))
```

1.17.9 SLOC

(SLOC(*text*,*string*))

Locates *string* in *text*, returning the position if the string is found, *0* if it is not found.
In the following example, the value of *vsloc1* is *3*.

```
SET VAR vsloc1 = (SLOC('ABCDE','C'))
```

The value of *vsloc2* in the following example is *0*, since the text string *X* does not exist in the text string *ABCDE*.

```
SET VAR vsloc2 = (SLOC('ABCDE','X'))
```

In the following example, the *custzip* column contains a customer's zip code, in which the first five characters might be followed by a dash and another four characters. If the first row contained the string *02178-5243*, *Sloc3* would capture the position of the dash (*6*), which is in the sixth position.

```
SET VAR v5zip = custzip IN customer WHERE COUNT = 1
SET VAR sloc3 = (SLOC(.v5zip,'-'))
```

1.17.10 SLOCP

(SLOCP(*TextNoteVarcharValue*,*string*,*occurrence*))

Locates the exact position of a given string and occurrence in a TEXT, NOTE or VARCHAR value, returning the position if the string is found, *0* if it is not found. Using -1 as the third parameter will return the LAST occurrence.

In the following example, the value of *vslocp1* is 1.

```
SET VARIABLE v1 VARCHAR = 'ABCDEABC_AB'
SET VARIABLE vslocp1 = (SLOCP(.v1,'AB',1))
```

In the following example, the value of *vslocp2* is 6.

```
SET VARIABLE v1 VARCHAR = 'ABCDEABC_AB'
SET VARIABLE vslocp2 = (SLOCP(.v1,'AB',2))
```

1.17.11 SMOVE

(SMOVE(*text*,*pos1*,*nchar*,*string*,*pos2*))

From *text*, starting at position *pos1*, moves *nchar* characters to *string* starting at position *pos2*.

In the example below, the value of *vsmove1* is *XBCDX*. The characters *BCD* in the first string are moved into the second through fourth positions in the string *XYZXX*.

```
SET VAR vsmove1 = (SMOVE('ABCDE',2,3,'XYZXX',2))
```

In the following example, the *custcity* column is 12 characters wide and contains the string *ANCHORAGE* in the first row. You can use SMOVE to fill in a blank (13 characters in the example above) in order to customize a report title. The value of *vfulltitle* is *ANCHORAGE WAREHOUSE*.

```
SET VAR vcity = custcity IN customer WHERE COUNT = 1
SET VAR vfulltitle = (SMOVE(.vcity,1,12,'           WAREHOUSE',1))
```

1.17.12 SOUNDEX

(SOUNDEX(*value*))

Converts a text value to the corresponding SOUNDEX code.

1.17.13 SPUT

(SPUT(*text*,*string*,*pos*))

Puts *string* into *text,* starting at *pos*, returning a text string.

The value of *vsput1* in the following example is *AXCDE*. The character *X* is put into the second position in the string *ABCDE*.

```
SET VAR vsput1 = (SPUT('ABCDE','X',2))
```

1.17.14 SQRT

(SQRT(*arg*))

Returns square root of *arg*. *Arg* must be a positive value with a DOUBLE, REAL, NUMERIC, or INTEGER data type.

In the following example, the value of *vsqrt* is *10*.

```
SET VAR vsqrt = (SQRT(100))
```

1.17.15 SRPL

(SRPL(*sourcestring*,*searchstring*,*replacestring*,[0|1]))

Enables searching for and replacing text within a specified string of text.

sourcestring - specifies a string of text to search
searchstring - specifies text to search for
replacestring - specifies the replacement text
flag - 0 = replacement occurs for every matching string
- 1 = replacement occurs for whole word matches only

The following "0 flag" example replaces *04/20/64* with *04-20-64*:

```
SET VAR vsrpl = (SRPL('04/20/64','/','-',0))
```

The following "1 flag" example replaces *Dear Joe* with *Dear Anne*:

```
SET VAR vsrpl = (SRPL('Dear Joe','Joe','Anne',1))
```

However, with this next "1 flag" example *DearJoe* remains the same as *DearJoe* is a whole word without a space.

```
SET VAR vsrpl = (SRPL('DearJoe','Joe','Anne',1))
```

1.17.16 SSTRIP

(SSTRIP(*source*, *chars*))

Strips characters from the *source* string, using *chars* as a comparison.

This function is CASE SENSITIVE.

In the following example, the value of *vsstrip* is *3935 O W P H.*

```
SET VAR vsstrip = (SSTRIP('3935 Old William Penn
Highway','abcdefghijklmnopqrstuvwxyz'))
```

Spaces are also recognized.

In the following example, the value of *vsstrip2* is *3935OWPH.*

```
SET VAR vsstrip2 = (SSTRIP('3935 Old William Penn Highway','
```

```
abcdefghijklmnopqrstuvwxyz'))
```

1.17.17 SSTRIPI

(SSTRIPI(*source*, *chars*))

Strips characters from the *source* string, using *chars* as a comparison.

This function is NOT CASE SENSITIVE.

In the following example, the value of *vsstripi* is *3935 , 15668.*

```
SET VAR vsstripi = (SSTRIPI('3935 Old William Penn Highway,
15668','abcdefghijklmnopqrstuvwxyz'))
```

Spaces are also recognized.

In the following example, the value of *vsstripi* is *3935,15668.*

```
SET VAR vsstripi = (SSTRIPI('3935 Old William Penn Highway, 15668','
abcdefghijklmnopqrstuvwxyz'))
```

1.17.18 SSUB

(SSUB(*text*,*n*))

Captures substring number *n* from *text*, returning a text string. Substrings in *text* are separated by a comma (or the current delimiter).

The SSUB function is often used with the CHOOSE command when capturing menu options from a pull-down menu.

In the following example, the value of *vssub* is *yearend*.

```
SET VAR vtext = 'reports,yearend'
SET VAR vssub = (SSUB(.vtext,2))
```

The following example shows that SSUB separates items based on a blank rather than the current delimiter when *n* is less than zero.

```
SET VAR vtext = 'reports yearend'
SET VAR vssub2 = (SSUB(.vtext,-2))
```

The following example also returns *yearend*. For this example, assume that *twodim* is a bar with a pull-down menu with the options *Edit* and *Enter* stored as text numbers according to their positions on the menu. The pop-up menu for both options contains a list of form names, so the second item in the CHOOSE variable will be a form name.

```
CHOOSE vtwodim FROM twodim
SET VARIABLE vbar = (SSUB(.vtwodim,1))
SET VARIABLE vpull = (SSUB(.vtwodim,2))
IF vbar = '1' THEN
   EDIT USING .vpull
ELSE
   ENTER .vpull
ENDIF
```

1.17.19 STRIM

(STRIM(*text*))

Trims trailing blanks from *text*, returning a text string.

In the following example, the value of *vstrim* is the text string *ABCDE* without the trailing blanks.

```
SET VAR vstrim = (STRIM('ABCDE   '))
```

1.18 T

1.18.1 TAN

(TAN(*angle*))

Returns the trigonometric tangent of *angle*.

In the following example, TAN defines a new column *newtan* for the *mytable* table. *Newtan* is a computed column providing the tangent of the angle in radians stored in the *oldangle* column.

```
ALTER TABLE mytable ADD newtan = (TAN(oldangle)) DOUBLE
```

1.18.2 TANH

(TANH(*angle*))

Returns the hyperbolic tangent of *angle*.

In the following example, the value of *vtanh* is *.7616*.

```
SET VAR vtanh = (TANH(1))
```

1.18.3 TDWK

(TDWK(*arg*))

Returns the day of the week as text, where *arg* is a value that has either a DATE or DATETIME data type.

In the following example, the value of *vtdwk* is *Saturday*.

```
SET VAR vtdwk = (TDWK('12/02/95'))
```

1.18.4 TERM1

(TERM1(*pv*,*int*,*fv*))

Returns the number of compounding periods (the term) for a return of future value *fv*, based on the present value, *pv*, and the interest rate, *int*.

In the following example, TERM1 returns the number of months your money must stay invested if you want to accumulate $10,000 on an initial investment of $5,000 at a compounded monthly rate of 1% (12% annually). The value of *vterm1* (the term) is *70*.

```
SET VAR vterm1 = (TERM1(5000,.01,10000))
```

1.18.5 TERM2

(TERM2(*pmt*,*int*,*fv*))

Returns the number of compounding periods (the term) for a return of future value *fv*, based on the payment, *pmt*, and interest rate, *int*.

In the following example, TERM2 returns the number of years you must make payments if you want to accumulate $75,000 by making annual installments of $2,000 at 8% annual interest. The value of *vterm2* (the term) is *18*.

```
SET VAR vterm2 = (TERM2(2000,.08,75000))
```

1.18.6 TERM3

(TERM3(*pmt*,*int*,*pv*))

Returns the number of periods (the term) for the present value, *pv*, to reach 0 based on the payment, *pmt*, and the interest rate, *int*.

In the following example, TERM3 returns the number of payments from a $15,000 annuity if the annual interest rate is 12.5% compounded monthly and you would like to receive $300 every month. The value of *vterm3* is *71*.

```
SET VAR vterm3 = (TERM3(300,(.125/12),15000))
```

1.18.7 TEXTRACT

(TEXTRACT(*datetime*))

Returns the time portion of DATETIME.

In the following example, the value of *vtextract* is *12:15:30.123*.

```
SET VAR vtextract = (TEXTRACT('08/09/95 12:15:30.123'))
```

1.18.8 TMON

(TMON(*arg*))

Returns the month name as text where *arg* is a value that has either a DATE or DATETIME data type.

In the following example, the value of *vtmon* is *November*.

```
SET VAR vtmon = (TMON('11/12/95'))
```

1.18.9 TRIM

(TRIM(*text*))

Trims leading and trailing blanks from *text*, returning a text string.

In the following example, the value of *vtrim* is the text string *ABCDE* without the leading and trailing blanks.

```
SET VAR vtrim = (TRIM('   ABCDE   '))
```

1.19 U

1.19.1 UDF (User-Defined Functions)

(UDF ('*exe-name*','*parameter-list*'))
(UDF ('*-exe-name*','*parameter-list*'))
(UDF ('*+exe-name*','*parameter-list*'))
(UDF('@dll-name','parameter-list'))

You can use a user-defined function (UDF) anywhere you can use a function in R:BASE:

- Forms

- Reports
- Entry/exit procedures
- Applications
- Commands

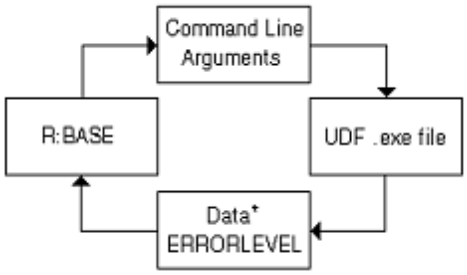
The arguments for a UDF are as follows:

@, +, -, or nothing
If you do not specify a plus or minus, the function is hidden when it runs; this setting is the default. A plus (+) runs the function in a normalized window. A minus (-) runs the function as a minimized icon. The at sign (@) is used to indicate that the UDF being called is a DLL. This only applies to Windows and is not available to the DOS version of R:BASE.

exe-name,dll-name
Specifies the name of the UDF (the name of the executable file into which you have compiled and linked your program). Under Windows this can be a Windows program. When using DLL's you MUST include the .DLL extension or the DLL may not be executed.

parameter-list
Specifies a text string. Multiple parameters must be separated by spaces, as shown:

```
SET VAR v1=(UDF('random','25 345'))
```



UDFs function as follows:

- R:BASE passes command line arguments to the RBGUDF.EXE file.
- The RBGUDF.EXE file completes its calculations and passes the data and the ERRORLEVEL to R:BASE.

This file can be found in the same directory as the main R:BASE executable.

When used with executables the parameter-list, from the UDF syntax, acts exactly as if you typed it at the a command prompt. Remember to account for the length of the executable name as well as the memory address (eight characters) plus the spaces between the .EXE name, the parameter list, and the address name. The parameter list can be space delimited with EXE UDF's.

You can write a UDF executable file in any language that allows writing to far pointers, such as Microsoft C. UDF executable files require protected mode libraries. You can return only 1,500 bytes of text data to R:BASE; that is, when you copy your results to either the address passed in or the shared-memory segment, you must not write more than 1,499 characters plus the NULL terminator (/0). If you write more than 1,500 total characters to the address you will corrupt the R:BASE variables and R:BASE might "hang."

If you forget the NULL terminator (/0) you will probably get high-order ASCII characters in your text variable. To pass back multiple values, delimit them with commas and then use other R:BASE functions to pull out the values. R:BASE stops looking at the returned text when it first encounters a NULL (/0). If your UDF executable file hangs, It is very likely that R:BASE will as well.

The size of a UDF is limited by the amount of available memory. Keep your executable files as small as possible. You will have to experiment to determine the maximum possible memory in a given situation.

You can have different amounts of available memory at different times in an R:BASE session. Because you had an amount of memory available at one point in an R:BASE session does not mean that you will have the same amount of memory available in your next R:BASE session.

If your UDF fails to write to the memory given to it, then the UDF function in R:BASE sets the variable to NULL (-0-). You could use this in your UDF executable file to show whether the UDF failed. UDFs treat error variables the same as R:BASE does; therefore, you can have your application tell you if the UDF was successful or not by using the R:BASE error variable and the compiler's function that allows the return of error variables, such as Microsoft C exit( ) function to return the ERRORLEVEL. R:BASE places the ERRORLEVEL value in the R:BASE error variable.

1.19.1.1 ERRORLEVEL

If your UDF fails to write to the memory given to it, then the UDF function in R:BASE sets the variable to NULL (-0-). You could use this in your RBGUDF.EXE file to show whether the UDF failed.

UDFs treat error variables the same as R:BASE does; therefore, you can have your application tell you if the UDF was successful or not by using the R:BASE error variable and the compiler's function that allows the return of error variables, such as Microsoft C exit( ) function to return the ERRORLEVEL. R:BASE places the ERRORLEVEL value in the R:BASE error variable.

1.19.1.2 Memory Issues

The size of a UDF is limited by the amount of available memory. Keep your executable files as small as possible. Depending on where the UDF is run and how much memory R:BASE has available to it at that moment, you could possibly run an executable file as large as 50K. You will have to experiment to determine the maximum possible memory in a given situation.

You can have different amounts of available memory at different times in an R:BASE session. Because you had an amount of memory available at one point in an R:BASE session does not mean that you will have the same amount of memory available in your next R:BASE session.

1.19.1.3 Returning Multiple Values

A UDF can return up to 1,500 characters. You can write a UDF that returns multiple values; then, you can write a command file that interprets coded strings and tells your application what was returned. For example, you could set up the following coded strings:

Coded String	What was done
100	Cube root of the input number
200	Standard deviation of input number

The UDF returns the information below to R:BASE with values for the coded strings 100 and 200 in comma-delimited format:

```
100,1.23456,200,2.34567
```

The command file would interpret that 1.23456 is the cube root of the input number, and 2.34567 is the standard deviation of the input number.

1.19.1.4 Sample UDF

The following example uses a UDF named dwrd to convert a currency value to words:

```
SET VAR v1=(UDF('dwrd','$1,456.99'))
```

If you execute the above command and then enter SHOW VARIABLE at the R> Prompt, you see the following output:

one thousand four hundred fifty-six dollars and ninety-nine cents

Examples

The following is a R:BASE command that calls the UDF named random.

```
SET VAR v1 = (UDF('random', '25 345'))
```

The following are the command line arguments for the UDF named random for DOS as seen by the UDF code:

```
DOS
argv[0] = random
argv[1] = 25
argv[2] = 345
argv[3] = A345BD3456
argc    = 4
```

Argument 0 is the name of the executable. Argument 1 is the first parameter. Argument 2 is the second parameter. Argument 3 is the memory address. This brings the argument count to 4.

The following example uses a UDF named DWRD.EXE to convert a currency value to words:

```
SET VAR v1=(UDF('dwrd','$1,456.99'))
SHOW VAR v1
```

Result: v1= one thousand, four hundred fifty six dollars and ninety nine cents

The following example uses the DWRD.DLL to accomplish the same function.

```
SET VAR v1=(UDF('@dwrd.dll','$1,456.99'))
SHOW VAR v1
```

Result: v1= one thousand, four hundred fifty six dollars and ninety nine cents

Source Code

What follows is the source code for the dwrd.exe sample UDF, shipped with DOS products, and the source code for the dwrd.dll sample UDF, shipped with Windows products.

```
/*
 *     Copyright 2001 by R:BASE Technologies, Inc.
 *    Author:  Wayne J. Erickson  12 Apr 1991
 *
 ************************************************************************
 * *    Routine:  dwrd2 *** DOS VERSION ***
 * *    Purpose:  convert a currency value to words
 *
 *     Input Parameters:
 *           name                Brief description
 *           -------------       ------------------------------------------------
 *           value               Currency string to be parsed
 *            Shared Segment         Shared memory segment to place the parsed string
 *
 ************************************************************************
 */

#include
#include
#include
```

```
void conout(char *, int);
void  crlf(void);
int lenstr(char *, int);
void dwrd(char *, char *);

cdecl  main(argc,argv)
int argc;
char * argv[];
{
      long int i4;
      char *pt;
      char wordnum[160];
      int lw;

      /* Get the argument count.  */
      if(argc <= 1) {
            conout("DWRD number pointer", 19);
            crlf();
            goto done;
            }

      /* Do the conversion */
      dwrd(argv[1], wordnum);

      /* Display the result */
      lw = strlen(wordnum);
      if (argc == 2) {
            conout(wordnum, lw);
            crlf();
      }

      /* See if there was an address to store the result */
      if(argc > 2) {
            i4 = atol(argv[2]);
            pt = (char *)i4;
            memcpy(pt, wordnum, lw+1);
      }
done:
      return(0);
}


/**** Start of DWRD function ****/
void dwrd(strin, strout)
char *strin;
char *strout;
{
      int lstr;
      static char numbers[20][10] = {
            "         ","one       ","two       ",
            "three     ","four      ","five      ",
            "six       ","seven     ","eight     ",
            "nine      ","ten       ","eleven    ",
            "twelve    ","thirteen  ","fourteen  ",
            "fifteen   ","sixteen   ","seventeen ",
            "eighteen  ","nineteen  "};

      static char tenvals[10][10] = {
            "no        ","ten       ","twenty    ",
            "thirty    ","forty     ","fifty     ",
            "sixty     ","seventy   ","eighty    ",
            "ninety    "};
      char blank = ' ';
```

```
        char comma = ',';
        char dsign = '$';
        char dot = '.';
        char minus = '-';
        char ch;
        char tmoney[16];
        int offset;
        int tens;
        int lw, ls, n, intch, l;

        /*  CONVERT THE NUMBER. */

        tens = 0;
        memset(tmoney, ' ', 16);
        lw = 0;
        lstr = strlen(strin);
        ls = 16 - lstr + 1;
        strcpy(&tmoney[ls-1],strin);

        /*  PICK OFF THE CHARACTERS. */

        n = ls - 1;
next_digit:
        n++;
        if(n > 16) goto cleanup;
        ch = tmoney[n-1];

        /*  SKIP THE COSMETIC CHARACTERS. */

        if((ch == blank) || (ch == comma) || (ch == dsign)) goto next_digit;
        if(ch == minus) {
                memcpy(&strout[(lw+1)-1], "minus ", 6);
                lw = lw + 6;
                goto next_digit;
                }
        if((n == 4) || (n == 8) || (n == 12)) tens = 1;

        /*  SPECIAL STUFF WHEN WE HIT THE DECIMAL POINT. */

        if(ch == dot) {
                tens = 1;
                if(lw == 0) {
                        memcpy(&strout[(lw+1)-1], "zero ", 5);
                        lw = lw + 5;
                        }
                memcpy(&strout[(lw+1)-1], "dollars and ", 12);
                lw = lw + 12;
                if(tmoney[(n+1)-1] == '0') {
                        if(tmoney[(n+2)-1] == '0') {
                                memcpy(&strout[(lw+1)-1], "zero ", 5);
                                lw = lw + 5;
                                n = n + 2;
                                }
                        }
                goto next_digit;
                }

        /*  CONVERT A CHARACTER INTO AN OFFSET. */

        intch = ch - 48;
        if(tens) {
                if(intch <= 1) {
                        offset = 0;
```

```
                if(intch == 1) offset = 10;
                n = n + 1;
                ch = tmoney[(n)-1];
                intch = ch - 48 + offset;
                }
        else {
                l = lenstr(tenvals[(intch+1) - 1], 10);
                if(l > 0) {
                        memcpy(&strout[(lw+1)-1], tenvals[(intch+1)-1], l+1);
                        lw = lw + l + 1;
                        }
                tens = 0;
                goto next_digit;
                }
        }
    l = lenstr(numbers[(intch+1)-1], 10);
    if(l > 0) {
            memcpy(&strout[(lw+1)-1], numbers[(intch+1)-1], l+1);
            lw = lw + l + 1;
            }
    tens = 0;
    if(((n == 3) || (n == 7) || (n == 11)) && (intch != 0)) {
            memcpy(&strout[(lw+1)-1], "hundred ", 8);
            lw = lw + 8;
            }
    if(n == 5) {
            memcpy(&strout[(lw+1)-1], "million ", 8);
            lw = lw + 8;
            }
    if(n == 9) {
            memcpy(&strout[(lw+1)-1], "thousand ", 9);
            lw = lw + 9;
            }
    goto next_digit;
cleanup:
    memcpy(&strout[(lw+1)-1], "cents", 5);
    lw = lw + 5;

    /* Add the null terminator */
    strout[(lw+1)-1] = '\0';

    /*    ALL DONE.  */

    return;
}
/**** End of DWRD function ****/


/**** Start of CONOUT function ****/
void conout(str, count)
char *str;
int count;
{
    int i;
    for (i = 0; i < count; i++) {
            putchar(str[i]);
            }
    return;
}
/**** End of CONOUT function ****/

/**** Start of CRLF function ****/
void crlf()
{
```

```
	putchar('\r');
	putchar('\n');
	return;
}
/**** End of CRLF function ****/

/**** Start of LENSTR function ****/
int lenstr(str, count)
char *str;
int count;
{
	int i;

	i = count - 1;
	while (i >= 0) {
		if (str[i] != ' ') break;
		i--;
		}
	return (i+1);
}
/**** End of LENSTR function ****/



#include
/*
 *     Copyright 2001 R:BASE Technologies, Inc.
 *     Author:  Wayne J. Erickson  18 October 1999
 *
 ************************************************************************
 *
 *     Routine:  dwrd  *** DLL VERSION ***
 *
 *     Purpose:  convert a currency value to words
 *
 *     Input Parameters:
 *           name                Brief description
 *           --------------      ------------------------------------------------
 *           value               Currency string to be parsed
 *            Shared Segment         Shared memory segment to place the parsed string
 *
 *
 ************************************************************************
 */

#include
void dwrd(char *, char *);
int lenstr(char *, int);

#pragma  hdrstop

//---------------------------------------------------------------------------
int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void*)
{
	return 1;
}

//---------------------------------------------------------------------------
extern "C" __declspec(dllexport) int WINAPI RbDllExec(const HWND dc,const char *in, int inlen, char
*out, int outlen)
{
	int returnValue = 0;
	char instr[80];
```

```
        char wordnum[160];

        // Put the input string in a null terminated string
        memcpy(instr, in, inlen);
        instr[inlen] = '\0';

        // Convert to a text string
        dwrd(instr, wordnum);

        // Return the result
        outlen = strlen(wordnum);
        if (outlen)
                memcpy(out, wordnum, outlen);
         // returnValue = MessageBox(dc, wordnum,"Result value",MB_OK);
        // returnValue = 0;
        return returnValue;
}

//---------------------------------------------------------------------------
extern "C" __declspec(dllexport) int WINAPI RbDllVersion(void)
{
  return 100;
}

/**** Start of DWRD function ****/
void dwrd(char *strin,char *strout)
{
        int lstr;
        static char numbers[20][11] = {
                "         ","one       ","two       ",
                "three     ","four      ","five      ",
                "six       ","seven     ","eight     ",
                "nine      ","ten       ","eleven    ",
                "twelve    ","thirteen  ","fourteen  ",
                "fifteen   ","sixteen   ","seventeen ",
                "eighteen  ","nineteen  "};

        static char tenvals[10][11] = {
                "no        ","ten       ","twenty    ",
                "thirty    ","forty     ","fifty     ",
                "sixty     ","seventy   ","eighty    ",
                "ninety    "};
        char blank = ' ';
        char comma = ',';
        char dsign = '$';
        char dot = '.';
        char minus = '-';
        char ch;
        char tmoney[16];
        int offset;
        int tens;
        int lw, ls, n, intch, l;

        /*  CONVERT THE NUMBER. */

        tens = 0;
        memset(tmoney, ' ', 16);
        lw = 0;
        lstr = strlen(strin);
        ls = 16 - lstr + 1;
        strcpy(&tmoney[ls-1],strin);

        /*  PICK OFF THE CHARACTERS. */
```

```
	n = ls - 1;
next_digit:
	n++;
	if(n > 16) goto cleanup;
	ch = tmoney[n-1];

	 /*  SKIP THE COSMETIC CHARACTERS. */

	if((ch == blank) || (ch == comma) || (ch == dsign)) goto next_digit;
	if(ch == minus) {
		memcpy(&strout[(lw+1)-1], "minus ", 6);
		lw = lw + 6;
		goto next_digit;
		}
	if((n == 4) || (n == 8) || (n == 12)) tens = 1;

	 /*  SPECIAL STUFF WHEN WE HIT THE DECIMAL POINT. */

	if(ch == dot) {
		tens = 1;
		if(lw == 0) {
			memcpy(&strout[(lw+1)-1], "zero ", 5);
			lw = lw + 5;
			}
		memcpy(&strout[(lw+1)-1], "dollars and ", 12);
		lw = lw + 12;
		if(tmoney[(n+1)-1] == '0') {
			if(tmoney[(n+2)-1] == '0') {
				memcpy(&strout[(lw+1)-1], "zero ", 5);
				lw = lw + 5;
				n = n + 2;
				}
			}
		goto next_digit;
		}

	 /*  CONVERT A CHARACTER INTO AN OFFSET. */

	intch = ch - 48;
	if(tens) {
		if(intch <= 1) {
			offset = 0;
			if(intch == 1) offset = 10;
			n = n + 1;
			ch = tmoney[(n)-1];
			intch = ch - 48 + offset;
			}
		else {
			l = lenstr(tenvals[(intch+1) - 1], 10);
			if(l > 0) {
				memcpy(&strout[(lw+1)-1], tenvals[(intch+1)-1], l+1);
				lw = lw + l + 1;
				}
			tens = 0;
			goto next_digit;
			}
		}
	l = lenstr(numbers[(intch+1)-1], 10);
	if(l > 0) {
		memcpy(&strout[(lw+1)-1], numbers[(intch+1)-1], l+1);
		lw = lw + l + 1;
		}
	tens = 0;
```

```
        if(((n == 3) || (n == 7) || (n == 11)) && (intch != 0)) {
                memcpy(&strout[(lw+1)-1], "hundred ", 8);
                lw = lw + 8;
                }
        if(n == 5) {
                memcpy(&strout[(lw+1)-1], "million ", 8);
                lw = lw + 8;
                }
        if(n == 9) {
                memcpy(&strout[(lw+1)-1], "thousand ", 9);
                lw = lw + 9;
                }
        goto next_digit;
cleanup:
        memcpy(&strout[(lw+1)-1], "cents", 5);
        lw = lw + 5;

        /* Add the null terminator */
        strout[(lw+1)-1] = '\0';

        /*    ALL DONE.  */

        return;
}
/**** End of DWRD function ****/

/**** Start of LENSTR function ****/
int lenstr(char *str, int count)
{
        int i;

        i = count - 1;
        while (i >= 0) {
                if (str[i] != ' ') break;
                i--;
                }
        return (i+1);
}
/**** End of LENSTR function ****/
```

Making and Setting DLL

You can use a DLL (Dynamic Link Library) which you make for use in R:BASE. DLLs which you make can be used column and variable objects in Forms, Reports and Labels.

If you make a DLL, you must follow some rules of making DLLs.

Rules of making DLLs

Value of DLL's version

- int WINAPI RbDllVersion(void);
- You must specify to return 100.

DLL name which is displayed in the Combo Box

- const char* WINAPI RbDllSpec(void);
- You designate the DLL function's name which is displayed in the Combo Box.

DLL's ID

- int WINAPI RbDllId(void);
- It is the identity number of the DLL. You designate to return a number form 2 to 127.

The function called by Form when drawing field

- int WINAPI RbDllDraw(const char*, const HDC, long, long, long, long)
- Return value : 0 success / Not 0 fails
- Argument : const char* Data strings
- const HDC Device Context handle
- long Left upper corner X position of object
- long Left upper corner Y position of object
- long Right lower corner X position of object
- long Right lower corner Y position of object

The function called by Report or Label when drawing field

- int WINAPI RbDllDrawP(const char*, const HDC, long, long, long, long)
- Return value : 0 success / Not 0 fails
- Argument : const char* Data strings
- const HDC Device Context handle
- long Left upper corner X position of object
- long Left upper corner Y position of object
- long Right lower corner X position of object
- long Right lower corner Y position of object

1.19.2 ULC

(ULC(*text*))

Converts *text* from uppercase to lowercase, returning a text string.

In the following example, the value of *vulc* is *abcde*. To ensure that your data is consistent, whether it is imported from outside R:BASE or entered through an R:BASE form, use ULC to convert text fields to lowercase.

```
SET VAR vulc = (ULC('ABCDE'))
```

1.20 W

1.20.1 WINUDF

(WINUDF('exe-name','parameter-list'))
(WINUDF('-exe-name','parameter-list'))
(WINUDF('+exe-name','parameter-list'))
(WINUDF('@dll-name','parameter-list'))

A new style of udf is available which should allow people to write UDF's in most any language they want. This results in a new function called WINUDF which has the same function parameters as the UDF function. The major difference is how R:BASE communicates with the WINUDF function. The exe program that WINUDF calls is passed one parameter which is the name of a file. This file has the parameters that should be passed to the WINUDF program. When the WINUDF program is completed, it is expected to write its results to the file that passed the original parameters.

Here are simple batch files that allow you to create WINUDF programs with at least 3 different C/C++ compilers. You could also use Visual Basic to create these new WINUDF's.

Using Visual Studio:

set path=c:\Program Files\Microsoft Visual Studio\VC98\Bin;c:\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin;%path%

```
set include=c:\Program Files\Microsoft Visual Studio\VC98\Include
set lib=c:\Program Files\Microsoft Visual Studio\VC98\Lib
cl /nologo /ML /W3 /Gm /GX /ZI /Od /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /D
"_MBCS" /GZ /c DWRD.C
LINK /MAP:DWRD.MAP /SUBSYSTEM:CONSOLE DWRD.OBJ
```

Using Borland:

```
set path=c:\f\bcc55\bin;%path%
set include=c:\f\bcc55\include
set lib=c:\f\bcc55\lib
bcc32 -Ic:\f\bcc55\include -Lc:\f\bcc55\lib -M DWRD.C
```

Using Microsoft C:

```
set  path=c:\f\msc6;C:\f\msc6\bin;c:\f\msc6\binb;%path%
set include=c:\f\msc6\include
set lib=c:\f\msc6\lib
ERASE SMALL.EXE
cl -c /Gc /Oas /AL /DLINT_ARGS DWRD.C
LINK @DWRD.LNK
EXEPACK DWRD.EXE SMALL.EXE
ERASE DWRD.EXE
REN SMALL.EXE DWRD.EXE
```

Examples:

```
SET VAR vTestA = (WINUDF('wscript.exe scrudf.vbs','The Power!'))
SET VAR vTestB = (WINUDF('wscript.exe scrudf.vbs','Oh Yes!'))
```

Index

- A -

ABS 3
ACOS 3
ADDDAY 3
ADDFRC 3
ADDHR 3
ADDMIN 3
ADDMON 3
ADDSEC 4
ADDYR 4
AINT 4
Aligning Decimals 34
AND 9
ANINT 4
ANSI 9
ASIN 4
ATAN 4
ATAN2 4
AUTOCOMMIT 9
AUTODROP 9
Autonumber 57
AUTOSKIP 10

- B -

BELL 10
BLANK 10
Block 15
BRND 5
BUILD 10

- C -

CASE 10
CHAR 5
Check Message Status 37
CHKCUR 5
CHKFILE 5
CHKFUNC 5
CHKKEY 6
CHKVAR 6
CLEAR 10
CLIPBOARDTEXT 10
COLOR 10
COMPUTER 11
CONNECTIONS 11
COS 6
COSH 6
CTR 6
CTXT 7
CURRDIR 11
CURRDRV 11
CURRENCY 11
CURRENTPRINTER 11
CURRNUMALOC 47
CURSORCOL 47
CURSORROW 47
CVAL 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
CVTYPE 24

- D -

DATABASE 11
DATE 12
DATE CENTURY 12
DATE FORMAT 12
DATE SEQUENCE 12
DATE YEAR 12
DATETIME 24
DBPATH 12
DBSIZE 47
DEBUG 12
DELFUNC 25
DELIMIT 13
DEXTRACT 25
DIM 25
DISKSPACE 47
DLCALL 25
DLFREE 30
DLLOAD 31
DNW 31
DRIVES 13
DWE 31
DWRD 31

- E -

ECHO 13

EDITOR 13
ENVVAL 32
EOFCHAR 13
EQNULL 13
ERROR 14
ERROR DETAIL 14
Error Message 37
ERROR VARIABLE 14
ERRORLEVEL 69
ESCAPE 14
EXP 32
EXPLODE 14

- F -

FASTFK 14
FASTLOCK 14
FEEDBACK 14
FILENAME 32
FILES 15
FINDFILE 32
FIXED 15
FLOAT 32
FORM_CONTROL_TYPE 48
FORM_DIRTY_FLAG 48
FORMAT 33, 34, 35
Formatting Currency 35
Formatting Text 35
Function 61, 62
Function Categories 2
Functions 2, 3, 4, 5, 6, 7, 24, 25, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 78
FV1 36
FV2 36

- G -

GETDATE 36
GetDriveReady 37
GetIPAddress 37
GETKEY 36
GetLock 38
GetMACAddr 39
GETVAL 37, 38, 39, 40, 41
GetVolumeID 40

- H -

HEADINGS 15
HTML 41

- I -

ICAP 41
ICAP1 41
ICAP2 41
ICHAR 41
IDAY 42
IDIM 42
IDOY 42
IDQUOTES 15
IDWK 42
IFEQ 42
IFEXISTS 43
IFGT 43
IFLT 43
IFNULL 43
IFRC 43
IFWINDOW 43
IHASH 44
IHR 45
ILY 45
IMIN 45
IMON 45
INSERT 15
INT 45
INTENSITY 15
INTERVAL 15
ISALPHA 46
ISDIGIT 46
ISEC 46
ISLOWER 46
ISRUNTIME 48
ISSPACE 46
ISTAT 46, 47, 48, 49, 50, 51, 52
ISTR 52
ISUPPER 52
ITEMCNT 52
IWOY 53
IYR 53
IYR4 53

- J -

JDATE 54

- L -

Last 15
LAST ERROR 15
LastBlock 15
LastBlockTable 15
LASTKEY 54
LAVG 55
LAYOUT 16
LIMITNUMALLOC 48
LINEEND 16
LINES 16
LJS 55
LMAX 55
LMIN 55
LOG 56
LOG10 56
LOOKUP 16
LTRIM 56
LUC 56

- M -

MAC Address 39
MANOPT 16
MANY 16
MAXFREE 48
MAXNUMALLOC 49
MAXTRANS 16
MDI 16
MEMORY 49
Memory Issues 69
MESSAGES 17
MIRROR 17
MOD 56
MOUSECOL 49
MOUSEROW 49
MULTI 17

- N -

NAME 17
NETUSER 17
NEXT 57
NINT 57
NOTE_PAD 17
NULL 17

- O -

OFFMESS 17
OLDLINE 17
ONELINE 18

- P -

PAGECOL 49
PAGEMODE 18
PAGEROW 49
PASSTHROUGH 18
PLATFORM 18
PlayAndExit 40
PlayAndWait 41
PLUS 18
PMT1 57
PMT2 57
port 18
ports 18
POSFIXED 18
PRINTERS 18
PRN_COLLATION 19
PRN_COLORMODE 19
PRN_COPIES 19
PRN_DUPLEXMODE 19
PRN_ORIENTATION 19
PRN_QUALITY 19
PRN_SIZE 19
PRN_SOURCE 19
PRN_STATUS 19
Punctuating Long Numbers 35
PV1 58
PV2 58

- Q -

QUALCOLS 19
QUOTES 20

- R -

RANDOM 58
RATE1 58
RATE2 58
RATE3 59
RB1SIZE 49
RB2SIZE 50
RB3SIZE 50
RB4SIZE 50
RDATE 59
REFRESH 20
Returning Multiple Values 69
REVERSE 20
RJS 59
ROUND 59
ROWLOCKS 20
RTIME 60
RTRIM 60
RULES 20
RWP 60
RX1SIZE 50
RX2SIZE 50
RX3SIZE 51
RX4SIZE 51

- S -

Sample UDF 69
SCRATCH 20
SCREENSIZE 20
SELMARGIN 20
SEMI 20
SERVER 20
SFIL 61
SGET 61
SIGN 61
SIN 61
SINGLE 20
SINH 61
SKEEP 61
SKEEPI 62
SLEN 62
SLOC 62
SLOCP 63
SMOVE 63
SORT 21
SORTMENU 21
SOUNDEX 63
SPUT 63
SQRT 64
SRPL 64
SSTRIP 64, 65
SSUB 65
STATICDB 21
STRIM 66

- T -

TAN 66
TANH 66
TDWK 66
TERM1 66
TERM2 66
TERM3 67
TEXTRACT 67
TIME 21
TIMEFORMAT 21
TIMEOUT 21
TIMESEQUENCE 21
TMON 67
TOLERANCE 21
TOTALALLOC 51
TOTALFREE 51
TOTALLOCKS 51
TOTALREADS 52
TOTALWRITES 52
TRACE 21
TRANSACT 21
TRIM 67

- U -

UDF 67, 69
ULC 78
USER 21
USERAPP 22
User-Defined Functions 67, 69, 78
USERDOMAIN 22
USERID 22

- V -

VERIFY 22
VERSION 22
VERSION BITS 22
VERSION BUILD 22
VERSION SYSTEM 23

- W -

WAIT 23
WALKMENU 23
WHILEOPT 23
WIDTH 23
WINBEEP 23
WINDOWSPRINTER 23
WINUDF 78
WRAP 23
WRITECHK 23

- Z -

ZERO 23
ZOOMEDIT 24

Notes