

RETURN TO MAIN MENU

Recital/4GL Functions

Recital/4GL Functions

Recital Corporation,
100 Cummings Center, Suite 318J
Beverly, MA 01915

Recital may have patents and/or patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.

COPYRIGHT ©1988-2006 Recital Corporation. All rights reserved. All Recital products are trademarks or registered trademarks of Recital Corporation, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Last Updated June, 2006

INDEX

&	1
AADD()	2
AAVERAGE()	3
ABROWSE()	4
ABS()	7
ACC()	8
ACCESS()	9
ACHOICE()	10
AClass()	12
ACOL()	13
ACOPY()	14
ACOS()	15
ACTIVEPID()	16
ADATABASES()	17
ADDBS()	18
ADDPROPERTY()	19
ADEL()	20
ADESC()	21
ADIR()	22
AELEMENT()	23
AFIELDS()	24
AFILL()	26
AFONT()	27
AINS()	28
ALEN()	29
ALIAS()	30
ALLTRIM()	31
ALTD()	32
AMAX()	33
AMEMBERS()	34
AMIN()	35
AMPM()	36
AROW()	37
ASC()	38
ASCAN()	39
ASIN()	40
ASIZE()	41
ASORT()	42
ASTORE()	43
ASTRING()	44
ASUBSCRIPT()	45
ASUM()	46
AT()	47
ATAN()	48
ATN2()	49
ATNEXT()	50
AVGVALUES()	51

BAR()	52
BARCOUNT()	53
BARPROMPT()	54
BASENAME()	55
BETWEEN()	56
BIN2I()	57
BIN2L()	58
BIN2W()	59
BINCLOSE()	60
BINCREATE()	61
BINOPEN()	62
BINREAD()	63
BINSEEK()	64
BINWRITE()	65
BITAND()	66
BITCLEAR()	67
BITLSHIFT()	68
BITNOT()	69
BITOR()	70
BITRSHIFT()	71
BITSET()	72
BITTEST()	73
BITXOR()	74
BLINK()	75
BOF()	76
BOLD()	77
BRIDGE()	78
CAGR()	79
CALC()	80
CANCELPID()	81
CANDIDATE()	82
CAPSLOCK()	83
CAST()	84
CATALOG()	86
CDOW()	87
CDX()	88
CEILING()	89
CENTER()	90
CERROR()	91
CHANGE()	92
CHR()	93
CHROVERLAP()	94
CHRTRAN()	95
CLEARRESULTSET()	96
CLOSEMAIL()	97
CMONTH()	98
CNTVALUES()	99
COL()	100
COLLATE()	101
COMPLETED()	102
COMPOBJ()	103

CONNECTED()	105
COUNTMAIL()	109
COPYFILEFROM()	106
COPYFILETO()	107
COS()	108
CREATEOBJECT()	110
CTOD()	111
CTRL()	113
CURDIR()	114
CURRENCY()	115
CURSEQNO()	116
CURSORNAME()	117
DATABASE()	118
DATE()	119
DATETIME()	120
DAY()	121
DAYS()	122
DBEDIT()	123
DBF()	125
DBFILTER()	126
DBRELATION()	127
DBRSELECT()	128
DBUSED()	129
DBXDESCEND()	130
DEFAULT()	131
DELETED()	132
DELETEMAIL()	133
DESCEND()	134
DESCENDING()	135
DIFFERENCE()	136
DIR()	137
DISKSPACE()	138
DMY()	139
DODEFAULT()	140
DOLEVEL()	141
DOSERROR()	142
DOW()	143
DTOC()	144
DTOM()	145
DTOR()	146
DTOS()	147
DOV()	148
EDITFIELD()	149
ELAPTIME()	150
EMPTY()	151
EOF()	152
EPOCH()	153
ERRNO()	154
ERROR()	155
ERRORLEVEL()	156
ETOS()	157

EXCLUSIVE()	158
EXEC()	159
EXECSRIPT()	160
EXP()	161
EXPRCHECK()	162
FCLOSE()	163
FCOUNT()	164
FCREATE()	165
FDATE()	166
FEOF()	167
FERROR()	168
FFLUSH()	169
FGETS()	170
FIELD()	171
FIELDNAME()	172
FILE()	173
FILECOUNT()	174
FILEINFO()	175
FILETOSTR()	176
FILETYPE()	177
FILTER()	178
FINDCURSOR()	179
FIXED()	180
FKLABEL()	181
FKMAX()	182
FLDCOUNT()	183
FLDLIST()	184
FLOAT()	185
FLOCK()	186
FLOOR()	187
FMT()	188
FONTMETRIC()	189
FOPEN()	191
FOR()	192
FOUND()	193
FPUTS()	194
FREAD()	195
FREADSTR()	196
FSEEK()	197
FSIZE()	198
FTIME()	199
FULLPATH()	200
FV()	201
FWRITE()	202
GATEWAY()	203
GETENV()	204
GETFILE()	205
GETFONT()	206
GETGID()	207
GETGRNAM()	208
GETLOCALHOST()	209

GETLOG()	210
GETNO()	211
GETPID()	212
GETPROT()	213
GETREMOTEADDR()	214
GETREMOTEHOST()	215
GETREMOTEUSER()	216
GETRESULTSET()	217
GETSIG()	218
GETUID()	219
GOMONTH()	220
GOTO()	221
HARDCR()	222
HEADER()	223
HOME()	224
HOUR()	225
HOURS()	226
HSCROLL()	227
I2BIN()	228
ICACHE()	229
ICASE()	230
ID()	231
IF()	232
IFILECOUNT()	233
IIF()	234
INDEXEXT()	235
INDEXKEY()	236
INDEXORDER()	237
INKEY()	238
INLIST()	240
INSMODE()	241
INT()	242
INUSE()	243
IOSTATS()	244
ISALPHA()	245
ISBLANK()	246
ISCOLOR()	247
ISDIGIT()	248
ISFILTERED()	249
ISLOCKED()	250
ISLOWER()	251
ISMARKED()	252
ISMIRAGE()	253
ISMOUSE()	254
ISPRINTER()	256
ISSERVER()	257
ISUPPER()	258
KEY()	259
KEYMATCH()	260
L2BIN()	261
LASTKEY()	262

LASTREC()	263
LEFT()	264
LEN()	265
LENNUM()	266
LIKE()	267
LKSYS()	268
LINENO()	269
LOCK()	270
LOG()	271
LOG10()	272
LOOKUP()	273
LOWER()	274
LPAD()	275
LTOS()	276
LTRIM()	277
LUPDATE()	278
MAIL()	279
MAILCLOSE()	280
MAILCOUNT()	281
MAILDELETE()	282
MAILERROR()	283
MAILHEADER()	284
MAILNODENAME()	285
MAIOPEN()	286
MAILREAD()	287
MAILSEND()	288
MAILUSERNAME()	290
MAX()	291
MAXCOL()	292
MAXROW()	293
MAXVALUES()	294
MCOL()	295
MDX()	296
MDY()	297
MEMLINES()	298
MEMOEDIT()	299
MEMOLINE()	301
MEMOREAD()	302
MEMORY()	303
MEMOSAY()	304
MEMOTRAN()	305
MEMOWRITE()	306
MENU()	307
MENUITEM()	308
MESSAGE()	309
MESSAGEBOX()	310
MIN()	312
MINUTE()	313
MINUTES()	314
MINVALUES()	315
MLCOUNT()	316

MLINE()	317
MOD()	318
MONTH()	319
MROW()	320
MTOD()	321
MTOS()	322
NDX()	323
NETERR()	324
NETNAME()	325
NEWOBJECT()	326
NETWORK()	327
NEXTKEY()	328
NFCOUNT()	329
NUMLOCK()	330
NVL()	331
OBJECTREAD()	332
OBJECTTYPE()	333
OBJECTWRITE()	334
OCCURS()	335
ON()	336
OPENMAIL()	337
ORDER()	338
OS()	339
PAD()	340
PADC()	341
PADL()	342
PADPROMPT()	343
PADR()	344
PARAMETERS()	345
PATH()	346
PAYMENT()	347
PCOL()	348
PCOUNT()	349
PERCENT()	350
PEXIST()	351
PI()	352
PMT()	353
POPUP()	354
PRINTFILE()	355
PRINTSCREEN()	357
PRINTSTATUS()	358
PRMBAR()	359
PROCLIBS()	360
PROCLINE()	361
PROCNAME()	362
PROGRAM()	363
PROMPT()	364
PROPER()	365
PROW()	366
PUTENV()	367
PUTFILE()	368

PUTLOG()	369
PV()	370
QUARTER()	371
RAND()	372
RANK()	373
RAT()	374
READEXIT()	375
READINSERT()	376
READKEY()	377
READMAIL()	378
READMODE()	379
READVAR()	380
RECCOUNT()	381
RECNO()	382
RECSIZE()	383
RELATION()	386
REMOVEPROPERTY()	385
REPLACE()	387
REPLICATE()	388
RESTSCREEN()	389
REVERSE()	390
RIGHT()	391
RLOCK()	392
RLOOKUP()	393
ROLLBACK()	394
ROUND()	395
ROW()	396
RPAD()	397
RTOD()	398
RTRIM()	399
RUN()	400
SAVESCREEN()	401
SCHEME()	402
SCOLS()	403
SCROLL()	404
SEC()	405
SECONDS()	406
SECS()	407
SEEK()	408
SELECT()	409
SENDMAIL()	410
SEQNO()	412
SET()	413
SETCANCEL()	414
SETCOLOR()	415
SETPRC()	416
SETPROT()	417
SETRESULTSET()	418
SHOWDOCUMENT()	419
SIGN()	421
SIN()	422

SOUNDEX()	423
SPACE()	424
SPAWNPID()	425
SQLVALUES()	426
SQRT()	427
SROWS()	428
STOD()	429
STR()	430
STRCOMPARE()	431
STREXTRACT()	432
STRTOFILE()	433
STRTRAN()	435
STRZERO()	436
STUFF()	437
SUBSTR()	438
SUMVALUES()	439
SYS()	440
TAG()	442
TAGCOUNT()	443
TAGNO()	444
TAN()	445
TARGET()	446
TEXTEDIT()	447
TIME()	449
TIMESTAMP()	450
TMPNAM()	451
TONE()	452
TRANSFORM()	453
TRIM()	454
TSPOS()	455
TSTRING()	456
TSWORD()	457
TTOC()	459
TTOD()	460
TXNISOLATION()	461
TXNLEVEL()	462
TYPE()	463
UNDERLINE()	465
UNIQUE()	466
UNIQUEROWID()	467
UPDATED()	468
UPPER()	469
USED()	470
USER()	471
USERLOG()	472
VAL()	473
VALIDTIME()	474
VARREAD()	475
VARTYPE()	476
VERSION()	478
VTOD()	479

WCOLS()	480
WEEK()	481
WEXIST()	483
WFONT()	484
WINDOW()	485
WONTOP()	486
WORKAREA()	487
WOUTPUT()	488
WROWS()	489
WTITLE()	490
WVISIBLE()	491
XMLCOUNT()	492
XMLCREATEDTD()	493
XMLFIRST()	494
XMLNEXT()	495
XMLVALIDATE()	496
YEAR()	497

Functions by Category

Applications

&	DOLEVEL()	EXEC()
EXECSCRIPT()	ICASE()	IF()
IIF()	PARAMETERS()	PCOUNT()
PEXIST()	PROCLINE()	PROCNAME()
PROGRAM()	RUN()	

Array Processing

AADD()	AAVERAGE()	ABROWSE()
ACHOICE()	ACOL()	ACOPY()
ADEL()	ADESC()	ADIR()
AELEMENT()	AFIELDS()	AFILL()
AINS()	ALEN()	AMAX()
AMIN()	AROW()	ASCAN()
ASIZE()	ASORT()	ASTORE()
ASTRING()	ASUBSCRIPT()	ASUM()

ASCII File Access

FCLOSE()	FCREATE()	FEOF()
FERROR()	FFLUSH()	FGETS()
FOPEN()	FPUTS()	FREAD()
FREADSTR()	FSEEK()	FWRITE()

Binary File Access

BINCLOSE()	BINCREATE()	BINOPEN()
BINREAD()	BINSEEK()	BINWRITE()

Bitwise Operations

BITAND()	BITCLEAR()	BITLSHIFT()
BITNOT()	BITOR()	BITRSHIFT()
BITSET()	BITTEST()	BITXOR()

Data Connectivity

BRIDGE()	CLEARRESULTSET()	CONNECTED()
CURSORNAME()	FINDCURSOR()	GATEWAY()
GETLOCALHOST()	GETREMOTEADDR()	GETREMOTEHOST()
GETREMOTEUSER()	GETRESULTSET()	SETRESULTSET()
SQLVALUES()	UNIQUEROWID()	

Databases

ADATABASES()	DATABASE()	DBUSED()
--------------	------------	----------

Date and Time Data

AMPM()	CDOW()	CMONTH()
CTOD()	CTOT()	DATE()
DATETIME()	DAY()	DAYS()
DMY()	DOW()	DTOC()
DTOM()	DTOS()	DTOV()
ELAPTIME()	EPOCH()	GOMONTH()
HOUR()	HOURS()	MDY()
MINUTE()	MINUTES()	MONTH()
MTOD()	QUARTER()	SEC()
SECONDS()	SECS()	STOD()
TIME()	TIMESTAMP()	TSTRING()
TTOC()	TTOD()	VALIDTIME()
VTOD()	WEEK()	YEAR()

Disk and File Utilities

ADDBS()	BASENAME()	COPYFILEFROM()
COPYFILETO()	CURDIR()	DEFAULT()
DISKSPACE()	FDATE()	FILE()
FILECOUNT()	FILEINFO()	FSIZE()
FTIME()	FULLPATH()	GETFILE()
HEADER()	HOME()	IFILECOUNT()
LUPDATE()	MAIL()	NETNAME()
NETWORK()	OS()	PATH()
PROCLIBS()	PUTFILE()	RECSIZE()
SHOWDOCUMENT()	TMPNAM()	

Environment

ACCESS()	ACTIVEPID()	CANCELPID()
EXCLUSIVE()	GETENV()	GETGID()
GETGRNAM()	GETLOG()	GETPID()
GETPROT()	GETUID()	ID()
ISMIRAGE()	ISSERVER()	MEMORY()
ON()	PUTENV()	PUTLOG()
SET()	SETPROT()	SPAWNPID()
SYS()	USED()	VERSION()
WORKAREA()		

Error Handling and Debugging

CERROR()	DOSERROR()	ERRNO()
ERROR()	ERRORLEVEL()	GETSIG()
LINENO()	MESSAGE()	NETERR()
USERLOG()		

Expressions and Type Conversion

ASC()	BETWEEN()	BIN2I()
BIN2L()	BIN2W()	CAST()
CHR()	CHROVERLAP()	CTOD()
CTOT()	CTRL()	DESCEND()
DTOC()	DTOM()	DTOR()
DTOS()	DOV()	EMPTY()
ETOS()	EXPRCHECK()	HARDCR()
I2BIN()	ISBLANK()	ISNULL()
L2BIN()	LEN()	LTOS()
MTOD()	MTOS()	NVL()
RANK()	STOD()	STR()
STRCOMPARE()	STRZERO()	TSTRING()
TTOC()	TTOD()	TYPE()
VAL()	VARTYPE()	VTOD()

Fields and Records

AVGVALUES()	BOF()	CNTVALUES()
DELETED()	EOF()	FCOUNT()
FIELD()	FIELDNAME()	FLDCOUNT()
FLDLIST()	FOUND()	GOTO()
LASTREC()	LKSYS()	LOOKUP()
MAXVALUES()	MINVALUES()	NFCOUNT()
RECCOUNT()	RECNO()	REFERENCES()
REPLACE()	RLOOKUP()	SUMVALUES()

Fonts

AFONT()	FONTMETRIC()	GETFONT()
WFONT()		

Indexing

CANDIDATE()	CDX()	COLLATE()
DBXDESCEND()	DESCENDING()	FOR()
ICACHE()	INDEXEXT()	INDEXKEY()
INDEXORDER()	KEY()	KEYMATCH()
MDX()	NDX()	ORDER()
SEEK()	TAG()	TAGCOUNT()
TAGNO()	UNIQUE()	

Information Center

CATALOG()		
-----------	--	--

Input / Output

ACC()	BLINK()	BOLD()
CALC()	MESSAGEBOX()	REVERSE()
TONE()	UNDERLINE()	

Keyboard Events

INKEY()	INSMODE()	LASTKEY()
NEXTKEY()		

Mail

CLOSEMAIL()	COUNTMAIL()	DELETEMAIL()
-------------	-------------	--------------

MAILCLOSE()	MAILCOUNT()	MAILDELETE()
MAILERROR()	MAILHEADER()	MAILNODENAME()
MAILOPEN()	MAILREAD()	MAILSEND()
MAILUSERNAME()	OPENMAIL()	READMAIL()
SENDMAIL()		

Manual Locking

CHANGE()	FLOCK()	ISLOCKED()
LOCK()	RLOCK()	

Memos

MEMLINES()	MEMOEDIT()	MEMOLINE()
MEMOREAD()	MEMOSAY()	MEMOTRAN()
MEMOWRITE()	MLCOUNT()	MLINE()

Menus

BAR()	BARCOUNT()	BARPROMPT()
MENU()	MENUITEM()	PAD()
PADPROMPT()	POPUP()	PRMBAR()
PROMPT()		

Numeric Data

ABS()	ACOS()	ASIN()
ATAN()	ATN2()	CAGR()
CEILING()	COS()	CURRENCY()
EXP()	FIXED()	FLOAT()
FLOOR()	FV()	INT()
LENNUM()	LOG()	LOG10()
MAX()	MIN()	MOD()
PAYMENT()	PERCENT()	PI()
PMT()	PV()	RAND()
ROUND()	RTOD()	SIGN()
SIN()	SQRT()	TAN()

Objects

AClass()	ADDPROPERTY()	AMEMBERS()
COMPOBJ()	CREATEOBJECT()	DODEFAULT()
NEWOBJECT()	OBJECTREAD()	OBJECTTYPE()
OBJECTWRITE()	REMOVEPROPERTY()	

Performance and Optimization

IOSTATS()		
-----------	--	--

Printing

ISPRINTER()	PCOL()	PRINTFILE()
PRINTSCREEN()	PRINTSTATUS()	PROW()
SETPRC()		

Screen Forms

COL()	DBEDIT()	EDITFIELD()
FKLABEL()	FKMAX()	FMT()
GETNO()	HSCROLL()	ISCOLOR()
ISMOUSE()	MAXCOL()	MAXROW()
MCOL()	MROW()	READEXIT()
READINSERT()	READKEY()	READMODE()
READVAR()	RESTSCREEN()	ROW()
SAVESCREEN()	SCHEME()	SCOLS()
SCROLL()	SETCOLOR()	SROWS()
TEXTEDIT()	UPDATED()	VARREAD()

Screen Windows

WCOLS()	WEXIST()	WINDOW()
WONTOP()	WOUTPUT()	WROWS()
WTITLE()	WVISIBLE()	

String Data

ALLTRIM()	AT()	ATNEXT()
CENTER() CENTRE()	CHRTRAN()	DIFFERENCE()
FILETOSTR()	INLIST()	ISALPHA()
ISDIGIT()	ISLOWER()	ISUPPER()
LEFT()	LIKE()	LOWER()
LPAD()	LTRIM()	OCCURS()
PADC()	PADL()	PADR()
PROPER()	RAT()	REPLICATE()
RIGHT()	RPAD()	RTRIM()
SOUNDEX()	SPACE()	STREXTRACT()
STRTOFILE()	STRTRAN()	STUFF()
SUBSTR()	TRANSFORM()	TRIM()
UPPER()		

Table Basics

ALIAS()	CURRSEQNO()	DBF()
DBFILTER()	DBRELATION()	DBRSELECT()
FILTER()	INUSE()	ISFILTERED()
RELATION()	SELECT()	SEQNO()
TARGET()	USED()	

Text Searching

TSPOS()	TSWORD()	
---------	----------	--

Transaction Processing

COMPLETED()	ISMARKED()	ROLLBACK()
TXNISOLATION()	TXNLEVEL()	

Xbase Compatibility

ALTD()	CAPSLOCK()	FILETYPE()
NUMLOCK()	SETCANCEL()	

XML

XMLCOUNT()	XMLCREATEDTD()	XMLFIRST()
XMLNEXT()	XMLVALIDATE()	

&

Class

Applications

Purpose

Function to perform macro substitution

Syntax

&<memvar> | (<exp>)

See Also

STORE, PRIVATE, PUBLIC, SET MACROS

Description

The & macro function substitutes the contents of the specified <memvar> into the command line. To use a macro in the middle of a word, it is necessary to end the variable name with a '!'. Any type of memory variable can be substituted as a macro. The & macro function can also substitute the result of an expression into the command line. The <exp> must be enclosed in round brackets.

Example

```
subscript = 10
i10i = 5
? i&subscript.i
5
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AADD()

Class

Array Processing

Purpose

Function to add a new element to the end of an array

Syntax

AADD(<array name>,<expr>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM(), AVERAGE()

Description

The AADD() function adds an element to the end of array specified in <array name>. The size of the <array name> array increases by one. The value of the new element is set to <expr>.

Example

```
declare array1[3]
array1 = "test"
? alen(array1)
3
aadd(array1,"newtest")
? array1[4]
newtest
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AAVERAGE()

Class

Array Processing

Purpose

Function to calculate the average of element values in an array

Syntax

AAVERAGE(<array>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM(), AVERAGE()

Description

The AAVERAGE() function calculates the average of all the numeric elements in the specified <array> and returns the result.

Example

```
use payroll
declare number[reccount(),1]
copy to array number fields pay_value
? aaverage(number)
1741.01
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ABROWSE()

Class

Array Processing

Purpose

Function to browse a two-dimensional array in a spreadsheet style format

Syntax

ABROWSE(<expN1>, <expN2>, <expN3>, <expN4>, <expC1> [,<expC2>] [,<expC3>] [,<expC4>]

See Also

ACOL(), AROW() DBEDIT()

Description

The ABROWSE() function browses a two dimensional array in a spreadsheet style format. This function is particularly useful with arrays that have been FETCHED or LOADED from SQL tables from a remote server.

The coordinates of the spreadsheet are defined with expressions <expN1> and <expN2>, which represent starting and ending row and column numbers respectively. The expression <expC1> is the name of the two dimensional array to be browsed. By default, only the [ABANDON] and [EXIT SAVE] keys and the cursor navigation keys are active within ABROWSE(). The cursor navigation keys are [CURSOR LEFT], [CURSOR RIGHT], [CURSOR UP], [CURSOR DOWN], [PAGE UP] and [PAGE DOWN]. The optional expression <expC2> is the name of a User Defined Function (UDF) to handle any other keystrokes. Note: if a UDF is specified, it will also trap the [ABANDON] and [EXIT SAVE] keys, so provision should be made to allow exit from the ABROWSE().

The AROW() and ACOL() functions return array row and column coordinates for use inside the UDF. When not used as part of a UDF, the AROW() and ACOL() functions can provide coordinates to edit the array. Two parameters are passed to the UDF: the current ABROWSE() status (see the table below) and the current column number starting from column 1.

Status	Description
0	ABROWSE() is idle, no keystrokes are pending.
1	An attempt was made to move beyond the top of the array.
2	An attempt was made to move beyond the bottom of the array.
3	The array is empty.
4	A key other than a cursor navigation key was pressed. The LASTKEY() function can be used to check the key pressed.

The UDF must return one of the following values:

Value	Description
0	Quit ABROWSE().
1	Continue ABROWSE().
2	Reread data, repaint the screen and continue.

The ABROWSE() function handles picture formatting for each column in the spreadsheet with an array represented by the optional expression <expC3>. Each value in array <expC3> should contain a picture formatting expression that corresponds to a column in array <expC1>.

Column headings may be optionally specified for the spreadsheet by using an array specified with

<expC4>. Each value in array <expC4> should contain a column heading that corresponds to a column in array <expC1>.

Example

```
function ab_udf
parameters ab_status, ab_col
do case
  case ab_status = 0
    ab_action = 1
  case ab_status = 1
    @ 0,0 clear to 0,79
    @ 0,0 say "Attempt to move past the top of file."
    ab_action = 1
  case ab_status = 2
    @ 0,0 clear to 0,79
    @ 0,0 say "Attempt to move past the end of file."
    ab_action = 1
  case ab_status = 3
    && array is empty
    ab_action = 0
  case ab_status = 4
    do case
      case lastkey() = 27
        ab_action = 0
      case lastkey() = 13
        @ 0,0 clear to 0,79
        m_get = ora_rows[arow(), acol()]
        @ 0,0 get m_get
        ora_rows[arow(), acol()] = m_get
        ab_action = 2
      otherwise
        @ 0,0 clear to 0,79
        @ 0,0 say "ASCII of key pressed is str(lastkey(),3)
        ab_action = 1
    endcase
  otherwise
    ab_action = 1
endcase
return ab_action

exec sql
  declare employees read only cursor for
  select * from emp;
exec sql
  open employees;
exec sql
  fetch employees into array ora_rows, ora_headings;
exec sql
  close employees;
exec sql
  drop cursor employees;
clear
set pckey on
abrowse(1,1,23,78,ora_rows,"ab_udf",ora_headings)
```

Products

Recital Terminal Developer

ABS()

Class

Numeric Data

Purpose

Function to return absolute value

Syntax

ABS(<expN>)

See Also

SET DECIMALS, INT(), ROUND()

Description

The ABS() function returns the absolute value of the numeric expression <expN>.

Example

? abs(-1)

1

? abs(-5.6)

5.6

? abs(2.5)

2.5

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ACC()

Class

Input/Output

Purpose

Function to return report accumulator

Syntax

ACC()

See Also

CALC, IIF(), REPORT

Description

The ACC() function, which is only valid in report formats, returns the contents of the specified report accumulator. There are 20 accumulators available in a report format (.frm file). All accumulators are cleared to zero at the start of the report (REPORT FORM) command. Accumulators 1 to 10 are cleared after each sub-total break. This function is useful if used in conjunction with the CALC() function which determines how accumulation takes place. It can be used to provide horizontal totaling in reports.

Example

acc(1) + order_val * discount

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ACCESS()

Class

Environment

Purpose

Function to return user access group

Syntax

ACCESS()

See Also

CREATE, MODIFY STRUCTURE, GETGID(), GETUID(), GETENV(), GETPID(), GETPROT(), SETPROT(), STR()

Description

The ACCESS() function returns the group number to which the user has been assigned by the system manager.

Example

```
if access() > 100
    dialog box "Sorry, access denied."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ACHOICE()

Class

Array Processing

Purpose

Function to display an array of choices in a popup menu

Syntax

ACHOICE(<expN1>, <expN2>, <expN3>, <expN4>, <array1> [,<array2>] [,<expC>] [,<expN5>] [,<expN6>])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, AFILL, DBEDIT, ABROWSE

Description

The ACHOICE() function executes a popup menu using an array of character strings as choices. The screen is automatically saved and restored if SET SCREENMAP is ON. The menu is framed in a box where <expN1> to <expN4> represent the coordinates <row>, <col>, <endrow>, <endcol>. The menu border can be disabled with the SET BORDER TO NONE command.

The array <array1> is the name of an array where all the elements are character strings. These character strings make up the menu items. The ACHOICE() function displays a scrollable menu of up to 1000 elements.

The optional <array2> is used in parallel with <array1>. If an element in <array2> is .F. (or an expression that evaluates to .F.), the corresponding element in <array1> will be included in the menu but not available for selection.

By default, only the [RETURN], [ABANDON] and [EXIT SAVE] keys and the cursor navigation keys are active within ACHOICE(). The cursor navigation keys are [CURSOR LEFT], [CURSOR RIGHT], [CURSOR UP], [CURSOR DOWN], [PAGE UP] and [PAGE DOWN]. The optional expression <expC> is the name of a User Defined Function (UDF) to handle any other keystrokes.

The ACHOICE () function automatically passes three parameters to the UDF: the current ACHOICE() status (see the table below), the current element number in the array, and the relative position within the menu window.

Status	Description
0	ACHOICE() is idle, no keystrokes are pending.
1	An attempt was made to move beyond the top of the array.
2	An attempt was made to move beyond the bottom of the array.
3	A key was pressed that cannot be automatically handled by ACHOICE(). The LASTKEY() function can be used to check the key pressed.
4	No selectable items.

The UDF must return one of the following values:

Value	Description
0	Quit ACHOICE(), returning 0.
1	Make selection, returning index of current item.
2	Continue selection process.

3	Go to the next item whose first character matches the last key pressed.
---	---

If the optional <expN5> is specified, the highlight bar will be displayed on <expN5> when ACHOICE() is activated. If <expN5> is not specified, the default is the first menu item.

The optional expression <expN6> may be used to specify an initial relative starting row in the window. ACHOICE() returns the position of the selected array element, or 0 if the [ABANDON] key was pressed.

Example

```
function ach_udf
parameters ach_mode, ach_element, ach_row
do case
  case ach_mode = 0
    ach_action = 2
  case ach_mode = 1
    @ 0,0 CLEAR TO 0,79
    @ 0,0 SAY "Attempt to move past the top of file."
    ach_action = 2
  case ach_mode = 2
    @ 0,0 CLEAR TO 0,79
    @ 0,0 SAY "Attempt to move past the end of file."
    ach_action = 2
  case ach_mode = 4
    // no item selectable
    ach_action = 0
  case ach_mode = 3
    // go to first element starting with this letter
    ach_action = 3
  otherwise
    ach_action = 2
endcase
return ach_action

clear
use cust index cust.ndx
declare cust_array[reccount(),1]
copy to array cust_array fields last_name
private m_var
m_var = achoice(1,1,23,78,cust_array,, "db_udf")
```

Products

Recital Mirage Server, Recital Terminal Developer

AClass()

Class
Objects

Purpose

Function to place the class name of an object and its ancestors into a variable array.

Syntax

AClass(<array>,<object>)

See Also

DEFINE CLASS

Description

The AClass() function places the class name of an object and its ancestors into a variable one-dimensional array. The name of the object is specified in <object>. The name of the array is specified in <array>. If the array does not exist, it is created. If the array is smaller or larger than required, it is resized. The object class name is placed in the first element of the array, the class name of its immediate parent in the second element and so on through the class hierarchy.

The AClass() function returns the number of class names placed in the array. If the array cannot be created, the AClass() function will return 0.

Example

```
nHierarchy = aclass(aHierarchy,myObject)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ACOL()

Class

Array Processing

Purpose

Function to return a column coordinate from an array

Syntax

ACOL()

See Also

ABROWSE(), AROW()

Description

The ACOL() function returns a column coordinate from an array. The ACOL() function may be used in conjunction with the AROW() function as part of a UDF for the ABROWSE() function. The ABROWSE() function browses a two-dimensional array in spreadsheet style format, and accepts an optional UDF to define keystrokes.

When not used in a UDF for the ABROWSE() function, the ACOL() and AROW() functions may be used to return the position in the array on exit. See the ABROWSE() entry for an example of the use of the AROW() and ACOL() functions within an ABROWSE() user defined function.

Example

```
use customer.rdb
declare acust[lastrec(),fcount()]
copy to array acust
clear
abrowse(1,1,23,78,acust)
dialog box acust[arow(),acol()]
```

Products

Recital Terminal Developer

ACOPY()

Class

Array Processing

Purpose

Function to copy elements from one array to another

Syntax

ACOPY(<array1>,<array2> [,<expN1> [,<expN2> [,<expN3>]]])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The ACOPY() function allows the transfer of elements between arrays, enabling arrays to be subdivided or combined. The destination array must have sufficient elements to receive all those transferred. All elements of the source array <array1> will be copied to the target array <array2> unless <expN1> is entered to specify the element from which to start copying. The specification of <expN2> sets the number of elements to be copied sequentially. The starting element in the target array can be set with <expN3>.

Example

```
declare overview[4000]
declare north[2000]
// Copy first 2000 elements from source array
// starting from position 1 in the target array
acopy(north,overview,1,2000,1)
declare south[1500]
// Copy first 1500 elements from source array
// starting from position 2001 in the target array
acopy(south,overview,1,1500,2001)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ACOS()

Class

Numeric Data

Purpose

Function to calculate and return the angle size in radians for any given cosine value

Syntax

ACOS(<expN>)

See Also

TAN(), EXP(), LOG(), COS(), PI(), SIN(), ASIN(), ATAN(), ATN2(), DTOR(), RTOD(), SET DECIMALS

Description

The ACOS() function calculates the angle size of a cosine value. The cosine value is represented by the numeric expression <expN>. The numeric expression must fall between -1.0 and +1.0 inclusive. The angle size is returned as a numeric in radians. The SET DECIMALS command determines the number of decimals displayed in the returned angle size.

Example

set decimal to 4

?acos(0.7071)

0.7854

set decimals to 2

?rtod(acos(cos(dtor(30))))

30.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ACTIVEPID()

Class

Environment

Purpose

Function to return activity of identified process

Syntax

ACTIVEPID(<expN>)

See Also

CANCELPID(), SPAWNPID(), SPAWN

Description

The ACTIVEPID() function returns .T. if the specified process ID, <expN> is still active and .F. otherwise. The SPAWNPID() function is used to get the process ID <expN> of the last process spawned with the SPAWN command.

Example

```
spawn db program
? activepid(spawnpid())
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ADATABASES()

Class

Databases

Purpose

Function to place the names of all open databases and their paths into a variable array.

Syntax

ADATABASES(<array>)

See Also

CLOSE DATABASES, DISPLAY STATUS, LIST STATUS, OPEN DATABASE, USE, ADIR(), ALIAS(), DATABASE(), DBF(), DBUSED(), GETENV(), USED(), SET SQL

Description

The ADATABASES() function is used to place the names of all open databases and their paths into a variable array. The name if the array is specified in <array>. If the array does not exist, it is created. If the array is smaller or larger than required, it is resized. The array is two-dimensional with two columns. The first column contains the name of an open database, the second the path for that database.

The ADATABASES() function returns the number of database names added to the array. If no databases are open or the array cannot be created, the ADATABASES() function returns 0.

NOTE: The ADATABASES() function operates on databases, not tables.

Databases in Recital are implemented as directories containing files that correspond to the tables and associated files in the database. Operating System file protection can be applied individually to the files for added security. The directories are sub-directories of the Recital data directory. The environment variable / symbol DB_DATADIR points to the current Recital data directory and can be queried using the GETENV() function. Files from other directories can be added to the database using the ADD TABLE command or via the database catalog and SET AUTOCATALOG functionality.

Databases can be opened using the SQL USE command, with SQL set to MYSQL, or using the SQL OPEN DATABASE command.

Example

```
VFP/SQL> OPEN DATABASE hr EXCLUSIVE
VFP/SQL> nDatabases = adatabases(aDBCNames)
VFP/SQL> CLOSE DATABASES
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ADDBS()

Class

String Data

Purpose

Function to add a backslash to a path expression if required.

Syntax

ADDBS(<expC>)

See Also

BASENAME(), FULLPATH(), SET FULLPATH

Description

The ADDBS() function adds a backslash to a path expression if required. The path name is specified in <expC>.

Example

```
rename (addbs(getenv([DB_CLASSDIR]))+ "myclass.cls") (addbs(getenv([DB_CLASSDIR]))+ ;  
    "myoldclass.cls")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ADDPROPERTY()

Class

Objects

Purpose

Function to add a property to an existing object

Syntax

ADDPROPERTY(<object-name> ,<expC>[,<exp>])

See Also

DEFINE CLASS...ENDDEFINE, WITH...ENDWITH, CREATEOBJECT(), NEWOBJECT(), REMOVEPROPERTY()

Description

The Visual FoxPro compatible ADDPROPERTY() function is used to add a property to an existing object. It returns .T. (True) if the property was successfully added and .F. (False) otherwise.

Parameter	Description
<object-name>	The name of the object.
<expC>	The name of the property to be added.
<exp>	The value to assign to the property being added. This is optional: if omitted and the property being added already exists, the property value is unchanged, if omitted and the property is new, the value is initialized to .F. (False).

Properties can be removed using the REMOVEPROPERTY() function.

All classes have an inbuilt ADDPROPERTY 'factory method'. This can be used as an alternative to the ADDPROPERTY() function to add properties to an object at runtime.

Example

```
define class myclass as custom
myprop = "Hello World"
enddefine
```

```
myobject = createobject("myclass")
MessageBox(myobject.myprop)
addproperty(myobject, "myprop2", "goodbye")
// Or: myobject.addproperty("myprop2", "goodbye")
MessageBox(myobject.myprop2)
removeproperty(myobject, "myprop2")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ADEL()

Class

Array Processing

Purpose

Function to delete an element from an array

Syntax

ADEL(<array>,<expN>[,2])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADIR(), AFIELDS(), AFILL(), AINS(), ALN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The ADEL() function deletes the element at position <expN> of the specified <array>. All elements higher than the deleted element will be shifted down and the last element in the <array> will become .F.. The total number of elements in the <array> will remain the same.

For two-dimensional arrays, the <expN> refers to an entire row of elements, rather than an individual element. If the '2' parameter is included, the <expN> refers to an entire column of elements.

Example

```
element_no = ascan(flist,"New information")
// If element exists
if m.element_no > 0
    adel(flist,m.element_no)
endif
//Another Example
set compatible to foxpro
DIMENSION array1(3,3)
array1(1,1) = "one"
array1(1,2) = 1
array1(1,3) = CTOD("01/01/2001")
array1(2,1) = "two"
array1(2,2) = 2
array1(2,3) = CTOD("02/02/2002")
array1(3,1) = "three"
array1(3,2) = 3
array1(3,3) = CTOD("03/03/2003")
//Delete row 2
ADEL(array1,2)
//Delete column 1
ADEL(array1,1,2)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ADESC()

Class

Array Processing

Purpose

Function to fill an array with field descriptions

Syntax

ADESC(<array>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, SET FIELDS, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASTORE(), ASUM(), DBEDIT()

Description

The ADESC() function fills the <array> with the field descriptions for the table in the current workarea. The ADESC() function returns the number of fields in the current workarea. The array must have been previously declared. The field descriptions will be copied into the array elements until either the array is full or all the descriptions are copied. The SET FIELDS command can be used to restrict the active fields and therefore determine which descriptions are copied into the array.

Example

```
declare headings[fcount()]\nadesc(headings)\ndbedit(1, 1, 10, 79, "", "", "", headings)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ADIR()

Class

Array Processing

Purpose

Function to return number of files matching a file pattern

Syntax

ADIR(<skeleton> [,<array1> [,<array2> [,<array3> [,<array4> [,<array5>]]]])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The ADIR() function returns the number of files which match the specified file skeleton. The following 'wild card' characters can be used:

Character	Description
?	Matches any one character.
%	Matches any one character.
*	Matches zero or more characters.

The ADIR() function may also be used to load directory information into arrays.

Parameters	Description
<skeleton>	The character string specifying the file skeleton.
<array1>	The name of a pre-declared array in which to load the character string file names that match the specified skeleton.
<array2>	The name of a pre-defined array in which to load the numeric type file size in bytes of the files that match the specified skeleton.
<array3>	The name of a pre-defined array in which to load the date type creation date of the files that match the specified skeleton.
<array4>	The name of a pre-defined array in which to load the character type creation time of the files matching the specified skeleton.
<array5>	The name of a pre-defined array. This option has been added for compatibility purposes only. An element containing the character string "A" will be loaded for each file that matches the specified skeleton.

If the compatibility setting is set to FoxPro, SET COMPATIBLE TO FOXPRO, ADIR() is fully compatible with FoxPro syntax and behavior.

Example

```
declare files[adir("*.dbf")]
nTables = adir("*.dbf",files)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AELEMENT()

Class

Array Processing

Purpose

Function to return the number of an array element from the element subscripts

Syntax

AELEMENT(<array>, <expN1> [,<expN2>])

See Also

ADEL(), ADIR(), AFIELDS(), AINS(), ALEN(), ASCAN() ASORT(), DECLARE

Description

The AELEMENT() function returns the element number from a two-dimensional array. The <array> name of a two-dimensional array and the row <expN1> and column <expN2> must be specified. From the row and column values of the two-dimensional array AELEMENT() will return the element number.

Example

```
declare test[4,4]
? aelement(test, 3,2)
```

10

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AFIELDS()

Class

Array Processing

Purpose

Function to fill a series of arrays with table structure details

Syntax

```
AFIELDS(<array1> [,<array2> [,<array3> [,<array4>]])  
AFIELDS(<array> [,<workarea | alias>])
```

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, SET COMPATIBLE, SET FIELDS, SET FILETYPE, AAVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The AFIELDS() function returns the number of fields in the table in the current workarea and loads the field names into an array, <array1>. The field types, field lengths and field decimals can also be loaded into individual arrays.

Parameters	Description
<array1>	The name of a pre-declared array in which to load the character string field name.
<array2>	The name of a pre-defined array in which to load the field data types as character strings.
<array3>	The name of a pre-defined array in which to load the field lengths as numeric values.
<array4>	The name of a pre-defined array in which to load the field decimal places as numeric values. If the field is not a numeric data type or has no decimal places, then zero is returned.

The SET FIELDS command can be used to restrict the active fields and therefore determine which fields are copied into the array.

With SET COMPATIBLE set to FoxPro, Foxbase or VFP, the AFIELDS() function syntax conforms to Visual FoxPro behavior. The information detailed in the table below is loaded into the 18 columns of the single array <array>. Each row in the array corresponds to a field in the table. If the optional <workarea | alias> is specified, then the function will operate in the required location, otherwise it will operate in the current workarea.

Column	Information	Data Type
1	Field name	Character
2	Field type: C = Character Y = Currency D = Date T = DateTime B = Double F = Float G = General I = Integer L = Logical M = Memo N = Numeric Q = Varbinary V = Varchar and Varchar (Binary) W = Blob	Character
3	Field width	Numeric
4	Decimal places	Numeric
5	Null values allowed	Logical
6	Code page translation not allowed	Logical
7	Field validation expression	Character
8	Field validation text	Character
9	Field default value	Character
10	Table validation expression	Character
11	Table validation text	Character
12	Long table name	Character
13	Insert trigger expression	Character
14	Update trigger expression	Character
15	Delete trigger expression	Character
16	Table comment	Character
17	NextValue for autoincrementing	Numeric
18	Step for autoincrementing	Numeric

Example

```
use patrons
declare tab1(12),tab2(12),tab3(12),tab4(12)
fields = afields(tab1,tab2,tab3,tab4)
```

//VFP example

```
set compatible to vfp
use orders in 0
use products in 0
nFIELDS = AFIELDS(array1,"orders")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AFILL()

Class

Array Processing

Purpose

Function to fill a defined section of an array with an expression

Syntax

AFILL(<array> , <exp> [,<expN1> [,<expN2>]])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The AFILL() function fills the elements of a previously declared <array> with the specified <exp>. The starting element in the array can be specified with <expN1> and the number of elements to fill with <expN2>. If <expN1> and <expN2> are not specified, all elements will be filled. The AFILL() function returns .T. if successful and .F. otherwise. All array elements can also be assigned a value using the = operator.

Example

```
declare matrix[100]
afill(matrix,"good",51,100)
? matrix[45]
```

.F.

```
? matrix[75]
```

good

```
? afill(matrix,"bad",1,50)
```

.T.

```
? matrix[45]
```

bad

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AFONT()

Class

Fonts

Purpose

Function to load available font name information into an array

Syntax

AFONT(<array-name> [,<expC> [,<expN>]])

See Also

DECLARE, DIMENSION, PRIVATE, PUBLIC, RELEASE, ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALEN(), ASCAN(), ASORT(), FONTMETRIC(), GETFONT(), WFONT()

Description

The AFONT() function loads the names of available fonts into a pre-declared array.

Parameters	Description
<array-name>	The name of a pre-declared array in which to load font information. The array will be resized (truncated or increased to fit).
<expC>	The <expC> can be used to optionally specify a particular font. This is included for language compatibility only.
<expN>	The <expN> can be used to optionally specify a particular font size. This is included for language compatibility only.

Example

```
declare a_font[10]
afont(a_font)
mchoice = 0
@10,10 get mchoice from a_font function "&"
read
```

Products

Recital Mirage

AINS()

Class

Array Processing

Purpose

Function to insert an element into an array

Syntax

AINS(<array>, <expN> [, <exp>])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The AINS() function inserts an element into the <array> at the position of <expN>. The element will be defined as .F. unless an optional <exp> is included. All elements higher than the inserted element are shifted up and the contents of the last element are overwritten. The total number of elements in the array remains the same. The array must have been previously declared.

Example

```
declare table[10]
ains(table, 3, "hello world")
// Another Example
declare flist[5]
...
if ascan(flist,"new information") = 0
    ains(flist,1,"new information")
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ALEN()

Class

Array Processing

Purpose

Function to return the number of elements, rows or columns in an array

Syntax

ALEN(<array> [,<expN>])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The ALEN() function returns the number of elements in the previously declared <array>. If the optional <expN> is specified, ALEN() can also return the number of rows or the number of columns in the array.

<expN>	Returns
0	Number of elements
1	Number of rows
2	Number of columns

If no <expN> is specified, the number of array elements is returned.

Example

```
declare aNames[10,12]
? alen(aNames)
    120
? alen(aNames,0)
    120
? alen(aNames,1)
    10
? alen(aNames,2)
    12
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ALIAS()

Class

Table Basics

Purpose

Function to return an alias name

Syntax

ALIAS([<expN>])

See Also

SELECT, USE, SELECT(), WORKAREA(), READVAR(), DBF()

Description

The ALIAS() function returns, as a character string in upper case, the alias name of the table in the currently selected workarea, or a null string if no table is active. The optional parameter <expN> can be used to specify the workarea number in which the ALIAS() function should operate.

Example

use demo

? alias()

CUSTOMER

? alias(2)

ACCOUNTS

? dbf()

customer.rdb

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ALLTRIM()

Class

String Data

Purpose

Function to remove leading and trailing spaces

Syntax

ALLTRIM(<expC>)

Description

The ALLTRIM() function trims leading and trailing spaces from the character expression <expC>. This function should be used in conjunction with the RPAD() function when used in index expressions.

See Also

TRIM(), RTRIM(), LTRIM() LPAD(), RPAD(), STRTRAN()

Example

```
name=" Peter "
```

```
? len(name)
```

```
15
```

```
? name
```

```
Peter
```

```
? alltrim(name)
```

```
Peter
```

```
? len(alltrim(name))
```

```
5
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ALTD()

Class

Xbase Compatibility

Purpose

Language compatibility only

Syntax

ALTD()

See Also

SET COMPATIBLE

Description

This function has been added for language compatibility only. The ALTD() function always returns .T..

Example

? altd()

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AMAX()

Class

Array Processing

Purpose

Function to return the highest element value in an array

Syntax

AMAX(<array>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The AMAX() function evaluates all the numeric elements in the specified <array> and returns the highest value.

Example

```
use payroll
declare number[reccount(),1]
copy to array number fields total
? amax(number)
8165.19
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AMEMBERS()

Class

Objects

Purpose

Function to load the properties of an object into an array

Syntax

AMEMBERS(<array-name>,<object-ref>)]

See Also

CLASS, DEFINE CLASS, WITH...ENDWITH, ACLASS(), COMPOBJ(), CREATEOBJECT(), NEWOBJECT(), OBJECTREAD(), OBJECTTYPE(), OBJECTWRITE()

Description

The AMEMBERS() function is used to load the names of the properties of the specified object into an array and return the number of properties loaded (and hence the number of array elements).

The <array-name> array will be created if it does not already exist.

Example

```
class myclass
property last_name
property first_name
property email
endclass
```

```
omyclass = new myclass()
dialog box str(amembers(myarray,omyclass))  && displays 3
display memory    && shows the array contents
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AMIN()

Class

Array Processing

Purpose

Function to return the lowest element value in an array

Syntax

AMIN(<array>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), ASCAN(), ASORT(), ASUM()

Description

The AMIN() function evaluates all the numeric elements in the specified <array> and returns the lowest value.

Example

```
use payroll
declare number[reccount(),1]
copy to array number fields total
? amin(number)
```

16.23

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AMPM()

Class

Date and Time Data

Purpose

Function to return 12-hour clock time

Syntax

AMPM()

See Also

SET CLOCK, SET CLOCKRATE, DAYS(), ELAPTIME(), HOURS(), MINUTES(), SECONDS(), SECS(), TSTRING(), VALIDTIME(), TIME()

Description

The AMPM() function returns the time based on a 12 hour clock as a character string followed by “am” or “pm”.

Example

? ampm()

07:30:24 pm

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AROW()

Class

Array Processing

Purpose

Function to return a row coordinate from a two dimensional array

Syntax

AROW()

See Also

ABROWSE(), ACOL()

Description

The AROW() function returns a row coordinate from a two-dimensional array. The AROW() function may be used in conjunction with the ACOL() function as part of a UDF for the ABROWSE() function. The ABROWSE() function browses a two-dimensional array in spreadsheet style format, and accepts an optional UDF to define keystrokes. When not used in a UDF for the ABROWSE() function, the AROW() and ACOL() functions may be used to return the position in the array on exit. See the ABROWSE() entry for an example of the use of the AROW() and ACOL() functions within an ABROWSE() user defined function.

Example

```
use customer.rdb
declare acust[lastrec(),fcount()]
copy to array acust
clear
abrowse(1,1,23,78,acust)
dialog box acust[arow(), acol()]
```

Products

Recital Mirage Server, Recital Terminal Developer

ASC()

Class

Expressions and Type Conversion

Purpose

Function to convert an ASCII character to a number

Syntax

ASC(<expC>)

See Also

CHR()

Description

The ASC() function returns the numeric value of the first ASCII character from the character expression <expC>.

Example

```
? asc("123")
```

49

```
? asc("Newcastle")
```

78

```
nVar = asc("Newcastle")
```

78

```
? type("nVar")
```

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASCAN()

Class

Array Processing

Purpose

Function to search an array for an expression

Syntax

ASCAN(<array>, <exp>, [<expN>])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASORT(), ASUM()

Description

The ASCAN() function searches an array for the expression specified in <exp>. The expression may consist of any data type. It returns the element number at which the expression is located, or 0 if not found. The element number returned is the first occurrence of the string found. The SET EXACT command does not alter the find condition because the ASCAN() function works as if SET EXACT were OFF. The optional expression <expN> specifies the position at which to start the scan. If the <expN> is not specified then the search starts at the first element.

Example

```
use accounts
declare names[reccount(),1]
copy to array names fields company
element_no=ascan(names,"OK Cleaning Company")
if element_no > 0
    ? names[m->element_no]
OK Cleaning Company
else
    dialog box "The company was not found."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASIN()

Class

Numeric Data

Purpose

Function to return the angle size of any given sine value

Syntax

ASIN(<expN>)

See Also

TAN(), EXP(), LOG(), COS(), PI(), SIN(), ACOS(), ATAN(), ATN2(), DTOR(), RTOD(), SET DECIMAL

Description

The ASIN() function returns the angle size of sine value. The sine value is represented by the numeric expression <expN>. The numeric expression must fall between -1.0 and +1.0 inclusive. The angle size is returned as a numeric in radians. The SET DECIMALS command determines the number of decimals displayed in the returned angle size.

Example

set decimals to 4

?asin(0.7071)

0.7854

set decimal to 2

?rtod(asin(sin(dtor(30))))

30.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASIZE()

Class

Array Processing

Purpose

Function to resize an array

Syntax

ASIZE(<array-name>,<expN>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT(), ASUM()

Description

The ASIZE() function can be used to resize an array. The <array-name> is the name of an existing array to resize. The <expN> is the number of rows the array should be given. For a one-dimensional array the number of rows is the same as the number of elements. If the <expN> is smaller than the original number of rows, the contents of rows higher than <expN> are lost.

Example

```
declare array1[100]
```

```
? alen(array1)
```

```
100
```

```
asize(array1,200)
```

```
? alen(array1)
```

```
200
```

```
declare array2[100,5]
```

```
? alen(array2)
```

```
500
```

```
? alen(array2,1)
```

```
100
```

```
asize(array2, alen(array2,1)+1)
```

```
? alen(array2)
```

```
505
```

```
? alen(array2,1)
```

```
101
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASORT()

Class

Array Processing

Purpose

Function to sort the elements of a specified array

Syntax

ASORT(<array> [,<expN1>, [<expN2>]])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASUM()

Description

The ASORT() function sorts all the character elements in the specified <array>. The sort can start at element <expN1> and finish at element <expN2>. The ASORT() function will stop sorting when it finds a non-character element.

Example

```
use accounts
declare names[reccount(),1]
copy to array names fields company
asort(names)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASTORE()

Class

Array Processing

Purpose

Function to fill an array with character strings that are separated with a specified character

Syntax

ASTORE(<array>, <expC1>, <expC2>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADESC(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), AMAX(), AMIN(), ASCAN(), ASORT(), ASTRING(), ASUM(), DBEDIT()

Description

The ASTORE() function fills the <array> with character strings from <expC1> which are separated by the character specified in <expC2>. The array must have been previously declared. The character strings are copied into the array elements until either all the elements are filled, or all the strings are copied. The length of the array is automatically shortened to the number of elements stored in it. The ASTORE() function returns the number of character strings copied into the array. The ASTORE() function is most commonly used in combination with menu commands where multiple selections can be made.

Example

```
declare aNums[10]
? alen(aNums)
    10
nelements = ASTORE(aNums, "one, two, three",',')
? nelements
    3
? alen(aNums)
    3
// Another Example
menu fields select '+'
declare aMenu[512]
nelements = astore(aMenu, menuitem(), '+')
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASTRING()

Class

Array Processing

Purpose

Function to return an array as a character string, with the elements separated by a comma or specified character

Syntax

ASTRING(<array>[, <expC>])

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADESC(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), AMAX(), AMIN(), ASCAN(), ASORT(), ASTORE(), ASUM(), DBEDIT()

Description

The ASTRING() function reads the array <array> and returns the elements as a character string. The elements are comma-separated unless <expC> is specified, in which case they are separated by <expC>.

Example

```
menu fields select '+'  
declare aMenu[512]  
nelements = astore(aMenu, menuitem(), "+")  
backtostring = astring(aMenu, "+")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASUBSCRIPT()

Class

Array Processing

Purpose

Function to return the row or column number of an array element from the element number

Syntax

ASUBSCRIPT(<array>, <expN1> ,<expN2>)

See Also

ADEL(), ADIR(), AFIELDS(), AINS(), ALLEN(), ASCAN() ASORT(), DECLARE, AELEMENT()

Description

The ASUBSCRIPT() function returns the array <array> row or column element reference for a specified array element. The parameter <expN1> defines the array element index position within the array. The parameter <expN2> identifies whether the array's row or column reference should be returned. If <expN2> is a 1, the row number is returned. If <expN2> is a 2, the column number is returned.

Example

```
use state.rdb
private aRECORDS[50,fcount()]
copy to array aRECORDS
nELEM = ascan(aRECORDS,'California')
? asubscript(aRECORDS,nELEM,1)
```

6

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ASUM()

Class

Array Processing

Purpose

Function to calculate the sum of element values in an array

Syntax

ASUM(<array>)

See Also

APPEND FROM ARRAY, COPY TO ARRAY, DECLARE, DIMENSION, GATHER, PRIVATE, PUBLIC, RELEASE, RESTORE FROM, SAVE TO, SCATTER, AVERAGE(), ACHOICE(), ACOPY(), ADEL(), ADIR(), AFIELDS(), AFILL(), AINS(), ALLEN(), AMAX(), AMIN(), ASCAN(), ASORT()

Description

The ASUM() function calculates the sum of all the numeric elements in the specified <array> and returns the total.

Example

```
use payroll
declare aNumber[reccount(),1]
copy to array aNumber fields pay_amount
? asum(aNumber)
127641.73
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AT()

Class

String Data

Purpose

Function to search for a substring

Syntax

AT(<expC1>, <expC2> | <memofield> [, <expN>])

See Also

ATNEXT(), LEFT(), RIGHT(), STREXTRACT(), STUFF(), STREXTRACT(), SUBSTR(), STRTRAN()

Description

AT() is the substring search function. It returns a number signifying the starting position of <expC1> in <expC2> or in the specified memo field. If the substring is not contained within <expC2> or <memofield>, then the function returns the value 0. The leftmost character of a string is in character position 1. The AT() function will return the starting position of the Nth occurrence of <expC1> when the optional numeric expression <expN> is specified. The AT() function is particularly useful in conjunction with the LEFT() or SUBSTR() functions for locating starting points for extracting text from a string.

Example

```
? at("is", "Recital is good")
```

9

```
cString1 = "is"
```

```
cString2 = "Recital is good"
```

```
? at(cString1, cString2)
```

9

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ATAN()

Class

Numeric Data

Purpose

Function to return the angle size of a given tangent value

Syntax

ATAN(<expN>)

See Also

TAN(), EXP(), LOG(), COS(), PI(), SIN(), ACOS(), ASIN(), ATN2(), DTOR(), RTOD(), SET DECIMALS

Description

The ATAN() function returns the angle size, in radians, of a tangent value. The tangent value is represented by the numeric expression <expN>. The angle size is returned as a numeric in radians. The SET DECIMALS command determines the number of decimals displayed in the returned angle size.

Example

set decimals to 4

?atan(.7071)

0.6155

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ATN2()

Class

Numeric Data

Purpose

Function to return the angle size for the specified cosine and sine values of a given point

Syntax

ATN2(<expN1>, <expN2>)

See Also

TAN(), EXP(), LOG(), COS(), PI(), SIN(), ACOS(), ASIN(), ATAN(), DTOR(), RTOD(), SET DECIMALS

Description

The ATN2() function returns the angle size, in radians, for the specified cosine and sine values of a given point. <expN1> and <expN2> represent the cosine and sine values, respectively. The angle size is returned as a numeric in radians. The SET DECIMALS command determines the number of decimals displayed in the returned angle size.

Example

set decimals to 4

?atn2(0.5000,0.8660)

0.5236

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ATNEXT()

Class

String Data

Purpose

Function to search a character string for the specified occurrence of a substring

Syntax

ATNEXT(<expC1>, <expC2> [, <expN>])

See Also

AT(), LEFT(), RIGHT(), STREXTRACT(), STUFF(), STREXTRACT(), SUBSTR(), STRTRAN(), RAT()

Description

ATNEXT() is a substring search function which returns a number signifying the start position of a specified occurrence of <expC1> in <expC2>. The <expN> specifies the occurrence of <expC1> in <expC2>. The ATNEXT() function returns the value 0 if the specified occurrence does not exist. If <expN> is not specified, ATNEXT() checks for the first occurrence of <expC1>.

The leftmost character of a string is in character position 1. The ATNEXT() function is particularly useful in conjunction with the LEFT() or SUBSTR() functions for locating starting points for extracting text from a string.

Example

```
? atnext("is", "The fact is that Recital is good", 2)
```

26

```
cString1 = "is"
```

```
cString2 = "The fact is that Recital is good"
```

```
? atnext(cString1, cString2, 2)
```

26

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

AVGVALUES()

Class

Fields and Records

Purpose

Function to return the average value of a column for a matching series of keys

Syntax

AVGVALUES(<expN> [, <for condition> [, <key-expression>]])

See Also

CNTVALUES(), SUMVALUES(), MINVALUES(), MAXVALUES(), SQLVALUES()

Description

The AVGVALUES() function returns the average value of <expN> for the matching series of keys. The required <expN> parameter must be a column name from a table. If none of the optional parameters is specified, then the average value for the number of keys matching the current key is returned. An optional <for condition> can be specified to restrict the rows included in the operation. You can also optionally specify a <key-expression> to be used instead of the current key. After completion, all record pointers and indexes are returned to the original position.

Example

```
use customer order title
// Average the balance column for
// the number of records matching the current key
total_balance = avgvalues(balance)
// Average the balance column for
// the number of records matching the key "Mr"
m_male = avgvalues(balance, .T., "Mr")
// Average the balance column for
// the number of records matching
// the current key with an expiry date < today
m_expired = avgvalues(balance, expiry < date())
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BAR()

Class

Menus

Purpose

Function to return the most recently selected menu bar number

Syntax

BAR()

See Also

POPUP(), PROMPT(), DEFINE POPUP, ACTIVATE POPUP, MENU(), MENUITEM(), SET COMPATIBLE

Description

The BAR() function returns the most recently selected bar number from an Xbase style pop-up menu. If the pop-up menu has been defined using the DEFINE BAR command, BAR() returns the bar number. If the pop-up prompts have been defined using the DEFINE POPUP command, BAR() returns the line number of the prompt, starting with number 1 as the first line within the pop-up screen area. The BAR() function will return 0 if there is no active or defined pop-up, or if the [ABANDON] key was pressed from within a pop-up menu.

The command SET COMPATIBLE must be set to “dbase4” when using Xbase style menus.

Example

procedure action

do case

 case bar() = 1

 do editfunc

 case bar() = 2

 do deletfunc

 case bar() = 3

 do addfunc

endcase

return

Products

Recital Mirage Server, Recital Terminal Developer

BARCOUNT()

Class

Menus

Purpose

Function to return the number of bars in a pop-up Xbase style menu

Syntax

BARCOUNT([<expC>])

See Also

ACTIVATE POPUP, BAR(), DEACTIVATE POPUP, DEFINE BAR, DEFINE POPUP, ON BAR, ON SELECTION BAR

Description

The BARCOUNT() function returns the number of defined bars in a pop-up Xbase style menu. These types of menus are created with the DEFINE POPUP and DEFINE BAR commands. The BARCOUNT() function returns the number of bars in the currently active pop-up. The optional <expC> allows the BARCOUNT() function to operate on the named defined pop-up.

Example

```
define menu main message "Welcome to the Main Menu"
define pad view;
    prompt "View";
    at 0,7;
    message str(barcount() + " ways to view records.",3)
define pad custom;
    prompt "Customize";
    at 0,18;
    message str(barcount() + " ways to customize the work surface.",3)
```

Products

Recital Mirage Server, Recital Terminal Developer

BARPROMPT()

Class

Menus

Purpose

Function to return the prompt text of a bar in a pop-up Xbase style menu

Syntax

BARPROMPT(<expN>[,<expC>])

See Also

BAR(), BARCOUNT(), DEFINE BAR, DEFINE POPUP

Description

The BARPROMPT() function returns the text associated with the prompt qualifier in a menu bar in a pop-up dBASE style menu. These types of menus are created with the DEFINE POPUP and DEFINE BAR commands. The bar is identified by the number that was assigned to it with the DEFINE BAR command, or with the DEFINE POPUP...PROMPT FIELD | PROMPT FILES | PROMPT STRUCTURE command. The BARPROMPT() works on the active pop-up menu unless another pop-up is specified with the optional character expression <expC>.

Example

```
?barprompt(4, "Accounts")
```

Kramer

Products

Recital Mirage Server, Recital Terminal Developer

BASENAME()

Class

Disk and File Utilities

Purpose

Function to return the base filename from a given file specification

Syntax

BASENAME(<expC>)

See Also

GETENV(), GETLOG(), DBF(), NDX (), FMT(), SET FULLPATH

Description

The BASENAME() function returns the base filename from the file specification <expC>. This function is very useful when used in conjunction with the command, SET FULLPATH ON. The BASENAME() functions always returns a character string without changing the case.

Example

```
? dbf()
```

```
accounts.dbf
```

```
set fullpath on
```

```
? dbf()
```

```
/usr/recital/UD/demo/accounts.dbf
```

```
? basename(dbf())
```

```
accounts.dbf
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BETWEEN()

Class

Expressions and Type Conversion

Purpose

Function to determine whether a specified expression is between two other specified expressions

Syntax

BETWEEN(<exp1>,<exp2>,<exp3>)

See Also

CHROVERLAP(), EMPTY(), EVALUATE(), EXPRCHECK(), INLIST(), ISALPHA(), ISBLANK(), ISDIGIT(), TYPE()

Description

The BETWEEN() function returns true (.T.) if the value of a specified expression lies in the range delimited by two other expressions all of the same data type. The BETWEEN() function operates on Character, Date and Numeric expressions.

Parameters	Description
<exp1>	The expression to be checked.
<exp2>	Expression representing the range minimum.
<exp3>	Expression representing the range maximum.

The <exp1> must be equal to or more than <exp2> and equal to or less than <exp3> for BETWEEN() to return true (.T.). If <exp1> is not within this range, BETWEEN() returns false (.F.).

Example

? between(date(),date()-10,date()+10)

.T.

? between("A","A","Z")

.T.

? between(10,1,9)

.F.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BIN2I()

Class

Expressions and Type Conversion

Purpose

Function to convert a binary encoded string to a numeric

Syntax

`BIN2I(<expC>)`

See Also

`CREATE BRIDGE`, `BIN2L()`, `BIN2W()`, `BINCLOSE()`, `BINCREATE()`, `BINOPEN()`, `BINREAD()`, `BINSEEK()`, `BINWRITE()`, `I2BIN()`, `L2BIN()`

Description

The `BIN2I()` function converts a binary encoded string, formatted as a 16-bit signed integer, into an integer. The `<expC>` is a string, two bytes in length, containing a binary numeric value. All binary conversion functions may be used in conjunction with the binary file functions.

Example

```
? bin2i(i2bin(987))  
    987
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BIN2L()

Class

Expressions and Type Conversion

Purpose

Function to convert a binary encoded string to a numeric

Syntax

`BIN2L(<expC>)`

See Also

`CREATE BRIDGE`, `BIN2I()`, `BIN2W()`, `BINCLOSE()`, `BINCREATE()`, `BINOPEN()`, `BINREAD()`, `BINSEEK()`, `BINWRITE()`, `I2BIN()`, `L2BIN()`

Description

The `BIN2L()` function converts a binary encoded string, formatted as a 32-bit signed integer, into a numeric value. The `<expC>` is a string, four bytes in length, containing a binary numeric value. All the binary conversion functions may be used in conjunction with the binary file functions.

Example

```
? bin2l(l2bin(98765))
    98765
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BIN2W()

Class

Expressions and Type Conversion

Purpose

Function to convert a binary encoded string to a numeric

Syntax

`BIN2W(<expC>)`

See Also

`CREATE BRIDGE`, `BIN2I()`, `BIN2L()`, `I2BIN()`, `L2BIN()`, `BINREAD()`, `BINWRITE()`, `BINCREATE()`, `BINOPEN()`, `BINCLOSE()`

Description

The `BIN2W()` function converts a binary encoded string, formatted as a 16-bit long signed integer, into a numeric value. The `<expC>` is a string, two bytes in length, containing a binary numeric value

Example

```
nVar = bin2w(binVar)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BINCLOSE()

Class

Binary File Access

Purpose

Function to close a file opened for binary file access

Syntax

BINCLOSE([<expN>])

See Also

BIN2I(), BIN2L(), BIN2W(), BINCREATE(), BINOPEN(), BINREAD(), BINSEEK(), BINWRITE(), I2BIN(), L2BIN(), W2BIN()

Description

The BINCLOSE() function is used to close a binary file which was opened with the BINCREATE() or BINOPEN() functions. Associated buffers are written to the disk as the file is closed. <expN> is the file descriptor returned when the file was opened with either the BINCREATE() or BINOPEN() functions. The binary conversion functions can be used in conjunction with all the binary file functions.

Example

```
fd = binopen("file.obj")
count = binread(fd, 4)
count = bin2l(count)
count = l2bin(count + 1)
binseek(fd, 512, 0)
binwrite(fd, count)
binclose(fd)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BINCREATE()

Class

Binary File Access

Purpose

Function to create a binary file

Syntax

`BINCREATE([<expC>])`

See Also

`BIN2I()`, `BIN2L()`, `BIN2W()`, `BINCLOSE()`, `BINOPEN()`, `BINREAD()`, `BINSEEK()`, `BINWRITE()`, `I2BIN()`, `L2BIN()`, `W2BIN()`

Description

The `BINCREATE()` function creates a new binary file. The `<expC>` is the name of the file to be created. `BINCREATE()` leaves the created file open, and returns a file descriptor when successful, or a -1 if an error occurs. Since the file descriptor is used to identify an open binary file, always assign the `BINCREATE()` return value to a memory variable.

Example

```
fd = bincreate("file.obj")
count = binread(fd, 4)
count = bin2l(count)
count = l2bin(count + 1)
binseek(fd, 512, 0)
binwrite(fd, count)
binclose(fd)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BINOPEN()

Class

Binary File Access

Purpose

Function to open a binary file

Syntax

BINOPEN([<expC>,<expN>])

See Also

BIN2I(), BIN2L(), BIN2W, BINCLOSE(), BINCREATE(), BINREAD(), BINSEEK(), BINWRITE(), I2BIN(), L2BIN(), W2BIN()

Description

The BINOPEN() function opens an existing binary file. The <expC> is the name of the file to be opened. The <expN> is the mode in which the file will be opened. The open mode options are:

Mode	Description
0	Read-only
1	Write-only
2	Read/write

BINOPEN() returns a file descriptor if it has opened the file successfully, or a -1 if unsuccessful. Since the file descriptor is used to identify open binary files, the BINOPEN() return value should always be assigned to a memory variable. The binary conversion functions may be used in conjunction with the binary file functions.

Example

```
fd = binopen("file.obj")
count = binread(fd, 4)
count = bin2l(count)
count = l2bin(count + 1)
binseek(fd, 512, 0)
binwrite(fd, count)
binclose(fd)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BINREAD()

Class

Binary File Access

Purpose

Function to read a character string from a binary file

Syntax

`BINREAD([<expN1>,<expN2>])`

See Also

`BIN2I()`, `BIN2L()`, `BIN2W`, `BINCLOSE()`, `BINCREATE()`, `BINOPEN()`, `BINSEEK()`, `BINWRITE()`, `I2BIN()`, `L2BIN()`, `W2BIN()`

Description

The `BINREAD()` reads a character string from a binary file that was opened with the `BINCREATE()` or `BINOPEN()` functions. The `<expN1>` is the file descriptor which was returned when the file was opened with either the `BINCREATE()` or `BINOPEN()` functions. The `<expN2>` value represents the number of bytes to read. The `BINREAD()` function reads the specified number of bytes from the file starting at the current position of `BINSEEK()`. `BINREAD()` returns the character string in the specified length of `<expN2>` if successful, or a string of length zero if unsuccessful. The binary conversion functions may be used in conjunction with the binary file functions.

Example

```
fd = binopen("file.obj")
count = binread(fd, 4)
count = bin2l(count)
count = l2bin(count + 1)
binseek(fd, 512, 0)
binwrite(fd, count)
binclose(fd)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BINSEEK()

Class

Binary File Access

Purpose

Function to position the file pointer in binary file

Syntax

`BINSEEK([<expN1C>,<expN2>,<expN3>])`

See Also

`BIN2I()`, `BIN2L()`, `BIN2W()`, `BINCLOSE()`, `BINCREATE()`, `BINOPEN()`, `BINREAD()`, `BINWRITE()`, `I2BIN()`, `L2BIN()`, `W2BIN()`

Description

The `BINSEEK()` function is used to position to a byte offset within a binary file opened with the `BINCREATE()` or `BINOPEN()` functions. The `<expN1>` is the file descriptor which was returned when the file was opened with either the `BINCREATE()` or `BINOPEN()` functions. The `<expN2>` is the byte offset to move the file pointer based on the value defined by `<expN3>`. The value of `<expN3>` represents a positioning mode that defines the position from which to start the byte offset. The positioning mode options are:

Offset	Description
0	Start from the beginning of file
1	Start from the current position
2	Start from the end of file

The `BINSEEK()` function returns the new file position relative to the beginning of the file. The binary conversion functions may be used in conjunction with the binary file functions.

Example

```
fd = binopen("file.obj")
count = binread(fd, 4)
count = bin2l(count)
count = l2bin(count + 1)
binseek(fd, 512, 0)
binwrite(fd, count)
binclose(fd)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BINWRITE()

Class

Binary File Access

Purpose

Function to write an expression to a binary file

Syntax

`BINWRITE(<expN>,<expC>)`

See Also

`BIN2I()`, `BIN2L()`, `BIN2W`, `BINCLOSE()`, `BINCREATE()`, `BINOPEN()`, `BINREAD()`, `BINSEEK()`, `I2BIN()`, `L2BIN()`, `W2BIN()`

Description

The `BINWRITE()` is used to write a character expression to a binary file, and returns the number of bytes written. The `<expC>` specifies the character expression to write to the binary file. The `<expN>` is the file descriptor of the file to write to. The file descriptor is obtained when the binary file is opened, as the return value from either the `BINCREATE()` or `BINOPEN()` functions. The `BINWRITE()` function writes the character expression starting at the position returned by the `BINSEEK()` function. The binary conversion functions may be used in conjunction with all the binary file functions.

Example

```
fd = binopen("file.obj")
count = binread(fd, 4)
count = bin2l(count)
count = l2bin(count + 1)
binseek(fd, 512, 0)
binwrite(fd, count)
binclose(fd)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITAND()

Class

Bitwise Operations

Purpose

Function to perform bitwise AND operation

Syntax

BITAND(<expN1>,<expN2>[,...<expN26>])

See Also

BITCLEAR(), BITLSHIFT(), BITNOT(), BITOR(), BITRSHIFT(), BITSET(), BITTEST(), BITXOR()

Description

The BITAND() function performs a bitwise AND operation on the specified numeric parameters. Up to 26 parameters can be specified. These parameters, if not integers, will be converted to integer values before the operation takes place.

BITAND() compares each bit in turn of <expN1> and <expN2>. If both bits are 1, the corresponding bit in the result is set to 1, otherwise the result bit is 0. If <expN3> is specified, the initial result is compared bit by bit with <expN3> and a new result evaluated. This new result is then compared with <expN4>, if specified, and so on.

<expN1> bit	<expN2> bit	Result bit
0	0	0
0	1	0
1	1	1
1	0	0

Example

```
x = 3      && 0011
y = 6      && 0110
z = 7      && 0111
? bitand(x,y)
    2      && 0010
? bitand(x,y,z)
    2      && 0010
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITCLEAR()

Class

Bitwise Operations

Purpose

Function to clear a specified bit in a numeric value

Syntax

BITCLEAR(<expN1>,<expN2>)

See Also

BITAND(), BITLSHIFT(), BITNOT(), BITOR(), BITRSHIFT(), BITSET(), BITTEST(), BITXOR()

Description

The BITCLEAR() function clears the specified bit <expN2> in a numeric value <expN1> and returns the new value. If <expN1> and <expN2> are not integers, they will be converted to integer values before the clear takes place. The bit position, <expN2>, can range from 0 (rightmost bit) to 31.

Example

```
x = 6          && 0110
y = 1
? bitclear(x,y)
  4    && 0100
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITLSHIFT()

Class

Bitwise Operations

Purpose

Function to shift the bits in a numeric value a specified number of places to the left

Syntax

BITLSHIFT(<expN1>,<expN2>)

See Also

BITAND(), BITCLEAR(), BITNOT(), BITOR(), BITRSHIFT(), BITSET(), BITTEST(), BITXOR()

Description

The BITLSHIFT() function shifts the bits in the numeric value <expN1> the specified number of places to the left <expN2> and returns the new value. If <expN1> and <expN2> are not integers, they will be converted to integer values before the shift takes place.

Example

```
x = 6      && 0110
y = 1
? bitlshift(x,y)
    12      && 1100
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITNOT()

Class

Bitwise Operations

Purpose

Function to perform bitwise NOT operation

Syntax

BITNOT(<expN>)

See Also

BITAND(), BITCLEAR(), BITLSHIFT(), BITOR(), BITRSHIFT(), BITSET(), BITTEST(), BITXOR()

Description

The BITNOT() function performs a bitwise NOT operation on the specified numeric parameter. If <expN> is not an integer, it will be converted to an integer value before the operation takes place.

BITNOT() switches each bit in turn of <expN1>. Each 1 becomes a 0, and each 0 becomes a 1.

<expN> bit	Result bit
0	1
1	0

Example

```
x = 3      && 0...0011
? bitnot(x)
-4    && 1...1100
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITOR()

Class

Bitwise Operations

Purpose

Function to perform bitwise OR operation

Syntax

BITOR(<expN1>,<expN2>[,...<expN26>])

See Also

BITAND(), BITCLEAR(), BITLSHIFT(), BITNOT(), BITRSHIFT(), BITSET(), BITTEST(), BITXOR()

Description

The BITOR() function performs a bitwise OR operation on the specified numeric parameters. Up to 26 parameters can be specified. These parameters, if not integers, will be converted to integer values before the operation takes place.

BITOR() compares each bit in turn of <expN1> and <expN2>. If either bit is 1, the corresponding bit in the result is set to 1: if both bits are 0, the result bit is 0. If <expN3> is specified, the initial result is compared bit by bit with <expN3> and a new result evaluated. This new result is then compared with <expN4>, if specified, and so on.

<expN1> bit	<expN2> bit	Result bit
0	0	0
0	1	1
1	1	1
1	0	1

Example

```
x = 3      && 0011
y = 6      && 0110
z = 7      && 0111
? bitor(x,y)
    7      && 0111
? bitor(x,y,z)
    7      && 0111
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITRSHIFT()

Class

Bitwise Operations

Purpose

Function to shift the bits in a numeric value a specified number of places to the right

Syntax

BITRSHIFT(<expN1>,<expN2>)

See Also

BITAND(), BITCLEAR(), BITLSHIFT(), BITNOT(), BITOR(), BITSET(), BITTEST(), BITXOR()

Description

The BITRSHIFT() function shifts the bits in the numeric value <expN1> the specified number of places to the right <expN2> and returns the new value. If <expN1> and <expN2> are not integers, they will be converted to integer values before the shift takes place.

Example

```
x = 6      && 0110
y = 1
? bitrshift(x,y)
      3      && 0011
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITSET()

Class

Bitwise Operations

Purpose

Function to set a specified bit in a numeric value

Syntax

BITSET(<expN1>,<expN2>)

See Also

BITAND(), BITCLEAR(), BITLSHIFT(), BITNOT(), BITOR(), BITRSHIFT(), BITTEST(), BITXOR()

Description

The BITSET() function sets the specified bit <expN2> in a numeric value <expN1> to 1 and returns the new value. If <expN1> and <expN2> are not integers, they will be converted to integer values before the clear takes place. The bit position, <expN2>, can range from 0 (rightmost bit) to 31.

Example

```
x = 6          && 0110
y = 3
? bitset (x,y)
    14          && 1110
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITTEST()

Class

Bitwise Operations

Purpose

Function to test the value of a specified bit in a numeric value

Syntax

BITTEST(<expN1>,<expN2>)

See Also

BITAND(), BITCLEAR(), BITLSHIFT(), BITNOT(), BITOR(), BITRSHIFT(), BITSET(), BITXOR()

Description

The BITTEST() function tests the value of the specified bit <expN2> in a numeric value <expN1>. If the bit is 1, BITTEST() returns True (.T.), otherwise it returns False (.F.). If <expN1> and <expN2> are not integers, they will be converted to integer values before the clear takes place. The bit position, <expN2>, can range from 0 (rightmost bit) to 31.

Example

x = 6 && 0110

y = 3

z = 2

? bittest (x,y)

.F.

? bittest (x,z)

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BITXOR()

Class

Bitwise Operations

Purpose

Function to perform bitwise XOR (exclusive OR) operation

Syntax

BITXOR(<expN1>,<expN2>[,...<expN26>])

See Also

BITAND(), BITCLEAR(), BITLSHIFT(), BITNOT(), BITRSHIFT(), BITSET(), BITTEST(), BITXOR()

Description

The BITXOR() function performs a bitwise XOR (exclusive OR) operation on the specified numeric parameters. Up to 26 parameters can be specified. These parameters, if not integers, will be converted to integer values before the operation takes place.

BITXOR() compares each bit in turn of <expN1> and <expN2>. If only one of the bits is 1, the corresponding bit in the result is set to 1, otherwise the result bit is 0. If <expN3> is specified, the initial result is compared bit by bit with <expN3> and a new result evaluated. This new result is then compared with <expN4>, if specified, and so on.

<expN1> bit	<expN2> bit	Result bit
0	0	0
0	1	1
1	1	0
1	0	1

Example

```
x = 3      && 0011
y = 6      && 0110
z = 7      && 0111
? bitxor(x,y)
    5      && 0101
? bitxor(x,y,z)
    2      && 0010
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BLINK()

Class

Input/Output

Purpose

Function to display blinking text

Syntax

BLINK(<expC>)

See Also

BOLD(), REVERSE(), UNDERLINE(), MESSAGE, SET MESSAGE, SETSCREENIO

Description

The BLINK() function displays the specified character expression <expC> flashing on the screen. SET SCREENIO must be ON (default is off), for the BLINK() function to operate.

Example

```
if value < 0 or value > 10
    @23,0 say blink("Out of range")
endif
```

Products

Recital Terminal Developer

BOF()

Class

Fields and Records

Purpose

Function to check if the record pointer is at the beginning of file

Syntax

BOF([<workarea | alias>])

See Also

EOF(), FOUND()

Description

The BOF() function returns .T. if an attempt is made to skip the record pointer before the first logical record of the currently selected table. If the optional <workarea | alias> is specified, then the function will operate in the required location.

Example

use orders

? bof()

.F.

skip -1

? bof()

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

BOLD()

Class

Input/Output

Purpose

Function to display highlighted text

Syntax

BOLD(<expC>)

See Also

BLINK(), REVERSE(), UNDERLINE(), MESSAGE, SET MESSAGE, SET SCREENIO

Description

The BOLD() function displays the specified character expression <expC> in bold face on the screen. SET SCREENIO must be ON (default is off), for the BOLD() function to operate.

Example

@2,1 say bold("PATRON INFORMATION")

Products

Recital Mirage Server, Recital Terminal Developer

BRIDGE()

Class

Data Connectivity

Purpose

Function to return type of Replaceable Database Driver (RDD) in the active workarea

Syntax

BRIDGE()

See Also

CREATE BRIDGE, MODIFY BRIDGE, SET RELATION, SET VIEW, USE

Description

The BRIDGE() function returns the type of RDD in the active workarea. The BRIDGE() function returns a RDD type in a character string. If the active workarea contains a native Recital table or is empty, BRIDGE() returns "".

Example

?bridge()

XBASE

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CAGR()

Class

Numeric Data

Purpose

Function to calculate compound annual growth rate

Syntax

CAGR(<expN1>,<expN2>,<expN3>)

See Also

PAYMENT(), PMT(), FV(), PV(), LOG10(), PI()

Description

The CAGR() function calculates compound annual growth rate where the present <expN1> is the starting figure, the future <expN2> is the target figure at the end of the period, and years <expN3> is the growth period in years.

Example

```
? cagr(5000,10000,5)  
0.15
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CALC()

Class

Input/Output

Purpose

Function to perform report calculation

Syntax

CALC(<memvar>, <expN>)

See Also

ACC(), REPORT

Description

The CALC() function evaluates the specified numeric expression <expN>, places the result in the designated accumulator <memvar>. The return value of the CALC() function is the result of the numeric expression <expN>. This function is only valid in report format <.frm> files. The CALC() function should be used in conjunction with the ACC() function.

There are 20 accumulators available, which are cleared to zero at the start of the report (REPORT FORM command). The first 10 accumulators are cleared on each sub-total break, and the last 10 are carried through to the end of the report. These accumulators are useful for providing horizontal totaling facilities.

Example

```
calc(1, acc(1) + order_value)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CANCELPID()

Class

Environment

Purpose

Function to cancel a sub-process

Syntax

CANCELPID(<expN>)

See Also

ACTIVEPID(), SPAWNPID(), SPAWN

Description

The CANCELPID() function returns .T. if the specified process ID <expN> could be “killed” and .F. otherwise. The SPAWNPID() function is used to get the process ID of the last process spawned.

Example

```
spawn db program
m_activepid = spawnpid()
if activepid(m->m_activepid)
    if cancelpid(m->m_activepid)
        dialog box "Process Canceled."
    else
        dialog box "Process couldn't be Canceled."
    endif
else
    dialog box "There is no Process Active."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CANDIDATE()

Class

Indexing

Purpose

Function to determine whether an index tag is a candidate index tag.

Syntax

CANDIDATE([<expN1>][,<expN2> | <expC>])

See Also

ALIAS(), SELECT(), TAG(), TAGCOUNT(), TAGNO()

Description

The CANDIDATE() function is used to determine whether an index tag is a candidate index tag. The number of the index tag to be examined can optionally be specified in <expN1>. If <expN> is not specified, the CANDIDATE() function will operate on the master index tag of the currently active table. If the tag number, <expN>, is specified, the CANDIDATE() function can be used on tables other than the currently active table by specifying either the relevant workarea number, <expN2>, or the alias name of the table, <expC>.

NOTE: A candidate index tag is an index tag that is a likely candidate to become the primary index tag as it does not contain any null or duplicate values.

Example

```
if tagcount() > 0
  for I = 1 to tagcount()
    ? "Candidate status for " + tag() + " is " + Itos(candidate(i))
  next
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CAPSLOCK()

Class

Xbase Compatibility

Purpose

Language compatibility only

Syntax

CAPSLOCK([<expL>])

See Also

SET COMPATIBLE

Description

This function has been added for language compatibility only. The CAPSLOCK() function always returns .F..

Example

? capslock()

.F.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CAST()

Class

Expressions and Type Conversion

Purpose

Function to convert the data type of an expression

Syntax

CAST(<exp> AS <expC>[(<expN1>[, <expN2>]]) [NULL | NOT NULL])

See Also

DTOC(), DTOS(), ETOS(), LTOS(), MTOS(), STR()

Description

The CAST() function converts the expression in <exp> to the data type specified in <expC> and returns the result. The <expC> can be the full data type name, e.g. Character or supported abbreviation, e.g. C or Char. For data types requiring width and precision information, these are specified in <expN1> and <expN2> respectively.

Data Type Abbreviations:

Abbreviations	Data Type	Width Required	Precision Required
B	TINYINT/DOUBLE	No	No
C, Char	CHARACTER	Yes	No
D	DATE	No	No
F	FLOAT	Yes	No, defaults to 0
G	LONG VARBINARY/GENERAL	No	No
I, Int	INTEGER	Yes	No, defaults to 0
L	LOGICAL/BIT	No	No
M	LONG VARCHAR/MEMO	No	No
N, Num	NUMERIC	Yes	No, defaults to 0
P	PACKED DECIMAL	Yes	No, defaults to 0
Q	QUAD	Yes	No, defaults to 0
R	REAL	Yes	No, defaults to 0
S	SHORT	Yes	No, defaults to 0
T	DATETIME	No	No
V	VAXDATE	No	No
Y	CURRENCY	No	No
Z	ZONED	Yes	No, defaults to 0

If the specified width, <expN1> is less than that of <exp>, the result will be truncated. If greater, the result will be padded. Precision may be lost if the precision specified in <expN2> is less than that of <exp>.

The optional NULL | NOT NULL determines whether null values are permitted or not.

Example

```
> m_var = "date"  
> ? cast("12/12/2005" as (m_var))  
12/12/2005
```

```
> open database southwind  
> set sql on  
Recital/SQL> select cast(limit-balance as numeric(10,2)) from example;
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CATALOG()

Class

Information Center

Purpose

Function to return the name of the currently active catalog file

Syntax

CATALOG()

See Also

DESIGN, INFO, SET CATALOG, SET DESIGN, SET TITLE

Description

The CATALOG() function returns the name of the currently active catalog file. If no catalog file is in use, the CATALOG() function returns a null string.

A catalog file is an object used to group and access files that pertain to a single application. Catalogs allow you to define your work environment by adding only those files that are needed. The Recital Information Center displays catalogs of files by listing filenames in “panels” of the following categories: Data, Text, Form, Report, Label, and Program.

Example

set catalog to customer

?catalog()

customer.cat

Products

Recital Terminal Developer

CROW()

Class

Date and Time Data

Purpose

Function to return the character day of week

Syntax

CROW(<expD> | <expT>)

See Also

AMPM(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EPOCH(), GOMONTH(), HOUR(), HOURS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), TIME(), TIMESTAMP(), TSTRING(), TOC(), TOD(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

CROW() returns the name of the day of the week from the specified date expression <expD> or datetime expression <expT> as a character string.

Example

store crow(date()) to dayofweek

? dayofweek

Sunday

dayofweek = crow(date())

? dayofweek

Sunday

dayofweek = crow(datetime())

? dayofweek

Sunday

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CDX()

Class

Indexing

Purpose

Function to return the active multiple index filename

Syntax

CDX([<expN>[,<alias>]])

See Also

FOR(), KEY(), NDX(), ORDER(), SET(), TAG(), TAGCOUNT(), TAGNO(), SET COMPATIBLE, SET FILECASE, SET FULLPATH

Description

The CDX() function returns the name of the currently open multiple index file. Used without any parameters, the CDX() function will return the name of the multiple index file in the current workarea. If there is no .dbx file in the workarea, the CDX() function will return a null string. The optional numeric expression, <expN>, may be used to return the .dbx file name for a specific tag number. This parameter is needed if more than one .dbx file is open in a workarea. You may optionally specify an alias name to use the CDX() function in other workareas. The <alias> may be a workarea number or letter (1-DB_MAXWKA, a-z) or the table alias name.

If SET COMPATIBLE is set to FOXPRO or VFP the CDX() return value format differs in the following way: if SET FULLPATH is ON the full path to the index file is returned, if SET FILECASE is OFF then the return value is converted to upper case (Windows only).

Example

```
?cdx(1, account)
invoice.dbx
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CEILING()

Class

Numeric Data

Purpose

Function to return the smallest integer that is greater than or equal to a given value

Syntax

CEILING(<expN>)

See Also

SIGN(), FLOOR(), ISDIGIT()

Description

The CEILING() function is used to return the smallest integer that is greater than or equal to a given value in the numeric expression <expN>.

Example

? ceiling(354.89)

355

? ceiling(354.19)

355

? ceiling(-354.89)

-354

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CENTER() | CENTRE()

Class

String Data

Purpose

Function to center a character expression

Syntax

CENTER(<expC>,<expN>)

CENTRE(<expC>,<expN>)

See Also

REVERSE(), UNDERLINE(), BOLD(), MESSAGE, SET MESSAGE

Description

The CENTER() function centers the character expression <expC> into a character string which is <expN> characters wide. Note that if the MESSAGE or SET MESSAGE TO commands are used, the message text is automatically centered.

Example

```
? center("Recital", 20)
```

Recital

```
? "" + center("Recital", 20) + ""
```

```
' Recital '
```

```
heading = center("Recital", 20)
```

```
? type("heading")
```

C

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CERROR()

Class

Error Handling and Debugging

Purpose

Function to return the number of the last error encountered during a compile operation

Syntax

CERROR()

See Also

ERROR(), MESSAGE(), COMPILE, SET COMPILE, SET DEVELOPMENT, DO, SET PROCEDURE

Description

The CERROR() function returns the number of the last error message encountered during a compile operation. If no compiler errors are found, the CERROR() function returns a zero. The COMPILE command is used to compile program files, however, other commands may also cause a program file to be compiled. These commands include: DO, SET PROCEDURE and MENU FORMAT.

Example

```
procedure comp_err
on error
if cerror() > 0
    dialog box "Compile Error#" +;
        str(cerror(),4)
endif
return
```

```
on error do com_err
compile proclib.prg
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CHANGE()

Class

Manual Locking

Purpose

Determine whether a record has been updated since it was last read

Syntax

CHANGE()

See Also

GATHER, READ, REPLACE , SET LOCKTYPE

Description

The CHANGE() function can be used in conjunction with SET LOCKTYPE TO OPTIMISTIC to determine whether a record has been updated since it was last read. If the records has been updated by another process since it was last read, the CHANGE() function will return .T. (TRUE), otherwise it will return .F. (FALSE).

Example

```
set locktype to optimistic
use customer
store automem
@1,1 get m.name
@2,1 get m.address
@3,1 get m.state
read
if not change()
    replace customer.name with m.name,;
    customer.address with m.address,;
    customer.state with m.state
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CHR()

Class

Expressions and Type Conversion

Purpose

Function to convert a number to an ASCII character

Syntax

CHR(<expN>)

See Also

ASC(), CTRL()

Description

The CHR() function returns the ASCII character denoted by the numeric value <expN>. CHR(0) should not be used to convert a field that is part of an index expression, as this will damage the index when the record containing that field is read.

Example

? chr(66)

B

? chr(7)

bell = chr(7)

? type("bell")

C

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CHROVERLAP()

Class

Expressions and Type Conversion

Purpose

Function to test for a character existing at the same position within two strings

Syntax

CHROVERLAP(<expC1>, <expC2> [, <expC3>])

See Also

AT(), GETGID(), GETUID(), STRCOMPARE(), STREXTRACT(), SUBSTR()

Description

The CHROVERLAP() function tests for a character existing at the same position within two strings. The <expC1> defines the source string and <expC2> the test string. The CHROVERLAP() function tests each character within the test string against the equivalent character in the source string, up to the length of the source string. By default, the CHROVERLAP() function will test for the existence of the upper case letter 'Y'. The optional <expC3> can be used to test for an alternative letter. The CHROVERLAP() function returns .T. if it finds a matching occurrence of the test character within both strings.

This function is particularly useful for controlling users' access rights to modules within an application. The example below shows how users' privileges can be stored in a string and checked using the CHROVERLAP() function.

Example

```
cUserPrivs = "YYNYNNY"
if chroverlap(cUserPrivs, "NNNNYNNN")
    SysMaintenanceMenu
else
    dialog box [You are not authorized to perform this operation.]
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CHRTRAN()

Class

String Data

Purpose

Function to search for and replace text within a character string or memo field

Syntax

CHRTRAN(<expC1> | <memofield>, [<expC2>, <expC3>, [<expN1> [,<expN2>]]])

See Also

AT(), ATNEXT(), STREXTRACT(), SUBSTR(), TRIM(), STUFF() STR(), STRZERO(), MTOS(), RAT()

Description

The CHRTRAN() function will search <expC1> or <memofield> and replace text within <expC1> or <memofield> with <expC3> wherever the occurrence of <expC2> is found. The optional <expN1> specifies the start position for the search in <expC1>. If this is not specified then the default is 1. The optional <expN2> specifies the number of occurrences of <expC2> to replace with <expC3>. If this is not specified, then all occurrences starting from <expN1> are replaced. This function can be used to remove blank spaces in character strings. The CHRTRAN() function returns a character expression that contains the result of the string translation.

The CHRTRAN() function is synonymous with the STRTRAN() function.

Example

```
? chrtran("Hello World", "ello", "i")
Hi World
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CLEARRESULTSET()

Class

Data Connectivity

Purpose

Function to clear the marker from an SQL cursor previously marked as a resultset

Syntax

CLEARRESULTSET()

See Also

GETRESULTSET(), SETRESULTSET(), SQL SELECT

Description

The CLEARRESULTSET() function clears the marker from an SQL cursor previously marked as a resultset by the SETRESULTSET() function. The SETRESULTSET() function is particularly used in returning a resultset from a stored procedure in SQL client/server applications.

CLEARRESULTSET() will return the workarea number of the SQL cursor from which the marker was cleared, or 0 (zero) if no SQL cursor was marked as a resultset.

Example

```
function GetExampleCursor
lparameters lcAccountNo
select * from example where account_no = lcAccountNo into cursor curExample
return setresultset("curExample")

open database southwind
GetExampleCursor("00050")
select * from curexample
? "Cleared resultset marker in work area #" + ltrim(str(clearresultset()))
? iif(getresultset() > 0, "Resultset available in work area #" + ltrim(str(getresultset())) ,
    "No resultsets available")
?
close databases
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CLOSEMAIL()

Class

Mail

Purpose

Function to disconnect from mail server

Syntax

CLOSEMAIL()

See Also

COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The CLOSEMAIL() function will disconnect you from the mail server. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() (POP3 only) or MAILNODENAME() function to check if you are connected.

If successful the CLOSEMAIL() will return .T., otherwise .F.. The MAILERROR() function can be used to return the error message if the CLOSEMAIL() fails.

Example

closemail()

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CMONTH()

Class

Date and Time Data

Purpose

Function to return the character month from a specified date or datetime

Syntax

CMONTH(<expD> | <expT>)

See Also

AMPM(), CDOW(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EPOCH(), GOMONTH(), HOUR(), HOURS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

CMONTH() is the character month function. It returns the name of the month from the specified date <expD> or the specified datetime <expT> as a character string.

Example

```
? cmonth(datetime())
```

October

```
? cmonth(date())
```

October

```
store cmonth(date()) to month
```

```
? month
```

October

```
? type("month")
```

C

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CNTVALUES()

Class

Fields and Records

Purpose

Function to return the number of records matching a series of keys

Syntax

CNTVALUES([<for condition> [, <key expression>]])

See Also

SUMVALUES(), AVGVALUES(), MINVALUES(), MAXVALUES(), SQLVALUES()

Description

The CNTVALUES() function returns the number of records matching a series of keys. If no parameters are specified then the number of keys matching the current key is returned. An optional <for condition> can be specified to restrict the key count. You can also optionally specify a <key-expression> to perform the count on instead of the current key.

After completion, all record pointers and indexes are returned to their original positions.

Example

```
use customer order title
// Number of records matching the current key
count = cntvalues()
// Number of records matching the key "Mr"
m_male = cntvalues(.T., "Mr")
// Number of records matching the current key with an expiry date < today
m_expired = cntvalues(expiry < date())
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

COL()

Class

Screen Forms

Purpose

Function to return screen column position

Syntax

COL()

See Also

ROW(), PROW(), PCOL(), SET DEVICE, SCOLS(), SROWS()

Description

The COL() function returns the current column position of the cursor on the screen. The current cursor position is updated whenever you issue the @...SAY...GET or MENU TO commands. Normal output using other commands does not have an effect. The main use of the COL() function is to calculate relative cursor addressing when designing forms and menus. If you have issued the SET DEVICE TO PRINT command, the printer column position is updated rather than the screen column position, see PCOL() for details.

Example

```
@1,1
do while col() < 20
    @1, col()+1 say "-"
enddo
```

Products

Recital Mirage Server, Recital Terminal Developer

COLLATE()

Class

Indexing

Purpose

Function to translate character expression into its correct collating sequence

Syntax

COLLATE(<expC>)

See Also

INDEX ON, SET LANGUAGE TO

Description

The COLLATE() function is used in conjunction with the SET LANGUAGE TO command for sorting or indexing. The SET LANGUAGE TO command allows use of foreign language character sets by using the character translation table specified in the “terminals” directory. The COLLATE() function then recognizes foreign characters and places the specified character expression, <expC>, correctly within the existing sequence. The COLLATE() function can translate and sort key expressions so that the INDEX ON command creates index files which include foreign characters.

Example

```
set language to french
use customers
index on collate(custname) to frenchcust
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

COMPLETED()

Class

Transaction Processing

Purpose

Function to determine whether an error occurred during a multi-statement transaction

Syntax

COMPLETED()

See Also

SET ROLLBACK, BEGIN TRANSACTION, ISMARKED(), RESET IN, ROLLBACK, ROLLBACK(), END TRANSACTION

Description

The COMPLETED() function returns .T. if no errors occurred during processing of any command issued between the BEGIN TRANSACTION and END TRANSACTION commands.

Example

```
procedure recovery
rollback
if rollback()
    dialog box "Rollback was ok."
else
    dialog box "Rollback failed."
endif
return

use setcomm
on error do recovery
begin transaction
    reset in comm_2
    delete first 15
    insert
    replace all t1 with (t2*t3)/100
    list
end transaction
if completed()
    dialog box "Transaction completed OK"
else
    dialog box "Errors occurred during transaction"
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

COMPOBJ()

Class
Objects

Purpose
Function to compare two objects

Syntax
COMPOBJ(<object-ref1>,<object-ref2>)

See Also
CLASS, DEFINE CLASS, WITH...ENDWITH, ACLASS(), AMEMBERS(), CREATEOBJECT(),
NEWOBJECT(), OBJECTREAD(), OBJECTTYPE(), OBJECTWRITE()

Description
The COMPOBJ() function is used to compare the property name and property values of two objects. The <object-ref1> is a reference to the first object to be compared and the <object-ref2> is a reference to the second object. If the property names and values of the two objects are identical, COMPOBJ() will return .T. (true), if not it will return .F. (false).

Example
class myclass1
property firstname
property lastname
endclass

class myclass2
property firstname
property lastname
endclass

class myclass3
property firstname
property lastname
property midinitial
endclass

o1 = new myclass1()
o2 = new myclass2()
o3 = new myclass3()

o1.firstname = "John"
o1.lastname = "Johnson"

o2.firstname = "Jack"
o2.lastname = "Johnson"

o3.firstname = "John"
o3.lastname = "Johnson"


```
? compobj(o1,o2)
// Returns .F. as firstname property has different values

o2.firstname = "John"
? compobj(o1,o2)
// Returns .T. as properties and values identical

? compobj(o1,o3)
// Returns .F. as although firstname and lastname property is identical, o3 has a midinitial property and o1
// does not
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CONNECTED()

Class

Data Connectivity

Purpose

Function to determine whether there is a gateway connection active in a workarea

Syntax

CONNECTED([<workarea | alias>])

See Also

GATEWAY(), SET GATEWAY, USED()

Description

The CONNECTED() function returns .T. if the workarea specified by <workarea | alias> is connected to a Recital Database Gateway. If there is no gateway connection in the specified workarea then .F. is returned. If no <workarea | alias> is specified then the CONNECTED() function defaults to the current workarea.

Example

set gateway to "ora@sales:scott/tiger.tcpip"

? connected()

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

COPYFILEFROM()

Class

Disk and File Utilities

Purpose

Function to copy a file from the client to the server

Syntax

COPYFILEFROM(<expC1>, <expC2>, <expC3>, <expL>)

See Also

COPYFILETO(), SHOWDOCUMENT()

Description

The COPYFILEFROM() function is used to copy files from the client to the server. It will return TRUE if the file transfer is successful, otherwise it will return FALSE.

Keyword	Description
<expC1>	The file transfer type: "BINARY" or "TEXT".
<expC2>	The name of the server file to be created.
<expC3>	The name of the client file to be transferred.
<expL>	If .T. (TRUE), the file transfer status is displayed in the status bar.

Example

```
if copyfilefrom("TEXT", "/usr/recital/UAS/mirage/mirage_demo/mirage_demo.prg ", ;  
    " C:\Program Files\Recital\UAS\Mirage\Mirage_demo\Mirage_demo.prg ", .T.)  
    dialog box "File transfer successful"  
else  
    dialog box "File transfer failed"  
endif
```

Products

Recital Mirage Server

COPYFILETO()

Class

Disk and File Utilities

Purpose

Function to copy a file from the server to the client

Syntax

COPYFILETO(<expC1>, <expC2>, <expC3>, <expL>)

See Also

COPYFILEFROM(),SHOWDOCUMENT()

Description

The COPYFILETO() function is used to copy files from the server to the client. It will return TRUE if the file transfer is successful, otherwise it will return FALSE.

Keyword	Description
<expC1>	The file transfer type: "BINARY" or "TEXT".
<expC2>	The name of the server file to be transferred.
<expC3>	The name of the client file to be created.
<expL>	If .T. (TRUE), the file transfer status is displayed in the status bar.

Example

```
if copyfileto("TEXT","/usr/recital/UAS/mirage/mirage_demo/mirage_demo.prg ", ;
    " C:\Program Files\Recital\UAS\Mirage\Mirage_demo\Mirage_demo.prg ",.T.)
    dialog box "File transfer successful"
else
    dialog box "File transfer failed"
endif
```

Products

Recital Mirage Server

COS()

Class

Numeric Data

Purpose

Function to return cosine value

Syntax

COS(<expN>)

See Also

EXP(), LOG(), LOG10(), SIN(), TAN(), RTOD(), DTOR(), PI(), SET DECIMALS

Description

The COS() function returns the cosine of the angle <expN> expressed in radians. The DTOR() function can be used for converting degrees to radians. The SET DECIMALS command determines the number of decimal places returned.

Example

set decimals to 6

? cos(dtor(30))

0.866025

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

COUNTMAIL()

Class

Mail

Purpose

Function to return the number of messages on the currently connected mail server

Syntax

COUNTMAIL([<array>])

See Also

CLOSEMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The COUNTMAIL() function returns the number of messages on the connected mail server. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() (POP3 only) or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<array>	No	None	The name of an array to contain the size in bytes of each mail message. No pre-declaration is required.

Example

```
value = countmail(message_size)
```

```
? value
```

```
6
```

```
? message_size[1]
```

```
365
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CREATEOBJECT()

Class

Objects

Purpose

Function to create a new object based on a defined class.

Syntax

CREATEOBJECT(<expC> [,<exp1>,<exp2>...)

See Also

DEFINE CLASS...ENDDEFINE, WITH...ENDWITH, ADDPROPERTY(), NEWOBJECT(), REMOVEPROPERTY()

Description

The Visual FoxPro compatible CREATEOBJECT() function is used to create a new object based on a defined class. The name of the class is specified in <expC>. Any parameters required for the object creation are specified in a comma-separated list after the class name.

The CREATEOBJECT() function returns a reference to the newly created object.

Example

```
myObject = createobject("MyClass")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CTOD()

Class

Expressions and Type Conversion

Purpose

Function to convert character to date

Syntax

CTOD(<expC>)

See Also

CDOW(), CMONTH(), DATE(), DAY(), DMY(), DOW(), DTOC(), DTOS(), MDY(), MONTH(), STOD(), TIME(), WEEK(), YEAR(), SET DATE, SET CENTURY

Description

The CTOD() function is the character to date conversion function. It converts the character expression specified to a date variable. The character expression must be correctly formatted based on the current SET DATE, SET MARK and SET CENTURY settings. For example, the default settings, SET DATE AMERICAN and SET CENTURY ON, require a date in the format "MM/DD/YYYY". If <expC> is an invalid date format the CTOD() function will return an empty date.

Example

```
mdate = ctod("11/10/2003")
```

```
? mdate
```

```
11/10/2003
```

```
? type("mdate")
```

```
D
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CTOT()

Class

Expressions and Type Conversion

Purpose

Function to convert character to datetime

Syntax

CTOT(<expC>)

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The CTOT() function is the character to datetime conversion function. It converts the character expression specified to a datetime variable. The character expression must be correctly formatted based on the current SET DATE, SET MARK, SET SECONDS and SET CENTURY settings. For example, the default settings, SET DATE AMERICAN, SET SECONDS ON and SET CENTURY ON, require a datetime in the format “MM/DD/YYYY HH:MM:SS AM/PM”. If <expC> is an invalid datetime format the CTOT() function will return an empty datetime.

Example

```
mdate = ctot("01/21/2004 01:44:44 PM")
```

```
? mdate
```

```
01/21/2004 01:44:44 PM
```

```
? type("mdate")
```

```
T
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CTRL()

Class

Expressions and Type Conversion

Purpose

Function to return the numeric value of a control character

Syntax

CTRL(<expC>)

See Also

READKEY(), INKEY(), LASTKEY(), NEXTKEY(), SET KEY, @...MENU, @...GET

Description

The CTRL() function returns the numeric value of the specified control character. This function is extremely useful, when used in conjunction with the LASTKEY(), NEXTKEY() or READKEY() functions, for creating terminal independent programs.

Example

```
if readkey()=ctrl('g')
    set message to "Operation canceled."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CURDIR()

Class

Disk and File Utilities

Purpose

Function to return current device and directory

Syntax

CURDIR([<expN>])

See Also

DIR (), SET DEFAULT, SYS()

Description

The CURDIR() function returns the name of the current directory. The optional <expN> is used to the return the current device specification as well as the directory on OpenVMS. The <expN> must return a 1. The CURDIR() function always returns a character string without changing the case.

Example

```
? curdir()
```

```
[USERS.RECITAL]
```

```
?curdir(1)
```

```
DIA1:[USERS.RECITAL]
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CURRENCY()

Class

Numeric Data

Purpose

Function to prefix numeric value with currency character

Syntax

CURRENCY(<expN>)

See Also

SET CURRENCY

Description

The CURRENCY() function returns the specified <expN> as a character string prefixed with the current currency character which can be set with the command SET CURRENCY TO <expC>. The default for SET CURRENCY is “\$”.

Example

set currency to “\$”

? currency(88.00)

\$88.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CURRSEQNO()

Class

Table Basics

Purpose

Function to return the current 'SEQNO' value

Syntax

CURRSEQNO([<workarea | alias>])

See Also

ZAP, SEQNO(), SET SEQNO

Description

The CURRSEQNO() function returns the current 'SEQNO' value. The SEQNO value is a unique sequence number generated on a per table basis. The CURRSEQNO() function differs from the SEQNO() function in that CURRSEQNO() returns the current sequence number for a table and SEQNO() increases the current sequence number by one and then returns the value. Including the optional <workarea | alias> parameter will cause the CURRSEQNO() function to operate on the specified table. If there is no active table the CURRSEQNO() function will return 0.

A ZAP operation will reset the sequence number to 0. The sequence number of a table can be reset with the command SET SEQNO TO <expN>.

Example

use newtable

? seqno()

1

? seqno()

2

? currseqno()

2

? currseqno()

2

set seqno to 0

? currseqno()

0

? seqno()

1

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

CURSORNAME()

Class

Data Connectivity

Purpose

Function to return the name of the cursor open in a workarea with an active gateway

Syntax

CURSORNAME([<workarea | alias>])

See Also

CLOSE, DECLARE CURSOR, DROP CURSOR, FETCH, FINDCURSOR(), OPEN CURSOR, GATEWAY()

Description

The CURSORNAME() function returns the cursor name corresponding to the workarea specified by <workarea | alias>. If there is no cursor open, or no gateway active in the specified workarea then an empty string is returned. If no <workarea | alias> is specified then the CURSORNAME() function defaults to the current workarea.

Example

```
set gateway to "ora@sales:scott/tiger"
```

```
exec sql
```

```
    declare employees cursor for
    select empno, ename, job
    from emp
    order by deptno;
```

```
exec sql
```

```
    open employees;
```

```
? cursorname()
```

```
EMPLOYEES
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DATABASE()

Class

Databases

Purpose

Function to return the name of the currently open database

Syntax

DATABASE()

See Also

CLOSE DATABASES, DISPLAY STATUS, LIST STATUS, OPEN DATABASE, USE, ADATABASES(), ALIAS(), DBF(), DBUSED(), GETENV(), USED(), SET FILECASE, SET SQL

Description

The DATABASE() function returns the name of the currently open database or a null string if none is open.

Databases in Recital are implemented as directories containing files that correspond to the tables and associated files in the database. Operating System file protection can be applied individually to the files for added security. The directories are sub-directories of the Recital data directory. The environment variable / symbol DB_DATADIR points to the current Recital data directory and can be queried using the GETENV() function. Files from other directories can be added to the database using the ADD TABLE command or via the database catalog and SET AUTOCATALOG functionality.

Databases can be opened using the SQL USE command, with SQL set to MYSQL, or using the SQL OPEN DATABASE command.

Example

```
VFP/SQL> OPEN DATABASE hr
```

```
VFP/SQL> ? database()
```

```
hr
```

```
VFP/SQL> CLOSE DATABASES
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DATE()

Class

Date And Time Data

Purpose

Function to return current system date or a date from specified year, month and day values

Syntax

DATE([<expN1>, <expN2>, <expN3>])

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DATE() function returns the current system date as a date type. The display format of dates is affected by the SET CENTURY, SET DATE, SET EPOCH and SET MARK set commands.

The optional <expN1>,<expN2>,<expN3> can be used to specify numeric years, months and days and return a valid corresponding date. If any parameter is invalid, an empty date is returned.

<expN1>	A valid number of years, -100 (1900) to 900 (2900)
<expN2>	A valid number of months, 1 to 12
<expN3>	A valid number of days, 1 to 31

Example

```
? date()
04/04/2004
mdate = date()
? mdate
04/04/2004
? type("mdate")
D
? date(4,4,4)
04/04/2004
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DATETIME()

Class

Date And Time Data

Purpose

Function to return current system date and time or a datetime from specified date and time values

Syntax

DATETIME([<expN1>, <expN2>, <expN3> [, <expN4> [, <expN5> [, <expN6>]]]])

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TROC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DATETIME() function returns the current system date and time as a datetime type. The display format of the date part of datetimes is affected by the SET CENTURY, SET DATE, SET EPOCH and SET MARK set commands. The SET SECONDS set command determines whether seconds are displayed in the time part of datetimes. The SET HOURS set command determines whether hours are shown in 24 hour format or in 12 hour format with AM | PM postfix.

The optional parameters can be used to specify numeric years, months, days, hours, minutes and seconds and return a valid corresponding datetime. If any date parameter is invalid, an empty datetime is returned. If any time parameter is invalid, an out of range error is generated.

<expN1>	A valid number of years, -100 (1900) to 900 (2900)
<expN2>	A valid number of months, 1 to 12
<expN3>	A valid number of days, 1 to 31
<expN4>	A valid number of hours, 0 to 23
<expN5>	A valid number of minutes, 0 to 59
<expN6>	A valid number of seconds, 0 to 59

Example

? datetime()

04/04/2004 11:54:45 AM

mdatetime = datetime()

? mdatetime

04/04/2004 11:54:45 AM

? type("mdatetime")

T

? datetime(4,4,4,4,4,4)

04/04/2004 04:04:04 AM

? datetime(4,4,4,14)

04/04/2004 02:00:00 PM

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DAY()

Class

Date and Time Data

Purpose

Function to return the day of the month from a specified date or datetime

Syntax

DAY(<expD> | <expT>)

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EPOCH(), GOMONTH(), HOUR(), HOURS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DAY() function returns the day of the month from the specified date expression <expD> or datetime expression <expT> as a numeric value.

Example

```
? day({01/01/2004})
1
store day({01/01/2004}) to m_Day
? m_Day
1
m_Day = day(date())
? type("m_Day")
N

? day({10/10/2004 10:15:43 AM})
10
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DAYS()

Class

Date and Time Data

Purpose

Function to return the number of whole days in a period expressed in seconds

Syntax

DAYS(<expN>)

See Also

SET CLOCK, SET CLOCKRATE, TIME(), VALIDTIME(), SECS(), TSTRING(), ELAPTIME(), HOURS(), MINUTES(), SECONDS(), AMPM()

Description

The DAYS() function returns the number of whole days in a specified period expressed in seconds from the given <expN>. The DAYS() function will return a value of zero if <expN> is less than 43200 (12 hours). For values between 43200 and 129599 (35 hours, 59 minutes, 59 seconds), DAYS() will return 1. For values of 36 hours to 59 hours, 59 minutes, 59 seconds the DAYS() function will return 2. This pattern continues as the number of seconds gets larger. The TSTRING() function can be used to return the duration of any remaining portions of days.

Example

```
? days(300033)
```

3

```
m_secs=357812
```

```
? "The period was "+alltrim(str(days(m->m_secs)))+;
```

```
" days and "+tstring(m->m_secs) +" hours"
```

The period was 4 days and 03:23:32 hours

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DBEDIT()

Class

Screen Forms

Purpose

Function to display expressions and records from one or more workareas in a browse style worksurface

Syntax

```
DBEDIT(<expN1>,<expN2>,<expN3>,<expN4>[,<array1>  
[, <expC1>[, <array2>[, <array3>[, <expC2>  
[, <expC3>[, <expC4>[, <array4>[, <expC5>  
[, <expC6>]]]]]]]]))
```

See Also

FUNCTION, BROWSE, MENU BROWSE, DEFINE TABLE, ABROWSE(), ACHOICE()

Description

The DBEDIT() function is used to display expressions and records from one or more workareas in a browse style worksurface.

Parameters	Description
<expN1>	The starting row number of the window.
<expN2>	The starting column number of the window.
<expN3>	The ending row number of the window.
<expN4>	The ending column number of the window.
<array1>	The name of an array containing expressions of field names from active workareas. If the array is not specified all fields from the active workarea will be displayed.
<expC1>	The name of a user-defined function (UDF) to execute when a key is pressed. This expression must not contain parameters or parentheses.
<array2>	The name of an array that contains PICTURE strings to format each column.
<array3>	The name of an array which contains headings for each column. Headings may be placed on more than one line if a semicolon is used as a line separator.
<expC2>	A character to be used for separating the headings from the field display.
<expC3>	A character to be used to separate the columns.
<expC4>	A character to be used to separate the column footers from the field display.
<array4>	The name of an array that contains column footers. Footers may be placed on more than one line if a semicolon is used as a line separator.
<expC5>	A FILTER condition passed as a character string.
<expC6>	A WHILE condition passed as a character string.

DBEDIT() passes two parameters to the user-defined function (UDF) <expC2>. The first parameter passed is the status of the DBEDIT() function. The second parameter is a number representing the ordinal position of the field. The following table summarizes the possible values that can be passed.

Status	Description
0	DBEDIT() is idle, no keystrokes are pending.
1	An attempt was made to move beyond the top of the file.
2	An attempt was made to move beyond the bottom of the file.
3	The table is empty.
4	A key other than a cursor navigation key was pressed. The LASTKEY() function can be used to check the key pressed.

The UDF must return one of the following values:

Value	Description
0	Quit DBEDIT().
1	Continue DBEDIT().
2	Reread data, repaint the screen and continue.

Example

```
use demo
declare flds[fcount()]
declare col_header[fcount()]
declare col_footer[fcount()]
declare pic[fcount]

for i=1 to fcount()
    // Load field names
    flds[i] = field(i)
    // Load headings
    col_header[i] = field(i)
    // Load footers
    col_footer[i] = field(i)
next

clear
dbedit(4,1,23,71,flds,"db_udf",pic,col_header,;
    chr(205),chr(186),chr(205),col_footers)
```

Products

Recital Terminal Developer

DBF()

Class

Table Basics

Purpose

Function to return the table file name

Syntax

DBF([<workarea | alias>])

See Also

ALIAS(), FCOUNT(), FIELD(), FILTER(), FMT(), INDEXKEY(), NDX (), READVAR(), SELECT(), WORKAREA(), SET COMPATIBLE, SET FILECASE, SET FULLPATH

Description

The DBF() function returns the file name of the currently active table or a null string if none is active. If the optional <workarea | alias> is specified, then the function will return the table name from the specified workarea. The DBF() function returns a character string in lower case, including the file extension. If the command FULLPATH is set ON, then the DBF() function will return the node, disk and directory name with the table name.

If SET COMPATIBLE is set to FOXPRO or VFP the DBF() return value format differs in the following way: if SET FULLPATH is ON the full path to the database table is returned, if SET FILECASE is OFF then the return value is converted to upper case (Windows only).

Example

```
use patrons
```

```
? dbf()
```

```
patrons.dbf
```

```
dbfname = dbf()
```

```
? dbfname
```

```
patrons.dbf
```

```
? len(dbfname)
```

```
11
```

```
use
```

```
dbfname = dbf()
```

```
? len(dbfname)
```

```
0
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DBFILTER()

Class

Table Basics

Purpose

Function to return the filter expression

Syntax

DBFILTER([<workarea | alias>])

See Also

SET FILTER, SET DELETED, SET VIEW, FILTER()

Description

The DBFILTER() function will return, as a character string, the current filter condition or a null string if no filter is set. If the <workarea | alias> is specified, then the function will return the active filter condition for the specified alias. The command SET FILTER ON | OFF has no affect on the DBFILTER() function.

Example

```
use patrons
set filter to event="BALLET"
? "[" + dbfilter() + "]"
[event="BALLET"]
? type("dbfilter()")
C
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DBRELATION()

Class

Table Basics

Purpose

Function to return table-linking expression

Syntax

DBRELATION(<expN>)

See Also

TARGET(), RELATION(), SET RELATION, CREATE VIEW, CREATE BRIDGE, SET VIEW, USE, DBRSELECT()

Description

The DBRELATION() function is synonymous with the RELATION() function. The DBRELATION() function returns the linking expression of a specified relation in the current workarea. The <expN> specifies the position of the desired linking expression from the list of previously defined relations. By default, the Recital environment supports 20 workareas but this can be increased, by setting the environment symbol DB_MAXWKA to a higher value, which allows for up to (DB_MAXWKA-1) relationships. The DBRELATION() function always returns a character string in lower case. If there is no linking expression defined for the <expN> selected a null string will be returned.

Example

```
use shows in 2 index pcode
use patrons in 1
set relation to patron_code into shows
? relation(1)
```

patron_code

```
? target(1)
```

2

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DBRSELECT()

Class

Table Basics

Purpose

Function to return the target workarea of a relation

Syntax

DBRSELECT(<expN>)

See Also

SET RELATION, CREATE VIEW, CREATE BRIDGE, SET VIEW, USE, DBRELATION()

Description

The DBRSELECT() function is synonymous with the TARGET() function. The DBRSELECT() function returns the numeric target workarea of a specified relation defined in the current workarea. By default, the Recital environment supports 20 workareas but this can be increased, by setting the environment symbol DB_MAXWKA to a higher value, which allows for up to (DB_MAXWKA-1) relationships. The <expN> is the position of the corresponding relation in the list of previously defined relations. If there is no relation defined that matches the <expN> on the active table the DBRSELECT() function will return 0.

Example

```
use shows in 2 index people
use patrons in 1
set relation to patron_code into shows
? dbrelation(1)
```

patron_code

```
? dbrselect(1)
```

2

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DBUSED()

Class

Databases

Purpose

Function to check whether a database is open.

Syntax

DBUSED(<expC>)

See Also

CLOSE DATABASES, DISPLAY STATUS, LIST STATUS, OPEN DATABASE, USE, ADATABASES(), ADIR(), ALIAS(), DATABASE(), DBF(), GETENV(), USED(), SET FILECASE, SET SQL

Description

The DBUSED() function is used to check whether the database whose name is specified in <expC> is open. If the database is open, DBUSED() returns True (.T.), if not it returns False (.F.).

NOTE: The DBUSED() function operates on databases, not tables.

Databases in Recital are implemented as directories containing files that correspond to the tables and associated files in the database. Operating System file protection can be applied individually to the files for added security. The directories are sub-directories of the Recital data directory. The environment variable / symbol DB_DATADIR points to the current Recital data directory and can be queried using the GETENV() function. Files from other directories can be added to the database using the ADD TABLE command or via the database catalog and SET AUTOCATALOG functionality.

Databases can be opened using the SQL USE command, with SQL set to MYSQL, or using the SQL OPEN DATABASE command.

Example

```
VFP/SQL> OPEN DATABASE hr EXCLUSIVE
```

```
VFP/SQL> ? dbused("hr")
```

```
.T.
```

```
VFP/SQL> CLOSE DATABASES
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DBXDESCEND()

Class

Expressions and Type Conversion

Purpose

Function used in the creation and searching of descending order tag indexes

Syntax

DBXDESCEND(<exp>)

See Also

DESCENDING(), FIND, INDEX ON, SEEK

Description

The DBXDESCEND() function is used in the creation and searching of descending order tag indexes. It can be used on CHARACTER, NUMERIC, LOGICAL and DATE fields. For character fields, the function operates by subtracting the ASCII code for each character from 255. For numeric fields, the sign is reversed. Logical Falses become .T. (true) and logical Trues become .F. (false). The DBXDESCEND() function can be used to search in DATE indexes created with the DESCENDING keyword, but use outside the context of an index search will result in an Invalid date being returned.

For index tags created using the DESCENDING keyword, the DBXDESCEND() function must be used in SEEK or FIND index searches.

Example

```
index on name tag descendname descending
seek dbxdescend("Smith")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DEFAULT()

Class

Disk and File Utilities

Purpose

Function to return current device and directory

Syntax

DEFAULT([<expN>])

See Also

DIR(), PATH(), SYS(), SET DEFAULT, SET PATH

Description

The DEFAULT() function returns the name of the current directory. The optional <expN> is used to the return the current device specification as well as the directory on OpenVMS. The <expN> must return a 1. The DEFAULT() function always returns a character string without changing the case.

Example

? default()

[USERS.RECITAL]

? default(1)

DIA1:[USERS.RECITAL]

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DELETED()

Class

Fields and Records

Purpose

Function to return the deletion mark status

Syntax

DELETED([<workarea | alias>])

See Also

SET DELETED, DELETE, RECALL, PACK, SET FILTER

Description

The DELETED() function returns .T. if the current record in the currently selected workarea is marked for deletion. If the optional <workarea | alias> is specified, then the function operate in the specified workarea.

Example

use patrons

delete

? deleted()

.T.

recall

? deleted()

.F.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DELETEMAIL()

Class

Mail

Purpose

Function to delete the specified mail message

Syntax

DELETEMAIL(<expN>)

See Also

CLOSEMAIL(), COUNTMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The DELETEMAIL() function will delete the mail message specified by number. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() (POP3 only) or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<expN>	Yes	None	Numeric value to specify the mail message number to be deleted.

Example

```
deletemail(1)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DESCEND()

Class

Expressions and Type Conversion

Purpose

Function to return a character expression in inverse character sequence

Syntax

DESCEND(<expC>)

See Also

FIND, INDEX ON, SEEK, SORT, SEEK()

Description

The DESCEND() function returns the character expression <expC> in inverse character sequence. This function operates by subtracting the code for each character from 255. It is most useful for creating indexes in descending order of key. To search in such indexes, the DESCEND() function must be used.

Example

index on descend(name) to descendname
seek descend("Smith")

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DESCENDING()

Class

Indexing

Purpose

Function to determine whether an index tag was created with the DESCENDING keyword

Syntax

DESCENDING([[.dbx filename> | <expC>], <expN> [, <alias>]])

See Also

DBXDESCEND(), FOR(), MDX(), ORDER(), TAG(), TAGCOUNT(), TAGNO(), UNIQUE(), FIND, INDEX ON, SEEK, SET INDEX TO

Description

The DESCENDING() function returns a logical true (.T.) if the index tag specified by the tag number <expN> was created with the DESCENDING keyword. If the index tag was not created with the DESCENDING keyword, the DESCENDING() function returns a logical false (.F.). The DESCENDING keyword creates an index tag in descending (Z-A, 10-1) rather than ascending (A-Z, 1-10) order. The optional <.dbx filename> causes the DESCENDING() function to check the specified tag in that multiple index file. You may specify a character expression that returns the name of a valid multiple index file. If no multiple index filename is specified, the DESCENDING() function checks the currently open multiple index file. If no multiple index file is open, the DESCENDING() function returns .F.. The DESCENDING() function returns a logical false (.F.) if the index is a single index (.ndx) file. You may specify an alias name to check for descending index tags in other workareas. The [<alias>] may be a workarea number from 1 up to DB_MAXWKA, an alias name which has been created with the USE command, or A through Z, excluding M.

For index tags created using the DESCENDING keyword, the DBXDESCEND() function must be used in SEEK or FIND index searches.

Example

```
set view to accounting
?descending(3)
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DIFFERENCE()

Class

String Data

Purpose

Function to return an integer signifying the phonetic difference between two character strings

Syntax

DIFFERENCE(<expC1>,<expC2>)

See Also

SOUNDEX()

Description

The DIFFERENCE() function returns an integer signifying the phonetic difference between two character strings. The DIFFERENCE() converts the specified character expressions, <expC1> and <expC2> into SOUNDEX codes and calculates the phonetic difference between the two. The DIFFERENCE() function returns an integer between 0 and 4. A return value of 4 represents a close match, and 0 is returned when the two character expression have no letters in common. The DIFFERENCE() function returns a 1 when the two expressions have one letter in common.

Example

```
proc sound_like
dialog get sound;
    label "SPELL SEARCH";
    help "Enter a name"
menu browse name;
    for difference(name,sound) >= 2;
        label "Phonetically Matching Names";
    quit clear
if not empty(menuitem())
    seek menuitem()
    edit
endif
return

use customer
do sound_like
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DIR()

Class

Disk and File Utilities

Purpose

Function to perform directory search

Syntax

DIR(<skeleton>,<expL1>,<expL2>)

See Also

FILE(), MENU FILES, ADIR()

Description

The DIR() function scans the current directory returning the names of files which match the pattern <skeleton>. If <expL1> is .T., then the scan starts at the first file in the directory otherwise it continues from the last file returned. If <expL2> is .T., then characters following ';' in the file name will be discarded, as will the ';'. The <expL2> or "trim" parameter is only of use in OpenVMS where the file names returned can have version numbers.

Example

```
// Delete existing text document
procedure del_files
parameter m_confirm
set message to "Operation in progress."
file_name=dir("*.txt",.T.,.T.)
do while not empty(file_name)
    if m_confirm
        dialog message "Erase &file_name.? (Y/N)"
    endif
    if lastkey()=asc('Y') or not m_confirm
        set message to "Deleting &file_name."
        erase &file_name
    endif
    file_name=dir("*.txt",.F.,.T.)
enddo
return

del_files(.T.)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DISKSPACE()

Class

Disk and File Utilities

Purpose

Function to return available disk space

Syntax

DISKSPACE()

See Also

HEADER(), RECCOUNT(), RECSIZE(), SET DEFAULT

Description

The DISKSPACE() function returns the available space on the default disk drive. The available space is represented in bytes as a numeric value. You can set the default drive (OpenVMS) and directory with the SET DEFAULT TO command.

Example

```
? diskspace()
```

```
60754290
```

```
// Example of a routine within a program
```

```
use patrons
```

```
size = header()+reclsize()*reccount()
```

```
if size > diskspace()
```

```
    dialog box "Insufficient disk space."
```

```
    return
```

```
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DMY()

Class

Date and Time Data

Purpose

Function to return a date or datetime as a character string

Syntax

DMY(<expD> | <expT>)

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DMY() function returns the specified date expression <expD> or date part of the datetime expression <expT> as a character string in the format: day, month name and year. If CENTURY is OFF, then a 2-digit year is returned.

Example

```
? dmy({04/04/2005})
```

4 April 2005

```
? dmy({04/04/2005 06:12:45 PM})
```

4 April 2005

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DODEFAULT()

Class

Objects

Purpose

Function to call the parent class method from within a sub-class

Syntax

DODEFAULT()

See Also

DEFINE CLASS...ENDDEFINE, WITH...ENDWITH, ADDPROPERTY(), CREATEOBJECT(), NEWOBJECT(), REMOVEPROPERTY()

Description

The Visual FoxPro compatible DODEFAULT() function is used to call the parent class method from within a sub-class. It can only be used within a class method. It calls the parent class method of the same name as the calling method. It allows a sub-class to perform the default parent class behavior along with additional behavior specific to that particular sub-class.

Example

```
class Box
procedure Draw
    messagebox("This is the parent Draw Method")
endproc && Draw
endclass
```

```
class Dialog1 of Box
procedure Draw
    messagebox("This is the object Draw Method")
    // call the Draw method of the Box parent class
    dodefault()
endproc && Draw
endclass
```

```
oDIALOGOK = createobject("Dialog1")
oDIALOGOK.Draw()
release oDIALOGOK
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DOLEVEL()

Class

Applications

Purpose

Function to return the 'do' level of the currently executing program or procedure

Syntax

DOLEVEL()

See Also

DEBUG, DO, MESSAGE, RESUME, SUSPEND, MESSAGE(), PATH(), PROCLIBS(), PROCLINE(), PROCNAME(), PROGRAM(), SYS(), SET DEBUG, SET DOHISTORY, SET ECHO, SET HISTORY

Description

The DOLEVEL() function returns the 'do' level of the currently executing program or procedure. Issued at the command prompt, the DOLEVEL() function returns 0. Issued in a master or calling program, DOLEVEL() returns 1. Subsequent called programs or procedures are at a 'do' level based on the program nesting.

Example

```
> ? dolevel()
0
//master.prg
procedure subproc1
? dolevel()
?
return

? dolevel()
subproc1()
//end of master.prg
> do master
1
2
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DOSERROR()

Class

Error Handling and Debugging

Purpose

Function to return Operating System dependent error number

Syntax

DOSERROR()

See Also

ERROR(), MESSAGE(), ON ERROR, ERRNO()

Description

The DOSERROR() function returns the operating system dependent error number for the last error encountered. This is useful for revealing an operating system error which may have caused an error reported by the ERROR() function. The DOSERROR() function is synonymous with the ERRNO() function.

Example

```
if doserror() > 0
    err = doserror()
    set message to "System error: &err"
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DOW()

Class

Date and Time Data

Purpose

Function to return numeric day of the week from a specified date

Syntax

DOW(<expD> | <expT> [,<expN>])

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EPOCH(), GOMONTH(), HOUR(), HOURS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DOW() function returns a number representing the day of the week from the given date expression <expD> or datetime expression <expT>.

Return Value	Day of Week
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

The optional <expN> is used to specify an alternative first day of the week.

Example

```
set date american
```

```
? dow({02/29/2004})
```

```
1
```

```
? dow({02/29/2004},2)
```

```
7
```

```
store dow({02/29/2004}) to dayofweek
```

```
? dayofweek
```

```
1
```

```
dayofweek = dow({02/29/2004 11:35:27 A.M.})
```

```
? dayofweek
```

```
1
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DTOC()

Class

Expressions and Type Conversion

Purpose

Function to perform date to character conversion

Syntax

DTOC(<expD> | <expT> [,1])

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DTOC() function converts the date expression <expD> or date part of the datetime expression <expT> to a character string in the format of the current SET DATE, SET MARK and SET CENTURY settings. For example, the default settings, SET DATE AMERICAN, and SET CENTURY ON, will return a date in the format “MM/DD/YYYY”.

If the optional 1 is specified, the date will be returned in DTOS() format, suitable for index key purposes.

Example

```
? dtoc({02/29/2004})
02/29/2004
? dtoc({02/29/2004 10:34:21 A.M.})
02/29/2004
? dtoc({02/29/2004},1)
20040229
store dtoc({02/29/2004}) to m_date
? m_date
02/29/2004
? type("m_date")
C
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DTOM()

Class

Expressions and Type Conversion

Purpose

Function to convert a date to DD-Mmm-YYYY format

Syntax

DTOM(<expD>)

See Also

AMPM(), CDOW(), CTOD(), CMONTH(), DATE(), DAY(), DMY(), DOW(), DTOC(), DTOS(), DTOV(), MDY(), MONTH(), MTOD(), SET CENTURY, SET DATE, SET MARK, STOD(), TIME(), VTOD(), YEAR()

Description

The DTOM() function is used to convert date expressions to DD-Mmm-YYYY format. If <expD> is not a valid date expression, DTOM() will return an empty string. The DD-Mmm-YYYY format is the calendar date format in Recital Internet Developer and Recital Mirage.

Example

```
? dtom({ 10/10/2000 })
```

10-Oct-2000

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DTOR()

Class

Expressions and Type Conversion

Purpose

Function to convert degrees to radians

Syntax

DTOR(<expN>)

See Also

RTOD(), COS(), SIN(), TAN(), EXP(), LOG(), LOG10(), PI()

Description

The DTOR() function converts the number of degrees in the expression <expN> to radians.

Example

```
? dtor(90)  
1.57
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DTOS()

Class

Expressions and Type Conversion

Purpose

Function to perform date to string conversion

Syntax

DTOS(<expD> | <expT>)

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The DTOS() function converts the date expression <expD> or date part of the datetime expression <expT> to a character string in the format “YYYYMMDD”. This function is particularly useful when creating indexes with mixed data types. The SET DATE, SET MARK and SET CENTURY settings have no effect on this function.

Example

```
? dtos({02/29/2004})
```

```
20040229
```

```
? dtos({02/29/2004 11:18:54 PM})
```

```
20040229
```

```
store dtos({02/29/2004})to m_date
```

```
? m_date
```

```
20040229
```

```
? type("m_date")
```

```
C
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

DTOV()

Class

Expressions and Type Conversion

Purpose

Function to convert Recital dates to OpenVMS VAXdates

Syntax

DTOV(<expD>)

See Also

VTOD(), SET VAXTIME, DAY(), CDOW(), CTOD(), CMONTH(), DATE(), DOW(), DTOC(), MONTH(), TIME(), WEEK(), YEAR(), STOD(), DTOS(), DMY(), MDY()

Description

The DTOV () function converts Recital dates to VAXdates. A VAXdate data type is a character string with a length of 23. The first 11 characters are the date in the format “DD-MMM-YYYY”. The remaining characters are the optional time string of HH:MM:SS.00, which can be turned off with the SET VAXTIME command. Date arithmetic cannot be performed on VAXdates.

Parameters	Description
<expD>	The valid date to be converted to a VAXdate

Example

```
? dtov({02/02/2000}+7)
9-FEB-2000 00:00:00.00
```

Products

All OpenVMS

EDITFIELD()

Class

Screen Forms

Purpose

Function to return or alter the current EDITFIELD setting

Syntax

EDITFIELD([<expL>])

See Also

APPEND, CHANGE, CREATE SCREEN, EDIT, MODIFY SCREEN, FILETYPE(), FMT(), INDEXEXT(), READEXIT(), READINSERT(), SET CLIPPER, SET CLIPPER5, SET COMPATIBLE, SET EDITFIELD, SET FILECASE, SET FILETYPE, SET FORMAT, SET INDEXEXT, SET MEMOEXT, SET PCEDIT, SET PCEXACT, SET PCFILTER, SET PCGRAPHICS, SET PCKEYS, SET PCLOCKING, SET PCPICTURE, SET PCSAYS, SET PCUNIQUE, DB_FOXPLUSBUGS, DB_FOXPROKEYS, DB_SAMBA

Description

The EDITFIELD() function returns .T. if EDITFIELD is ON and .F. if EDITFIELD is OFF. If EDITFIELD is ON, when the user partially overwrites a field the entire field value is saved when the [RETURN] key is pressed, not just the user's input. When EDITFIELD is OFF, the default, the contents of the field up to the current cursor position are saved.

The EDITFIELD setting can be altered using the SET EDITFIELD ON | OFF command, or by using the optional <expL> in the EDITFIELD() function. The <expL> logical expression will SET EDITFIELD ON if .T. and SET EDITFIELD OFF if .F..

Example

```
lEditfield = editfield()
set editfield on
edit
editfield(lEditfield)
```

Products

Recital Terminal Developer

ELAPTIME()

Class

Date and Time Data

Purpose

Function to return time difference between two time strings

Syntax

ELAPTIME(<timestring1>,<timestring2>)

See Also

SET CLOCK, SET CLOCKRATE, TIME(), VALIDTIME(), TSTRING(), DAYS(), SECS(), HOURS(), MINUTES(), SECONDS(), AMPM()

Description

The ELAPTIME() function returns a time string in the form “HH:MM:SS” which is the difference between <timestring2> and <timestring1>. This function can be very useful when used in the Application Data Dictionary to define a total work time field.

Example

```
? elaptime("10:10:10","11:11:11")
```

01:01:01

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EMPTY()

Class

Expressions and Type Conversion

Purpose

Function to check for empty value

Syntax

EMPTY(<expC> | <expD> | <expT> | <expL> | <memofield> | <expN>)

See Also

CTOD(), CTOT(), IIF(), SPACE(), TYPE(), SET CENTURY, SET DATE, SET MARK, SET SECONDS

Description

The EMPTY() function returns .T. if the specified expression is 'empty'.

Data Type	Empty
Character	Space(0), "", [], ', no text entered
Date	CTOD(""), {}, {00/00/0000}, { / / } and other SET CENTURY, SET MARK and SET DATE variations
Datetime	CTOT(""), { / / : : AM} and other SET CENTURY, SET MARK, SET SECONDS and SET DATE variations
Logical	.F.
Memo	No data entered
Numeric	0

Example

```
if empty(name + address)
    dialog box "Name and address not specified."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EOF()

Class

Fields and Records

Purpose

Function to check if the record pointer is at the end of file

Syntax

EOF([<workarea | alias>])

See Also

BOF(), FOUND(), SET DELETED, DELETE, RECALL, PACK

Description

The EOF() function returns .T. when the last logical record of the table is passed. If the optional <workarea | alias> is specified, then the function will operate in the required location. The EOF() function is set to .T. if SEEK, FIND, LOCATE or CONTINUE operations are not successful.

Example

```
use catalogue
do while not eof()
    m_file=lower(trim(file) + "." + filetype)
    if not file("&m_file")
        delete
    endif
    skip
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EPOCH()

Class

Date and Time Data

Purpose

Function to return the current EPOCH setting

Syntax

EPOCH()

See Also

SET EPOCH, SET CENTURY, SET DATE, SET MARK

Description

The EPOCH() function returns the current SET EPOCH setting as a numeric 4 digit year. The SET EPOCH TO <expN> command allows for the specification of the starting year of a one hundred-year period. The one hundred-year period can span two consecutive centuries. Two-digit year dates (SET CENTURY OFF) can be used and still assigned the correct century provided that they fall within this one-hundred year period. For example, SET EPOCH TO 1995 would specify a one hundred-year period from 1995 to 2094. Any two-digit years less than 95 would be 21st century, any two-digit years of 95 or over would be 20th century.

10/10/97 = October 10th 1997

10/10/27 = October 10th 2027

By default, EPOCH is set to 1900.

Example

```
? epoch()
```

```
1900
```

```
set epoch to 1995
```

```
? epoch()
```

```
1995
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ERRNO()

Class

Error Handling and Debugging

Purpose

Function to return Operating System dependent error number

Syntax

ERRNO()

See Also

ERROR(), MESSAGE(), ON ERROR

Description

The ERRNO() function returns the Operating System dependent error number for the last error encountered. This is useful for revealing an Operating System error which may have caused an error reported by the ERROR() function.

Example

```
if errno()>0
    err = errno()
    set message to "System error: &err"
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ERROR()

Class

Error Handling and Debugging

Purpose

Function to determine the last error encountered

Syntax

ERROR()

See Also

ERRNO(), MESSAGE(), ON ERROR

Description

The ERROR() function returns a number representing the last error encountered. A zero return value indicates that no error has been encountered. The MESSAGE() function can be used to get further information.

Example

```
? error()  
0
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ERRORLEVEL()

Class

Error Handling and Debugging

Purpose

Return or set an exit value

Syntax

ERRORLEVEL([<expN>])

See Also

QUIT

Description

The ERRORLEVEL() function is used to return or set the exit value of a program. The <expN> is the error level setting that will be returned to the operating system when QUIT is executed. When no level is specified, ERRORLEVEL() returns the current setting. This function is useful for setting up serial program execution.

Example

```
oldlevel = errorlevel(10)
quit
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ETOS()

Class

Expressions and Type Conversion

Purpose

Function to evaluate an expression and return the result as a string

Syntax

ETOS(<expr> [, <expN>])

See Also

DTOC(), DTOS(), LTOS(), MTOS(), STR()

Description

The ETOS() function evaluates the expression in <expr> and returns the result as a string. The <expr> can be any valid string, date, datetime, logical or numeric expression.

Where <expr> is a date or datetime expression, the string returned will conform to the current SET DATE, SET SEPARATOR, SET SECONDS and SET CENTURY settings, in the same format as DTOC() or TTOC().

The optional <expN> is used to specify the length of the string returned. If the length is shorter than that required by <expr>, strings, dates and datetimes are truncated to the right and numerics truncated to the left.

Example

? etos({03/29/2003})

03/29/2003

? etos({03/29/2003},5)

03/29

? etos(10 * 10)

100

? etos(0.45,3)

.45

? etos("Hello" + " " + "World")

Hello World

? etos("Hello" + " " + "World",5)

Hello

? etos(.F.)

F

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EXCLUSIVE()

Class

Environment

Purpose

Function to return access status for a table

Syntax

EXCLUSIVE([<workarea | alias>])

See Also

SET EXCLUSIVE, USE EXCLUSIVE, MODIFY STRUCTURE, PACK, ZAP, REINDEX, ON ERROR

Description

The EXCLUSIVE() function returns .F. if the table open in the currently selected workarea is available for shared access, and .T. if it is open for private use. If the optional <workarea | alias> is specified, the function will operate in the specified location. The MODIFY STRUCTURE, PACK, REINDEX, INDEX ON <exp> TAG and ZAP are the only commands that require exclusive use of the table. For reasons of safety, the INDEX ON ...TO <.ndx> command should also be used only with exclusive use of the table.

Example

```
if not exclusive()  
    dialog box "Command not available."  
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EXEC()

Class

Applications

Purpose

Function to execute a list of commands

Syntax

EXEC(<expC1> [,<expC2>...])

See Also

ALIAS, KEYWORD, DO, RUN, RUN(), SPAWN, EXECSCRIPT()

Description

The EXEC() function executes a list of commands. Each command is specified by <expC1>, <expC2>, and so on. This function is useful for consolidating command lines in UDCs (User Defined Commands) which have been defined with the ALIAS command. The EXEC() function returns a logical true (.T.) if all the specified commands execute without error, or a logical false (.F.) if an error occurs.

Example

```
alias lf;  
"iif(run('ls -l %1 > run.tmp') = 0,;  
    exec('textedit([run.tmp],0,0,20,79,.f.,[FILE LISTING])','erase run.tmp'), .F.)"
```

If

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EXECSCRIPT()

Class

Applications

Purpose

Function to run multiple lines of code

Syntax

EXECSCRIPT(<expC1> [, <expC2> ...])

See Also

COMPILE, ON ERROR, EXEC()

Description

The EXECSCRIPT() function runs multiple lines of code. The lines of code are contained in <expC>, which can be a text constant, a character variable, or a character or memo field. The EXECSCRIPT() function, unlike EXEC() can handle blocks of code, such as FOR...NEXT or DO WHILE...ENDDO loops. Individual lines of code must be separated by a CHR(13) carriage return character.

The EXECSCRIPT() function returns a logical true (.T.) unless its contents specifically return false (.F.).

Example

// Character field

set sql on

set sql to VFP

create table scripts (script char(200))

use scripts

append blank

replace script with "for i=1 to 10" + CHR(13) + "?i" + CHR(13) + "endfor" + CHR(13)

execscript(script)

// Text constant

execscript("for i=1 to 10" + CHR(13) + "?i" + CHR(13) + "endfor" + CHR(13))

// Memory variable

m_script = "do while .T." + CHR(13) + "inkey(0)" + CHR(13) + "if lastkey() = ctrl('G')" + CHR(13) ;
+ "exit" + CHR(13) + "endif" + CHR(13) + "enddo"

execscript(m_script)

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EXP()

Class

Numeric Data

Purpose

Function to return exponential value

Syntax

EXP(<expN>)

See Also

SET DECIMALS, SET FIXED, LOG(), LOG10(), SIN(), COS(), TAN(), RTOD(), DTOR(), PI()

Description

The EXP() function returns the value of e to the power <expN>.

Example

set decimals to 3

? exp(1.000)

2.718

eval = exp(1.000)

? eval

2.718

? type("eval")

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

EXPRCHECK()

Class

Expressions and Type Conversion

Purpose

Function to check expression for errors

Syntax

EXPRCHECK(<expC>)

See Also

ERROR(), MESSAGE()

Description

The function EXPRCHECK() evaluates the Recital/4GL expression specified in the string <expC>, and returns the appropriate Recital error number, or 0 if the expression evaluated correctly.

Example

```
function CheckExpr
parameter querystring
if exprcheck(querystring) > 0
    dialog box "Error in Query String: " + message()
    m_ret = .F.
else
    m_ret = .T.
endif
return m_ret
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FCLOSE()

Class

ASCII File Access

Purpose

Function to close an open text file

Syntax

FCLOSE(<expN>)

See Also

FCREATE(), FREADSTR(), FOPEN(), FWRITE(), FERROR(), TEXTEDIT(), MEMOEDIT(), MEMOREAD(), MEMOWRITE()

Description

The FCLOSE() function will close an open text file, writing the associated buffers to disk. The <expN> is the file pointer obtained from FCREATE() or FOPEN(). The file pointer must be assigned when the file is first opened using the FOPEN() or FCREATE() functions. If an error occurs during the operation -1 is returned by the FERROR() function.

Example

```
use accounts
scatter to flist
m_total=len(flist)
fp=fopen("new.txt")
if ferror()=-1
    dialog box "The file could not be created."
else
    for n=1 to m_total
        fwrite(m->fp,flist[n],80)
    next
endif
fclose(fp)
if ferror()=-1
    dialog box "The file could not be closed."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FCOUNT()

Class

Fields and Records

Purpose

Function to count fields contained in a table

Syntax

FCOUNT([<workarea | alias>])

See Also

NFCOUNT(), DBF(), NDX(), FMT(), FILTER(), FIELD(), DISPLAY STRUCTURE, AFIELDS(), SET FIELDS

Description

The FCOUNT() function returns the number of fields in the active table. If the optional <workarea | alias> is specified, then the function will operate in the required location. The command SET FIELDS will affect the value returned by the FCOUNT() function.

Example

```
use accounts
do while i <= fcount()
    ? field(i)
    ++ i
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FCREATE()

Class

ASCII File Access

Purpose

Function to create an ASCII text file

Syntax

FCREATE(<expC>)

See Also

FOPEN(), FREADSTR(), FWRITE() FERROR(), FCLOSE() TEXTEDIT(), MEMOREAD(), MEMOWRITE()

Description

The FCREATE() function will create a new ASCII text file or truncate an existing file to zero length. The <expC> is the name of the file to be created. When FCREATE() successfully creates a new file it leaves the file open and returns the numeric file pointer. If an error occurs, FCREATE() returns a file pointer with a value of -1. Since the file pointer is required in order to identify an open file to other file functions, always assign the return value to a memory variable. If an error occurs during the operation, -1 is returned by the FERROR() function.

Example

```
use acc_log
scatter to flist
m_total=len(flist)
fp=fopen("new.txt")
if ferror()=-1
    dialog box "The file could not be created."
else
    for n=1 to m_total
        fwrite(fp,flist[n],80)
    next
endif
fclose(fp)
if ferror()=-1
    dialog box "The file could not be closed."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FDATE()

Class

Disk and File Utilities

Purpose

Function to return the last modification date of a file

Syntax

FDATE(<expC>)

See Also

DISKSPACE(), FILE(), SET CENTURY, SET DATE, SET MARK, SET PATH, LUPDATE(), FTIME()

Description

The FDATE() function returns, from the operating system, the date that the specified file <expC> was last modified. The search will be carried out in the current directory and any directories specified with the SET PATH TO command. The character expression <expC> may include the full pathname, but no 'wild card' characters. The FDATE() function returns the date in the format as specified by the SET CENTURY, SET DATE, and SET MARK commands. If the file specified by <expC> is open, the FDATE() function returns the last modification date since the last time the file was closed.

Example

```
? fdate("main.prg")
```

05/12/1999

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FEOF()

Class

ASCII File Access

Purpose

Function to check if the file pointer is at the end of an ASCII file

Syntax

FEOF(<expN>)

See Also

FCLOSE(), FCREATE(), FERROR(), FFLUSH(), FGETS(), FOPEN(), FPUTS(), FREAD(), FSEEK(), FWRITE()

Description

The FEOF() returns a logical true (.T.) if the file pointer is at the end of the specified ASCII file. The ASCII file is specified by its file handle <expN>. The file handle should be assigned to a memory variable when the file is opened with the FCREATE() or FOPEN() functions.

Example

```
use accounts
fp=fopen("existing.file")
if fp > 0
    do while not feof(fp)
        ? fgets(fp)
    enddo
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FERROR()

Class

ASCII File Access

Purpose

Function to return the status of the last text-file operation

Syntax

FERROR()

See Also

FOPEN(), FCREATE(), FREADSTR() FWRITE(), FCLOSE() TEXTEDIT(), MEMOREAD(), MEMOWRITE()

Description

The FERROR() function returns an integer representing the status of the last text file operation. It returns – 1 for failure, 0 for success. This function should always be used in conjunction with the other text file functions to check for errors.

Example

```
use accounts
fp=fopen("existing.file")
if ferror()=-1
    dialog box "The file could not be opened."
else
    string = freadstr(fp,80)
    do while not empty(string)
        ? string
        string = freadstr(fp,80)
    enddo
    ?
    fclose(fp)
    if ferror()=-1
        dialog box "The file could not be closed."
    endif
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FFLUSH()

Class

ASCII File Access

Purpose

Function to flush internal buffer

Syntax

FFLUSH(<expN>)

See Also

FCLOSE(), FOPEN(), FWRITE(), FLUSH, SET TBUFSIZE

Description

The FFLUSH() function works in conjunction with the FOPEN() and FWRITE() functions to flush the internal buffer. The <expN> is the file handle returned by the FOPEN() function. The FFLUSH() function is used to bypass the I/O buffer and commit open text files to the current device. The FCLOSE() function also flushes associated buffers as it closes the specified file.

Example

```
fwrite( fp,"Hello World")  
fflush(fp)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FGETS()

Class

ASCII File Access

Purpose

Function to read and return a line of text from an ASCII file

Syntax

FGETS(<expN1>[,<expN2>][,<expC>])

See Also

FCLOSE(), FCREATE(), FERROR(), FEOF(), FFLUSH(), FOPEN(), FPUTS(), FREAD(), FSEEK(), FWRITE()

Description

The FGETS() function reads and returns a line of text from the file specified by <expN>, an ASCII file handle. The file handle is returned from the FCREATE() or FOPEN() functions. The FGETS() function reads a line of text from the current position of the record pointer. You may optionally specify the number of bytes to read from that point with the numeric expression <expN2>. The number of bytes may be from 0 to 254. If a number is not specified, the FGETS() function will read 254 bytes or to the end line marker. The optional character expression <expC> defines an end of line terminator to determine where the FGETS() function will stop reading. The character expression <expC> must evaluate to one or two characters.

Example

```
fp = fopen("names.txt")
count = 0
do while not feof()
    if fgets(fp,20) = "Smith"
        ++count
    endif
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FIELD()

Class

Fields and Records

Purpose

Function to return field name

Syntax

FIELD(<expN1>[,<workarea | alias>[,<expN3>]])

See Also

AFIELD(), DBF(), FIELDNAME(), FILTER(), FMT(), INDEXKEY(), NDX()

Description

The FIELD() function is synonymous with the FIELDNAME() function. It returns the name of the field <expN1> in the currently selected table. The optional <workarea | alias> parameter can be used to operate on the table in the specified workarea number, or with the specified table alias. If the optional <expN3> is specified and evaluates to 1, the field data type initial (e.g. 'C' for character), the field width and the field value will also be returned along with the fieldname in a comma separated string.

If <expN1> exceeds the number of fields in the table, or there is no currently selected table, then FIELD() returns a null string. Field offsets start at 1. The FIELD() function always returns a character string in upper case.

Example

```
use payroll
declare fname[fcount()]
for n=1 to fcount()
    fname[n] = field(n)
    ?fname[n]
next
?

use demo
go top
? field(1,1,1)
ACCOUNT_NO,C,5,00046
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FIELDNAME()

Class

Fields and Records

Purpose

Function to return field name

Syntax

FIELDNAME(<expN1>[,<workarea | alias>[,<expN3>]])

See Also

AFIELD(), DBF(), FIELD(), FILTER(), FMT(), INDEXKEY(), NDX()

Description

The FIELDNAME() function is synonymous with the FIELD() function. It returns the name of the field <expN> in the table currently selected. The optional <workarea | alias> parameter can be used to operate on the table in the specified workarea number, or with the specified table alias. If the optional <expN3> is specified and evaluates to 1, the field data type initial (e.g. 'C' for character), the field width and the field value will also be returned along with the fieldname in a comma separated string.

If <expN1> exceeds the number of fields in the table, or there is no currently selected table, then FIELDNAME() returns a null string. Field offsets start at 1. The FIELDNAME() function always returns a character string in upper case.

Example

```
use payroll
declare fname[fcount()]
for n=1 to fcount()
    fname[n] = fieldname(n)
    ?fname[n]
next
?
```

```
use demo
go top
? fieldname(1,1,1)
ACCOUNT_NO,C,5,00046
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FILE()

Class

Disk and File Utilities

Purpose

Function to check whether file exists

Syntax

FILE(<expC>)

See Also

DIR(), SET PATH, MENU FILES, SET FILECASE

Description

The FILE() function returns .T. if the file exists and .F. otherwise. FILE() checks for the existence of the file in the current directory, but not in the PATH (specified with the SET PATH TO command). FILE() does not check the access permission of the specified file for writing, it only checks that the file can be opened for reading. If no file extension is specified in the filename <expC>, then the file extension “.dbf” is concatenated to the end of the filename. This means that FILE() cannot be used to check for the existence of files without extensions.

Example

```
if file("names.ndx")
    erase names.ndx
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FILECOUNT()

Class

Disk and File Utilities

Purpose

Function to count files matching pattern

Syntax

FILECOUNT(<skeleton>)

See Also

MENU FILES, ADIR(), IFILECOUNT(), FILE()

Description

The FILECOUNT() function returns the number of files which match the specified pattern <skeleton>. The following 'wild card' characters can be used:

Character	Description
?	Matches any one character.
%	Matches any one character.
*	Matches zero or more characters.

Example

```
nfiles = filecount("*.dbf")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FILEINFO()

Class

Disk and File Utilities

Purpose

Function to return information about a file

Syntax

FILEINFO(<expC>)

See Also

GETENV(), GETLOG(), DBF(), NDX(), FMT(), SET FULLPATH

Description

The FILEINFO() function returns a comma-separated string containing information about the file specified in <expC>.

The following information is returned:

- Day of the month of file last update date
- Month of file last update date
- Year of file last update date
- Hours of file last update time
- Minutes of file last update time
- Seconds of file last update time
- File attribute string
- File size in bytes

Example

```
? fileinfo("db.exe")
```

30,9,2003,20,12,23,A,4731284

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FILETOSTR()

Class

String Data

Purpose

Function to read a file into a text string

Syntax

FILETOSTR(<expC>)

See Also

AT(), ATNEXT(), FCLOSE(), FCREATE(), FERROR(), FOPEN(), FREADSTR(), FWRITE(),
MEMOREAD(), MEMOWRITE(), SUBSTR(), STUFF(), STR(), STREXTRACT(), STRTOFILE(),
STRTRAN(), STRZERO(), TEXTEDIT(), TRIM()

Description

The FILETOSTR() function reads the specified file and returns its contents as text.

Parameter	Description
<expC>	Name of the file to be read. This should include the path if not in the current directory.

Example

```
myString = filetostr("myfile.txt")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FILETYPE()

Class

Xbase Compatibility

Purpose

Function to return the filetype of the active table or the table in a specified workarea

Syntax

FILETYPE([<expN>])

See Also

COPY, CREATE, INDEX, USE, INDEXEXT(), READEXIT(), READINSERT(), SET CLIPPER, SET CLIPPER5, SET COMPATIBLE, SET EDITFIELD, SET FILECASE, SET FILETYPE, SET INDEXEXT, SET MEMOEXT, SET PCEDIT, SET PCEXACT, SET PCFILTER, SET PCGRAPHICS, SET PCKEYS, SET PCLOCKING, SET PCPICTURE, SET PCSAYS, SET PCUNIQUE, DB_FOXPLUSBUGS, DB_FOXPROKEYS, DB_SAMBA

Description

The FILETYPE() function returns the filetype of the active table. The optional <expN> can be used to specify the workarea of an open table for which the filetype is returned.

Filetype
Recital
Clipper 87
dBASE III+
dBASE IV
FoxPlus
FoxPro
Visual FoxPro

If there is no active table, or no open table in the workarea specified by <expN>, the FILETYPE() function returns an empty string.

Example

use demo

? filetype()

Recital

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FILTER()

Class

Table Basics

Purpose

Function to return filter expression

Syntax

`FILTER([<workarea | alias>])`

See Also

SET FILTER, SET DELETED, SET VIEW, DBFILTER()

Description

The FILTER() function returns the current filter condition as a string. If no filter is set, the FILTER() function returns a null string. If the optional <workarea | alias> is specified, then the function will return the active filter condition for the specified <workarea | alias>. Filters are set for a workarea using the SET FILTER command. The FILTER() has no effect on the SET FILTER command.

Example

```
use patrons
```

```
set filter to event="BALLET"
```

```
? "["+filter()+"]"
```

```
[event="BALLET"]
```

```
? type("filter()")
```

```
C
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FINDCURSOR()

Class

Data Connectivity

Purpose

Returns the workarea number corresponding to the specified gateway cursor

Syntax

FINDCURSOR(<expC>)

See Also

CLOSE, DECLARE CURSOR, CURSORNAME(), DROP CURSOR, FETCH, OPEN CURSOR, GATEWAY()

Description

The FINDCURSOR() function returns a workarea number corresponding to the open gateway cursor specified by <expC>.

If the cursor name specified is not open, or no gateway is active, then a value of -1 is returned.

Example

```
select 3
set gateway to "ora@sales:scott/tiger"
exec sql
    declare employees cursor for
    select empno, ename, job
    from emp
    order by deptno;
exec sql
    open employees;
```

```
? findcursor(cursorname())
```

```
3
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FIXED()

Class

Numeric Data

Purpose

Function to return a numeric value

Syntax

FIXED(<expN>)

See Also

FLOAT(), BIN2I(), BIN2L(), BIN2W(), I2BIN(), L2BIN()

Description

The FIXED() function returns the <expN> as a numeric value. This function has been added for compatibility purposes.

Example

```
? fixed(2.5)  
2.5
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FKLABEL()

Class

Screen Forms

Purpose

Function to return function key label

Syntax

FKLABEL(<expN>)

See Also

FKMAX(), SET FUNCTION, SET KEY

Description

The FKLABEL() function returns the label associated with the function key <expN> as a character string. If <expN> is out of the range of the available function keys, then a null string is returned.

Example

```
? fklable(2)
```

F2

Products

Recital Terminal Developer

FKMAX()

Class

Screen Forms

Purpose

Function to return maximum available function keys

Syntax

FKMAX()

See Also

FKLABEL(), SET FUNCTION, SET KEY

Description

The FKMAX() function returns the number of function which are available for use with the SET FUNCTION command.

Example

```
? fkmax()
```

18

Products

Recital Mirage Server, Recital Terminal Developer

FLDCOUNT()

Class

Fields and Records

Purpose

Function to count fields contained in a table

Syntax

FLDCOUNT([<workarea | alias>])

See Also

NFCOUNT(), DBF(), NDX(), FMT(), FILTER(), FIELD(), DISPLAY STRUCTURE, AFIELDS(), SET FIELDS

Description

The FLDCOUNT() function returns the number of fields in the active table. If the optional <workarea | alias> is specified, then the function will operate in the required location. The command SET FIELDS will affect the value returned by the FLDCOUNT() function.

Example

```
use accounts
do while i <= fldcount()
    ? field(i)
    ++ i
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FLDLIST()

Class

Fields and Records

Purpose

Function to return a comma separated list of field names

Syntax

FLDLIST([<workarea | alias>])

See Also

ADESC(), AFIELDS(), DBF(), FIELD(), FIELDNAME(), FILTER(), FLDCOUNT(), FMT(), INDEXKEY(), NDX()

Description

The FLDLIST() function returns a string containing a comma separated list of field names. The optional <workarea | alias> parameter can be used to operate on the table in the specified workarea number, or with the specified table alias. If the optional <workarea | alias> is not specified, FLDLIST() operates on the currently selected table. If there is no table selected or open with the specified alias name or in the specified workarea, FLDLIST() returns an error.

Example

```
proc ExcelReport
  select * from example order by state save as datafile&(getpid())
  use datafile&(getpid()).dbf in 0
  copy to datafile&(getpid()).txt type delimited
  fieldList = fldlist()
  use
  select example
  showDocument("DemoReportData.xls?command=create;row=3;col=1;" +
    "title=<Demo Excel Report>;titleColor=23;columnTitles=&fieldList;" +
    "columnTitlesColor=24;stripeColor=20;subtotal=8;" +
    "subtotalcolumns=10,11,12;" +
    "columnFormat=||||||$#,##0.00|$#,##0.00|$#,##0.00|;" +
    "subtotalBackColor=24;subtotalForeColor=55;" +
    "subtotalTitle=Sub-total for state: ", "_blank", "datafile&(getpid()).txt")
endproc
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FLOAT()

Class

Numeric Data

Purpose

Function to return a numeric value

Syntax

FLOAT(<expN>)

See Also

FLOAT(), BIN2I(), BIN2L(), BIN2W(), I2BIN(), L2BIN()

Description

The FLOAT() function returns the <expN> as a numeric value. This function has been added for compatibility purposes.

Example

```
? float(2.5)
      2.5
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FLOCK()

Class

Manual Locking

Purpose

Function to lock file

Syntax

FLOCK([<workarea | alias>])

See Also

RLOCK(), LOCK(), ACCESS(), SET EXCLUSIVE, USE EXCLUSIVE, LOCKF, LOCKR

Description

The FLOCK() function attempts to logically lock the currently selected table. If successful, it returns .T. and the table is locked. If the table is already locked by another user, it returns .F. If the optional <workarea | alias> name is specified the function will operate in the required location. The FLOCK() function does NOT change the open status of the table to 'exclusive'.

The Recital/4GL performs automatic record locking, which makes the use of the FLOCK() function unnecessary. The commands that require EXCLUSIVE use of the table are MODIFY STRUCTURE, REINDEX, PACK, INDEX ON <exp> TAG, and ZAP. For reasons of safety, the INDEX command should also be used only with exclusive use of the table.

Example

```
do while not flock()
    @23,0 say "Database in use. please wait."
    sleep 2
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FLOOR()

Class

Numeric Data

Purpose

Function to return the largest integer that is less than or equal to a given number

Syntax

FLOOR(<expN>)

See Also

SIGN(), CEILING(), ISDIGIT()

Description

The FLOOR() function is used to return the largest integer that is less than or equal to <expN>.

Example

? floor(354.89)

354

? floor(354.19)

354

? floor(-354.89)

-355

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FMT()

Class

Screen Forms

Purpose

Function to return the name of an active format file

Syntax

FMT([<workarea | alias>])

See Also

CREATE SCREEN, ALIAS(), DBF(), FCOUNT(), FIELD(), FIELDNAME(), INDEXKEY(), NDX (), SET AUTOFORMAT, SET COMPATIBLE, SET FILECASE, SET FORMAT, SET FULLPATH

Description

The FMT() function returns the name of the currently active screen format file or a null string, if none is active. If the optional <workarea | alias> is specified, then the function will operate in the required workarea. The FMT() function always returns a character string in lower case including the file extension of the active format file.

If SET COMPATIBLE is set to FOXPRO or VFP the FMT() return value format differs in the following way: if SET FULLPATH is ON the full path to the format file is returned, if SET FILECASE is OFF then the return value is converted to upper case (Windows only).

Example

```
? fmt()
```

accounts.fmt

Products

Recital Mirage Server, Recital Terminal Developer

FONTMETRIC()

Class

Fonts

Purpose

Function to return font attribute information

Syntax

FONTMETRIC(<expN1> [,<expC1>,<expN2> [,<expC2>]])

See Also

AFONT(), GETFONT(), WFONT()

Description

The FONTMETRIC() function returns attribute information about the current or specified font.

Parameters	Description
<expN1>	The attribute about which information is returned. Please see table below for attribute numbers and descriptions.
<expC1>	The <expC1> can be used to optionally specify a particular font name.
<expN2>	The <expN> is used to specify the font size for the font name in <expC1>.
<expC2>	The <expC2> can be used to optionally specify a font style code for the font name in <expC1> and font size in <expN2>.

The FONTMETRIC() function is included for language compatibility only and the return values for each attribute are shown in the table below.

Code	Attribute	Return Value
1	Character height (pixels)	16
2	Character ascent (pixels)	13
3	Character descent (pixels)	3
4	Leading (pixels)	3
5	Extra leading (pixels)	0
6	Average character width (pixels)	7
7	Maximum character width (pixels)	14
8	Font weight	400
9	Italic	0
10	Underlined	0
11	Strikeout	0
12	First character	32
13	Last character	255
14	Default character	129
15	Word break character	32
16	Pitch and family	33
17	Character set	0
18	Overhang	0
19	Horizontal aspect	120
20	Vertical aspect	120

Font Styles:

Code	Style
B	Bold
I	Italic
N	Normal
O	Outline
Q	Opaque
S	Shadow
-	Strikeout
T	Transparent
U	Underline

Example

? fontmetric(20,“Monospaced”,12,“B”)

120

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FOPEN()

Class

ASCII File Access

Purpose

Function to open an existing ASCII text file

Syntax

FOPEN(<expC> [,expN>])

See Also

FCREATE(), FCLOSE(), FERROR(), FREADSTR(), FWRITE(), MEMOREAD(), MEMOWRITE(), TEXTEDIT(),

Description

The FOPEN() function opens an existing ASCII text file. It returns a numeric file pointer when the file is opened successfully, or a -1 if unsuccessful. The <expC> is the name of the ASCII file to open. Since the file pointer is required to identify an open file to other file functions, always assign the return value to a memory variable. The optional <expN> determines the file access mode:

<expN>	Access Mode
	If not specified, read-only
0	Read-only
1	Write only. Existing contents are deleted.
2	Append. Text may be added to the end of the existing contents.

If an error occurs, -1 is returned by the FERROR() function. The FCLOSE() function is used to close a file which has been opened with FOPEN().

Example

```
use accounts
fp=fopen("existing.file")
if ferror()=-1
    dialog box "The file could not be opened."
else
    string = freadstr(fp,80)
    do while not empty(string)
        ? string
        string = freadstr(fp,80)
    enddo
    ?
endif
fclose(fp)
if ferror()=-1
    dialog box "The file could not be closed."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FOR()

Class

Indexing

Purpose

Function to return the FOR condition that was used to create an index tag

Syntax

FOR([[<.dbx filename>], <expN> [, <alias>]])

See Also

DESCENDING(), INDEX, KEY(), MDX(), ORDER(), SET INDEX TO, TAG(), TAGCOUNT(), TAGNO(), UNIQUE()

Description

The FOR() function returns the FOR condition which was used to create an index tag. If the tag was not created with a FOR condition, or does not exist, the FOR() function returns a null string. With no parameters, the FOR() function operates in the current workarea on the master index.

The optional <.dbx filename> causes the FOR() function to check the specified .dbx file for the tag specified by the tag number <expN>. You may specify a character expression that returns the name of a valid multiple index file. If no multiple index filename is specified, the FOR() function checks the currently open multiple index file. If no multiple index file is open, the FOR() function returns a null string.

You may optionally specify an alias name to use the FOR() function in other workareas. The <alias> may be a workarea number from 1 up to DB_MAXWKA, an alias name which has been created with the USE command, or A through T, excluding M. If no alias name is specified, the FOR() function looks in the currently active workarea.

Example

set view to accounting

? for(3)

acct = "CAR"

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FOUND()

Class

Fields and Records

Purpose

Function to return whether a record was found

Syntax

FOUND([<workarea | alias>])

See Also

LOCATE, CONTINUE, SEEK, FIND, EOF(), BOF(), RLOOKUP(), LOOKUP(), SEEK()

Description

The FOUND() function returns .T. if a FIND, SEEK, LOCATE or CONTINUE operation was successful. The FOUND() function returns .F. if one of the above operations fails. If the optional <workarea/alias> is specified, then the function will operate in the required workarea. The FOUND() function should be used with the SEEK, FIND, LOCATE or CONTINUE commands for testing the success or failure of the operation.

Example

```
use accounts
set index to acc_no
seek "00751"
if found()
    change noclear
else
    dialog box "Customer not found."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FPUTS()

Class

ASCII File Access

Purpose

Function to write a character string to an ASCII file

Syntax

FPUTS(<expN1>,<expC1>[,<expN2>][,<expC2>])

See Also

FCLOSE(), FCREATE(), FEOF(), FERROR(), FFLUSH(), FGETS(), FOPEN(), FREAD(), FSEEK(), FWRITE()

Description

The FPUTS() function writes the specified character string to an ASCII file that was created by the FCREATE() function, or opened with the FOPEN() function. The numeric expression <expN1> is the file handle returned by either the FCREATE() or FOPEN() functions, and is used to specify the file that the FPUTS() function writes to.

The character expression <expC1> is the character string that FPUTS() will write to the file. The character string will be placed at the current position of the file pointer. After the string is written to the file, the file pointer is repositioned past the last character of <expC1>.

The FPUTS() function will terminate <expC1> with a default carriage return/line feed on OpenVMS, and a line feed on UNIX/Linux. You may optionally specify a different terminator with character expression <expC2>. Character expression <expC2> must evaluate to one or two characters. The optional numeric expression <expN2> specifies the number of bytes that the FPUTS() function will write from <expC1>. This number must be between 0 and 254.

The FPUTS() function returns the number of bytes, including the end of line terminator, that was written to the file. The FPUTS() function will return a zero if the write was unsuccessful.

Example

```
fp=fopen("names.txt")
count=0
do while .not eof()
    count= count + fputs(fp,trim(first) + trim(last))
    skip
enddo
dialog box str(count,5) + "written."
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FREAD()

Class

ASCII File Access

Purpose

Function to read and return a line of text from an ASCII file

Syntax

FREAD(<expN1>,<expN2>)

See Also

FCLOSE(), FCREATE(), FERROR(), FEOF(), FFLUSH(), FGETS(), FOPEN(), FPUTS(), FREADSTR(), FSEEK(), FWRITE()

Description

The FREAD() function reads and returns a line of text from the specified ASCII file. The ASCII file is specified by the file handle <expN1>. The file handle is created when the ASCII file is created with the FCREATE() function, or opened with the FOPEN() function. The FREAD() reads a line of text from the current position of the file pointer. You must specify the number of bytes to read from that position with the numeric expression <expN2>. The number of bytes may be from 0 to 254.

Example

```
fp = fopen("names.txt")
?fread(fp,80)
Smith,Bill
fclose(fp)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FREADSTR()

Class

ASCII File Access

Purpose

Function to return a character string from a text file

Syntax

FREADSTR(<expN1>, <expN2>)

See Also

FOPEN(), FCREATE(), FWRITE(), FERROR(), FCLOSE(), TEXTEDIT(), MEMOREAD(), MEMOWRITE()

Description

The FREADSTR() function returns the next line of a text file, opened by FOPEN(), as a character string. The <expN1> is the file pointer obtained from FOPEN(). The maximum number of bytes to read is specified by the value in <expN2>. The FREADSTR() function sequentially reads each line from the opened text file each time it is used on the file. If an error occurs during the operation, -1 is assigned to the FERROR() function. The file pointer must be assigned when the file is first opened using the FOPEN() function.

Example

```
use accounts
fp=fopen("existing.txt")
if ferror()=-1
    dialog box "The file could not be opened."
else
    string = freadstr(fp,80)
    do while not empty(string)
        ? string
        string = freadstr(fp,80)
    enddo
endif
fclose(fp)
if ferror()=-1
    dialog box "The file could not be closed."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FSEEK()

Class

ASCII File Access

Purpose

Function to position to a byte offset within a text file

Syntax

FSEEK(<expN1>, <expN2>, <expN3>)

See Also

FCLOSE(), FCREATE(), FCLOSE(), FERROR(), FREADSTR(), FWRITE()

Description

The FSEEK() function is used to position to a byte offset within a text file. The <expN1> is a valid file handle returned from the FOPEN() function. The <expN2> is the byte offset position to move the file pointer based on value defined by <expN3>. This can be a positive or negative number depending on the direction of pointer movement. The <expN3> defines the positions from which to start the byte offset, and can be any of the following three options:

Offset	Description
0	Start from start of file
1	Start from current position
2	Start from EOF

FSEEK() returns the new position of the file pointer relative to the beginning of the file. The beginning of the file is 0 regardless of the direction of movement.

Example

```
fp = fopen("names.txt")
string = freadstr(fp,80)
fseek(fp, 1024,0)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FSIZE()

Class

Disk and File Utilities

Purpose

Function to return size of file

Syntax

FSIZE(<expC>)

See Also

FILEINFO(), GETENV(), GETLOG(), DBF(), NDX(), FMT(), SET FULLPATH

Description

The FSIZE() function returns the size in bytes of the file specified in <expC> as a numeric.

Example

```
? fsize("db.exe")
```

4731284

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FTIME()

Class

Disk and File Utilities

Purpose

Function to return the time a file was last modified

Syntax

FTIME(<expC>)

See Also

DISKSPACE(), FILE(), SET CENTURY, SET DATE, SET MARK, SET PATH, LUPDATE(), FDATE()

Description

The FTIME() function returns, from the operating system, the time that the specified file was last modified. The file name is specified as character expression <expC> and may contain the full pathname. The file will be searched for in the current directory and any directories specified with the SET PATH command. The character expression <expC> may not contain wildcard characters. If the file specified by <expC> is open, the FTIME() function returns the latest modification date since the last time the file was closed.

Example

```
? ftime("main.prg")
```

11:51:38

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FULLPATH()

Class

Disk and File Utilities

Purpose

Function to return the path for the specified file

Syntax

FULLPATH(<filename>)

See Also

SET PATH, SET FULLPATH, BASENAME()

Description

The FULLPATH() function returns the path containing the specified <filename>. The path is return in a character string. FULLPATH() searches for <filename> in the current path which has been set with the SET PATH TO command.

Example

```
?FULLPATH("accounts.dbf")  
\usr\recital\UD\demo\accounts.dbf
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FV()

Class

Numeric Data

Purpose

Function to return future value of a periodic investment

Syntax

FV(<expN1>,<expN2>,<expN3>)

See Also

PAYMENT(), PMT(), PV(), CAGR(), LOG10(), PI()

Description

The FV() function calculates the future value of an investment, where <expN1> is the payment per period, <expN2> is the periodic interest rate and <expN3> is the number of payments.

Example

? fv(1000,.10,20)

57275.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

FWRITE()

Class

ASCII File Access

Purpose

Function to write a character expression to an ASCII file

Syntax

FWRITE(<expN1>, <expC> [,<expN2>])

See Also

FOPEN(), FCREATE(), FDATE(), FREAD(), FREADSTR(), FERROR(), FCLOSE(), FTIME(), TEXTEDIT(), MEMOREAD(), MEMOWRITE()

Description

The FWRITE() function writes the character expression <expC> to a specified text file. The <expN1> is the file pointer returned by either the FOPEN() or FCREATE() functions. The optional <expN2> specifies the maximum number of bytes to write from the buffer.

The effect of FWRITE() is to write new lines sequentially to the opened text file each time it is used on the file. If an error occurs during the operation -1 is assigned to the FERROR() function. The file pointer must be assigned when the file is first opened using the FOPEN() or FCREATE() functions.

Example

```
use acc_log
scatter to flist
m_total=alen(flist)
fp=fcreate("new.txt")
if ferror()=-1
    dialog box "The file could not be created."
else
    for n=1 to m_total
        fwrite(fp,flist[n],80)
    next
endif
fclose(fp)
if ferror()=-1
    dialog box "The file could not be closed."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GATEWAY()

Class

Data Connectivity

Purpose

Returns information about a gateway connection

Syntax

GATEWAY([<expN>])

See Also

CREATE GATEWAY, MODIFY GATEWAY, SET GATEWAY TO, START DATABASE, STOP DATABASE, CONNECTED()

Description

The GATEWAY() function returns specified information about the connection to the Recital Database Gateway in the current workarea. The following table lists the optional information that may be requested.

Value	Description
	Returns the server type, Recital, Oracle, ODBC, Informix, DB2, etc
0	Returns the server type, Recital, Oracle, ODBC, Informix, DB2, etc
1	Returns the node name that the gateway is connected to.
2	Returns the user name that was used to connect to the gateway.
3	Returns the password associated with the user name.
4	Returns the database connected to on the server.

Example

login "oracle"

?gateway()

Oracle

ORA

? gateway(1)

sales

? gateway(2)

scott

? gateway(3)

tiger

? gateway(4)

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETENV()

Class

Environment

Purpose

Function to get environment information

Syntax

GETENV(<expC>)

See Also

ACCESS(), GETUID(), GETGID(), GETPID(), GETLOG(), PUTLOG(), PUTENV()

Description

The GETENV() function returns the value of the environment variable <expC>. Its main purpose is to pass information into Recital (e.g. from DCL command procedures, or UNIX/Linux shell scripts). In UNIX, Linux and Windows it returns the value of an environment variable. In OpenVMS, it returns the value of a DCL symbol. When used in conjunction with the PUTENV() function, the GETENV() function is very useful for passing parameters to Recital programs spawned in the background.

Example

```
? getenv("file")  
/usr/recital/patrons.dbf
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETFILE()

Class

Disk and File Utilities

Purpose

Function to display an Open dialog allowing a filename to be selected and returned.

Syntax

GETFILE([<expC1>] [,<expC2>] [,<expC3>] [,<expN1>])

See Also

PUTFILE()

Description

The GETFILE() function displays an Open dialog allowing a filename to be selected and returned. The dialog allows navigation through directories and shows a list of relevant files in that directory. The cursor keys, Return key and tab key can be used to navigate the different sections under a character mode environment. If the user selects a file, the GETFILE() function returns the name of that file. If no file is selected, the GETFILE() function returns an empty string "".

Parameters	Requirement	Description
<expC1>	No	A file extension skeleton. If specified only files with this extension are shown
<expC2>	No	The text to display at the top of the dialog. If not specified, 'Open' is displayed.
<expC3>	No	The text to display on the Ok button. This is ignored, the button is always labeled 'Ok'.
<expN1>	No	The buttons set to be displayed. This is ignored, the buttons are always Ok, Cancel, Reset, Back and Help.

Example

```
cProgramSelected = getfile("prg","Please select a program")
```

Products

Recital Mirage Server, Recital Terminal Developer

GETFONT()

Class

Fonts

Purpose

Function to return the chosen font name, size and style

Syntax

GETFONT()

See Also

AFONT(), FONTMETRIC(), WFONT()

Description

The GETFONT() function is included for language compatibility. It returns the current font name, size and style in a comma separated string.

Font Styles:

Code	Style
B	Bold
I	Italic
N	Normal
O	Outline
Q	Opaque
S	Shadow
-	Strikeout
T	Transparent
U	Underline

Example

```
? getfont()
```

Monospaced,8,N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETGID()

Class

Environment

Purpose

Function to get group identity

Syntax

GETGID()

See Also

CREATE, MODIFY STRUCTURE, GETUID(), GETENV(), GETPID(), ACCESS(), PUTENV(), STR()

Description

The GETGID() function returns the group number of the user. In UNIX/Linux, this is the group number of the logged in user from the /etc/passwd file. In OpenVMS, this is the group number extracted from the UIC of the logged in user from the system authorization file.

The GETGID() function returns a decimal value. OpenVMS group user IDs are stored in octal, however you may use the STR() function to convert the decimal value to octal.

Example

```
use accounts
@2,3 get name
@3,3 get rates;
    when getgid() > 100 and getgid() < 200
read
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETGRNAM()

Class

Environment

Purpose

Function to get group name

Syntax

GETGRNAM()

See Also

GETGID(), GETUID(), GETENV(), GETPID(), ACCESS(), PUTENV()

Description

The GETGRNAM() function returns the group name of the current user.

Example

```
use accounts
@2,3 get name
@3,3 get rates;
    when getgrnam() = "recital"
read
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETLOCALHOST()

Class

Data Connectivity

Purpose

Function to return the hostname of the server that is running the Recital Database/Mirage Server

Syntax

GETLOCALHOST()

See Also

GETREMOTEADDR(), GETREMOTEHOST(), GETREMOTEUSER(), ISMIRAGE(), ISSERVER()

Description

The GETLOCALHOST() function returns the hostname of the server that is running the Recital Database Server or Recital Mirage Server. The address is returned as a character string.

An empty string will be returned if the function is not being run via the Recital Database Server or Recital Mirage Server or if the server does not have a hostname defined.

Example

```
mCurrentHostName = getlocalhost()
```

Products

Recital Database Server, Recital Mirage Server

GETLOG()

Class

Environment

Purpose

Function to get value from a defined logical name

Syntax

GETLOG(<expC1>, [<expC2>])

See Also

PUTLOG(), GETENV(), PUTENV()

Description

Used to manipulate process wide logical names, the GETLOG() function translates the specified logical name <expC1>, and returns the equivalence string. The optional <expC2> is the name of the logical table where the specified logical name is stored. If no logical table name is specified the GETLOG() function returns values from the logicals stored in the LNM\$PROCESS_TABLE. These can be displayed in OpenVMS by entering the OpenVMS command SHOW LOGICAL/PROCESS.

On other operating systems this function operates in the same way as GETENV() and translates environment variables. The GETLOG() function always returns a character string without changing the case.

Example

```
// Display the system printer name
system_printer = getlog("sys$print","lnm$process_table")
@03,30 say "system Printer:"
@03,30 say system_printer
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETNO()

Class

Screen Forms

Purpose

Function to return the number of the current GET

Syntax

GETNO()

See Also

@...GET

Description

The GETNO() function returns the number of the currently active GET. You may use a function which returns a number with the VALID option of the @...GET command to activate a specific GET for selective processing in a form. The GETNO() function returns a numeric data type value.

Example

```
function chk_country
if getno() = 1
  do case
    case country = "USA"
      // Move to the state field
      m_return = 2
    case country = "CANADA"
      // Move to the province field
      m_return = 4
    otherwise
      dialog box "INVALID COUNTRY"
      m_return = .F.
    endcase
  else
    dialog box "ERROR"
  endif
return m_return

@02,10 get country valid chk_country()
@03,10 get state
@04,10 get zip
@05,10 get province
read
```

Products

Recital Mirage Server, Recital Terminal Developer

GETPID()

Class

Environment

Purpose

Function to get process identity

Syntax

GETPID()

See Also

GETUID(), GETENV(), GETGID(), ACCESS(), TMPNAM(), PUTENV()

Description

The GETPID() function returns a unique system wide number which represents the identity of the user process. This function will return the same value while you remain logged in. This can be used, among other purposes, to create system-wide temporary files. Note that each time you log into the system, you will have a different process identity.

Example

```
? getpid()
```

```
14586
```

```
pid = getpid()
```

```
? type("pid")
```

```
N
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETPROT()

Class

Environment

Purpose

Function to return current file-protection mask

Syntax

GETPROT()

See Also

CREATE, MODIFY STRUCTURE, GETGID(), GETUID(), SETPROT()

Description

Used to manipulate file protection masks, the GETPROT() function returns the current file protection mask which will be used when creating any new files. If the protection mask has not been changed with the SETPROT()function, the default protection mask is returned. The string returned by GETPROT() is known as the file protection mask. It is the same on all operating systems, however on non-OpenVMS platforms the 'System' mask and the 'Delete' mask are ignored. The GETPROT() function always returns a character string in upper case.

Example

```
old_mask = getprot()
setprot("S:RWED,O:RWED,G:RW,W:RW")
pack
setprot(old_mask)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETREMOTEADDR()

Class

Data Connectivity

Purpose

Function to return the IP address of the client that is connected to the Recital Database/Mirage Server

Syntax

GETREMOTEADDR()

See Also

GETLOCALHOST(), GETREMOTEHOST(), GETREMOTEUSER(), ISMIRAGE(), ISSERVER()

Description

The GETREMOTEADDR() function returns the IP address of the client that is connected to the Recital Database Server or Recital Mirage Server. The address is returned as a character string in 999.999.999.999 format.

An empty string will be returned if the function is not being run via the Recital Database Server or Recital Mirage Server.

Example

```
mCurrentIP = getremoteaddr()
```

Products

Recital Database Server, Recital Mirage Server

GETREMOTEHOST()

Class

Data Connectivity

Purpose

Function to return the hostname of the client that is connected to the Recital Database/Mirage Server

Syntax

GETREMOTEHOST()

See Also

GETLOCALHOST(), GETREMOTEADDR(), GETREMOTEUSER(), ISMIRAGE(), ISSERVER()

Description

The GETREMOTEHOST() function returns the hostname of the client that is connected to the Recital Database Server or Recital Mirage Server. The address is returned as a character string.

An empty string will be returned if the function is not being run via the Recital Database Server or Recital Mirage Server or if the client does not have a hostname defined.

Example

```
mCurrentHostName = getremotehost()
```

Products

Recital Database Server, Recital Mirage Server

GETREMOTEUSER()

Class

Data Connectivity

Purpose

Function to return the user name of the client that is connected to the Recital Database/Mirage Server

Syntax

GETREMOTEUSER()

See Also

GETLOCALHOST(), GETREMOTEADDR(), GETREMOTEHOST(), ISMIRAGE(), ISSERVER()

Description

The GETREMOTEUSER() function returns the user name used by the current client to connect to the Recital Database Server or Recital Mirage Server. The user name is returned as a character string.

An empty string will be returned if the function is not being run via the Recital Database Server or Recital Mirage Server.

Example

```
mCurrentUser = getremoteuser()
```

Products

Recital Database Server, Recital Mirage Server

GETRESULTSET()

Class

Data Connectivity

Purpose

Function to return the workarea number of an SQL cursor previously marked as a resultset

Syntax

GETRESULTSET()

See Also

CLEARRESULTSET(), SETRESULTSET(), SQL SELECT

Description

The GETRESULTSET() function returns the workarea number of an SQL cursor previously marked as a resultset by the SETRESULTSET() function. The SETRESULTSET() function is particularly used in returning a resultset from a stored procedure in SQL client/server applications.

GETRESULTSET() will return 0 (zero) if no SQL cursor is currently marked as a resultset. The marker can be cleared from an SQL cursor using the CLEARRESULTSET() function.

Example

```
function GetExampleCursor
```

```
lparameters lcAccountNo
```

```
select * from example where account_no = lcAccountNo into cursor curExample
```

```
return setresultset("curExample")
```

```
open database southwind
```

```
GetExampleCursor("00050")
```

```
? "Returned resultset is in work area #" + ltrim(str(getresultset()))
```

```
set sql off
```

```
select getresultset()
```

```
display all
```

```
? "Cleared resultset marker in work area #" + ltrim(str(clearresultset()))
```

```
? iif(getresultset() > 0, "Resultset available in work area #" + ltrim(str(getresultset()));
```

```
    "No resultsets available")
```

```
?
```

```
close databases
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETSIG()

Class

Error Handling and Debugging

Purpose

Retrieve the last signal number received

Syntax

GETSIG()

See Also

ERROR(), ON FINISH, ON TERMINATION

Description

The GETSIG() function returns the last signal number received, or zero if none has been received. This can be used in an ON FINISH procedure to determine if the user exited by themselves or they received a signal to terminate.

Example

```
procedure on_finish
if getsig() != 0
    dialog box "Signal received. signo = "+alltrim(str(getsig())) label "ON FINISH"
elseif error() > 0
    dialog box "Error received, error = "+alltrim(str(error())) label "ON FINISH"
else
    dialog box "Successful exit" label "ON FINISH"
endif
return

on finish do on_finish
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GETUID()

Class

Environment

Purpose

Function to get user identity

Syntax

GETUID()

See Also

CREATE, MODIFY STRUCTURE, GETGID(), GETENV(), GETPID(), ACCESS(), PUTENV()

Description

The GETUID() function returns the user number. In UNIX/Linux, this is the user number of the logged in user from the /etc/passwd file. In OpenVMS this is the user number extracted from the UIC of the logged in user from the system authorization file. The GETUID() returns a decimal value. OpenVMS user IDs are stored in octal. You may use the STR() function to convert the OpenVMS value to octal when using the GETUID() function.

Example

```
use accounts
@2,3 get name
@3,3 get rates;
    when getuid() > 100 and getuid() < 200
read
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GOMONTH()

Class

Date and Time Data

Purpose

Function to return a date that is a specified number of months before or after a particular date or datetime

Syntax

GOMONTH(<expD> | <expT>, <expN>)

See Also

CDOW(), CMONTH(), CTOD(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), EPOCH(), MDY(), MONTH(), MTOD(), QUARTER(), STOD(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK

Description

The GOMONTH() function returns a date which is <expN> months before or after the date expression <expD> or datetime expression <expT>. If <expN> is a negative number, the GOMONTH() function returns a date that is before <expD> or <expT>. If <expN> is a positive number, the GOMONTH() function returns a date that is after <expD> or <expT>.

Example

```
?gomonth({04/14/2004}, 4)
```

08/14/2004

```
? gomonth({01/21/2004 03:18:33 PM}, -12)
```

01/21/2003

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

GOTO()

Class

Fields and Records

Purpose

Function to position record pointer in the specified work area

Syntax

GOTO(<workarea | alias>, <expN>)

See Also

GOTO, CREATE, MODIFY STRUCTURE, RLOOKUP(), LOOKUP()

Description

The GOTO() function positions the record pointer in the specified <workarea | alias> and returns a .T. if successful. This function is most useful in the Applications Data Dictionary where it can provide an additional link to another data file whenever needed. The <expN> is the record number you wish to move the record pointer to.

Example

Name = IIF(GOTO(supplier, RECNO()), suppliers->name, "??????")

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

HARDCR()

Class

Expressions and Type Conversion

Purpose

Function to convert carriage returns

Syntax

HARDCR(<expC> | <memofield>)

See Also

MEMOEDIT(), MEMOTRAN(), MEMOWRITE(), MEMOREAD(), MEMOLINE(), TEXTEDIT(),
MLCOUNT(), MLINE(), LEN()

Description

The HARDCR() function converts all the 'soft' carriage returns (ASCII 141) in the specified <expC> or <memofield>, to 'hard' carriage returns (ASCII 13). The HARDCR() function returns the <expC> or <memofield> as a character string, after conversion.

Example

cFormat = hardcr(notepad)

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

HEADER()

Class

Disk and File Utilities

Purpose

Function to return the table header size

Syntax

HEADER()

See Also

DISKSPACE(), RECSIZE(), RECCOUNT()

Description

The HEADER() function returns the size of a table header. The table file header contains information about the table, among other things, the field definitions, and the field descriptions. When used with the RECSIZE() function and the RECCOUNT() function, the HEADER() function allows you to calculate the space which your table occupies. The total space needed to backup your table can thus be calculated.

Example

use mytable

size = header() + reccount() * recsize()

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

HOME()

Class

Disk and File Utilities

Purpose

Function to return the directory where the Recital software is installed

Syntax

HOME()

See Also

CURDIR(), DEFAULT()

Description

The HOME() function returns a character string representing the drive and directory where the Recital software is installed.

Example

```
// UNIX  
?home()  
/usr/recital/UD/
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

HOUR

Class

Date and Time Data

Purpose

Function to return the numeric hours from a specified datetime

Syntax

Hour(<expT>)

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOURS(), MINUTE(), MINUTES(), SEC(), SECONDS(), SECS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The HOUR() function returns the hours from the specified datetime expression <expT> as a numeric value.

Example

```
? hour({ 10/10/2004 10:15:43 AM})
```

10

```
m_Hour = hour(datetime())
```

```
? type("m_Hour")
```

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

HOURS()

Class

Date and Time Data

Purpose

Function to extract number of hours from a time string

Syntax

HOURS(<time-string>)

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), MINUTE(), MINUTES(), SEC(), SECONDS(), SECS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The HOURS() function extracts the hours from the specified time-string and returns the number of hours as a numeric value.

Example

```
? hours("10:00:00")
```

10

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

HSCROLL()

Class

Screen Forms

Purpose

Function to designate a screen area to scroll horizontally

Syntax

HSCROLL(<expN1>,<expN2>,<expN3>,<expN4>,<expN5>)

See Also

SET SCROLL, TEXTEDIT(), SCROLL(), SCROLL

Description

The HSCROLL() function designates an area of the screen to scroll left, scroll right, or blank out. This function can be used to emulate windows. <expN1>, <expN2>, <expN3> and <expN4> are the beginning row, beginning column, ending row, and ending column screen coordinates of the area. <expN5> is the number of columns to scroll. A value greater than zero scrolls right. A value less than zero scrolls to the left. A value of zero clears the specified scroll area.

Example

```
// Create window with text
@05,35,09,46
@06,36 say "ABCDEFGHJIJ"
@07,36 say "abcdefghij"
@08,36 say "0123456789"
// Scroll slowly one column at a time until
// the window is clear.
for i = 1 to 10
    hscroll(6,36,8,45,1)
    sleep 1
next
```

Products

Recital Mirage Server, Recital Terminal Developer

I2BIN()

Class

Expressions and Type Conversion

Purpose

Function to convert numeric to binary encoded character string

Syntax

I2BIN(<expN>)

See Also

CREATE BRIDGE, BIN2I(), BIN2L(), BIN2W(), BINCLOSE(), BINCREATE(), BINOPEN(), BINREAD(), BINSEEK(), BINWRITE(), L2BIN()

Description

The I2BIN() function converts a numeric value into a binary encoded character string formatted as an unsigned 16-bit integer. The <expN> is any numeric value. Any decimal places are ignored.

This function can be used to construct composite index keys of mixed data types in OpenVMS RMS files. With the exception of the BIN2W(), all binary conversion functions may be used in conjunction with the binary file access functions.

Example

```
? bin2i(i2bin(987))  
    987
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ICACHE()

Class

Indexing

Purpose

Function to return optimum ICACHE size for a given index expression

Syntax

ICACHE(<exp>[, <expN>])

See Also

SET ICACHE, SET DCACHE, INDEX, REINDEX

Description

The ICACHE() function returns the optimum size of the index buffer cache for an index created on the specified expression. The <exp> can either be a numeric value representing the width of the index key, or the index expression. This function is used in conjunction with the SET ICACHE TO command. The command SET ICACHE TO OPTIMUM can be used instead of the ICACHE() function to automate defining index buffer caches.

If the optional <expN> is specified and is greater than zero (0), the required memory allocation is returned.

Example

```
use names index names
size = icache(indexkey(1))
set icache to size
reindex
use names
set icache to icache(name+left(address, 5))
index on name+left(address, 5) to names
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ICASE()

Class

Applications

Purpose

Function to execute an immediate case statement

Syntax

ICASE(<expL1>, <exp1>[, <expL2>, <exp2>] [...] [, <exp-otherwise>)

See Also

DO CASE, IF, IF()

Description

The ICASE() function operates as an immediate DO CASE statement. It evaluates the specified conditions <expL1> to <expLn> and returns the matching expression <exp1> to <expn> for the first condition that evaluates to True (.T.). If none of the conditions evaluates to True, the 'otherwise' expression, <exp-otherwise>, is returned. If there is no otherwise expression specified, and none of the conditions evaluates to True, ICASE() returns a null (.NULL.).

Example

```
accept "Enter a command: " to command
&(icase(upper(command) = "BROWSE", "browse", upper(command) = "DIR", "dir",;
    "Set message to [Unknown command.]"))
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ID()

Class

Environment

Purpose

Function to return the login name of the current user

Syntax

USER()

See Also

DISPLAY USER, LIST USER, GETPID(), GETUID(), GETGID(), USER(), SET COMPATIBLE

Description

The ID() function returns the login name of the current user. The login name is assigned by the operating system. This function is synonymous with the USER() function.

If COMPATIBLE is set to VFP, the ID() function is compatible with Visual FoxPro and returns user and machine information in the following format:

MACHINEID # userid

Example

?id()

william

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

IF()

Class

Applications

Purpose

Function to execute an immediate if

Syntax

IF(<expL>, <exp1>, <exp2>)

See Also

ACC(), CALC(), ICASE(), IF

Description

The IF() function returns the value of the expression <exp1> or the value of the expression <exp2>, depending on the result of the expression <expL>. If the expression <expL> returns .T., the IF() function returns the value of <exp1>. If the expression <expL> returns .F., the IF() function returns the value of <exp2>. The expressions <exp1> and <exp2> may themselves contain IF() functions. The IF() function may be used anywhere that a normal expression can be used (e.g. in SQL SELECT column expressions).

The IF() function is synonymous with the IIF() function.

Example

event = "drama"

? if(event = "drama", "It's for Drama", "It's not for Drama")

It's for Drama

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

IFILECOUNT()

Class

Disk and File Utilities

Purpose

Function to return the number of index files matching the pattern

Syntax

IFILECOUNT(<skeleton>)

See Also

NDX(), FILECOUNT(), DIR(), FILE(), MENU FILES

Description

The IFILECOUNT() function returns the number of index files which match the specified <skeleton> and are valid for the currently selected table.

Example

```
? ifilecount("*.ndx")  
5
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

IIF()

Class

Applications

Purpose

Function to execute an immediate if

Syntax

IIF(<expL>, <exp1>, <exp2>)

See Also

ACC(), CALC(), ICASE(), IF

Description

The IIF() function returns the value of the expression <exp1> or the value of the expression <exp2>, depending on the result of the expression <expL>. If the expression <expL> returns .T., the IIF() function returns the value of <exp1>. If the expression <expL> returns .F., the IIF() function returns the value of <exp2>. The expressions <exp1> and <exp2> may themselves contain IIF() functions. The IIF() function may be used anywhere that a normal expression can be used (e.g. as part of expressions in REPORT, DICTIONARY and LABEL formats).

Example

```
event = "drama"  
? iif(event = "drama", "It's for Drama", "It's not for Drama")
```

It's for Drama

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

INDEXEXT()

Class

Indexing

Purpose

Function to return the default index extension

Syntax

INDEXEXT()

See Also

INDEX, REINDEX, DBF(), FILETYPE(), FMT(), INDEXKEY(), INDEXORDER(), NDX(), READEXIT(), READINSERT(), SET CLIPPER, SET CLIPPER5, SET COMPATIBLE, SET EDITFIELD, SET FILECASE, SET FILETYPE, SET INDEX, SET INDEXEXT, SET MEMOEXT, SET PCEDIT, SET PCEXACT, SET PCFILTER, SET PCGRAPHICS, SET PCKEYS, SET PCLOCKING, SET PCPICTURE, SET PCSAYS, SET PCUNIQUE, DB_FOXPLUSBUGS, DB_FOXPROKEYS, DB_SAMBA

Description

The INDEXEXT() function returns the current default index extension. The default extension is “.ndx”, but this can be changed with the SET INDEXEXT command.

Example

? indexext()

.ndx

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

INDEXKEY()

Class

Indexing

Purpose

Function to return the index key expression

Syntax

INDEXKEY(<expN>)

See Also

LEN(), FIELD(), FILTER(), DBF(), NDX (), FMT(), FCOUNT(), INDEXORDER(), INDEXEXT()

Description

The INDEXKEY() function is synonymous with the KEY() function. The INDEXKEY() function returns the index key expression for index <expN>, or a null string if no index file exists. When used in conjunction with the INDEXORDER() function, the INDEXKEY() function will return the index key expression of the master index. The INDEXKEY() function always returns a character string in lower case.

Example

```
use accounts index acc_no, date_paid
? indexkey(1)
acc_no + dtos(date_rec)
set order to 2
? indexkey(indexorder())
dtos(date_paid) + str(amo_paid,11,2)
index on lower(left(company,20)) to company
? indexkey(1)
lower(left(company,20))
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

INDEXORDER()

Class

Indexing

Purpose

Function to return the currently selected master index number

Syntax

INDEXORDER([<workarea | alias>])

See Also

INDEX, SET INDEX, CLOSE INDEX, USE, SEEK, FIND, REINDEX, CREATE VIEW, SET ORDER, SET ICACHE, SET DCACHE, SET CACHELOAD, INDEXKEY(), NDX (), IFILECOUNT(), RLOOKUP(), LOOKUP(), SYS(14), INDEXEXT(), ORDER()

Description

The INDEXORDER() function returns the currently selected master index number as specified with the SET ORDER TO command. If no index files are selected, it returns -1. If index file are open, and SET ORDER has been set to 0, the INDEXORDER() function returns 0. If the optional <workarea | alias> is specified, then the function will operate in that <workarea | alias>.

When used in conjunction with the NDX() or INDEXKEY() functions, the INDEXORDER() function will return the index file name or index key of the master index. The INDEXORDER() function is synonymous with the ORDER() function.

Example

use patrons index name, event

? indexorder()

1

set order to 2

? indexorder()

2

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

INKEY()

Class

Keyboard Events

Purpose

Function to return key press

Syntax

INKEY([<expN>[,<expC>]])

See Also

SET PCKEYS, READKEY(), SET KEY, SLEEP, ON ESCAPE, ON KEY, ON MOUSE, MESSAGE, SET MESSAGE, WAIT, INPUT, CHR(), LASTKEY(), MCOL(), MROW()

Description

The INKEY() function returns zero if no key has been pressed on the keyboard, or a non-zero value which is the ASCII code for the key if one has been pressed.

<expN>

If the optional parameter <expN> is included then the INKEY() function will wait <expN> number of seconds for a key to be pressed. If no key is pressed during that time, the program will continue to execute and INKEY() will return a value of 0. If a key is pressed during the specified time, the program will continue to execute and INKEY() will return the ASCII code for the key pressed.

If the value of zero is specified for <expN>, then the INKEY() function will wait until a key is pressed on the keyboard. The command SET PCKEYS is used to change the return value of the INKEY() function. If it is set ON, then the return value will match an IBM PC keyboard. If INKEYDELAY is set OFF, then DO WHILE loops that wait for user input with the INKEY() function may be very CPU intensive. The SET INKEYDELAY command controls the wait period for the INKEY() function.

<expC>

A second parameter, <expC> is available in Recital Mirage. If <expC> is equal to "M", the INKEY() function will trap and return mouse clicks. The numeric value returned indicates the mouse operation.

INKEY() Return Value	Mouse Operation
151	LEFTCLICK
152	RIGHTCLICK
153	DBLCLICK
154	CTRL+LEFTCLICK
155	CTRL+RIGHTCLICK
156	CTRL+DBLCLICK
157	SHIFT+LEFTCLICK
158	SHIFT+RIGHTCLICK
159	SHIFT+DBLCLICK
160	ALT+LEFTCLICK
161	ALT+RIGHTCLICK
162	ALT+DBLCLICK

INKEY(<expN>,"M") can also be used in conjunction with the ON MOUSE command to run a command when a mouse operation takes place. The command run can be a call to a procedure or User Defined Function if required. The MROW() and MCOL() functions can be called to determine the current row and column positions at the time of the mouse operation. The LASTKEY() function can be called to determine the type of operation. LASTKEY() returns the same numeric values for mouse operations as shown in the table above.

Example

```
do while inkey()=0
    @0,68 say ampm()
    sleep 2
enddo
```

```
// Another Example
? "Press any key..."
ch = inkey(0)
```

```
// Recital Mirage Example
? "Click to continue..."
mouse_op = inkey(0,"M")
if mouse_op = 151
    // left click
    //...
elseif mouse_op = 152
    //right click
    //...
endif
```

Products

Recital Mirage Server, Recital Terminal Developer

INLIST()

Class

String Data

Purpose

Function to search a list of expressions

Syntax

INLIST(<exp1>,<exp2>[,<exp3>...])

See Also

LOOKUP(), RLOOKUP()

Description

The INLIST() function searches a list of expressions and returns a logical true (.T) if the specified expression is found within that list. The expression to search for is represented by <exp1>. Expressions following <exp1> make up the list in which the INLIST() function will search. All the expressions specified must be the same data type: character, numeric, logical, date.

Example

```
do case
  case inlist(cState, "CA", "OR", "WA", "NV")
    cTimezone = "Pacific"
  case inlist(cState,"MT", "ID", "WY", "UT", "CO",;
    "AZ", "NM")
    cTimezone = "Mountain"
endcase
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

INSMODE()

Class

Keyboard Events

Purpose

Function to return current state of insert mode

Syntax

INSMODE([<expL>])

See Also

READINSERT(), SET READINSERT

Description

The INSMODE function returns a logical true (.T.) if insert mode is on, or a logical false (.F) if insert mode is off. You may optionally set insert mode on or off by specifying logical expression <expL>. If <expL> is .F., insert mode is set off. If <expL> is .T. then insert mode is set on.

Example

```
?insmode()
```

```
.T.
```

Products

Recital Mirage Server, Recital Terminal Developer

INT()

Class

Numeric Data

Purpose

Function to return integer

Syntax

INT(<expN>)

See Also

ROUND(), SET DECIMALS

Description

The INT() function converts the numeric expression <expN> to an integer by discarding all digits to the right of the decimal point.

Example

? int(2.56)

2

? int(-2.56)

-2

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

INUSE()

Class

Table Basics

Purpose

Function to determine if a table is open

Syntax

INUSE(<expC>)

See Also

USE, SET EXCLUSIVE, SET DCACHE, SET CACHELOAD, DBF(), NDX(), NETERR(), USED()

Description

The INUSE() function checks if a table with the alias name specified in <expC> is open in any workarea. The INUSE() functions returns .T. if a table is open, .F. if not. The alias can be specified in the USE command. If not specified, the table basename is used

Example

```
// The demo view bridge opens
// the tables customer, accounts,
// state and products in workareas 1 – 4
use demo
? inuse("customer")
.T.
? inuse("product")
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

IOSTATS()

Class

Performance and Optimization

Purpose

Function to monitor table and index file I/O operations

Syntax

IOSTATS(<expC> [, <workarea | alias>])

See Also

CLEAR IOSTATS, SET ICACHE, SET DCACHE

Description

The IOSTATS() function provides a facility for monitoring table and index file I/O operations. This function is very useful for performance tuning. If the optional <workarea | alias> is omitted, then the value returned is the total for all active workareas.

The <expC> is a character expression denoting any of the following operations:

Item	Action
Reads	# of row reads for tables
Updates	# of row updates for tables
Deletes	# of row deletes for tables
Recalls	# of row recalls for tables
Appends	# of row inserts for tables
DcacheReads	# of cache reads for tables
DcacheWrites	# of cache writes for tables
IReads	# of index reads for tables
IWrites	# of index writes for tables
IcacheReads	# of index cache reads for tables
IcacheWrites	# of index cache writes for tables
Opens	# of table opens
Closes	# of table closes
KeyStrokes	# of key strokes

OPENS, CLOSES, and KEYSTROKES are general items and are not specific to individual workareas. KEYSTROKES are only applicable Recital Terminal Developer.

Example

```
// Print total keystrokes  
? iostats("keystrokes")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISALPHA()

Class

String Data

Purpose

Function to test for an alphabetic character

Syntax

ISALPHA(<expC>)

See Also

ISLOWER(), ISUPPER(), UPPER(), LOWER()

Description

The ISALPHA() function returns .T. if the first character of the character expression <expC> is alphabetic (i.e. lies within A-Z or a-z).

Example

```
store "965.23" to value
// If alpha assign 0
if isalpha(m->value)
    value = 0
else
    // convert to numeric
    value = val(m->value)
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISBLANK()

Class

Expressions and Type Conversion

Purpose

Function to check for empty value

Syntax

ISBLANK(<expC> | <expD> | <expT> | <expL> | <memofield> | <expN>)

See Also

CTOD(), CTOT(), IIF(), SPACE(), TYPE(), SET CENTURY, SET DATE, SET MARK, SET SECONDS

Description

The ISBLANK() function returns .T. if the specified expression is 'empty'.

Data Type	Empty
Character	Space(0), "", [], ', ', no text entered
Date	CTOD(""), {}, {00/00/0000}, { / / } and other SET CENTURY, SET MARK and SET DATE variations
Datetime	CTOT(""), { / / : : AM} and other SET CENTURY, SET MARK, SET SECONDS and SET DATE variations
Logical	.F.
Memo	No data entered
Numeric	0

Example

```
if isblank(name + address)
    dialog box "Name and address not specified."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISCOLOR()

Class

Screen Forms

Purpose

Function to test for color monitor

Syntax

ISCOLOR()

See Also

REVERSE(), BOLD(), UNDERLINE(), SETCOLOR(), SET COLOR, @...FILL, @...GET, @...TO

Description

The ISCOLOR() function will return .T. if the current terminal definition file supports color, otherwise .F..

Example

```
if iscolor()
    set color to GR/B,W/R
else
    set color to W+
endif
```

Products

Recital Mirage Server, Recital Terminal Developer

ISDIGIT()

Class

String Data

Purpose

Function to check to see if the first character in a given character expression is a digit

Syntax

ISDIGIT(<expC>)

See Also

ISALPHA(), ISLOWER(), ISUPPER()

Description

The ISDIGIT() function will return .T. if the first character of the given character expression is a value within the range of 0 to 9, and .F. otherwise.

Example

? isdigit("9 Belmont Ave")

.T.

? isdigit("Nine Belmont Ave")

.F.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISFILTERED()

Class

Table Basics

Purpose

Function to test if table has an active filter expression

Syntax

ISFILTERED([<workarea | alias>])

See Also

SET FILTER, SET DELETED, SET VIEW, DBFILTER(), FILTER()

Description

The ISFILTERED() function tests whether a table has an active filter expression. If the optional <workarea | alias> is specified, then the function will operate on the specified <workarea | alias>. ISFILTERED() returns .T. (True) if a filter is active in the current or specified table and .F. if there is no active filter or no active table in the current or specified workarea.

Example

```
use patrons
set filter to event="BALLET"
? isfiltered()
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISLOCKED()

Class

Manual Locking

Purpose

Function to test if current record is locked

Syntax

ISLOCKED()

See Also

ISALPHA(), ISUPPER(), UPPER(), ISLOWER(), LOWER(), LOCK(), LKSYS()

Description

The ISLOCKED() function returns .T. if the current record is locked by the current user, and .F. otherwise.. The ISLOCKED() function does not determine whether the current record is locked by another user, the LKSYS() function can be used for this purpose. The ISLOCKED() function is useful in validation and hot-key procedures.

Note: Recital automatically performs file and record locking, so in most cases the ISLOCKED() function need not be used with manual locking commands or functions.

Example

? islocked()

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISLOWER()

Class

String Data

Purpose

Function to test for lower case character

Syntax

ISLOWER(<expC>)

See Also

ISALPHA(), ISUPPER(), UPPER(), LOWER()

Description

The ISLOWER() function returns .T. if the first character of the character expression <expC> is lower case and .F. otherwise.

Example

```
use prospect  
replace all company with proper(company);  
    for islower(company)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISMARKED()

Class

Transaction Processing

Purpose

Function to verify if journaling is in operation

Syntax

ISMARKED()

See Also

SET ROLLBACK, BEGIN TRANSACTION, RESET IN, ROLLBACK, ROLLBACK(), END TRANSACTION , COMPLETED()

Description

The ISMARKED() function returns .T. if the current workarea is marked as having a transaction in progress and .F. if not.

Example

```
procedure recovery
rollback
if rollback()
    dialog box "Rollback was ok."
else
    dialog box "Rollback not completed."
endif
return

use setcomm
on error do recovery
if ismarked()
    dialog box "Other transaction in progress."
    return
endif
begin transaction
    delete first 15
    insert
    replace all t1 with (t2*t3)/100
    list
end transaction
if completed()
    dialog box "Transaction completed. No errors."
else
    dialog box "Errors occurred during transaction"
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISMIRAGE()

Class

Environment

Purpose

Used to determine whether an application is being run via the Recital Mirage Server

Syntax

ISMIRAGE()

See Also

ISSERVER()

Description

The ISMIRAGE() function returns .T. (TRUE) if the current application code is being run via the Recital Mirage Server. If the application is being run on any other client, ISMIRAGE() will return .F. (FALSE). This allows developers to customize code for the particular environment yet share the same code across all environments.

Example

```
if ismirage()
    set dialog on
    set message on
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISMOUSE()

Class

Screen Forms

Purpose

Function to return .T. if a mouse is detected

Syntax

ISMOUSE()

See Also

SET MOUSE, SET NAVIGATE

Description

The ISMOUSE() function will return .T. if a mouse is detected, otherwise .F..

Example

```
?ismouse()
```

```
.T.
```

Products

Recital Mirage Server, Recital Terminal Developer

ISNULL()

Class

Expressions and Type Conversion

Purpose

Function to test if an expression evaluates to NULL

Syntax

ISNULL(<expr>)

See Also

EMPTY(), ISALPHA(), ISBLANK(), ISDIGIT()

Description

The ISNULL() function will return True (.T.) if the specified expression, <expr> is NULL otherwise False (.F.).

Example

```
select account_no, isnull(last_name) from customer;
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISPRINTER()

Class

Printing

Purpose

Function to test if printer is ready

Syntax

ISPRINTER()

See Also

PRINT STATUS(), SET PRINTER, PRINT, SYS(), PRINTSCREEN()

Description

The ISPRINTER() function will return .T. if the printer is on line and ready to receive data, otherwise .F..

Example

```
? isprinter()
```

```
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISSERVER()

Class

Environment

Purpose

Used to determine whether an application is being run via the Recital Database or Mirage Servers

Syntax

ISSERVER()

See Also

ISMIRAGE()

Description

The ISSERVER() function returns .T. (TRUE) if the current application code is being run via the Recital Database or Mirage Servers. If the application is being run on Recital Terminal Developer, ISSERVER() will return .F. (FALSE). This allows developers to customize code for the particular environment yet share the same code across all environments.

Example

```
store 1 to mimagebtn
if issERVER()
    @23,46 say "ImageButton: "get mimagebtn function "*HNB *okbtn;*cancelbtn;*savebtn;*addbtn";
    valid v_imagebtn() size 2,2,-1
else
    @23,46 say "ImageButton: "get mimagebtn function "*HNB ok,cancel;save;add";
    valid v_imagebtn() size 2,2,-1
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ISUPPER()

Class

String Data

Purpose

Function to test for upper case character

Syntax

ISUPPER(<expC>)

See Also

ISALPHA(), ISLOWER(), LOWER(), UPPER()

Description

The ISUPPER() function returns .T. if the first character of the character expression <expC> is upper case and .F. otherwise.

Example

```
use prospect  
replace all company with proper(company);  
    for not isupper(company)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

KEY()

Class

Indexing

Purpose

Function to return the index key expression

Syntax

KEY(<expN>)

See Also

LEN(), FIELD(), FILTER(), DBF(), NDX(), FMT(), FCOUNT(), INDEXORDER(), INDEXEXT()

Description

The KEY() function is synonymous with the INDEXKEY() function. The KEY() function returns the index key expression for index <expN>, or a null string if no index file exists. When used in conjunction with the INDEXORDER() function, the KEY() function will return the index key expression of the master index. The KEY() function always returns a character string in lower case.

Example

```
use accounts index acc_no, date_paid
? key(1)
acc_no + dtos(date_rec)
set order to 2
? key(indexorder())
dtos(date_paid) + str(amo_paid,11,2)
index on lower(left(company,20)) to company
? key(1)
lower(left(company,20))
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

KEYMATCH()

Class

Indexing

Purpose

Function to determine whether a specified key value already exists

Syntax

KEYMATCH(<exp> [<index number>[,<workarea>]])

See Also

@...GET, CREATE, MODIFY STRUCTURE, DISLAY STATUS, LIST STATUS, INDEX ON, SET ORDER

Description

The KEYMATCH() function returns TRUE (.T.) if the specified key value <exp> exists in an open index. The KEYMATCH() function will search the current ndx (single index) file, or the controlling tag of a dbx (multiple index) file, unless the number of another index file is specified. The optional <index number> clause can contain the number of an ndx file in a list of open indexes, or the number of a tag in an open .dbx file. If you are specifying a tag in a dbx file other than the currently active dbx file, you must specify the dbx file name and then the tag number.

Index numbers can be listed using the DISPLAY | LIST STATUS command. The optional <workarea> clause is used to check indexes in other workareas. When using the <workarea> clause, the <index number> clause must also be used.

KEYMATCH() ignores the SET DELETED ON|OFF set command and always takes records marked for deletion into account when determining whether the key value exists. This ensures that subsequent recalling of deleted records does not compromise the integrity of unique index keys.

Example

```
m_val=part_no
if keymatch(m_val)
    dialog box "This part number already exists!"
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

L2BIN()

Class

Expressions and Type Conversion

Purpose

Function to convert numeric to binary encoded string

Syntax

L2BIN(<expN>)

See Also

CREATE BRIDGE, BIN2I, BIN2L, BIN2W, BINCLOSE(), BINCREATE(), BINOPEN(), BINREAD(), BINSEEK(), BINWRITE(), I2BIN

Description

The L2BIN() function converts a numeric value into a binary encoded character string formatted as a signed 32-bit integer. The <expN> is any numeric value. Any decimal places are ignored. This function can be used to construct composite index keys of mixed data types in OpenVMS RMS files. With the exception of the BIN2W() function, all binary conversion functions may be used in conjunction with the binary file functions.

Example

```
? bin2l(l2bin(98765))  
    98765
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LASTKEY()

Class

Keyboard Events

Purpose

Function to return ASCII code for last key pressed

Syntax

LASTKEY([<expN>])

See Also

SET PCKEYS, MESSAGE, READKEY, INKEY(), CTRL(), SET KEY, NEXTKEY()

Description

The LASTKEY() function returns the ASCII code for the last key which was pressed at the keyboard. If the optional <expN> is passed, the next LASTKEY() return value will be set to this value. This function is useful for returning the key pressed in response to the DIALOG MESSAGE command. The LASTKEY() function can also be used in conjunction with the CTRL() function for terminal independent programs. The CLEAR TYPEAHEAD commands clears the LASTKEY() function to zero.

When used in Recital Mirage in conjunction with INKEY(<expN>,"M"), LASTKEY() returns a numeric value indicating the last mouse operation as shown in the table below.

LASTKEY() Return Value	Mouse Operation
151	LEFTCLICK
152	RIGHTCLICK
153	DBLCLICK
154	CTRL+LEFTCLICK
155	CTRL+RIGHTCLICK
156	CTRL+DBLCLICK
157	SHIFT+LEFTCLICK
158	SHIFT+RIGHTCLICK
159	SHIFT+DBLCLICK
160	ALT+LEFTCLICK
161	ALT+RIGHTCLICK
162	ALT+DBLCLICK

Example

dialog message "Do you wish to continue? (Y or N)."

a = iif(lastkey()=asc("Y"),"program continuing", "program ending")

set message to "&a"

Products

Recital Mirage Server, Recital Terminal Developer

LASTREC()

Class

Fields And Records

Purpose

Function to return number of records in table

Syntax

LASTREC([<workarea | alias>])

See Also

RECSIZE(), DBF(), NDX(), HEADER(), FCOUNT(), INDEXKEY()

Description

The LASTREC() function returns a number representing the total number of records in the currently selected table. This number includes records that have been marked for deletion. If the optional <workarea | alias> is specified, then the function will operate in the required location.

Example

use accounts

? lastrec()

2500

? fcount()

18

declare aAccounts[reccount(),fcount()]

copy to array aAccounts for empty(paid_date)

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LEFT()

Class

String Data

Purpose

Function to extract substring from left

Syntax

LEFT(<expC>, <expN>)

See Also

SUBSTR(), RIGHT(), AT(), ATNEXT(), RAT(), STUFF(), STRTRAN(), LEN(), STREXTRACT()

Description

The LEFT() function extracts a substring from the left of the character expression <expC> of width <expN> characters wide. The LEFT() function can be used on character fields in index expressions to extract part of the field in the index key.

Example

```
? left("Christopher",5)
```

Chris

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LEN()

Class

Expressions and Type Conversion

Purpose

Function to return length of specified expression

Syntax

LEN(<exp>[,<expL>])

See Also

AT(), STREXTRACT(), SUBSTR(), LEFT(), RIGHT(), RAT(), STUFF(), STRTRAN()

Description

The LEN() function returns the output width of the specified expression <exp>. The expression can be of any data type, including memo fields and arrays. When checking the length of character expressions, the optional <expL> can be specified. If <expL> is specified and evaluates to .T. (true) , the LEN() function will return the length of the character expression after stripping any formatting options such as BOLD().

Example

? len(name)

20

? len(notepad)

213

? len("")

0

? len(bold("Hello"))

9

? len(bold("Hello"),.T.)

5

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LENNUM()

Class

Numeric Data

Purpose

Function to return the length of a numeric value

Syntax

LENNUM(<expN>)

See Also

INT(), ROUND()

Description

The LENNUM() function is used to return the length of a numeric value. Numeric values are a minimum length of 10. The decimal point and decimal places are included in the value returned.

Example

```
? lennum(123456789.123)
```

13

```
? lennum(879)
```

10

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LIKE()

Class

String Data

Purpose

Function to compare two strings for similarities

Syntax

LIKE(<expC1>, <expC2>)

See Also

SOUNDEX(), STRTRAN()

Description

The LIKE() function is used to compare two character strings for matching qualities. The <expC1> specifies the matching pattern to look for. The <expC2> specifies the string in which to look. The <expC1> accepts “wildcard” characters with which to construct the pattern. Wildcard characters may represent upper case or lower case characters, but specific characters are case sensitive. The Recital/4GL supports the following wildcard characters:

Characters	Description
?	Matches any one character
%	Synonymous with ?
*	Matches zero or more characters

If the LIKE() function is used in SQL mode (SET SQL ON or embedded EXEC SQL statements), the following wildcard characters are available:

Characters	Description
_	Matches any one character
%	Matches zero or more characters

The LIKE() function returns .T. if a matching pattern is found, or .F. otherwise.

Example

?like(“*Recital*”, “This is Recital”)

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LKSYS()

Class

Fields and Records

Purpose

Function to return lock status of current record

Syntax

LKSYS(<expN>)

See Also

ISLOCKED(), LOCKED(), FLOCK(), RLOCK(), LOCKR, LOCKF, UNLOCK

Description

The LKSYS() function returns information regarding the lock status of the current record. The parameter <expN> is used to specify what type of information is returned. The <expN> may be one of the following:

Value	Description
0	Current time if current record is locked
1	Current date if current record is locked
2	User name of person who has locked the record
3	Terminal name at which record is locked

The LYKSYS() function returns a null string ("") if the record is not locked.

Example

?lksys(0)

11:30:20

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LINENO()

Class

Error Handling and Debugging

Purpose

Function to return number of the next line

Syntax

LINENO()

See Also

SET DOHISTORY, SET DEBUG, SET ECHO, SET HISTORY TO, MESSAGE, SET HISTORY TO FILE, RESUME, SUSPEND, SET HISTORY, PROGRAM(), PROCLINE(), MESSAGE()

Description

The LINENO() function returns the number of the next command line to be executed in a program, procedure or UDF. This function can be used with ON ERROR routines to return the line number causing the error.

Example

```
m_lineno = lineno()
dialog box "The line is # &m_lineno."
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LOCK()

Class

Manual Locking

Purpose

Function to lock record

Syntax

LOCK([<workarea | alias>])

See Also

RLOCK(), FLOCK(), UNLOCK, LOCKR, LOCKF

Description

The LOCK() function attempts to lock the current record. If successful, it returns .T. and the record is locked. If the record is already locked by another user then it returns .F..

Note: Recital automatically performs file and record locking so, in most situations, this function is unnecessary. It is included for compatibility with programs written with other products. If the optional <workarea | alias> is specified, then the function will operate in that <workarea | alias>.

Example

```
do while not lock()
    set message to "Record is in use."
    sleep 2
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LOG()

Class

Numeric Data

Purpose

Function to return natural logarithm for a specified numeric expression

Syntax

LOG(<expN>)

See Also

LOG10(), SIN(), COS(), TAN(), EXP(), RTOD(), DTOR(), PI(), SET DECIMALS

Description

The LOG() function returns the natural logarithm of the given numeric expression <expN>. The SET DECIMALS command determines the number of decimal places returned. By default, SET DECIMALS is set to 2.

Example

set decimals to 5

? log(2.71828)

1.00000

? type("log(2.71828)")

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LOG10()

Class

Numeric Data

Purpose

Function to return base 10 logarithm for the specified numeric expression

Syntax

LOG10(<expN>)

See Also

LOG(), PAYMENT(), PMT(), FV(), PV(), CAGR(), PI(), RTOD(), DTOR(), EXP(), TAN(), COS(), SIN(), SET DECIMALS

Description

The LOG10(<expN>) function returns the base 10 log of the specified <expN>. The SET DECIMALS command determines the number of decimal places returned. By default, SET DECIMALS is set to 2.

Example

```
? log10(300)
2.48
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LOOKUP()

Class

Fields and Records

Purpose

Function to perform a cross-table lookup

Syntax

LOOKUP(<exp>, <key expression>, <workarea | alias>)

See Also

CREATE, MODIFY STRUCTURE, SET RELATION, RLOOKUP(), SEEK()

Description

The LOOKUP() function looks up the specified index <key expression> in the master index of specified <workarea | alias>. The LOOKUP() function then evaluates the expression <exp>, and returns the value of the <exp> if the <key expression> is found. The LOOKUP() function returns a null string if the <key expression> is not found. The lookup function can be very useful for dealing with calculated expressions in either the Application Data Dictionary or the Forms Designer.

Example

use depts in 2 index dnum

use funcs in 1 index funcno

? lookup(depts->dname,funcs->dnum,depts)

Research and Development

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LOWER()

Class

String Data

Purpose

Function to convert characters to lower case

Syntax

LOWER(<expC>)

See Also

UPPER(), ISLOWER(), ISUPPER(), TRANSFORM(), STRTRAN(), PROPER()

Description

The LOWER() function converts the characters in the character expression <expC> to lower case. This function is particularly useful if used in index keys and their corresponding SEEK expressions, since a search expression in lower case will match both upper and lower case characters.

Example

```
? lower("Recital")
```

recital

```
// Another Example
```

```
use accounts
```

```
index on lower(company) to company
```

```
replace company with "Company Name"
```

```
seek "company name"
```

```
? found()
```

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LPAD()

Class

String Data

Purpose

Function to pad out a character string to a defined length from the left

Syntax

LPAD(<expC1>,<expN>,[<expC2>])

See Also

STRZERO(), STR(), RPAD(), STRTRAN(), STUFF(), SUBSTR()

Description

The LPAD() function right justifies a character string <expC1> and pads out the left of the string, to a total length of <expN> characters, with blank spaces. The optional <expC2> may be used to pad <expC1> with a character string instead of blank spaces. The LPAD() function is useful for creating fixed length character strings by padding out the string with blanks. If the string being processed is longer than the specified total length, the string is truncated.

Example

```
? lpad("abc",6,"d")
```

dddabc

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LTOS()

Class

Expressions and Type Conversion

Purpose

Function to convert logical to string

Syntax

LTOS(<expL>)

See Also

DTOS(), STR(), STRZERO(), DTOC()

Description

The LTOS() function returns the logical expression <expL> as a character string “T” or “F”. It is particularly useful for building indexes on mixed data types.

Example

index on ltos(married)+upper(name) to married

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LTRIM()

Class

String Data

Purpose

Function to trim spaces from left

Syntax

LTRIM(<expC>)

See Also

TRIM(), RPAD(), RTRIM(), ALTRIM(), STRTRAN()

Description

The LTRIM() function removes leading spaces from the character expression <expC>.

Example

```
? ltrim("  Recital")
```

Recital

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

LUPDATE()

Class

Disk and File Utilities

Purpose

Function to return date of last table update

Syntax

LUPDATE([<workarea | alias>])

See Also

HEADER(), RECCOUNT(), RECSIZE(), DIR, ADIR()

Description

The LUPDATE() function returns the last date on which the table in the currently selected workarea was modified. If the optional <workarea | alias> is specified, then the function will operate in the required location. The LUPDATE() function will not return a new date unless the table has been closed since the last update.

Example

```
use patrons
```

```
? lupdate()
```

```
02/02/2000
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAIL()

Class

Disk and File Utilities

Purpose

Function to return the last mail message

Syntax

MAIL([<expN>])

See Also

SET MAIL, ON MAIL

Description

The MAIL() function returns the last mail message received at the current terminal. The optional parameter <expN> is a numeric expression that must equal 1. This option causes the MAIL() function to return the sender name. The SET MAIL command must be ON in order to trap mail messages. When SET MAIL is OFF, mail messages are ignored.

Example

```
// Trap mail messages
set mail on
on mail do mail_proc with mail(), mail(1)
use demo
browse
on mail
```

Products

Recital Terminal Developer (OpenVMS only)

MAILCLOSE()

Class

Mail

Purpose

Function to disconnect from mail server

Syntax

MAILCLOSE()

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILCLOSE() function will disconnect you from the mail server. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() (POP3 only) or MAILNODENAME() function to check if you are connected.

If successful the MAILCLOSE() will return .T., otherwise .F.. The MAILERROR() function can be used to return the error message if the MAILCLOSE() fails.

Example

```
mailclose()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILCOUNT()

Class

Mail

Purpose

Function to return the number of messages on the currently connected mail server

Syntax

MAILCOUNT([<array>])

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILCOUNT() function returns the number of messages on the connected mail server. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() (POP3 only) or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<array>	No	None	The name of an array to contain the size in bytes of each mail message. No pre-declaration is required.

Example

```
value = mailcount(message_size)
```

```
? value
```

```
6
```

```
? message_size[1]
```

```
365
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILDELETE()

Class

Mail

Purpose

Function to delete the specified mail message

Syntax

MAILDELETE(<expN>)

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILDELETE() function will delete the mail message specified by number. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() (POP3 only) or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<expN>	Yes	None	Numeric value to specify the mail message number to be deleted.

Example

```
maildelete(1)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILERROR()

Class

Mail

Purpose

Function to return last mail error

Syntax

MAILERROR()

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILERROR() function will return the last error message if there was one, or an empty string if no error occurred.

Example

```
? mailopen("mailserver.company.com","username","password")
```

.F.

```
? mailerror()
```

Host name 'mailserver.company.com' not found.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILHEADER()

Class

Mail

Purpose

Function to return specified header information

Syntax

MAILHEADER(<array1>, <expC1>)

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILHEADER() function will return the specified header information from an array that was already created by MAILREAD(). If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<array1>	Yes	None	The name of an array to contain the mail messages read by MAILREAD()
<expC2>	Yes	None	Specify the header information to return from the mail message. This can be any one of the following options: "RECEIVED" "RETURN-PATH" "MESSAGE-ID" "FROM" "TO" "SUBJECT" "DATE" "STATUS"

The MAILERROR() function can be used to return the error message if the MAILHEADER() fails.

Example

```
number = mailread(1, "mail_mes")
```

```
? number
```

```
30
```

```
? mailheader(mail_mes, "FROM")
```

```
sales@recital.com
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILNODENAME()

Class

Mail

Purpose

Function to return the node name of the currently connected mail server

Syntax

MAILNODENAME()

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILNODENAME() function will return the node name of the currently connected mail server, or an empty string if there is no connection.

Example

```
? mailopen("mailserver.company.com","username","password")
```

```
.T.
```

```
? mailnodename()
```

```
mailserver.company.com
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILOPEN()

Class

Mail

Purpose

Function to connect to the specified mail server

Syntax

MAILOPEN(<expC1>, <expC2>, <expC3> [, <expC4>])

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILOPEN() function is used to connect to the specified mail server. The mail type POP3 should be used for reading mail, and SMTP used for sending mail.

Parameters	Required	Default	Description
<expC1>	Yes	None	The node name or IP address of the mail server
<expC2>	Yes	None	The user name with which to connect to the mail server
<expC3>	Yes	None	The password for the user name
<expC4>	No	POP3	The mail protocol type. POP3 and SMTP are the currently supported protocols.

If successful the MAILOPEN() will return .T., otherwise .F.. The MAILERROR() function can be used to return the error message if the MAILOPEN() fails.

Example

```
// Open POP3 for reading mail
m_open = mailopen("mailserver.company.com", "username", "password")
if not m_open
    dialog box mailerror()
endif
mailclose()

// Open SMTP for sending
m_open = mailopen("mailserver.company.com", "username", "password", "SMTP")
if not m_open
    dialog box mailerror()
endif
mailclose()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILREAD()

Class

Mail

Purpose

Function to read the specified mail message

Syntax

MAILREAD(<expN>, <expC1>, [,<expC2>])

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILREAD() function will read the mail message specified by the number <message>, returning the number of elements in the array that it creates. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<expN>	Yes	None	Numeric value specifying the mail message to read
<expC1>	Yes	None	The name of the array to be created to contain the lines of the mail message
<expC2>	No	“ALL”	The text that will be read from the mail message. This can be any one of the following: “ALL” “HEADER” “BODY” “ATTACHMENT”

The MAILERROR() function can be used to return the error message if the MAILREAD() fails.

Example

```
number = mailread(1, “mail_mes”, “BODY”)
```

```
? number
```

```
30
```

```
? mail_mes[30]
```

This is the last line in the mail message.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILSEND()

Class

Mail

Purpose

Function to send mail

Syntax

MAILSEND([<expC1>,<expC2>, [<expC3>], [<expC4>], <expC5> [,<expC6> [, <expL1>]]
[, <expC7>])

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(),
MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(),
MAILUSERNAME(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILSEND() function is used for sending mail. If successful the MAILSEND() will return .T. or .F. otherwise. The MAILERROR() function can be used to return the error message if the MAILSEND() fails.

Parameters	Required	Default	Description
<expC1>	No	Your node	The from name of the sender
<expC2>	Yes	None	A semi-colon separated list of recipient names to receive the message
<expC3>	Yes	None	A semi-colon separated list of cc recipient names to receive the message
<expC4>	No	None	The subject of the message
<expC5>	Yes	None	The message to be sent. This can be a character string, a variable containing a character string or a character string containing the name of a text file.
<expC6>	No	None	A comma separated list of names of files to be sent as attachments
<expL1>	No	None	Open VMS only. If <expL1> is .T. (true), any attachments will be sent as ASCII, not binary. If <expL1> is .F. or omitted, all attachments will be sent as binary 64 bit encoded.
<expC7>	No	text/plain	Content type.

Example

```
// Open SMTP for sending
m_open = mailopen("mailserver.company.com","username","password","SMTP")
if not m_open
    dialog box mailerror()
    return
endif
```

```
fromname = "info@recital.com"
tonames = "fred@recital.com;sue@recital.com"
ccnames = "bert@recital.co.uk;linda@recital.co.uk"
subject = "For your information"
message = "Dear All" + chr(10) + "Here are the files you asked for." + chr(10) + ;
    "Best regards" + chr(10) + "Sam"
attachments = "info.doc, info.xls"
mailsend(fromname,tonames,ccnames,subject,message,attachments)
mailclose()
```

```
// Open SMTP for sending
m_open = mailopen("mailserver.company.com","username","password","SMTP")
if not m_open
    dialog box mailerror()
    return
endif
```

```
fromname = "info@recital.com"
tonames = "fred@recital.com"
ccnames = ""
subject = "For your information"
message = "email.htm"
attachments = ""
content = "test/html"
sendmail(fromname,tonames,ccnames,subject,message,attachments,.F.,content)
mailclose()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAILUSERNAME()

Class

Mail

Purpose

Function to return current user name

Syntax

MAILUSERNAME()

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), OPENMAIL(), READMAIL(), SENDMAIL()

Description

The MAILUSERNAME() function will return the name of the user who is currently connected to the mail server, or an empty string if there is no connection. This function only operates with POP3 connections.

Example

```
? mailopen("mailserver.company.com","username","password")
```

```
.T.
```

```
? mailusername()
```

```
username
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAX()

Class

Numeric Data

Purpose

Function to return maximum value function

Syntax

MAX(<exp1>,<exp2>)

See Also

MIN(), IIF(), AMAX(), AMIN(), MOD()

Description

The MAX() function returns the greater of the two expressions <exp1> and <exp2>. These expressions can be of numeric or date data types.

Example

```
? max(1, 3)
```

3

```
set date british
```

```
? max(ctod("15/10/1997"), ctod("14/10/1997"))
```

15/10/1997

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MAXCOL()

Class

Screen Forms

Purpose

Function to return the highest addressable column number

Syntax

MAXCOL()

See Also

ROW(), PROW(), PCOL(), COL(), MAXROW(), SCOLS(), SROWS()

Description

The MAXCOL() function returns the highest addressable column number for the present terminal. The return value is of numeric data type. Addressable column numbers range from 0 to 79.

Example

```
?maxcol()
```

```
79
```

Products

Recital Mirage Server, Recital Terminal Developer

MAXROW()

Class

Screen Forms

Purpose

Function to return the highest addressable row number

Syntax

MAXROW()

See Also

ROW(), PROW(), PCOL(), COL(), MAXCOL(), SCOLS(), SROWS()

Description

The MAXROW() function returns the highest addressable row number for the present terminal. The return value is numeric data type. The number of addressable rows is dependent on the setting of item 220 in the current Terminal Definition File. For terminals that support access to 25 lines, rows are addressable from 0 to 24. For terminals that support 24 lines, rows are addressable from 0 to 23, and MAXROW() will return 23.

Example

```
?maxrow()
```

24

Products

Recital Mirage Server, Recital Terminal Developer

MAXVALUES()

Class

Fields and Records

Purpose

Function to return the maximum numeric value for a matching series of keys

Syntax

MAXVALUES(<expN> [, <for condition>[, <key expression >]])

See Also

CNTVALUES(), SUMVALUES(), AVGVALUES(), MINVALUES(), SQLVALUES()

Description

The MAXVALUES() function returns the maximum value of the <expN> for a matching series of keys. The required <expN> parameter must be a column name from a table. If none of the optional parameters is specified, then the maximum value of keys matching the current key is returned. An optional <for condition> can be specified to restrict the rows included in the operation. You can also optionally specify a <key expression> to return the maximum value from instead of the current key.

After completion, all record pointers and indexes are returned to their original positions.

Example

```
use customer order title
// Maximum value of balance column for the number of records matching the current key
total_balance = maxvalues(balance)
// Maximum value of balance column for the number of records matching the key "Mr"
m_male = maxvalues(balance, .T., "Mr")
// Maximum value of balance column for the number of records matching the current key with an expiry
date < today
m_expired = maxvalues(balance, expiry<date())
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MCOL()

Class

Screen Forms

Purpose

Return the current mouse column position

Syntax

MCOL()

See Also

ON MOUSE, INKEY(), LASTKEY(), MROW()

Description

The MCOL() function returns the numeric value of the current mouse column position.

In Recital Terminal Developer, the MCOL() function always returns 0.

Example

```
procedure on_mouse
m_colpos = mcol()
m_rowpos = mrow()
m_mouseop = lastkey()
// process based on position and operation
return
```

```
on mouse do on_mouse
inkey(0,"M")
```

Products

Recital Mirage Server, Recital Terminal Developer

MDX()

Class

Indexing

Purpose

Function to return the active multiple index filename

Syntax

MDX([<expN>[,<alias>]])

See Also

FOR(), KEY(), NDX(), ORDER(), SET(), TAG(), TAGCOUNT(), TAGNO(), SET COMPATIBLE, SET FILECASE, SET FULLPATH

Description

The MDX() function returns the name of the currently open multiple index file. Used without any parameters, the MDX() function will return the name of the multiple index file in the current workarea. If there is no .dbx file in the workarea, the MDX() function will return a null string. The optional numeric expression, <expN>, may be used to return the .dbx file name for a specific tag number. This parameter is needed if more than one .dbx file is open in a workarea. You may optionally specify an alias name to use the MDX() function in other workareas. The <alias> may be a workarea number or letter (1-DB_MAXWKA, a-z) or the table alias name.

If SET COMPATIBLE is set to FOXPRO or VFP the MDX() return value format differs in the following way: if SET FULLPATH is ON the full path to the index file is returned, if SET FILECASE is OFF then the return value is converted to upper case (Windows only).

Example

?mdx(1, account)

invoice.dbx

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MDY()

Class

Date and Time Data

Purpose

Function to return a date as a character string

Syntax

MDY(<expD>)

See Also

DMY(), CDOW(), CTOD(), CMONTH(), DOW(), DTOC(), DAY(), MONTH(), TIME(), WEEK(), YEAR(), STOD(), DTOS(), AMPM(), SET CENTURY, SET DATE, VTOD(), DTOV(), SET EPOCH, EPOCH()

Description

The MDY() function returns a character string of the form of “Month name Day, Year” from the specified <expD>. If CENTURY is OFF, then a 2-digit year is returned.

Example

```
set century on
? mdy({04/04/1990})
April 4, 1990
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MEMLINES()

Class

Memos

Purpose

Function to return lines needed to display a memo field

Syntax

MEMLINES(<memofield>)

See Also

SET WP TO, SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MLINE(), MEMOEDIT(), TEXTEDIT(), MLCOUNT(), MEMOLINE(), MEMOREAD(), MEMOTRAN(), HARDCLR()

Description

The MEMLINES() function returns the number of lines needed to display the <memofield> as specified by the current MEMOWIDTH. The MEMOWIDTH can be reset with the command SET MEMOWIDTH.

Example

```
use prospect
set memowidth to 40
m_last = memlines(comp_hist)
declare lines[m->m_last]
for n=1 to m->m_last
    lines[m->n] = mline(comp_hist,m->n)
next
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MEMOEDIT()

Class

Memos

Purpose

Function to edit a memo field or character string in a pop-up window

Syntax

MEMOEDIT(<memofield | expC1>, [<expN1>, <expN2>, <expN3>, <expN4> [,<expL>, [<expC2>]]])

See Also

SET MEMOWINDOW, SET MEMOWIDTH, SET MEMOFORMAT, SET CLEAR, MEMOWRITE(), MEMLINES(), MLCOUNT(), MLINE(), TEXTEDIT(), MEMOLINE(), MEMOREAD(), MEMOTRAN(), HARDCR()

Description

The MEMOEDIT() function allows the <memo field> or character expression <expC1> to be edited in a 'pop-up' window. If a <memofield> is used with MEMOEDIT(), the function will return .T. if the memo was modified and .F. otherwise. If <expC1> is used in MEMOEDIT(), the function returns the character string.

Coordinates <row>, <col> to <endrow>, <endcol>, may be optionally specified by <expN1> to <expN4>. If no screen coordinates are specified, MEMOEDIT() will default to the current SET MEMOWINDOW specifications.

If the optional update parameter <expL> is .T., the memo can be modified. If <expL> is .F., the memo can only be viewed (e.g. in help messages). Record locks are automatically applied if UPDATE is ON.

The optional expression <expC2> is used to border the pop-up window and label it with the character expression.

Once a pop-up window has been activated, pressing the [HELP] key displays a menu of memo editing keys. These keys include facilities for reading from and writing to external files, and printing on the system printer. In UNIX/Linux, you may pop-up a command window by pressing the [MENUBAR] key and the letter {C} simultaneously. Operating system commands may be typed into the window, the results of which are read into the memo. You may stuff the keyboard with operating system commands by enclosing the command in single quotes.

The functions MEMOREAD() and MEMOWRITE() are used for reading text files to and from memo fields. The MEMOLINE() and MLINE() memo editing functions are used to extract a line of text from a memo field. Reading text files into memo fields is strongly recommended for enabling users to edit text files from within applications.

Text files can be edited directly within the same type of window as MEMOEDIT() by using the TEXTEDIT() function. Please note that memo-editing functions will not work with text files, and text-editing functions will not work with memo fields. An external Word Processor can be defined for the editing of memo fields when the [FIELD EDIT] key is pressed in forms (see the SET WP command for more information).

The SET MEMOFORMAT ON | OFF command is active with MEMOEDIT(), determining whether memo text is formatted during and after editing operations. If SET CLIPPER is ON then MEMOEDIT() provides full syntax capability with Clipper.

Example

memoedit(notes, 5, 10, 20, 70, .T., "Notepad")

Products

Recital Terminal Developer

MEMOLINE()

Class

Memos

Purpose

Function to extract a line of text from a memo field

Syntax

MEMOLINE(<memofield>, <expN1>, <expN2>[,<expN3>,<expL>])

See Also

SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MEMLINES(), MLCOUNT(), MLINE(), TEXTEDIT(), MEMOEDIT() MEMOREAD(), MEMOTRAN(), HARDCLR()

Description

The MEMOLINE() function extracts a formatted line of text from the specified <memofield>. The <expN1> is the number of characters per line and <expN2> is the line number to extract. The <expN3> is included for compatibility with the Clipper tab size setting, but its value is ignored. The <expL> determines whether word wrapping is on or off. If <expL> is .T. (True), and the line length indicated by <expN1> falls in the middle of a word, that word will not be returned until the next line is extracted. If <expL> is .F. (False), the number of characters specified by <expN1> is returned. The default setting of <expL> is .T. (True).

The length of the line extracted is not affected by the SET MEMOWIDTH command.

Example

```
use prospect
set scroll to 4,15
@4,0 say ""
m_last = memlines(comp_hist)
for n=1 to m->m_last
    m_line = memoline(comp_hist,40,m->n)
    ? m_line
next
set scroll to default
clear
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MEMOREAD()

Class

Memos

Purpose

Function to read an external file into a memo field

Syntax

MEMOREAD(<filename>, [<memofield>], [<expL>])

See Also

SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MEMLINES(), MLINE(), MEMOEDIT(), TEXTEDIT(), MLCOUNT(), MEMOLINE(), MEMOTRAN(), HARDCR()

Description

The MEMOREAD() function reads an external text file <filename> into a character string. If the optional <memofield> is specified then the contents of the text file are read into the MEMO field. The MEMOREAD() function returns .T. if the operation was successful. There is no limit to the size of the text file to be read into the memo field.

Setting <expL> to a value of TRUE will cause return characters to be retained during the MEMOREAD() even if MEMOFORMAT is ON. Setting the <expL> value to .F. will format the memo during the MEMOREAD() operation, stripping carriage return characters during input. The default setting for the <expL> parameter is .F..

Example

```
// To transfer company history notes
use prospect
memowrite("tempname.txt",comp_hist)
seek "00234"
if found()
    memoread("tempname.txt",comp_hist)
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MEMORY()

Class

Environment

Purpose

Function to return the amount of memory available

Syntax

MEMORY(<expN>)

See Also

DISKSPACE(), OS()

Description

The MEMORY() function will return the amount of available to the user on the system. On OpenVMS systems, it will always return a value of 100000.

Example

```
? memory(0)  
388608
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MEMOSAY()

Class

Memos

Purpose

Function to display contents of a memo field

Syntax

MEMOSAY(<memofield> | <expC>,[<expN1>, <expN2>, <expN3>, <expN4> [, <expC1>]])

See Also

SET PRERECORD, SET MEMOCLEAR, SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MEMLINES(), MLINE(), MEMOEDIT(), TEXTEDIT(), MLCOUNT(), MEMOLINE(), MEMOTRAN(), HARDCLR(), MEMOREAD()

Description

The MEMOSAY() function displays the contents of the <memofield> or <expC> in a window on the screen. The output is displayed in reverse video to interface consistently through forms. Coordinates <row>, <col> to <endrow>, <endcol>, may be optionally specified by <expN1> to <expN4>. If no screen coordinates are specified, MEMOSAY() will default to the current SET MEMOWINDOW coordinates. The optional <expC1> contains a label name for the memo if SET BORDER is set to SINGLE or DOUBLE. The MEMOSAY() functions returns .T. if successful and .F. otherwise.

Example

```
procedure saymemo
memosay(details,2,41,10,79,"Customer Details")
return
```

```
use accounts
set form to details
set prerecord to saymemo
edit
```

Products

Recital Terminal Developer

MEMOTRAN()

Class

Memos

Purpose

Function to translate carriage returns

Syntax

MEMOTRAN(<memofield> | <expC1> [,<expC2> [,<expC3>]])

See Also

SET MEMOWINDOW, SET MEMOWIDTH, HARDCLR(), MEMOWRITE() MEMOREAD(), MEMOEDIT(), MLINE(), TEXTEDIT(), MLCOUNT(), MEMOLINE(), MEMLINES()

Description

The MEMOTRAN() function translates all hard carriage returns (ASCII 13), within the <memofield> or <expC1>, into the character defined by <expC2>. If <expC2> is not specified, then it defaults to a semicolon. MEMOTRAN() also translates all line feed/soft carriage returns (ASCII 10), within the <memofield> or <expC1>, into the character defined by <expC3>. If <expC3> is not specified, then it defaults to a space.

Example

```
use accounts
m_var = memotran(product_des)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MEMOWRITE()

Class

Memos

Purpose

Function to write to an external file from a memo field

Syntax

MEMOWRITE(<filename>, <memofield> | <expC>)

See Also

SET MEMOWINDOW, SET MEMOWIDTH, MEMOREAD(), MEMLINES(), MLINE(), MEMOEDIT(), TEXTEDIT(), MLCOUNT(), MEMOLINE() MEMOTRAN(), HARDCLR()

Description

The MEMOWRITE() function writes an external text file <filename> from the specified field or character string. The <expC> is a character expression that returns a valid memo field name. MEMOWRITE() returns .T. if the operation was successful.

Example

```
// To transfer company history notes  
use prospect  
memowrite(tmpnam() + ".txt", comp_hist)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MENU()

Class

Menus

Purpose

Function to return selected menu number

Syntax

MENU()

See Also

@...MENU, MENU AT, MENU COMMAND, MENU FIELDS, MENU FILES, MENU FORMAT, MENU FRAME, MENU QUERY, MENU SCOPE, MENUITEM(), MENU BROWSE, MENU FROM

Description

The MENU() function returns the number of the menu item selected from a previously activated menu. This function is valid on any of the Recital 4GL menu commands and dBASE style menu commands. If no item was selected, MENU() returns -1. Menu items start at 0. This function is very useful when used in conjunction with Recital PULLDOWN menus. When used with the MENU BROWSE command, MENU() returns the record number of the selected record rather than its position in the menu. If COMPATIBLE is set ON, then the MENU() function will return the alphanumeric string for the name of the most recent menu that was activated. If there is no active menu, this function returns a null string. This is compatible with the Xbase style menus.

Example

```
menu browse customer, lastname  
? menu()
```

Products

Recital Mirage Server, Recital Terminal Developer

MENUITEM()

Class

Menus

Purpose

Function to return selected menu item

Syntax

MENUITEM()

See Also

MENU (), @...MENU, MENU, SET KEY, MENU QUERY, MENU AT, MENU FIELDS, MENU FILES, MENU SCOPE, MENU BROWSE, MENU FROM

Description

The MENUITEM() function returns the selected menu item from a previously activated menu as a character string. This function is valid on any of the Recital 4GL menu commands. The MENUITEM() returns the portion of the selected menu item that was displayed in the menu. If no item was selected, MENUITEM() returns a null string.

Example

```
// UDF to set a filter
function setfilter
save screen
menu query command ">"
m_filter = menuitem()
set filter to &m_filter
restore screen
return .T.
```

Products

Recital Mirage Server, Recital Terminal Developer

MESSAGE()

Class

Error Handling and Debugging

Purpose

Function to return last error encountered message

Syntax

MESSAGE([<expN>])

See Also

ERROR(), ERRNO(), ON ERROR

Description

The MESSAGE() function returns a character string describing the last error encountered. The optional parameter <expN> must result in 1, and when used will cause MESSAGE() to return the last program line which caused an error.

Example

```
set exclusive off
on error *
use payroll
if not used()
    msg=message()
    dialog box "&msg.. Press any key to continue."
endif
set exclusive on
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MESSAGEBOX()

Class

Dialogs

Purpose

Function to display a dialog box with user-defined elements

Syntax

MESSAGEBOX(<expC1> [, <expN> [, <expC2>]])

See Also

DIALOG BOX, DIALOG MESSAGE

Description

The MESSAGEBOX() function is used to display a dialog box with user-defined elements. The message, title, icon and buttons can all be defined.

Parameters	Required	Default	Description
<expC1>	Yes	None	The message to display in the message box.
<expN>	No	0	A number representing the buttons and icon to include in the message box and the specification of the default button. See the tables below for details.
<expC2>	No	Recital	The title of the message box.

Buttons

Value	Buttons
0	Ok
1	Ok and Cancel
2	Abort, Retry and Ignore
3	Yes, No and Cancel
4	Yes and No
5	Retry and Cancel

Icons (The Icon is ignored in products other than Recital Mirage)

Value	Icon
16	Stop road sign
32	Question mark
48	Exclamation mark
64	Information sign

Default

Value	Buttons
0	First button
256	Second button
512	Third button

The <expN> is the sum of the values to be specified, one from each table. For example, if <expN> is 292, the message box will have Yes and No buttons (4), a Question mark icon (32) and the second button (No) will be the default (256).

The MESSAGEBOX() function returns a number signifying the selected button:

Return Value	Buttons
1	Ok
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

For message boxes that include the Cancel button, pressing the Escape key will also cause the MESSAGEBOX() function to return 2.

Example

```
nReturn = messagebox("Are you sure?", 36, "Continue to next screen")
if nReturn = 6
    // user selected 'Yes', process accordingly
else
    // user selected 'No', process accordingly
endif
```

Products

Recital Mirage Server, Recital Terminal Developer

MIN()

Class

Numeric Data

Purpose

Function to return the minimum value

Syntax

MIN(<exp1>, <exp2>)

See Also

MAX(), IIF(), AMIN(), AMAX(), MOD()

Description

The MIN() function returns the lesser of two expressions <exp1> and <exp2>. These expressions can be of numeric or date data types.

Example

```
? min(1, 3)
```

1

```
? min(ctod("01/02/1997"), date())
```

01/02/1997

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MINUTE

Class

Date and Time Data

Purpose

Function to return the numeric minutes from a specified datetime

Syntax

MINUTE(<expT>)

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), HOURS(), MINUTES(), SEC(), SECONDS(), SECS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The MINUTE() function returns the minutes from the specified datetime expression <expT> as a numeric value.

Example

```
? minute({ 10/10/2004 10:15:43 AM})
```

15

```
m_Min = minute(datetime())
```

```
? type("m_Min")
```

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MINUTES()

Class

Date and Time Data

Purpose

Function to extract minutes from a specified time string

Syntax

MINUTES(<time-string>)

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), HOURS(), MINUTE(), SEC(), SECONDS(), SECS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The MINUTES() function extracts the number of minutes from a specified time string and returns it as a numeric value.

Example

? minutes("10:21:00")

21

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MINVALUES()

Class

Fields and Records

Purpose

Function to returns the minimum numeric value for a matching series of keys

Syntax

MINVALUES(<expN> [, <for condition> [, <key expr>]])

See Also

CNTVALUES(), SUMVALUES(), AVGVALUES(), MAXVALUES(), SQLVALUES()

Description

The MINVALUES() function returns the minimum value of <expN> for a matching series of keys. The required <expN> parameter must be a column name from a table. If none of the optional parameters are specified, then the minimum value of keys matching the current key is returned. An optional <for condition> can be specified to restrict the rows included in the operation. You can also optionally specify a <key expression> to be used instead of the current key.

After completion, all record pointers and indexes are returned to their original positions.

Example

use customer order title

// Minimum value of balance column for the number of records matching the current key

total_balance = minvalues(balance)

// Minimum value of balance column for the number of records matching the key "Mr"

m_male = minvalues(balance, .T., "Mr")

// Minimum value of balance column for the number of records matching the current key with an expiry date < today

m_expired = minvalues(balance, expiry<date())

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MLCOUNT()

Class

Memos

Purpose

Function to return the number of word-wrapped lines in a memo field

Syntax

MLCOUNT(<memofield>, <expN>)

See Also

SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MEMLINES(), MLINE(), MEMOEDIT(), TEXTEDIT(), MEMOLINE(), MEMOREAD(), MEMOTRAN(), HARDCLR()

Description

The MLCOUNT() function will return the number of word-wrapped lines in the specified <memofield> field where <expN> is the number of characters per line. The SET MEMOWIDTH command does not affect the length of word wrapping in the memo line count.

Example

```
use prospect
m_last = mlcount(comp_hist, 40)
declare lines[m->m_last]
for n=1 to m->m_last
    lines[m->n] = mline(comp_hist, m->n)
next
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MLINE()

Class

Memos

Purpose

Function to extract a line of text from a memo

Syntax

MLINE(<memofield>,<expN>)

See Also

SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MEMOEDIT(), TEXTEDIT(),
MEMLINES(), MLCOUNT(), MEMOLINE(), MEMOREAD(), MEMOTRAN(), HARDCLR()

Description

The MLINE() function extracts a line of text from the specified <memofield>. The <expN> specifies the line number in the range 1 to MEMLINES(). The length of the line extracted is set by the command SET MEMOWIDTH.

Example

```
use prospect
set scroll to 4,15
@4,0 say " "
set memowidth to 40
m_last = memlines(comp_hist)
for n=1 to m_last
    m_line = mline(comp_hist,n)
    ? m_line
next
set scroll to default
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MOD()

Class

Numeric Data

Purpose

Function to return division remainder

Syntax

MOD(<expN1>,<expN2>)

See Also

IIF(), MAX(), MIN()

Description

The MOD() returns the remainder after <expN1> has been divided by <expN2>.

Example

```
? mod(7,3)  
1.00
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MONTH()

Class

Date and Time Data

Purpose

Function to return month from a specified date or datetime

Syntax

MONTH(<expD> | <expT>)

See Also

CROW(), CMONTH(), CTOD(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), EPOCH(), GOMONTH(), MDY(), MTOD(), QUARTER(), STOD(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK

Description

The MONTH() function returns the numeric month of the year from the given date expression <expD> or datetime expression <expT>.

Example

? month(date())

10

? month(datetime())

10

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MROW()

Class

Screen Forms

Purpose

Return the current mouse row position

Syntax

MROW()

See Also

ON MOUSE, INKEY(), LASTKEY(), MCOL()

Description

The MROW() function returns the numeric value of the current mouse row position.

In Recital Terminal Developer, the MROW() function always returns 0.

Example

```
procedure on_mouse
m_colpos = mcol()
m_rowpos = mrow()
m_mouseop = lastkey()
// process based on position and operation
return
```

```
on mouse do on_mouse
inkey(0,"M")
```

Products

Recital Mirage Server, Recital Terminal Developer

MTOD()

Class

Expressions and Type Conversion

Purpose

Function to convert a date string in the format DD-Mmm-YYYY to a date data type

Syntax

MTOD(<expC>)

See Also

AMPM(), CDOW(), CTOD(), CMONTH(), DATE(), DAY(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), MDY(), MONTH(), SET CENTURY, SET DATE, SET MARK, STOD(), TIME(), VTOD(), YEAR()

Description

The MTOd() function is used to convert date strings in the format DD-Mmm-YYYY to date data types. The returned date will be in the format determined by SET CENTURY, SET DATE and SET MARK.

If <expC> is not a valid date string, MTOd() will return an empty date. The DD-Mmm-YYYY format is the calendar date format in Recital Internet Developer and Recital Mirage.

Example

```
set century on
? dtom("10-Oct-2000")
10/10/2000
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

MTOS()

Class

Expressions and Type Conversion

Purpose

Function to convert a memo field to character data type

Syntax

MTOS(<memofield>)

See Also

SET MEMOFORMAT, SET WP, SET MEMOWINDOW, SET MEMOWIDTH, MEMOEDIT(), MEMOSAY(), MEMLINES(), MEMOWRITE(), MLCOUNT(), MLINE(), TEXTEDIT(), MEMOLINE(), MEMOREAD(), MEMOTRAN(), HARDCR(), SET PRERECORD TO, SET MEMOCLEAR (), STRTRAN(), SUBSTR()

Description

The MTOS() function converts the memo field <memofield> to a character data type. This function is particularly useful for modifying fields globally. If the specified <memofield> is empty, the MTOS() function returns a null string ("").

Example

```
set memoformat off
string = mtos(notes)
string = string + chr(10) + "AT" + date() + " " + time()
replace notes with string
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NDX()

Class

Indexing

Purpose

Function to return index file name

Syntax

NDX(<expN>)

See Also

DBF(), FMT(), INDEXKEY(), INDEXORDER(), SET COMPATIBLE, SET FILECASE, SET FULLPATH

Description

The NDX() function returns, in lower case, the name of the index file <expN> for the table in the currently selected workarea. If the table is not indexed, then NDX() returns a null string. When used in conjunction with the INDEXORDER() function, the NDX() function will return the index file name of the master index.

If SET COMPATIBLE is set to FOXPRO or VFP the NDX() return value format differs in the following way: if SET FULLPATH is ON the full path to the index file is returned, if SET FILECASE is OFF then the return value is converted to upper case (Windows only).

Example

use accounts index acc_no,date_paid

? ndx(1)

acc_no.ndx

set order to 2

? ndx(indexorder())

date_paid.ndx

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NETERR()

Class

Error Handling and Debugging

Purpose

Function to return .T. if a table could not be shared

Syntax

NETERR()

See Also

ON ERROR, USE, SET EXCLUSIVE, SET DCACHE, SET CACHELOAD, USED()

Description

The NETERR() function will return .T. if a table could not be shared. This function must be used in conjunction with the ON ERROR command.

Example

```
procedure badfile
if neterr()
set message to "File not available now."
endif
return
on error do badfile
use patrons
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NETNAME()

Class

Disk and File Utilities

Purpose

Function to return system information

Syntax

NETNAME()

See Also

OS(), VERSION()

Description

The NETNAME() function returns the name of the machine. On UNIX/Linux systems, full 'uname -a' information is returned.

Example

? netname()

system=SCO_SV nodename=mynode version=3.2.2 machine=i386

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NEWOBJECT()

Class
Objects

Purpose

Function to create a new class or object from a .vcx visual class library or program.

Syntax

NEWOBJECT(<expC1> [,<expC2> [,<expC3> [,<exp1>,<exp2>, ...]])

See Also

DEFINE CLASS...ENDDEFINE, WITH...ENDWITH, ADDPROPERTY(), CREATEOBJECT(),
DODEFAULT(), REMOVEPROPERTY()

Description

The NEWOBJECT() function is used to create a new class or object from a .vcx visual class library or program. The class or object on which the new

Parameters	Required	Description
<expC1>	Yes	The class or object on which the new class or object is based.
<expC2>	No	The module that contains <expC1>.
<expC3>	No	The application (.exe, or .app) that contains <expC2>.
<exp1>,<exp2>,...	No	The parameters to be passed to the class or object's Init event procedure.

The NEWOBJECT() function returns a reference to the newly created class or object.

Example

```
mySpreadSheet = newobject("Excel.Sheet")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NETWORK()

Class

Disk and File Utilities

Purpose

Function to check for network

Syntax

NETWORK()

See Also

USED(), NETERR(), SYS()

Description

The NETWORK() function will return .T. if you are running a multi-user system, otherwise .F. On UNIX/Linux and OpenVMS, this function always returns .T. as these systems are multi-user.

Example

```
? network()
```

```
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NEXTKEY()

Class

Keyboard Events

Purpose

Function to return the integer numeric value of a key press

Syntax

NEXTKEY()

See Also

INKEY(), READKEY(), SET KEY, SLEEP, ON ESCAPE, ON KEY, LASTKEY(), SET PCKEYS

Description

The NEXTKEY() function returns an integer numeric value for the next key pressed at the keyboard, without removing it from the keyboard buffer. The command SET PCKEYS is used to change to return value of the NEXTKEY() function. When SET PCKEYS is OFF, the NEXTKEY() function returns the ASCII code for the key that was pressed. When SET PCKEYS is ON, the NEXTKEY() function returns the IBM PC keyboard value of the key that was pressed. If no key is pressed, the NEXTKEY() function returns 0.

Example

```
do while nextkey()=0
    @0,68 say ampm()
    sleep 2
enddo
```

Products

Recital Mirage Server, Recital Terminal Developer

NFCOUNT()

Class

Fields and Records

Purpose

Function to return numeric field count

Syntax

NFCOUNT([<workarea | alias>])

See Also

FCOUNT(), DBF(), NDX(), FMT(), FILTER(), FIELD(), DISPLAY STRUCTURE

Description

The NFCOUNT() function returns the number of numeric fields in the active table. If the optional <workarea | alias> is specified, then the function will operate in the required location.

Example

use accounts

? nfcunt()

3

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NUMLOCK()

Class

Xbase Compatibility

Purpose

Language compatibility only

Syntax

NUMLOCK([<expL>])

See Also

SET COMPATIBLE

Description

This function has been added for language compatibility only. The NUMLOCK() function always returns .F..

Example

? numlock()

.F.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

NVL()

Class

Expressions and Type Conversion

Purpose

Function to specify an alternative value for a null expression

Syntax

NVL(<expr1>,<expr2>)

See Also

ETOS(), ISNULL(), SET NULL

Description

The NVL() function evaluates the expression in <expr1>, and if the expression does not evaluate to NULL, the evaluated result is returned. If the expression in <expr1> does evaluate to NULL, the expression in <expr2> is evaluated. If <expr2> does not evaluate to NULL, the evaluated result is returned. If both <expr1> and <expr2> evaluate to NULL, the NVL() function returns NULL.

Example

```
set sql to vfp
set null on
CREATE TABLE nullon (firstname c(20), lastname c(20))
INSERT INTO nullon (lastname) VALUES ("Smith")
SELECT lastname, nvl(firstname,"Unknown") from nullon
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

OBJECTREAD()

Class

Objects

Purpose

Function to read an external binary file into an object field

Syntax

OBJECTREAD(<filename>,<object field>)

See Also

OBJECTWRITE(), OBJECTTYPE(), BINOPEN(), BINREAD(), BINWRITE(), BINCREATE(),
BINCLOSE()

Description

The OBJECTREAD() function reads an external binary file into a Recital Object field. It returns .T. if the operation was successful, and .F. otherwise.

The file to read in is specified in <filename>, and can be any valid Recital/4GL expression that returns a valid file name. The first three characters of the file extension are stored in the Object field as the Objects type, as returned by the OBJECTTYPE() function.

The field into which the file is placed is specified in <object field>, and must be a Recital Object field in the currently selected workarea.

Example

```
? objectread('images/brickwall.gif',WALLPAPER)  
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

OBJECTTYPE()

Class

Objects

Purpose

Function to return object file extension

Syntax

OBJECTTYPE(<memofield>)

See Also

OBJECTREAD(), OBJECTWRITE(), BINOPEN(), BINREAD(), BINWRITE(), BINCREATE(),
BINCLOSE()

Description

The OBJECTTYPE() function returns the file extension of the data stored in the memo field <memofield>. The following file types are supported:

File Extension	Data
gif	GIF Image
jpg	JPG Image
au	AU Sound
avi	AVI Video
mov	MOV Video

Example

```
objectread('brickwall.gif',IMAGES)
```

```
? objecttype(IMAGES)
```

gif

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

OBJECTWRITE()

Class

Objects

Purpose

Function to write an external binary file from an object field

Syntax

OBJECTWRITE(<filename>,<object field>[,<expL>])

See Also

OBJECTREAD(), OBJECTTYPE(), BINCREATE(), BINOPEN(), BINREAD(), BINWRITE(), BINCLOSE()

Description

The OBJECTWRITE() function writes an external binary file from a Recital Object field. The OBJECTWRITE() function returns .T. if the external binary file was created successfully, and .F. otherwise.

The name of the file to create can be specified in the parameter <filename>. This can be any valid Recital/4GL expression that returns a valid filename. The filename can also be an empty string, providing that the logical expression <expL> is true (.T.). In this case, a unique temporary file name will be generated and the Object written to this file, with the file name being returned from OBJECTWRITE().

The Recital object field containing the data to be written, is specified in the parameter <object field>, and must be a valid field in the currently selected workarea.

Example

```
? objectwrite("myicon.gif",ICONS)
```

.T.

```
? objectwrite("",PHOTO,.T.)
```

_0049e10001.gif

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

OCCURS()

Class

String Data

Purpose

Function to return the number of times a specified character expression occurs within another specified character expression.

Syntax

OCCURS(<expC1>,<expC2>)

See Also

\$, INLIST()

Description

The OCCURS() function returns the number of times the character expression <expC1> is found in character expression <expC2>. A value of zero is returned if <expC1> does not exist in <expC2>. The character matching is case sensitive.

Example

```
? occurs("a", "Recital DAO Classes for Java")
```

4

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ON()

Class

Environment

Purpose

Function to return the commands assigned to the ON commands

Syntax

ON(<expC1> [,<expC2>])

See Also

INKEY(), LASTKEY(), ON ERROR, ON KEY, ON KEY LABEL, ON PAGE, READKEY()

Description

The ON() function will return the name of a command assigned to an ON command. The particular ON command is specified with <expC1>.

ON Command	<expC1>
ON ERROR	ERROR
ON ESCAPE	ESCAPE
ON KEY	KEY
ON KEY LABEL	KEY
ON PAGE	PAGE

The optional <expC2> is used to return the command assigned to the specific key or key-combination in ON KEY LABEL.

Example

```
? on("KEY", "F7")
```

do mail

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

OPENMAIL()

Class

Mail

Purpose

Function to connect to the specified mail server

Syntax

OPENMAIL(<expC1> [, <expC2>, <expC3> , <expC4>])

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), READMAIL(), SENDMAIL()

Description

The OPENMAIL() function is used to connect to the specified mail server. The mail type POP3 should be used for reading mail, and SMTP used for sending mail.

Parameters	Required	Default	Description
<expC1>	Yes	None	The node name or IP address of the mail server
<expC2>	Yes	None	The user name with which to connect to the mail server
<expC3>	Yes	None	The password for the user name
<expC4>	No	SMTP	The mail protocol type. POP3 and SMTP are the currently supported protocols.

If successful the OPENMAIL() function will return .T., otherwise .F.. The MAILERROR() function can be used to return the error message if the MAILOPEN() fails.

Example

```
// Open POP3 for reading mail
m_open = openmail("mailserver.company.com", "username", "password", "POP3")
if not m_open
    dialog box mailerror()
endif
closemail()

// Open SMTP for sending
m_open = mailopen("mailserver.company.com")
if not m_open
    dialog box mailerror()
endif
closemail()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ORDER()

Class

Indexing

Purpose

Function to return the currently selected master index number

Syntax

ORDER([<workarea | alias>])

See Also

INDEX, SET INDEX, CLOSE INDEX, SEEK, FIND, REINDEX, SET ORDER, SET ICACHE, MENU INDEX, SET DCACHE, SET CACHELOAD, INDEXKEY(), KEY(), NDX(), IFILECOUNT(), RLOOKUP(), LOOKUP(), SYS(14)

Description

The ORDER() function returns the currently selected master index number as specified with the SET ORDER TO command. If no index files are active, it returns -1.

If the workarea number or alias name <workarea | alias> is specified, then the function will operate in the required location.

Example

use patrons index name, event

? order()

1

set order to 2

? order()

2

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

OS()

Class

Disk and File Utilities

Purpose

Function to return Operating System name

Syntax

OS()

See Also

VERSION(), GETENV()

Description

The OS() function returns a character string of mixed case containing the name of the operating system. In UNIX/Linux, the OS() command returns the CPU name as well as the operating system. This function is very useful for determining the operating system before using the RUN command.

Example

```
// On OpenVMS
```

```
? os()
```

OpenVMS

```
// On UNIX (SCO)
```

SCO Openserver 5

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PAD()

Class

Menus

Purpose

Function to return prompt PAD name

Syntax

PAD()

See Also

POPUP(), BAR(), PROMPT(), SET COMPATIBLE, DEFINE PAD, DEFINE MENU, ON PAD, ON SELECTION PAD, ACTIVATE MENU, DEACTIVATE MENU, RELEASE MENUS, CLEAR MENUS

Description

The PAD() function returns the name of the most recently selected PAD of the currently active Xbase style menu. The command SET COMPATIBLE must be ON when using Xbase style menus.

Example

? pad()

edit

Products

Recital Mirage Server, Recital Terminal Developer

PADC()

Class

String Data

Purpose

Function to pad out a string to a specified length

Syntax

PADC(<expC1>,<expN>)

See Also

STRZERO(), STR(), RPAD(), STRTRAN(), STUFF(), STREXTRACT(), SUBSTR(), LPAD(), PADR(), PADL()

Description

The PADC() function centers a character string <expC1> and pads out the right and left of the string, to a total length of <expN> characters with blanks. PADC() returns the new string. If the string being processed is longer than the specified total length, it is returned unchanged.

Example

```
? [""] + padc("abc",7) + [""]
```

```
“  abc  ”
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PADL()

Class

String Data

Purpose

Function to pad out a string to a specified length

Syntax

PADL(<expC1>,<expN>,<expC2>)

See Also

STRZERO(), STR(), RPAD(), STRTRAN(), STUFF(), STREXTRACT(), SUBSTR(), LPAD(), PADR()

Description

The PADL() function right justifies a character string <expC1> and pads out the left of the string, to a total length of <expN> characters. PADL() returns the new string. If <expC2> is not specified, <expC1> will be padded with blanks, otherwise the string will be padded with the character specified by <expC2>. If the string being processed is longer than the specified total length, it is truncated.

Example

```
? padl("abc",6,"X")
```

XXXabc

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PADPROMPT()

Class

Menus

Purpose

Function to return the prompt text of a pad in a Xbase style menu

Syntax

PADPROMPT(<expC1> [,<expC2>])

See Also

ACTIVATE MENU, DEACTIVATE MENU, DEFINE MENU, DEFINE PAD, ON PAD, ON SELECTION PAD

Description

The PADPROMPT() function returns the text associated with a menu bar in a pop-up Xbase style menu. These menus are created with the DEFINE MENU and DEFINE PAD commands. The pad is identified by <expC1> which represents the name that was assigned with the DEFINE PAD...PROMPT <expC> command. The PADPROMPT() function works on the active menu unless another menu is specified with the optional character expression <expC2>.

Example

```
?padprompt("SW", "Accounts")
```

Southwest

Products

Recital Mirage Server, Recital Terminal Developer

PADR()

Class

String Data

Purpose

Function to pad out a string to a specified length

Syntax

PADR(<expC1>,<expN>,<expC2>)

See Also

LPAD(), STUFF(), STRTRAN(), STR(), STRZERO(), PADL(), LPAD(), RPAD()

Description

The PADR() function left justifies a character string <expC1> and pads out the right of the string, to a total length of <expN> characters. PADR() returns the new string. If <expC2> is not specified, <expC1> will be padded with blanks, otherwise, the string will be padded with the character specified by <expC2>. If the string being processed is longer than the specified total length, it is truncated. This function is the same as RPAD() and is included for language compatibility.

Example

```
? rpad("abc",6,"X")
```

abcXXX

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PARAMETERS()

Class

Applications

Purpose

Function to return number of parameters passed

Syntax

PARAMETERS()

See Also

PROCEDURE, FUNCTION, SET PROCEDURE, PCOUNT(), RETURN

Description

The PARAMETERS() function is synonymous with the PCOUNT() function. The PARAMETERS() function returns the number of parameters passed to a procedure or function. These functions are useful for checking that the correct number of parameters has been passed to a procedure or function.

Example

```
function print
parameter m_file,m_filter,m_options
if parameters() <>3
    set message to "3 parameters must be passed."
    return .F.
else
    set filter to &m_filter
    report form &m_file &m_options to print
    set filter to
endif
return .T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PATH()

Class

Disk and File Utilities

Purpose

Function to return path setting

Syntax

PATH()

See Also

DEFAULT(), DIR(), GETENV(), GETLOG(), SYS(), SET DEFAULT, SET PATH

Description

The PATH() function returns the current path setting. This is the path set using the Recital/4GL SET PATH command. The GETENV() or GETLOG() functions should be used to obtain Operating System level settings.

Example

? path()

C:\Program files\Recital\Uas\Mirage\Mirage_demo

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PAYMENT()

Class

Numeric Data

Purpose

Function to calculate the payment on a loan

Syntax

PAYMENT(<expN1>,<expN2>,<expN3>)

See Also

PMT(), FV(), PV(), CAGR(), LOG10(), PI()

Description

The PAYMENT() function is synonymous with the PMT() function. It calculates the payment on a loan where <expN1> is the principal amount of the loan, <expN2> is the periodic interest rate and <expN3> is the number of payment periods of the loan. Interest rates must be converted to fractions of one (e.g. 25% = .25).

Example

? payment(100,10,1)

110.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PCOL()

Class

Printing

Purpose

Function to return printer column position function

Syntax

PCOL()

See Also

ROW(), SCOLS(), COL(), PROW(), SROWS(), SET DEVICE

Description

The PCOL() function returns the current column position of the printer. The current print position is updated whenever you issue @...SAY commands. Normal output using other commands does not have an effect. The main use of the PCOL() function is for calculating relative column addressing when designing formatted reports for output to the printer. The SET DEVICE TO PRINT command must be in effect for the print column position to be tracked.

Example

```
set print on
set device to print
@0, 0 say "Testing"
@0, pcol()+20 say "o.k."
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PCOUNT()

Class

Applications

Purpose

Function to return number of parameters passed

Syntax

PCOUNT()

See Also

PROCEDURE, FUNCTION, SET PROCEDURE, RETURN, PARAMETERS

Description

The PCOUNT() function is synonymous with the PARAMETERS() function. The PCOUNT() function returns the number of parameters passed to a procedure or function. These functions are useful for checking that the correct number of parameters has been passed to a procedure or function.

Example

```
function print
parameter m_file,m_filter,m_options
if pcount() <>3
    set message to "3 parameters must be passed."
    return .F.
else
    set filter to &m_filter
    report form &m_file &m_options to print
    set filter to
endif
return .T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PERCENT()

Class

Numeric Data

Purpose

Function to calculate percentage

Syntax

PERCENT(<expN1>,<expN2>)

See Also

SET DECIMALS, ROUND()

Description

The PERCENT() function calculates <expN1> as a percentage of <expN2>. The number of decimals returned is defined by the SET DECIMALS command.

Example

```
?percent(10,100)  
10.00
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PEXIST()

Class

Applications

Purpose

Function to determine whether a function or procedure is active

Syntax

PEXIST(<expC>)

See Also

FUNCTION, RETURN, PROCEDURE, SET PROCEDURE, DISPLAY PROCEDURE, LIST PROCEDURE

Description

The PEXIST() function determines whether the specified user defined function or procedure is active. The procedure or function is specified by the character expression <expC>. If the procedure or function specified by <expC> is active, the PEXIST() function returns .T., otherwise .F..

Example

```
if not pexist("myproc")
    set procedure to mylib
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PI()

Class

Numeric Data

Purpose

Function to return the value of pi

Syntax

PI()

See Also

PAYMENT(), PMT(), FV(), PV(), CAGR(), LOG10(), LOG(), RTOD(), DTOR(), EXP(), SIN(), COS(), TAN()

Description

The PI() function returns the value of pi to seven decimal places.

Example

? pi()

3.1415926

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PMT()

Class

Numeric Data

Purpose

Function to calculate the payment on a loan

Syntax

PMT(<expN1>,<expN2>,<expN3>)

See Also

PAYMENT(), FV(), PV(), CAGR(), LOG10(), PI()

Description

The PMT() function is synonymous with the PAYMENT() function. It calculates the payment on a loan where <expN1> is the principal amount of the loan, <expN2> is the periodic interest rate and <expN3> is the number of payment periods of the loan. Interest rates must be converted to fractions of one (e.g. 25% = .25).

Example

? pmt(100,10,1)

110.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

POPUP()

Class

Menus

Purpose

Function to return the name of the active pop-up menu

Syntax

POPUP()

See Also

PROMPT(), BAR(), CLEAR POPUP, RELEASE POPUP, SHOW POPUP, SET COMPATIBLE

Description

The POPUP() function returns the name of the last Xbase style pop-up menu activated, which has not been deactivated. The command SET COMPATIBLE must be set ON when using Xbase style menus.

Example

? popup()

popup_3

Products

Recital Mirage Server, Recital Terminal Developer

PRINTFILE()

Class

Printing

Purpose

Used to provide client-side printing for Windows Recital Mirage clients

Syntax

```
PRINTFILE(<expC1>[,<expC2> [ , <expC3> ] ] )
```

See Also

SET PRINTER, GETENV()

Description

The PRINTFILE() function is used to provide client-side printing for Windows Recital Mirage clients. When the PRINTFILE() function is called, the Windows Print Dialog will be displayed allowing the user to select a printer before printing the specified file.

Keyword	Description
<expC1>	The full path name of the file to be printed. The <expC1> can also contain printing options. See below for available options
<expC2>	Optional header text to be displayed on each page of the printout.
<expC3>	Optional footer text to be displayed on each page of the printout.

Printing options are separated from the file name by a question mark '?' and have an option=value syntax. The second and subsequent options must be preceded by an ampersand '&'. The options can include the following:

Option	Description
autoNumber	If set to <i>true</i> , the file printout will include line numbers.
bottomMargin	Bottom margin in pixels.
font	Font definition, e.g. font=Helvetica,plain,10
fontBold	If set to <i>true</i> , font will be bold.
fontItalic	If set to <i>true</i> , font will be italic.
fontSize	Font size.
leftMargin	Left margin in pixels.
tabs	Number of spaces per tab.
topMargin	Top margin in pixels.

Header and footer text is made up of three comma-separated sections. Text preceding the first comma is printed left justified on the page. Text following the first comma (and preceding the second) is printed centered on the page. Text following the second comma is printed right justified on the page. The header and footer text can also contain special options:

Option	Description
{DATE}	Replaced in the header or footer text with the current system date.
{FILE}	Replaced in the header or footer text with the file name.
{PAGE}	Replaced in the header or footer text with the current page number.

If “&printfile(‘%s’)” is placed in the Recital Mirage Server DB_PRINT environment variable, then SET PRINTER TO \\SPOOLER will automatically print the spooled file to a client-side printer after displaying the Windows Print Dialog allowing the user to select the printer that they want to print on.

```
putenv(“DB_PRINT”,“&printfile(‘%s’)”)
```

Example

```
printFile("c:\Program files\Recital\UAS\Mirage\Mirage_demo\mirage_demo.prg" + ;  
    "?tabs=4&topMargin=50&autoNumber=true",;  
    "{FILE},,{DATE}","-- Page {PAGE} --,")
```

Products

Recital Mirage Server

PRINTSCREEN()

Class

Printing

Purpose

Function to print the currently displayed screen

Syntax

PRINTSCREEN()

See Also

SET PRINTER TO, SYS(), PRINTSTATUS(), ISPRINTER(), SET KEY...TO

Description

The PRINTSCREEN() function prints a hard copy of the currently displayed screen. Intended for use within full screen forms, the PRINTSCREEN() function is most commonly used with hot keys. The SET PRINTER TO command may be used to direct printer output to a printer attached to a serial communications line, or to cause the printer output to be spooled to the system printer. The PRINTSCREEN() function always returns .T..

Example

```
procedure print_scr
set printer to \\spooler
printscreen()
set printer to
return
```

set key 7 to print_scr

Products

Recital Terminal Developer

PRINTSTATUS()

Class

Printing

Purpose

Function to check printer status

Syntax

PRINTSTATUS()

See Also

ISPRINTER(), SET PRINTER, PRINT, PRINTSCREEN(), SYS()

Description

The PRINTSTATUS() function will return .T. if the printer is on line and ready to receive data, or .F. otherwise. On UNIX/Linux and OpenVMS, this function always returns .T..

Example

```
? printstatus()
```

```
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PRMBAR()

Class

Menus

Purpose

Function to return the prompt text from a popup bar

Syntax

PRMBAR(<expC>,<expN>)

See Also

ACTIVATE POPUP, BAR(), DEFINE BAR, DEFINE POPUP, DIALOG FIELDS, DIALOG FILES LIKE, DIALOG QUERY, DIALOG SCOPE, MENU FIELDS, MENU FILES LIKE, MENU QUERY, MENU SCOPE, ON SELECTION POPUP, POPUP(), PROMPT(), SET COMPATIBLE

Description

The PRMBAR() function returns the prompt text from a specified bar of a specified popup. The popup is specified in <expC>. The popup must have been defined, but need not be active. The name of the active popup is returned by the POPUP() function. The bar number is specified in <expN>. The number of the last selected bar is returned by the BAR() function. If special characters such as hot keys '\<' and disabled items '\' are included in the prompts, these are removed by the PRMBAR() function. When a popup bar is a separator, '\-', PRMBAR() returns a null string.

Example

```
set compatible to foxpro
define popup popup_1;
    from 1,26
define bar 1 of popup_1;
    prompt "Red"
define bar 2 of popup_1;
    prompt "Yellow"
define bar 3 of popup_1;
    prompt "Blue"
on selection popup popup_1 dialog box "You chose " + prmbar("POPUP_1",bar())
activate popup popup_1
```

Products

Recital Mirage Server, Recital Terminal Developer

PROCLIBS()

Class

Disk and File Utilities

Purpose

Function to return procedure library setting

Syntax

PROCLIBS()

See Also

FUNCTION, PROCEDURE, PROCNAME(), PROGRAM(), SET PROCEDURE

Description

The PROCLIBS() function returns the current procedure library setting. Procedure libraries are set using the SET PROCEDURE or SET LIBRARY commands. The full filenames including the path of all active procedure libraries are returned by the PROCLIBS() function in a comma-separated string.

Example

```
set procedure to lib1
```

```
set procedure to lib2 additive
```

```
? proclibs()
```

```
/home/apps/test/lib1.dbo,/home/apps/test/lib2.dbo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PROCLINE()

Class

Applications

Purpose

Function to return currently executing line number

Syntax

PROCLINE([<expN>])

See Also

FUNCTION, MESSAGE, RESUME, SUSPEND, LINENO(), MESSAGE(), PROCEDURE(), PROGRAM(), SET DEBUG, SET DOHISTORY, SET ECHO, SET HISTORY

Description

The PROCLINE() function returns the number of the current line being executed. This function is particularly useful in conjunction with the ON ERROR command. If the currently executing line is within a procedure or function within a multi-procedure/function program or library, the line number returned is relative to the program or library as a whole. If the optional <expN> evaluates to 1, the line number returned is relative to the individual procedure or function. In this case, the PROCEDURE or FUNCTION declaration is treated as line 0 and the first executable line as line 1.

Example

```
m_procline = procline()
message "The last line number is &m_procline. Press any key."
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PROCNAME()

Class

Applications

Purpose

Function to return name of currently executing procedure

Syntax

PROCNAME([<expN>])

See Also

DEBUG, DO, MESSAGE, RESUME, SUSPEND, DOLEVEL(), MESSAGE(), PATH(), PROCLIBS(), PROCLINE(), SYS(), SET DEBUG, SET DOHISTORY, SET ECHO, SET HISTORY

Description

The PROCNAME() function returns the name of the currently executing program or procedure. This function is useful when used with the SET KEY TO command or ON ERROR routines. The PROCNAME() function always returns a character string in lower case. If a program is currently being executed, the “.prg” for program or “.dbo” for compiled program will be returned with the name of the program. Otherwise if a procedure or user defined function (UDF) is being executed, only the name will be returned.

<expN>

Specifying the optional <expN> causes PROCNAME() to return the name of the procedure or program at the <expN> level. If there is no program or procedure at the specified level, an empty string is returned. The name of the master or starting program is returned if <expN> is 0 or 1.

Example

```
m_procname = procname()
```

```
dialog message "The error occurred in the procedure &m_procname."
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PROGRAM()

Class

Applications

Purpose

Function to return the currently executing program name

Syntax

PROGRAM([<expN>])

See Also

DEBUG, DO, FUNCTION, MESSAGE, PROCEDURE, RESUME, SUSPEND, DOLEVEL(), LINENO(), MESSAGE(), PATH(), PROCLIBS(), PROCLINE(), PROCNAME(), SYS(), SET DEBUG, SET DOHISTORY, SET ECHO, SET HISTORY

Description

The PROGRAM() function returns the name of the currently executing program or procedure. This function is particularly useful when used with the SET KEY TO command or ON ERROR routines. The PROGRAM() function always returns a character string in upper case. If a program is currently being executed, the full path and the file extension, “.PRG” for program or “.DBO” for compiled program, will be returned with the name of the program. If a procedure or user defined function (UDF) is being executed, only the name will be returned.

<expN>

Specifying the optional <expN> causes PROGRAM() to return the name of the procedure or program at the <expN> level. If there is no program or procedure at the specified level, an empty string is returned. The name of the master or starting program is returned if <expN> is 0 or 1.

Example

```
m_program = program()
dialog message "The error occurred in the program; &m_program."
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PROMPT()

Class

Menus

Purpose

Function to return the prompt of the most recently selected pop-up or menu option

Syntax

PROMPT()

See Also

PAD(), BAR(), POPUP(), SET COMPATIBLE

Description

The PROMPT() function returns the prompt of the most recently selected Xbase style pop-up or menu option. If there are no active pop-ups or the [ABANDON] key was pressed, the PROMPT() function will return a null string. If the prompt has been defined using the DEFINE POPUP command, the PROMPT() function will return the selected menu option as a character string. The command SET COMPATIBLE must be set ON when using Xbase style menus.

Example

```
define popup popup_4;
    from 1,26
define bar 1 of popup_4;
    prompt "Bar 1"
define bar 2 of popup_4;
    prompt "Bar 2"
define bar 3 of popup_4;
    prompt "Bar 3"
on selection popup popup_4 set message to PROMPT()
set status on
activate popup popup_4
```

Products

Recital Mirage Server, Recital Terminal Developer

PROPER()

Class

String Data

Purpose

Function to convert words to lower case with the first character in upper case

Syntax

PROPER(<expC>)

See Also

TRIM(), LEN(), STREXTRACT(), SUBSTR(), UPPER(), LOWER(), ISUPPER(), ISLOWER()

Description

The PROPER() function converts the first character of the specified <expC> to upper case and the remaining characters of the word to lower case.

Example

```
? proper("ACCOUNTS")
```

Accounts

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PROW()

Class

Printing

Purpose

Function to return printer row position function

Syntax

PROW()

See Also

ROW(), SROWS(), COL(), PCOL(), SCOLS()

Description

The PROW() function returns the current row position of the printer. The current print position is updated whenever you issue @...SAY commands. Normal output using other commands does not have an effect. The main use of the PROW() function is to calculate relative row addressing when designing formatted reports for output to the printer. . The SET DEVICE TO PRINT command must be in effect for the print column position to be tracked.

Example

```
set print on
set device to print
@0,0 say "Line one"
@prow()+1,1 say "Line two"
@prow()+2,1 say "Line three"
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PUTENV()

Class

Environment

Purpose

Function to update the value of an environment symbol

Syntax

PUTENV(<expC1>,<expC2>)

See Also

GETENV()

Description

The PUTENV() function updates the value of <expC1> with the character expression <expC2>. The value of the symbol can be retrieved using the GETENV() function. If variable <expC1> does not exist, it is created. If the operation is successful, then PUTENV() returns .T., otherwise .F.. The PUTENV() function is very useful for passing parameters to Recital programs spawned in the background, when used in conjunction with the GETENV() function.

Example

```
set printer to \\spooler
putenv("db_print","/usr/recital/UD/print_labels")
list status to print
? getenv("db_print")
/usr/recital/UD/print_labels
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PUTFILE()

Class

Disk and File Utilities

Purpose

Function to display a Save As dialog allowing a filename to be selected or specified and returned.

Syntax

PUTFILE([<expC1>] [,<expC2>] [,<expC3>])

See Also

GETFILE()

Description

The PUTFILE() function displays a Save As dialog allowing a filename to be selected or specified and returned. The dialog allows navigation through directories and shows a list of relevant files in that directory. The cursor keys, Return key and tab key can be used to navigate the different sections under a Screen Forms. If the user selects or specifies a file, the PUTFILE() function returns the name of that file. If no file is selected or specified, the PUTFILE() function returns an empty string "".

Parameters	Required	Description
<expC1>	No	The text to display at the top of the dialog. If not specified, 'Save As' is displayed.
<expC2>	No	The default filename displayed in the text entry area..
<expC3>	No	A file extension skeleton. If specified only files with this extension are shown.

Example

```
cProgramSaveAs = putfile("Save the program","default.prg","prg")
```

Products

Recital Mirage Server, Recital Terminal Developer

PUTLOG()

Class

Environment

Purpose

Function to create or update a logical name with specified equivalence string

Syntax

PUTLOG(<expC1>,<expC2>, [<expC3>])

See Also

GETLOG(), GETENV(), PUTENV()

Description

Used to manipulate OpenVMS process wide logical names, the PUTLOG() function creates a logical name <expC1> with the equivalence string <expC2>. If the logical name <expC1> already exists, <expC2> will overwrite the existing equivalence string. The optional <expC3> may be used to specify the name of the logical table where the logical name should be created or updated. If no logical table is specified, the PUTLOG() function creates or updates values in the LNM\$PROCESS_TABLE. These values can be displayed in OpenVMS by entering the OpenVMS command SHOW LOGICAL/PROCESS. On other operating systems, this function operates in the same way as PUTENV().

Example

```
// To change the logical value of sys$print
menu at 2,35 to 8,45;
    with "LINE","LASER","PLOTTER";
    label "PRINTERS";
    clear;
    quit
if not empty(menuitem())
    putlog("sys$print", menuitem())
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

PV()

Class

Numeric Data

Purpose

Function to calculate the present value of equal regular payments on a loan

Syntax

PV(<expN1>,<expN2>,<expN3>)

See Also

PAYMENT(), PMT(), FV(), CAGR(), LOG10(), PI()

Description

The PV() function calculates the present value of equal regular payments on a loan, where <expN1> is the payment per period, <expN2> is the periodic interest rate and <expN3> is the number of equal payments.

Example

? pv(1000,.10,10)

6144.57

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

QUARTER()

Class

Date and Time Data

Purpose

Function to return year quarter for the specified date or datetime

Syntax

QUARTER(<expD> | <expT>[, <expN>])

See Also

CDOW(), CMONTH(), CTOD(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), EPOCH(), GOMONTH(), MDY(), MONTH(), MTOD(), STOD(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK

Description

The QUARTER() function returns the year quarter for the specified date expression <expD> or datetime expression <expT>. The optional <expN> is used to specify the number of an alternative starting month for the year; the default is 1 (January).

Example

? quarter({01/22/2004})

1

? quarter({^20040822 12:34:29 PM})

3

? quarter({01/22/2004},2)

4

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RAND()

Class

Numeric Data

Purpose

Function to return a random number

Syntax

RAND(<expN>)

See Also

TMPNAM(), SYS(), GETPID()

Description

The RAND() function returns a random number in the range 1 to 2147483647. The <expN> is included for FoxPro compatibility.

Example

? rand()

31466

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RANK()

Class

Expressions and Type Conversion

Purpose

Function to convert an ASCII character to a number

Syntax

RANK(<expC>)

See Also

CHR()

Description

The RANK() function returns the numeric value of the first ASCII character from the character expression <expC>.

Example

```
? rank("123")
```

49

```
? rank ("Newcastle")
```

78

```
nVar = rank ("Newcastle")
```

78

```
? type("nVar")
```

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RAT()

Class

String Data

Purpose

Function to search for a substring

Syntax

RAT(<expC1>, <expC2> | <memofield> [<expN>])

See Also

AT(), ATNEXT(), LEFT(), RIGHT(), STUFF(), SUBSTR()

Description

The RAT() function is a substring search function. It will search <expC2> or the specified memo field for the last occurrence of <expC1> and return the starting position as a numeric value. If the substring is not contained within the second character expression or memo field, then the function returns the value 0. The leftmost character of a string is in character position 1. The RAT() function will return the starting position of the specified occurrence of <expC1> when the optional numeric expression <expN> is used. The RAT() function is particularly useful in conjunction with RIGHT() or SUBSTR functions for locating starting points for text extraction.

Example

```
? rat("is", "is is is")
```

7

```
cString1 = "fa"
```

```
cString2 = "Recital is fast"
```

```
? rat(cString1, cString2)
```

12

```
?rat("Step", notes, 8)
```

24

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

READEXIT()

Class

Screen Forms

Purpose

Function to control read termination from keyboard

Syntax

READEXIT([<expL>])

See Also

SET KEY, SET PCKEYS

Description

The READEXIT() function allows you to enable or disable the termination of a read by using the [CURSOR UP] and [CURSOR DOWN] keys. Read termination is enabled if the result of the logical expression <expL> is .T. and disabled if .F. The READEXIT() function will return .T. if READEXIT is ON and .F. if READEXIT is OFF. The command PCKEYS must be set ON, when using the function.

Example

? readexit()

.T.

readexit(.F.)

? readexit()

.F.

Products

Recital Mirage Server, Recital Terminal Developer

READINSERT()

Class

Screen Forms

Purpose

Function to toggle insert mode on or off during a READ

Syntax

READINSERT(<expL>)

See Also

READEXIT(), KEYBOARD, REPLAY, SET READINSERT

Description

The READINSERT() function is used to toggle insert mode on or off prior to activating a form. Insert mode is enabled if the result of the logical expression <expL> is .T. and disabled if the result is .F.. The default for READINSERT() is .F., in which case the user must press the [INSERT] key to toggle insert mode on. When READINSERT() is .T., insert mode is automatically on for activated forms. The READINSERT() function will return .T. if READINSERT was ON and .F. if READINSERT was OFF. The command PCKEYS must be set ON when using the function.

Example

```
? readinsert()
```

```
.F.
```

```
// Toggle insert mode on
```

```
readinsert(.T.)
```

```
? readinsert()
```

```
.T.
```

Products

Recital Mirage Server, Recital Terminal Developer

READKEY()

Class

Screen Forms

Purpose

Function to read last key pressed

Syntax

READKEY()

See Also

INKEY(), CTRL(), SET KEY, LASTKEY(), NEXTKEY(), UPDATED(), SET PCKEYS

Description

The READKEY() function returns a number representing the ASCII code for the last key pressed from within a form or an @...GET. If the contents of the GET or memory variable were modified by the READ, then 256 is added to the return value. The CLEAR TYPEAHEAD commands clears the READKEY() function to zero.

Example

```
if readkey()=ctrl('g')
    set message to "Operation canceled."
endif
```

Products

Recital Mirage Server, Recital Terminal Developer

READMAIL()

Class

Mail

Purpose

Function to read the specified mail message

Syntax

READMAIL(<expN>, <expC1>, [,<expC2>])

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(), MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(), MAILSEND(), MAILUSERNAME(), OPENMAIL(), SENDMAIL()

Description

The READMAIL() function will read the mail message specified by the number <message>, returning the number of elements in the array that it creates. If you are not connected to a mail server, it will return an error. You can use the MAILUSERNAME() or MAILNODENAME() function to check if you are connected.

Parameters	Required	Default	Description
<expN>	Yes	None	Numeric value specifying the mail message to read
<expC1>	Yes	None	The name of the array to be created to contain the lines of the mail message
<expC2>	No	“ALL”	The text that will be read from the mail message. This can be any one of the following: “ALL” “HEADER” “BODY” “ATTACHMENT”

The MAILERROR() function can be used to return the error message if the READMAIL() fails.

Example

```
number = readmail(1, “mail_mes”, “BODY”)
```

```
? number
```

```
30
```

```
? mail_mes[30]
```

This is the last line in the mail message.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

READMODE()

Class

Screen Forms

Purpose

Function to return a character string describing the 'read' mode for the currently active form

Syntax

READMODE()

See Also

SET(), SYS(), SET PREFORM, SET PRERECORD, CREATE, @...GET, SET POSTRECORD, SET POSTFORM, MODIFY STRUCTURE

Description

The READMODE() function returns a character string describing the 'read' mode for the currently active form. Any of the following strings can be returned.

Readmode	Description
READ	READ command is active
EDIT	EDIT command is active
CHANGE	CHANGE command is active
INSERT	INSERT command is active
QUERY	QUERY command is active
APPEND	APPEND command is active
	No form active

The READMODE() function always returns a character string in upper case. This function is very useful when used in table and form triggers for determining the current read state for conditional coding.

Example

```
// Pre-record trigger procedure
procedure form_trigger
if readmode() = "APPEND"
    return
else
    //...commands
endif
return
```

```
use accounts
set form to details
set prerecord to form_trigger
edit
```

Products

Recital Mirage Server, Recital Terminal Developer

READVAR()

Class

Screen Forms

Purpose

Function to name the @...GET variable being processed by a READ

Syntax

READVAR([<expN>])

See Also

@...GET, DEFINE TABLE, RECNO(), PARAMETER, SET VALIDATE

Description

The READVAR() function returns the name of the current @...GET variable being processed by a READ command. The name of the variable is returned as an uppercase string. When used with table fields, the READVAR() function returns a blank when the table field is displayed, and the table name when it is activated. The READVAR() function can also be used to return the <variable> of the current MENU TO <variable> command.

If the optional <expN> is specified and is greater than zero (0) and if the current @...GET being processed is an object property, then the name of that object will be returned.

Example

```
// Pop-up choice list called from a field validation
procedure choices
parameters m_input
// If the help key is pressed
if lastkey()=ctrl('c')
    m_exp = iif(readvar()="ACC_NO","account","supplier")
    select codes
    save screen
    set message to "Select code, press ^C to abandon."
    menu browse &m_exp at 5,30 to 15,40;
        quit;
        clear
    restore screen
    // If a selection was made
    if not empty(menuitem())
        set fieldval to left(menuitem(),3)
        set validate on
    endif
    select accounts
endif
return
```

Products

Recital Mirage Server, Recital Terminal Developer

RECCOUNT()

Class

Fields And Records

Purpose

Function to return the number of records in table

Syntax

RECCOUNT([<workarea | alias>])

See Also

RECSIZE(), DBF(), NDX(), HEADER(), FCOUNT(), INDEXKEY()

Description

The RECCOUNT() function returns a number representing the total number of records in the currently selected table. This number includes records that have been marked for deletion. If the optional <workarea | alias> is specified, then the function will operate in the required location.

Example

use accounts

? reccount()

2500

? fcount()

18

declare aAccounts[reccount(),fcount()]

copy to array aAccounts for empty(paid_date)

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RECNO()

Class

Fields And Records

Purpose

Function to return current record number

Syntax

RECNO([<workarea | alias> [<expN>]])

See Also

BOF(), EOF(), FOUND(), RECCOUNT()

Description

The RECNO() function returns the record number of the current record in the currently selected table file. Note that when using indexed tables, RECNO() returns a physical record number in the table, and not a record number relative to the index. If the optional <workarea | alias> is specified, then the function will operate in the required location. The optional <expN> can be used as a softseek function when the value of 0 is specified. If a seek fails on an index table, RECNO(0) will return the record number for the record above the SEEK expression value with respect to the index.

Example

```
use accounts index acc_no
? recno()
  1
goto 35
? recno()
 35
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RECSIZE()

Class

Disk and File Utilities

Purpose

Function to return record size

Syntax

RECSIZE([<workarea | alias>])

See Also

RECCOUNT(), HEADER(), DBF(), NDX(), FIELD(), FCOUNT()

Description

The RECSIZE() function returns a number representing the size of each record in the currently selected table. The size will be the sum of all the field storage sizes plus one character for the deletion marker in the record. If the optional <workarea | alias> is specified, then the function will operate in the required location. When used with the HEADER() function and the RECCOUNT() function, the RECSIZE() function allows you to calculate the space which your table occupies.

Example

```
use accounts
if header() + reccount() * reccount() > 100000
    dialog box "Database too big for backup disk."
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

REFERENCES()

Class

Fields and Records

Purpose

Function to perform a cross-table lookup for a specified key expression

Syntax

REFERENCES(<key expression>, <workarea | alias> [,<tag name>])

See Also

SET RELATION, LOOKUP(), RLOOKUP(), SEEK()

Description

The REFERENCES() function looks up the specified <key expression> in the master tag index of the specified <workarea | alias>. The <workarea | alias> is the workarea or alias name of an open table. To search in a tag index which is not the current master index, the optional <tag name> parameter can be used. The tag name must be specified as a string.

The REFERENCES() function returns True (.T.) or False (.F.), depending on the success of the lookup operation.

Please see the RLOOKUP() function for cross-table lookups using single indexes (.ndx).

Example

```
use customer.rdb
index on account_no tag account_no
index on upper(last_name) tag uplast
index on zip tag zip
? references("STEREK",customer,"uplast")
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

REMOVEPROPERTY()

Class

Objects

Purpose

Function to remove a property from an existing object

Syntax

REMOVEPROPERTY(<object-name> ,<expC>)

See Also

DEFINE CLASS...ENDDEFINE, WITH...ENDWITH, ADDPROPERTY(), CREATEOBJECT(), DODEFAULT(), NEWOBJECT()

Description

The Visual FoxPro compatible REMOVEPROPERTY() function is used to remove a property from an existing object. It returns .T. (True) if the property was successfully removed and .F. (False) otherwise.

Parameter	Description
<object-name>	The name of the object.
<expC>	The name of the property to be removed.

Properties can be added using the ADDPROPERTY() function. All classes have an inbuilt ADDPROPERTY 'factory method'. This can be used as an alternative to the ADDPROPERTY() function to add properties to an object at runtime.

Example

```
define class myclass as custom
myprop = "Hello World"
enddefine
```

```
myobject = createobject("myclass")
MessageBox(myobject.myprop)
addproperty(myobject, "myprop2", "goodbye")
// Or: myobject.addproperty("myprop2", "goodbye")
MessageBox(myobject.myprop2)
removeproperty(myobject, "myprop2")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RELATION()

Class

Table Basics

Purpose

Function to return table-linking expression

Syntax

RELATION(<expN>)

See Also

TARGET(), DBRELATION(), SET RELATION, CREATE VIEW, CREATE BRIDGE, SET VIEW, USE, DBRSELECT()

Description

The RELATION() function is synonymous with the DBRELATION() function. The RELATION() function returns the linking expression of a specified relation in the current workarea. The <expN> specifies the position of the desired linking expression from the list of previously defined relations. By default, the Recital environment supports 20 workareas but this can be increased, by setting the environment symbol DB_MAXWKA to a higher value, which allows for up to (DB_MAXWKA-1) relationships. The RELATION() function always returns a character string in lower case. If there is no linking expression defined for the <expN> selected, a null string will be returned.

Example

```
use shows in 2 index pcode
use patrons in 1
set relation to patron_code into shows
? relation(1)
patron_code
? target(1)
  2
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

REPLACE()

Class

Fields and Records

Purpose

Function to replace fields in the current record buffer without re-reading the current record

Syntax

REPLACE(<field>,<exp>)

See Also

SET POSTRECORD, SET PRERECORD, CREATE, MODIFY STRUCTURE, REPLACE, @...GET

Description

The REPLACE() function allows fields in the current record buffer to be modified without re-reading the record. The REPLACE function will update the <field> with the value stored in <exp>. The <exp> must be the same data type as the <field> being replaced. When used in conjunction with the PRERECORD, POSTRECORD and table UPDATE trigger procedures, REPLACE() can be used to add timestamps, and other “hidden” information to records which are processed through forms. If any dictionary validation or security is in effect for the specified <field>, then this will be checked. If the field cannot be updated, .F. is returned, otherwise .T. is returned. The REPLACE() function is particularly useful when used in POSTRECORD trigger procedures which are activated in APPEND forms. The REPLACE() function replaces data in the active record buffer, these changes are not written to disk until action is taken to commit the record. The REPLACE command is unlike the REPLACE() function, in that changes to the active record buffer are written to disk immediately.

Example

```
procedure postrec
replace(time, time())
replace(use, getenv("username"))
replace(date, date())
return .T.
```

```
use accounts
set postrecord to postrec
append
```

Products

Recital Mirage Server, Recital Terminal Developer

REPLICATE()

Class

String Data

Purpose

Function to replicate characters

Syntax

REPLICATE(<expC>,<expN>)

See Also

SPACE(), REVERSE(), UNDERLINE(), BOLD(), BLINK()

Description

The REPLICATE() function makes <expN> copies of the character expression <expC>.

Example

```
? replicate("-", 20)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RESTSCREEN()

Class

Screen Forms

Purpose

Function to display a screen region, previously saved with SAVESCREEN(), to a specified screen area

Syntax

RESTSCREEN(<expN1>, <expN2>, <expN3>, <expN4>, <expC>)

See Also

SAVE SCREEN, SAVESCREEN(), RESTORE SCREEN, SET SCREENMAP, CHANGE, EDIT, QUERY, APPEND, BROWSE, @...SAY, @...MENU, MENU

Description

The RESTSCREEN() function displays a screen region previously saved with SAVESCREEN(). The numerical expressions <expN1-4> represents the coordinates of the screen, and must be at the same location that the memory variable <expC> was saved at with the SAVESCREEN() function. The RESTSCREEN() function refreshes only those portions of the screen which have changed since the screen was last saved. The character expression <expC> specifies a memory variable that contains the screen image saved with the SAVESCREEN() function. The command SET SCREENMAP must be ON for this function to work.

Example

```
// Clear area for pop up window
save_region = savescreen(1,2,10,20)
// Pop up a window
// Restore screen area
restscreen(1,2,10,20,save_region)
```

Products

Recital Mirage Server, Recital Terminal Developer

REVERSE()

Class

Input/Output

Purpose

Function to display text in reverse video

Syntax

REVERSE(<expC>)

See Also

BOLD(), BLINK(), UNDERLINE(), SET SCREENIO

Description

The REVERSE() function displays the specified character expression <expC> in reverse video on the screen. Key words in MESSAGE commands can be reversed to make them stand out. SET SCREENIO must be ON (default is off), for the REVERSE() function to operate.

Example

@3,3 say "press " + reverse("ENTER") + " key."

Products

Recital Mirage Server, Recital Terminal Developer

RIGHT()

Class

String Data

Purpose

Function to extract substring from right

Syntax

RIGHT(<expC>,<expN>)

See Also

LEFT(), STREXTRACT(), SUBSTR(), STRTRAN(), AT(), ATNEXT(), RAT()

Description

The RIGHT() function extracts a substring from the right of the character expression <expC> of width <expN> characters. This function can be used on character fields in index expressions to extract part of the field in the index key.

Example

? right("Recital Corporation",11)

Corporation

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RLOCK()

Class

Manual Locking

Purpose

Function to lock record

Syntax

RLOCK([<workarea | alias>])

See Also

LOCK(), FLOCK(), UNLOCK, LOCKR, LOCKF

Description

The RLOCK() function attempts to lock the current record. If successful, it returns .T. and the record is locked. If the record is already locked by another user then it returns .F.. Please note that Recital automatically performs file and record locking so, in most situations, this function is unnecessary. It is included for compatibility with programs written with other products. If the optional <workarea | alias> is specified, then the function will operate in the required location.

Example

```
do while not rlock()
    set message to "Record in use."
    sleep 2
enddo
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RLOOKUP()

Class

Fields and Records

Purpose

Function to perform a cross-table lookup for a specified key expression

Syntax

RLOOKUP(<key expression>, <workarea | alias> [,<expN> | <tag name>])

See Also

SET RELATION, LOOKUP(), SEEK()

Description

The RLOOKUP() function looks up the specified <key expression> in the master index of the specified <workarea | alias>. The <workarea | alias> is the workarea or alias name of an open table. To search in an index which is not the current master index, the optional <expN> | <tag name> parameter can be used. The <expN> is for single indexes and indicates the required index by the numerical index order in which the indexes were opened. For tag / multiple indexes, the tag name must be specified as a string.

The RLOOKUP() function returns True (.T.) or False (.F.), depending on the success of the lookup operation.

This function is especially useful when used in the Applications Data Dictionary (ADD) for maintaining referential integrity rules. See the CREATE command for details of the Applications Data Dictionary.

Example

use func in 1 index funcno

use depts in 2 index deptnum

seek "300"

? iif(rlookup(depts->funcno,func,1),func->dept_name,"No Department.")

Research and Development

use customer.rdb

index on account_no tag account_no

index on upper(last_name) tag uplast

index on zip tag zip

? rlookup("STEREK",customer,"uplast")

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ROLLBACK()

Class

Transaction Processing

Purpose

Function to verify success of a transaction rollback

Syntax

ROLLBACK()

See Also

SET ROLLBACK, BEGIN TRANSACTION, ISMARKED(), RESET IN, ROLLBACK, END TRANSACTION , COMPLETED()

Description

The ROLLBACK() function returns .T. if a rollback is successful and .F. if a rollback fails. This function is used in conjunction with the ROLLBACK command to determine if an attempted rollback and recovery was successful. Automatic rollback and recovery can be initiated within a BEGIN TRANSACTION ... END TRANSACTION block with the SET ROLLBACK ON command.

Example

```
procedure recovery
rollback
if rollback()
    dialog box "Rollback was ok"
else
    dialog box "Rollback not completed"
endif
return

use masterfile
on error do recovery
begin transaction
    delete first 15
    replace all t1 with (t2*t3)/100
    list
end transaction
if completed()
    dialog box "Transaction completed"
else
    dialog box "Errors occurred during transaction"
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ROUND()

Class

Numeric Data

Purpose

Function to return rounded numeric

Syntax

ROUND(<expN1> [,<expN2>])

See Also

SET FIXED, SET DECIMALS, INT(), TRANSFORM(), FLOOR(), CEILING()

Description

The ROUND() function rounds the numeric expression <expN1> to <expN2> decimal places. If <expN2> is omitted then it is taken as zero. If the SET FIXED command is ON, <expN2> can not be greater than the number of decimal places specified by the SET DECIMALS command.

Example

? round(3.467)

3.00

? round(3.467,1)

3.50

? round(3.467,2)

3.47

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

ROW()

Class

Screen Forms

Purpose

Function to return screen row position

Syntax

ROW()

See Also

PROW(), PCOL(), COL(), SCOLS(), SROWS()

Description

The ROW() function returns the current screen row as a number. The current cursor position is updated whenever you issue the @...SAY...GET or MENU TO commands. Normal output using other commands does not have an effect. The main use of the ROW() function is to calculate relative cursor addressing when designing forms and menus. If you have issued the SET DEVICE TO PRINT command, the printer column position is updated rather than the screen column position, see PROW() for details.

Example

```
@01,01 say "Hello"  
@row()+1,01 say "World"  
@row()+1,01 say "This is Recital"
```

Products

Recital Mirage Server, Recital Terminal Developer

RPAD()

Class

String Data

Purpose

Function to pad out a character string to a defined length from the right

Syntax

RPAD(<expC1>,<expN>,[<expC2>])

See Also

LPAD(), STUFF(), STRTRAN(), STR(), STRZERO(), PADR()

Description

The RPAD() function left-justifies a character string <expC1> and pads out the right of the string, to a total length of <expN> characters, with blank spaces. The optional <expC2> may be used to pad <expC1> with a character string instead of blank spaces. The RPAD() function is useful for creating fixed length character strings by padding them out with blanks. If the string being processed is longer than the specified total length, it is truncated. The RPAD() function is synonymous with the PADR() function.

Example

```
? rpad("abc",6,"d")
```

abcddd

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RTOD()

Class

Numeric Data

Purpose

Function to convert radians to degrees

Syntax

RTOD(<expN>)

See Also

DTOR(), COS(), SIN(), TAN(), PI(), LOG(), LOG10(), EXP()

Description

The RTOD() function converts the number of radians specified in the expression <expN> to degrees.

Example

```
? rtod(3.14159)
180.00
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RTRIM()

Class

String Data

Purpose

Function to remove trailing blanks

Syntax

RTRIM(<expC>)

See Also

TRIM(), LTRIM(), ALTRIM(), STRTRAN()

Description

The RTRIM() function is synonymous with the TRIM() function. This function removes trailing blank characters from the character expression <expC>. This function should be used in conjunction with the RPAD() function when used in index expressions.

Example

```
fname = "Christopher"
sname = "Thompson"
? len(fname)
20
? len(rtrim(fname))
11
? rtrim(fname) + " " + rtrim(sname)
Christopher Thompson
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

RUN()

Class

Applications

Purpose

Function to execute an operating system command or an external program

Syntax

RUN([<expl>,<expC> [,<expl2>])

See Also

ALIAS, DO, EXEC(), GETENV(), PUTENV(), RUN, SPAWN

Description

The RUN() function loads the specified operating system command or external program to be executed by the operating system. The operating system command or external program is specified with character expression <expC>. If the specified program is a batch file whose full path name is not the default path, and you are loading the program to the operating system via the command interpreter, <expC> must include the file extension. The RUN() function returns an operating system dependent numeric completion code.

Example

```
run("dir")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SAVESCREEN()

Class

Screen Forms

Purpose

Function to save a screen region to a memory variable

Syntax

SAVESCREEN(<expN1>,<expN2>,<expN3>,expN4>)

See Also

SET SCREENMAP, RESTSCREEN(), CHANGE, EDIT, QUERY, APPEND, BROWSE, @...SAY, @...MENU, MENU, SAVE SCREEN, RESTORE SCREEN

Description

The SAVESCREEN() function allows you to save a portion of a screen for redisplay later. The coordinates of the screen portion are specified in the numeric expressions <expN1-4> . The numeric expressions represent numbers for the beginning row, beginning column, ending row and ending column, respectively. When SCREENMAP is on, the SAVESCREEN() function allows an image of all or part of the current terminal display to be saved to a memory variable. All Recital output to the screen is stored as part of the screen image. No output from the RUN command is stored.

Example

```
// Clear area for pop-up window
save_win = savescreen(1,5,10,20)
// Pop up a window
// Restore screen area
restscreen(1,5,10,20,save_win)
```

Products

Recital Mirage Server, Recital Terminal Developer

SCHEME()

Class

Screen Forms

Purpose

Function to return the color codes for a specified color scheme

Syntax

SCHEME(<expN1>[, <expN2>])

See Also

SET COLOR

Description

The SCHEME() function returns a color code pair or the list of all the color code pairs for a specified color scheme. The color scheme is specified using the numeric expression, <expN1>. If only <expN1> is specified, the entire list of color code pairs for that scheme is returned. If <expN2> is specified, the color code pair at that position in the list will be returned.

Example

```
? scheme(1)
```

```
W/B,R/W,B+/B+,W+/RB,N/W,BG+/R,GR+/R,N+/N,N/N,N/N
```

```
? scheme(1,4)
```

```
W+/RB
```

Products

Recital Mirage Server, Recital Terminal Developer

SCOLS()

Class

Screen Forms

Purpose

Function to return the number of screen columns

Syntax

SCOLS()

See Also

SROWS(), ROW(), PROW(), PCOL, COL(), MAXROW(), MAXCOL()

Description

The SCOLS() function returns the number of columns being used on the current terminal. The return value is numeric data type. In Recital addressable column numbers range from 0 to 79, however some programs may change this number to 80 or more. SCOLS() may be used to determine how screen or printed displays are affected by such changes.

Example

```
?scols()
```

```
79
```

Products

Recital Mirage Server, Recital Terminal Developer

SCROLL()

Class

Screen Forms

Purpose

Function to designate a screen area to scroll

Syntax

SCROLL(<expN1>,<expN2>,<expN3>,<expN4>,<expN5>)

See Also

MENU BROWSE, SET SCROLL, TEXTEDIT()

Description

The SCROLL() function designates an area of the to scroll up, scroll down, or blank out. The coordinates of the screen portion are specified in the numeric expressions <expN1-4> . The numeric expressions represent numbers for the beginning row, beginning column, ending row and ending column, respectively. <expN5> is the number of rows to scroll. A number greater than zero scrolls up the specified number of rows. A value less than zero scrolls down the specified number of rows. A value of zero clears the specified scroll area.

Example

```
scroll(2,2,20,20,10)
```

Products

Recital Mirage Server, Recital Terminal Developer

SEC()

Class

Date and Time Data

Purpose

Function to return the numeric seconds from a specified datetime

Syntax

SEC(<expT>)

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), HOURS(), MINUTE(), MINUTES(), SECONDS(), SECS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The SEC() function returns the seconds from the specified datetime expression <expT> as a numeric value.

Example

```
? sec({ 10/10/2004 10:15:43 AM})
```

43

```
m_Sec = sec(datetime())
```

```
? type("m_Sec")
```

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SECONDS()

Class

Date And Time Data

Purpose

Function to extract seconds from a time string

Syntax

SECONDS([<time-string>])

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), HOURS(), MINUTE(), MINUTES(), SEC(), SECS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLIPPER, SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The SECONDS() function returns the seconds from the current time as a number. The SECONDS() function will also return the seconds from an optionally specified <time-string>. If the command SET CLIPPER is ON, the SECONDS() function operates in the same way as the SECS() function.

Example

? seconds("10:33:21")

21

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SECS()

Class

Date And Time Data

Purpose

Function to return the number of seconds since midnight

Syntax

SECS([<time-string>])

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), HOURS(), MINUTE(), MINUTES(), SEC(), SECONDS(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLIPPER, SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The SECS() function returns the number of seconds since midnight. The optional <time-string>, in the format HH:MM:SS, may be used to specify a time. If no <time-string> is passed, the SECS() function uses the current time. This function can be used to store time as seconds in a numeric field. The TSTRING() function is used to convert the seconds back to a time-string. If SET CLIPPER is ON, the SECS() function behaves like the SECONDS() function.

Example

```
? secs("10:10:10")
```

36610

```
// Another Example
```

```
use accounts
```

```
replace seconds with secs(time())
```

```
? seconds
```

39306

```
? tstring(seconds)
```

10:55:00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SEEK()

Class

Indexing

Purpose

Function to verify presence of an index key

Syntax

SEEK(<key expression> [,<workarea | alias> [,<exp>]])

See Also

SEEK, FIND, LOCATE, RLOOKUP(), LOOKUP()

Description

The SEEK() function returns .T. if the specified index <key expression> has been found in the master index of the current workarea or the optionally specified <workarea | alias>. If the optional order parameter <exp> is specified, that index will be searched instead of the master index.

Example

```
use accounts in 1
use supp in 2 index name
select 1
display name
BCD CORPORATION
? iif(seek(name,supp),supp_name,"***name not on file***")
BCD CORPORATION
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SELECT()

Class

Table Basics

Purpose

Function to return the current workarea number

Syntax

SELECT([<alias>])

See Also

SELECT, USE, ALIAS(), WORKAREA(), READVAR(), DBF()

Description

The SELECT() function returns the number of the currently selected workarea. If the optional <alias> is specified, the SELECT() function returns the workarea number for that alias. If the <alias> specified does not exist, the SELECT() function will return a 0.

Example

close all

? select()

1

use prefixes in 3

? select ("prefixes")

3

// Another Example

// to access each open workarea

m_count=0

select 1

do while not empty(alias())

 m_count = m->m_count + reccount()

 select select() + 1

enddo

dialog box "Total records in all tables is &m_count"

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SENDMAIL()

Class

Mail

Purpose

Function to send mail

Syntax

```
SENDMAIL(<expC1>,<expC2>,[<expC3>],[<expC4>],<expC5>[,<expC6>[,<expL1>]]  
[,<expC7>])
```

See Also

CLOSEMAIL(), COUNTMAIL(), DELETEMAIL(), MAILCLOSE(), MAILCOUNT(), MAILDELETE(),
MAILERROR(), MAILHEADER(), MAILNODENAME(), MAILOPEN(), MAILREAD(),
MAILSEND(), MAILUSERNAME(), OPENMAIL(), READMAIL()

Description

The SENDMAIL() function is used for sending mail. If successful the SENDMAIL() will return .T. or .F. otherwise. The MAILERROR() function can be used to return the error message if the SENDMAIL() fails.

Parameters	Required	Default	Description
<expC1>	No	Your node	The from name of the sender
<expC2>	Yes	None	A semi-colon separated list of recipient names to receive the message
<expC3>	Yes	None	A semi-colon separated list of cc recipient names to receive the message
<expC4>	No	None	The subject of the message
<expC5>	Yes	None	The message to be sent. This can be a character string, a variable containing a character string or a character string containing the name of a text file.
<expC6>	No	None	A comma separated list of names of files to be sent as attachments
<expL1>	No	None	Open VMS only. If <expL1> is .T. (true), any attachments will be sent as ASCII, not binary. If <expL1> is .F. or omitted, all attachments will be sent as binary 64 bit encoded.
<expC7>	No	text/plain	Content type

Example

```
// Open SMTP for sending
m_open = mailopen("mailserver.company.com","username","password","SMTP")
if not m_open
    dialog box mailerror()
    return
endif

fromname = "info@recital.com"
tonames = "fred@recital.com;sue@recital.com"
ccnames = "bert@recital.co.uk;linda@recital.co.uk"
subject = "For your information"
message = "Dear All" + chr(10) + "Here are the files you asked for." + chr(10) + ;
    "Best regards" + chr(10) + "Sam"
attachments = "info.doc, info.xls"
mailsend(fromname,tonames,ccnames,subject,message,attachments)
mailclose()

// Open SMTP for sending
m_open = mailopen("mailserver.company.com","username","password","SMTP")
if not m_open
    dialog box mailerror()
    return
endif

fromname = "info@recital.com"
tonames = "fred@recital.com"
ccnames = ""
subject = "For your information"
message = "email.htm"
attachments = ""
content = "test/html"
sendmail(fromname,tonames,ccnames,subject,message,attachments,.F.,content)
mailclose()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SEQNO()

Class

Table Basics

Purpose

Function to return a unique sequence number

Syntax

SEQNO([<workarea | alias>])

See Also

ZAP, REPLACE(), SET POSTRECORD, SET SEQNO

Description

The SEQNO() function returns the next unique sequence number for the current table. Automatic locking is performed during the operation of this function if the specified table is opened shareable. The optional <workarea | alias> will return the next unique sequence number from the specified table. If there is no active table the SEQNO() function will return 0.

The SEQNO() function guarantees a unique sequence number even in a multi-user environment. The sequence number will continue increasing even after a PACK operation. A ZAP operation will reset the sequence number to 0. The sequence number of a table can be reset with the command SET SEQNO TO <expN>.

Example

```
append blank
replace custno with seqno()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SET()

Class

Environment

Purpose

Function to check status of a set command

Syntax

SET(<expC> [, <expN>])

See Also

EXCLUSIVE(), SYS(), SET COMMANDS

Description

The SET() function returns .T. or .F. depending on the status of the system setting. SET() only returns information for SET parameters which can be ON or OFF. The character expression <expC> is the SET command keyword. This function can be used in conjunction with the SET command to toggle the various SET COMMANDS ON and OFF.

SET("COMPRESS") returns true if COMPRESS has been set on.

SET("FASTINDEX") returns true if FASTINDEX has been set on.

SET("FILECASE") returns true if FILECASE has been set on.

SET("PSHARE") returns true if PSHARE has been set on.

SET("NAVIGATE") returns true if NAVIGATE has been set on.

SET("SYSMENU") returns true if SYSMENU has been set on. It returns "ON" or "OFF" in Xbase compatibility mode.

If the optional <expN> is included and is greater than zero (0), additional information is returned for those set commands that have more than one setting. For example, the SET ALTERNATE command has two settings: SET ALTERNATE ON/OFF and SET ALTERNATE TO <filename>.

Example

```
? set("safety")
```

.F.

```
// Another Example
```

```
// toggle exclusive to the opposite setting
```

```
set exclusive(iif(set("exclusive"),.F.,.T.))
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SETCANCEL()

Class

Xbase Compatibility

Purpose

Function to enable or disable exit using Alt-C

Syntax

SETCANCEL([<expL>])

See Also

QUIT, SET ESCAPE

Description

The SETCANCEL() function has been included for language compatibility with other products only.

Example

```
setcancel(.F.)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SETCOLOR()

Class

Screen Forms

Purpose

Function to return the current color setting and optionally set a new color setting

Syntax

SETCOLOR([<color>])

See Also

@...FILL, @...TO, @...GET, SET COLOR

Description

The SETCOLOR() function is used to determine the current color settings defined with the SETCOLOR() function or the SET COLOR TO command. This function returns a character string in upper case. The optional <color> setting allows the colors to be changed in the same way that the SET COLOR TO <color> command works.

Example

```
oldcolor = setcolor("BR+/N")
```

Products

Recital Mirage Server, Recital Terminal Developer

SETPRC()

Class

Printing

Purpose

Function to set the internal PROW() and PCOL() values

Syntax

SETPRC(<expN1>, <expN2>)

See Also

SET DEVICE, @...SAY..., SET PRINT, SET PRINTER, PROW(), PCOL(), ROW(), COL()

Description

The SETPRC() function will set the internal PROW() and PCOL() values to the specified values. The <expN1> is the printer row position; <expN2> is the printer column position. This function can be useful when sending setup strings to the printer without changing the print head coordinates. It can also be used to suppress page ejects.

Example

```
set device to print
set print on
@0,0 say "printer setup string"
setprc(0,0) && does not send page eject
set print off
set device to screen
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SETPROT()

Class

Environment

Purpose

Function to change current file-protection mask

Syntax

SETPROT(<expC>)

See Also

GETPROT(), CREATE, MODIFY STRUCTURE

Description

The SETPROT() function changes the current file protection mask. The argument to SETPROT() is an <expC> in the same form that is returned by the GETPROT() function. This string is known as the file protection mask. It is the same on all systems, however on non-OpenVMS platforms, the 'System' mask and the 'Delete' item mask are ignored.

Example

```
old_mask = getprot()
setprot("S:RWED,O:RWED,G:RW,W")
use accounts
setprot(old_mask)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SETRESULTSET()

Class

Data Connectivity

Purpose

Function to mark an SQL cursor as a resultset

Syntax

SETRESULTSET(<expN>|<expC>)

See Also

CLEARRESULTSET(), GETRESULTSET(), SQL SELECT

Description

The SETRESULTSET() function marks an SQL cursor as a resultset. Any previous SQL cursor marker is cleared. The workarea number or alias name of the cursor should be specified in <expN> or <expC> respectively. The SETRESULTSET() function is particularly used in returning a resultset from a stored procedure in SQL client/server applications.

The GETRESULTSET() function can be used to return the workarea number of an SQL cursor marked as a resultset by SETRESULTSET(). The resultset marker can be cleared from an SQL cursor using the CLEARRESULTSET() function.

Example

```
function GetExampleCursor
lparameters lcAccountNo
select * from example where account_no = lcAccountNo into cursor curExample
return setresultset("curExample")

open database southwind
GetExampleCursor("00050")
select * from curexample
? "Cleared resultset marker in work area #" + ltrim(str(clearresultset()))
? iif(getresultset() > 0, "Resultset available in work area #" + ltrim(str(getresultset())));
? "No resultsets available")
?
close databases
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SHOWDOCUMENT()

Class

Disk and File Utilities

Purpose

Function to open a URL or a server-based file on a Mirage client

Syntax

SHOWDOCUMENT(<expC1>[, <expC2>[, <expC3>]])

See Also

COPYFILEFROM(), COPYFILETO()

Description

The SHOWDOCUMENT() function is used to open a URL or server-based file on a Mirage client. For server files or file:// URLs, the file can be of any type with a Windows file association on the client. For browser based URLs, the target frame can also be specified. For XML or XLS (Microsoft Excel) files, data can be dynamically loaded from a specified text file.

Parameter	Description
<expC1>	A URL or the name of a file on the server.
<expC2>	The target frame for browser based URLs. If omitted the document is loaded into a new browser window. Ignored for non-browser based URLs. Required if <expC3> is specified.
<expC3>	A templatefile for use with XML or XLS files. This is a text file containing comma separated data that is loaded into the XML or XLS file specified in <expC1>.

If <expC1> specifies an Excel spreadsheet file additional options can be included as arguments. The arguments follow the XLS file name and are preceded by a '?'. For further configuration, directives may be included in the template file. Please see the Mirage documentation for full details on the available option arguments and directives.

Example

// Accessing local files using '.pdf' and '.doc' file associations

```
showdocument("file://C:\Documents and Settings\All Users\Documents\Acrobat.pdf")
```

```
showdocument("file://C:\Documents and Settings\All Users\Documents\WordDoc.doc")
```

// Remote ftp URL

```
showdocument("ftp://ftp.recital.com/RecitalUpdate-Win.exe")
```

// Remote http URL

```
showdocument("http://www.recital.com", "_top")
```

// Loading a template of comma-separated text into an XML file

// and then displaying the XML file

```
showdocument("file://C:\Customers.xml", "_blank", "custdata.txt")
```

```
// Example to download an Excel file, load comma delimited data into it
// and then display it on the client in Microsoft Excel
open database southwind
use suppliers
copy to datafile&(getpid()).txt type delimited
showDocument("myExcelReport.xls?command=create;row=4;col=2", "_blank", "datafile&(getpid()).txt")
```

```
// Example to create an HTML file on the server and
// then invoke the browser on the client to display it
fp = fcreate("htmlfile&(getpid()).htm")
fwrite(fp, "<html>")
fwrite(fp, "<body>")
fwrite(fp, "<table>")
fwrite(fp, "<tr>")
fwrite(fp, "<td>Field1</td>")
fwrite(fp, "<td>Field2</td>")
fwrite(fp, "</tr>")
fwrite(fp, "</table>")
fwrite(fp, "</body>")
fwrite(fp, "</html>")
fclose(fp)
showDocument("htmlfile&(getpid()).htm")
```

Products

Recital Mirage Server

SIGN()

Class

Numeric Data

Purpose

Function to determine the sign of a numeric value

Syntax

SIGN(<expN>)

See Also

IIF(), INT(), ROUND()

Description

The SIGN() function is used to determine the sign of a numeric value. The SIGN() function will return the following values:

<expN> Value	Return Value
Positive	1
Negative	-1
Zero	0

Example

? sign(9876.78)

1

? sign(-87.9)

-1

? sign(0)

0

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SIN()

Class

Numeric Data

Purpose

Function to return the sine of a value

Syntax

SIN(<expN>)

See Also

COS(), TAN(), RTOD(), DTOR(), EXP(), LOG(), LOG10(), PI()

Description

The SIN() function returns the sine of the angle <expN> expressed in radians. The DTOR() function can be used for converting degrees to radians.

Example

set decimals to 6

? sin(dtor(45))

0.707107

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SOUNDEX()

Class

String Data

Purpose

Function to perform phonetic similarity evaluation

Syntax

SOUNDEX(<expC>)

Description

The SOUNDEX() function returns a code that represents the equivalent of <expC>. By comparing the codes that are returned for different words, it is possible to check whether different character strings sound alike. The SOUNDEX() function is particularly useful when used with indexes. The “?” operator, which appears in the query menu, operates a sounds-like comparison between two strings. The comparison returns .T. if two strings are phonetically similar.

Example

```
if soundex(“Jonas”) = soundex(“Jones”)
    ? “These names are phonetically similar.”
endif
index on soundex(name) to similars
display all for “Jones” ? name
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SPACE()

Class

String Data

Purpose

Function to generate string of spaces

Syntax

SPACE(<expN>)

See Also

REPLICATE(), LPAD(), RPAD()

Description

The SPACE() function generates a blank string of <expN> characters.

Example

store space(10) to line1,line2,line3

? len(line1)

10

? empty(line2)

.T.

? type("line3")

C

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SPAWNPID()

Class

Environment

Purpose

Function to return spawned process identity

Syntax

SPAWNPID()

See Also

SPAWN, ACTIVEPID(), CANCELPID()

Description

The SPAWNPID() function returns the identity of the last spawned process. The process identity returned can be used with the ACTIVEPID() or the CANCELPID() functions. A spawned process will only remain active while the user remains logged in to the system. All spawned processes will be terminated when the user logs out. See the RUN command for batch processing.

Example

```
spawn db program
m_activepid = spawnpid()
if activepid(m->m_activepid)
    if cancelpid(m->m_activepid)
        dialog box "Process Canceled "
    else
        dialog box "Unable to Cancel Process."
    endif
else
    dialog box "There is no Process Active".
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SQLVALUES()

Class

Data Connectivity

Purpose

Function to return the single row result of an SQL statement

Syntax

SQLVALUES(<SQL statement>)

See Also

CNTVALUES(), AVGVALUES(), MINVALUES(), MAXVALUES(), SUMVALUES()

Description

The SQLVALUES() function executes the <SQL statement> and returns the result as a string. The required <SQL statement> cannot return multiple rows. If a SELECT statement is specified it must be a singleton select.

The SQLVALUES() function can only be used with an active gateway connection.

Example

```
login "Oracle","node","user","Password"  
// Count the number of records in the employee table  
cTotal = sqlvalues("SELECT COUNT(*) FROM emp")  
total=val(cTotal)  
// Count the number of records in the employee table matching the key "Mr"  
cTotal = sqlvalues("SELECT COUNT(*) FROM emp WHERE title = 'Mr'")
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SQRT()

Class

Numeric Data

Purpose

Function to return square root

Syntax

SQRT(<expN>)

See Also

SIN(), COS(), TAN(), LOG(), INT(), LOG10(), PI(), DTOR(), RTOD()

Description

The SQRT() function returns the square root of the numeric expression <expN>. SQRT() works only with positive numbers.

Example

? sqrt(9.00)

3.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SROWS()

Class

Screen Forms

Purpose

Function to return the number of screen rows

Syntax

SROWS()

See Also

SCOLS(), ROW(), PROW(), PCOL, COL(), MAXROW(), MAXCOL()

Description

The SROWS() function returns the number of rows being used on the current terminal. The return value is of numeric data type. SROWS() may be used to determine how screen or printed listings will be affected by the row setting.

Example

```
?srows()  
25
```

Products

Recital Mirage Server, Recital Terminal Developer

STOD()

Class

Expressions and Type Conversion

Purpose

Function to perform string to date conversion

Syntax

STOD(<expC>)

See Also

CTOD(), DTOS()

Description

The STOD() function converts the character expression <expC>, in the format “YYYYMMDD”, to a date. This function needs the century included in the character string even if SET CENTURY is OFF.

Example

```
? dtos(date())
```

```
20001015
```

```
store dtos(date()) to mDate
```

```
? mDate
```

```
20001015
```

```
? type("mDate")
```

```
D
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STR()

Class

Expressions And Type Conversion

Purpose

Function to perform numeric to string conversion

Syntax

STR(<expN1> [,<expN2> [,<expN3> [,<expN4>]]])

See Also

STRZERO, VAL(), TRANSFORM(), INT()

Description

The STR() function converts the numeric expression <expN1> to a right justified character string of width <expN2>, rounded to <expN3> decimal places. If <expN2> is not specified, the string width defaults to 10. If <expN1> is wider than 10 or <expN2>, the STR() function returns a string of asterisks. If <expN3> is not specified, then <expN1> is treated as an integer.

The string conversion can be returned in octal, decimal or hexadecimal when the optional expression <expN4> is specified as 8, 10 or 16, respectively. If <expN4> is not specified, the string is returned in decimal. If hexadecimal is specified then all alpha characters returned are in lower case. This function can be used in index expressions to add numeric and character field types together.

Example

? str(1999.019,8,2)

1999.02

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STRCOMPARE()

Class

Expressions And Type Conversion

Purpose

Function to compare two strings

Syntax

STRCOMPARE(<expC1>,<expC2>)

See Also

STREXTRACT(), SUBSTR(), CHROVERLAP(), GETUID(), GETGID(), AT()

Description

The STRCOMPARE() function compares two strings, character by character, and returns true (.T.) if the characters in the first string <expC1>, are always greater than or equal to the equivalent characters in the second string <expC2>. The test continues up to the length of the shorter string.

This function is particularly useful in validation routines for controlling access to menu options.

Example

```
cUSERPRIVS = [YYNNNYNN]
if strcmpare(cUSERPRIVS,'NNNNNNNYN')
    SysMaint()
else
    return
endif
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STREXTRACT()

Class

String Data

Purpose

Function to perform string extraction from between specified delimiters

Syntax

STREXTRACT(<expC1>, <expC2> [,<expC3>[,<expN1>[,<expN2>]]])

See Also

AT(), ATNEXT(), LEFT(), MTOS(), RIGHT(), RPAD(), STRTRAN(), STUFF(), SUBSTR()

Description

The STREXTRACT() function extracts a string from <expC1> from between the delimiters specified in <expC2> and <expC3>. The occurrence to be extracted can also be specified.

Parameter	Description
<expC1>	The string to be searched
<expC2>	The start delimiter. If <expC2> is an empty string, the string is extracted from the start of <expC1> to the first occurrence of <expC3>.
<expC3>	The end delimiter. If omitted or empty, the string is extracted from <expC2> to the end of <expC1>
<expN1>	The occurrence of <expC2> at which to start the extraction. If omitted, the first occurrence is extracted.
<expN2>	Optional flags (see below).

Settings:

Bit	Value (additive)	Setting
0	1	Search is case-insensitive.
1	2	End delimiter is not required. If <expC3> is specified, but not found, the string will be extracted from <expC2> to the end of <expC1>
2	4	The delimiters are included in the return string.

Example

```
htmlstring1 = "<title>STREXTRACT()</title>"
```

```
? STREXTRACT(htmlstring1, "<title>", "</title>")
```

```
STREXTRACT()
```

```
? STREXTRACT(htmlstring1, "<title>", "</title>",1,4)
```

```
<title>STREXTRACT()</title>
```

```
? STREXTRACT(htmlstring1, "<", ">",2)
```

```
/title
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STRTOFILE()

Class

String Data

Purpose

Function to write a text string out to a file

Syntax

STRTOFILE(<expC1>, <expC2> [, <expL> | <expN2>])

See Also

AT(), ATNEXT(), FCLOSE(), FCREATE(), FERROR(), FILETOSTR(), FOPEN(), FREADSTR(), FWRITE(), MEMOREAD(), MEMOWRITE(), STREXTRACT(), SUBSTR(), STUFF(), STR(), STRTRAN(), STRZERO(), TEXTEDIT(), TRIM()

Description

The STRTOFILE() function writes the specified string out to the specified file and returns the number of bytes written as a numeric.

Parameter	Description
<expC1>	String expression to be written to the file
<expC2>	Name of the output file. This should include the path if not in the current directory. If it does not exist, it will be created.
<expL>	If True (.T.), the string will be appended to the file's previous contents. If False (.F.), the string will overwrite the file's previous contents. The default is False.
<expN>	Optional settings, see below.

The optional <expN> setting can be any of the following and used instead of <expL>:

Value	Setting
0	The string will overwrite the file's previous contents.
1	The string will be appended to the file's previous contents.
2	The Unicode BOM "FF FE" will be written to the start of the file. The string will overwrite the file's previous contents.
4	The UTF-8 BOM "EF BB BF" will be written to the start of the file. The string will overwrite the file's previous contents.

Example

```
myString = "Hello World"
```

```
// Create file if it does not exist, overwrite if it does
nBytes = strtofile(mystring, "myfile.txt")
```

```
// Append to file's previous contents
nBytes = strtofile(mystring, "myfile.txt",.T.)
```

```
// Create file if it does not exist, overwrite if it does
nBytes = strtofile(mystring, "myfile.txt",0)
```

```
// Append to file's previous contents
nBytes = strtofile(mystring, "myfile.txt",1)

// Create file if it does not exist, overwrite if it does. Include Unicode BOM
nBytes = strtofile(mystring, "myfile.txt",2)

// Create file if it does not exist, overwrite if it does. Include UTF-8 BOM
nBytes = strtofile(mystring, "myfile.txt",4)
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STRTRAN()

Class

String Data

Purpose

Function to search for and replace text within a character string or memo field

Syntax

STRTRAN(<expC1> | <memofield>, [<expC2>, <expC3>, [<expN1> [,<expN2>]]])

See Also

AT(), ATNEXT(), MTOS(), RAT(), STR(), STUFF(), STRZERO(), STREXTRACT(), SUBSTR(), TRIM(),

Description

The STRTRAN() function will search <expC1> or <memofield> and replace text within <expC1> or <memofield> with <expC3> wherever the occurrence of <expC2> is found. The optional <expN1> specifies the start position for the search in <expC1>. If this is not specified then the default is 1. The optional <expN2> specifies the number of occurrences of <expC2> to replace with <expC3>. If this is not specified, then all occurrences starting from <expN1> are replaced. This function can be used to remove blank spaces in character strings. The STRTRAN() function returns a character expression that contains the result of the string translation.

Example

```
? strtran("Hello World", "ello", "i")  
Hi World
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STRZERO()

Class

Expressions And Type Conversion

Purpose

Function to perform numeric to string conversion

Syntax

STRZERO(<expN1>, <expN2> [,<expN3>])

See Also

STR(), LPAD(), RPAD(), VAL(), TRANSFORM(), INT()

Description

The STRZERO() function converts the numeric expression <expN1> to a right justified character string of width <expN2>, rounded to <expN3> decimal places. If <expN2> is not specified, then a string of 10 characters is returned. If <expN3> is not specified, then <expN1> is treated as an integer. The STRZERO() function operates in the same way as the STR() function except that zeroes are placed at the start of the string instead of spaces. This function is very useful for storing numeric numbers in character fields so that they appear in the expected sequential order.

Example

```
? strzero(1234,8)
```

00001234

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

STUFF()

Class

String Data

Purpose

Function to perform character replacement

Syntax

STUFF(<expC1>, <expN1>, <expN2>, <expC2>)

See Also

AT(), ATNEXT(), STREXTRACT(), SUBSTR(), LEFT(), RIGHT(), STRTRAN(), LPAD(), RPAD()

Description

The STUFF() function returns a character string which is the result of replacing a string in <expC1> with <expC2>. The starting position for the replacement is <expN1>, and the number of characters to replace is <expN2>. It can also be used to remove text from a string.

Example

```
? stuff("Life is as you take it",9,13,"grea")
```

Life is great

```
? stuff("Friends are welcome",9,7,"")
```

Friends come

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SUBSTR()

Class

String Data

Purpose

Function to perform substring extraction

Syntax

SUBSTR(<expC> | <memofield>, <expN1> [,<expN2>])

See Also

AT(), ATNEXT(), LEFT(), LPAD(), MTOS(), RIGHT(), RPAD(), STREXTRACT(), STRTRAN(), STUFF()

Description

The SUBSTR() function extracts a substring from <expC> or <memofield> starting at position <expN1> of width <expN2>. If <expN1> is negative, the extraction starts from the right side of the string. If <expN2> is omitted, a substring will be extracted up to the end of <expC>. When used in conjunction with the AT() function, the SUBSTR() function is very useful for extracting selected information from character strings.

Example

```
m_username = "GTW , Gary T West "  
? substr(m->m_username,at(',',m->m_username)+1)
```

Gary T West

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SUMVALUES()

Class

Fields and Records

Purpose

Function to returns the total value of a column for a matching series of keys

Syntax

SUMVALUES(<expN> [, <for condition> [, <key expression>]])

See Also

CNTVALUES(), AVGVALUES(), MINVALUES(), MAXVALUES(), SQLVALUES()

Description

The SUMVALUES() function returns the total value of <expN> for a matching series of keys. The required <expN> parameter must be a column name from a table. If none of the optional parameters is specified, then the number of keys matching the current key is returned. An optional <for condition> can be specified to restrict the rows included in the sum operation. You can also optionally specify a <key expression> to perform the sum on instead of the current key.

After completion, all record pointers and indexes are returned to their original positions.

Example

use customer order title

```
// Sum the balance column for the number of records matching the current key
```

```
total_balance = sumvalues(balance)
```

```
// Sum the balance column for the number of records matching the key "Mr"
```

```
m_male = sumvalues(balance, .T., "Mr")
```

```
//Sum the balance column for the number of records matching the current key with an expiry date < today
```

```
m_expired = sumvalues(balance, expiry < date())
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

SYS()

Class

Environment

Purpose

Functions to return various system and miscellaneous information

Syntax

SYS(<expN>)

See Also

Description

The SYS() function returns miscellaneous information depending on <expN>. The arguments and operation of this function are as follows:

Parameter	Description
SYS(0)	Returns complete host system information
SYS(1)	Returns the current system date as a Julian day number in a character string
SYS(2)	Returns the number of seconds elapsed since midnight in a character string
SYS(3)	Returns a unique filename
SYS(5)	Returns the current default directory
SYS(6)	Returns the current PRINT device as set by the SET PRINTER TO command
SYS(7 [,w])	Returns the name of the current FORMAT file. If no workarea 'w' is specified the current workarea is used
SYS(9)	Returns the Recital license serial number
SYS(10,d)	Returns the numeric day number 'd' as a character string
SYS(11,s)	Returns a date or character string 's' as a character type Julian day number
SYS(12)	Returns remaining available memory in bytes
SYS(13)	Returns the status of the printer as "READY" or "OFF-LINE"
SYS(14,n[,w])	Returns the index expression for index 'n' in workarea 'w'. If 'w' is omitted then the current workarea is used
SYS(16)	Returns the name of the currently executing program
SYS(17)	Returns the CPU type number
SYS(18)	Returns the name of the GET field being entered
SYS(21)	Returns the order number of the master index. Note: if the master index is a tag, SYS(21) will always return 1.
SYS(22)	Returns the master index: the tagname or single index file name. If the master index is a tag, then SYS(22) returns the tag name in upper case. If the master index is a single index, then SYS(22) returns the index file name in lower case, including the file extension.
SYS(100)	Returns the status of the CONSOLE setting as "ON" or "OFF"
SYS(101)	Returns the DEVICE setting as "SCREEN" or "PRINT"
SYS(102)	Returns the PRINT setting as "ON" or "OFF"
SYS(103)	Returns the TALK setting as "ON" or "OFF"
SYS(2000,s[,1])	Returns the name of the first file matching the pattern 's'. If the

	third parameter is included the next matching file is returned
SYS(2001,s[,1])	Returns the current value of the set option 's'. The optional '1' can be used to return the character setting of certain SET COMMANDS, e.g. ALTERNATE
SYS(2002[,1])	Turns the cursor OFF (sys(2002)) or ON (sys(2002,1))
SYS(2003)	Returns the default directory
SYS(2004)	Returns the starting directory for the current process
SYS(2005)	Returns the currently available resource file
SYS(2006)	Returns "VGA/Color" for color terminals, "Mono" for non-color terminals.
SYS(2008 [,<expC> [,<expN>]])	Language compatibility only.
SYS(2010)	Return the maximum number of open files for the current user
SYS(2011)	Returns the current lock status, no lock is placed on the record by this command
SYS(2012, [<expN> <expC>])	Returns the memo field block size for dBASE and FoxPro database files. SET COMPATIBLE TO DBASE FOXPRO must be set.
SYS(2015)	Returns a unique 10-character string that begins with an underscore
SYS(2017)	Re display sign on screen
SYS(2019)	Get name and location of current configuration file
SYS(3000)	Returns the options string from the Recital license
SYS(5000)	Returns amount of currently allocated physical memory for strings
SYS(5001)	Returns amount of memory allocated for strings
SYS(5002)	Returns amount of currently allocated memory for indexes
SYS(5003)	Returns amount of currently allocated memory for symbol table nodes
SYS(6000,n)	Returns the assigned Hot-Key procedure for key 'n', as defined by the SET KEY TO command
SYS(6001)	Returns current DO level as a character string
SYS(9000)	Returns .T. (true) if the runtime environment is active, .F. (false) if the development environment is active.
SYS(9001)	Returns .T. (true) if the environment was invoked with the system windows defined, .F. (false) otherwise.

Example

```
? sys(5)
/usr/recital/UD/
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TAG()

Class

Indexing

Purpose

Function to return the name of an index file or index tag

Syntax

TAG([<.dbx filename>,<expN> [,<alias>]])

See Also

DBF(), KEY(), MDX(), NDX(), ORDER(), TAGCOUNT(), TAGNO()

Description

The TAG() function returns the tag name of the specified .dbx file, or the filename of a single (.ndx) file. With no parameters, the TAG() function operates in the current workarea on the master index. If no indexes are active, TAG() will return a null string. An error will be given if an invalid alias name is given.

Parameter	Description
<.dbx filename>	The .dbx file which contains the tag
<expN>	The number of the tag
<alias>	The workarea in which to operate. Specified in any of the following ways: A workarea number: 1-DB_MAXWKA A workarea letter: A-Z (a-z) excluding M (m). A table alias. The alias can be specified in the USE command. If not specified, the table basename is used.

Example

```
?tag(2)
```

```
LAST_NAME
```

```
tag("alternate.dbx",3)
```

```
ZIPCODE
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TAGCOUNT()

Class

Indexing

Purpose

Function to count the number of tags in the active .dbx file

Syntax

TAGCOUNT([<.dbx filename> [,<alias>]])

See Also

KEY(), MDX(), TAG(), TAGNO()

Description

The TAGCOUNT() function returns the number of tags in a particular .dbx file. With no parameters, the TAGCOUNT() function operates on the currently active .dbx file in the currently selected workarea. A .dbx file is active if one of the tags contained within the .dbx file is the master index order. The TAGCOUNT() function will return a zero if the file is a single (.ndx) file or if there are no indexes active.

Parameter	Description
<.dbx filename>	The .dbx file which contains the tag. This must be specified if the <alias> is used.
<alias>	The workarea in which to operate. Specified in any of the following ways: A workarea number: 1-DB_MAXWKA A workarea letter: A-Z (a-z) excluding M (m). A table alias. The alias can be specified in the USE command. If not specified, the table basename is used.

Example

```
use accounts
```

```
? tagcount()
```

```
4
```

```
set order to 0
```

```
?tagcount()
```

```
0
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TAGNO()

Class

Indexing

Purpose

Function to return the number of an index tag

Syntax

TAGNO(<tag name>[,<.dbx filename>[,<alias>]])

See Also

KEY(), MDX(), ORDER(), TAG(), SET INDEX, SET ORDER, TAGCOUNT()

Description

The TAGNO() function returns the tag number for the specified index tag. Tag numbers are assigned as they are added to multiple index files. The first tag added is number 1, the second is number 2, and so on. If more than one .dbx file is active, tag numbers are assigned sequentially starting with the first production index tag unless there are single (.ndx) index files active in the workarea. In this case, all .ndx files will be given a tag number first, then tags in the production index will be numbered. Non production .dbx files will be assigned tag numbers last. With no <alias> specified, the TAGNO() function operates in the currently selected workarea.

Parameter	Description
<tag name>	The name of the tag
<.dbx filename>	The .dbx file which contains the tag. Only required if the same tag name exists in multiple .dbx files or the <alias> is specified..
<alias>	The workarea in which to operate. Specified in any of the following ways: A workarea number: 1-DB_MAXWKA A workarea letter: A-Z (a-z) excluding M (m). A table alias. The alias can be specified in the USE command. If not specified, the table basename is used.

Example

```
?tagno("city")
```

3

```
?tagno("last_name", "customer.dbx", "customer")
```

1

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TAN()

Class

Numeric Data

Purpose

Function to return tangent

Syntax

TAN(<expN>)

See Also

SIN(), COS(), RTOD(), DTOR(), LOG(), LOG10(), PI(), EXP()

Description

The TAN() function returns the tangent of the angle <expN> expressed in radians. The DTOR() function can be used for converting degrees to radians.

Example

set decimals to 6

? tan(dtor(44))

0.965689

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TARGET()

Class

Table Basics

Purpose

Function to return target workarea of a relation

Syntax

TARGET(<expN>)

See Also

DBRSELECT(), SET RELATION

Description

The TARGET() function is synonymous with the DBRSELECT() function. The TARGET() function returns the target workarea of a specified relation defined in the current workarea. By default, the Recital environment supports 20 workareas but this can be increased, by setting the environment symbol DB_MAXWKA to a higher value, which allows for up to (DB_MAXWKA-1) relationships. The <expN> is the position of the corresponding relation in the list of previously defined relations. If there are no relations defined for the <expN> on the currently selected table, the TARGET() function will return 0.

Example

```
use shows in 2 index people
use patrons in 1
set relation to patron_code into shows
? target(1)
2
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TEXTEDIT()

Class

Screen Forms

Purpose

Function to edit a text file in a pop-up window

Syntax

TEXTEDIT(<filename>, [<expN1>, <expN2>, <expN3>, <expN4>, <expL1> [,<expC> [,<expL2> [,<expL3>]]])

See Also

FCLOSE(), FREADSTR(), FOPEN(), FWRITE(), SET MEMOWINDOW, SET MEMOWIDTH, MEMOWRITE(), MEMLINES(), MLCOUNT(), MEMOLINE() MLINE(), MEMOEDIT(), MEMOREAD(), MEMOTRAN(), HARDCR()

Description

The TEXTEDIT() function allows ASCII text file <filename> to be edited in a pop-up window. It returns .T. if the file was modified and .F. otherwise. Coordinates <row>, <col> to <endrow>, <endcol>, may be optionally specified as <expN1> to <expN4>. If no screen coordinates are specified, TEXTEDIT() will default to the current SET MEMOWINDOW coordinates. If the update parameter <expL1> is .T., then the <filename> can be modified. If <expL1> is .F., then the <filename> can only be viewed. By default, <expL1> is .T..

If the optional <expC> is specified, then the window is bordered and labeled with <expC>. The screen is automatically saved and then restored when SET MEMOCLEAR is ON. If the optional wrap parameter <expL2> is .T. or the [REFORMAT] key is pressed during the editing session, the text is word-wrapped when saved. If the optional <expL3> is .T., then the TEXTEDIT() displays a window in reverse video. Press the [HELP] key to obtain details of the keys that are available during the edit session.

The functions MEMOREAD() and MEMOWRITE() are used for reading text files to and from memo fields. The MEMOLINE() and MLINE() memo editing functions are used to extract a line of text from a memo field. Reading text files into memo fields is strongly recommended for enabling users to edit text files from within applications. Memo files can be edited directly within the same type of window as TEXTEDIT() by using the MEMOEDIT() function. Please note that memo-editing functions will not work with text files, and text-editing functions will not work with memo fields.

Example

```
// This procedure will
// be called if the [HELP] key is pressed
procedure helpme
set message to "Hit ^C for help, ^G to exit."
if varread() = "part_des"
    textedit("part_des.txt",5,10,20,70,.F.,"Part Description",.F.)
elseif varread() = "name"
    textedit("s_name.txt",5,10,20,70,.F.,"Supplier Name",.F.)
else
    dialog box "No help file available."
endif
return
```


Products

Recital Terminal Developer

TIME()

Class

Date and Time Data

Purpose

Function to return the current system time

Syntax

TIME([<expN>])

See Also

AMPM(), CTOT(), DATE(), DATETIME(), ELAPTIME(), HOUR(), HOURS(), MINUTE(), MINUTES(), SEC(), SECONDS(), SECS(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), VALIDTIME(), SET CLOCK, SET CLOCKRATE, SET SECONDS, SET VAXTIME

Description

The TIME() function returns the current system time as a character string in the format HH:MM:SS. The TIME() function always returns the time in 24 hour format, and is not affected by the SET HOURS TO [12/24] command. The optional numeric expression <expN> must result in a non-zero value, and when specified, the current time including hundredths of seconds is returned. This is provided for Xbase language compatibility. The TIME() function will always return hundredths of seconds as 00.

Example

? time()

17:47:24

? time(1)

17:47:31.00

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TIMESTAMP()

Class

Date and Time Data

Purpose

Function to return the current system date and time

Syntax

TIMESTAMP()

See Also

CROW(), DATE(), DAY(), DOW(), DTOS(), MONTH(), SECONDS(), SECS(), TIME(), WEEK(), YEAR(), STOD()

Description

The TIMESTAMP() function is used for returning the current system date and time in a fixed format. The return value is a 19-character string in the format “YYYY.MM.DD hh:mm:ss”. The return value is unaffected by the SET DATE, SET CENTURY and SET MARK settings.

Example

```
update_time = timestamp()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TMPNAM()

Class

Disk and File Utilities

Purpose

Function to return temporary file name function

Syntax

TMPNAM()

See Also

GETENV(), GETGID(), GETUID(), GETPID(), RAND()

Description

The TMPNAM() function returns a character string containing a unique temporary filename. Filenames returned by the TMPNAM() function have the extension “.tmp”. The TMPNAM() function is particularly useful in multi-user applications where a unique temporary file can be created for each process.

Example

```
name = tmpnam()  
report form Listing for event = “BALLET” to file &name  
print &name
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TONE()

Class

Input/Output

Purpose

Function to sound the terminal bell

Syntax

TONE(<expN1>, <expN2>)

See Also

SET BELL, CHR()

Description

The TONE() function sounds the bell on your terminal. It can be used for program controlled warning sounds. The frequency (in hertz) of the emitted sound is determined by the integer <expN1>. The duration of the sound is determined by <expN2> expressed in 1/18ths of a second.

Example

tone(340,5)

Products

Recital Mirage Server, Recital Terminal Developer

TRANSFORM()

Class

String Data

Purpose

Function to perform picture formatting

Syntax

TRANSFORM(<exp>, <expC>)

See Also

STR(), IIF()

Description

The TRANSFORM() function allows formatting of <exp> using a picture string <expC> in a similar way to the @...GET command. TRANSFORM() returns a character string formatted as requested. TRANSFORM() is particularly useful for redefining field widths when formatting reports or forms.

Example

```
total = 3001.23
? transform(total, "$$$$9.99")
$3001.23
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TRIM()

Class

String Data

Purpose

Function to remove trailing blanks

Syntax

TRIM(<expC>)

See Also

RTRIM(), LTRIM(), ALLTRIM()

Description

The TRIM() function is synonymous with the RTRIM() function. It removes trailing blank characters from the character expression <expC>. This function should only be used in conjunction with the RPAD() function when used in index expressions, since index keys must be fixed length.

Example

```
fname = "Christopher   "
```

```
sname = "Thompson   "
```

```
? len(fname)
```

```
20
```

```
? len(trim(fname))
```

```
11
```

```
? trim(fname)++ "++trim(sname)
```

```
Christopher Thompson
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TSPOS()

Class

Text Searching

Purpose

Function to search a Text Search Index for one or more words

Syntax

TSPOS(<expC>,<expN>)

See Also

INDEX, KEY(), LOWER(), NDX(), TSWORD(), UPPER(), SET INDEX, SET ORDER, SET
TSLENGTH, DB_TSINDEX

Description

The TSPOS() function is used to search the currently active Text Search Index. If the search is successful, the record pointer is positioned to the matching record and the record number is returned. If no matching record is found, the TSPOS() function returns -1.

The <expC> is the word list for which to search. Words should be space-separated. The <expN> is the occurrence number for which to search. If <expN> is 0, the index is searched for the first match to a new search or the next match based on a previous search.

The UPPER() or LOWER() functions can be used when creating indexes and when conducting index searches to make them case-insensitive.

The DB_TSINDEX environment variable / symbol must be set to “ON” when building or using Text Search Indexes.

Example

use example

index on tsword(first_name+last_name,1) to namesearch

? tspos(“John”,0)

42

? tspos(“John”,0)

43

? tspos(“John”,0)

-1

? tspos(“John”,1)

42

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TSTRING()

Class

Data Type Conversion

Purpose

Function to convert seconds to a time-string

Syntax

TSTRING(<expN>)

See Also

TIME(), VALIDTIME(), SECS(), DAYS(), ELAPTIME(), HOURS(), MINUTES(), SECONDS(), AMPM()

Description

The TSTRING() function converts <expN> seconds to a time-string in the format “HH:MM:SS”. The DAYS() function can be used to return the duration of any remaining days in the <expN> seconds.

Example

? tstring(36610)

10:10:10

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TSWORD()

Class

Text Searching

Purpose

Function to extract a word from a text string or create the key for a Text Search Index

Syntax

TSWORD(<expC>,<expN>)

See Also

INDEX, KEY(), LOWER(), MTOS(), NDX(), TSPOS(), UPPER(), SET INDEX, SET ORDER, SET TSLength, DB_TSINDEX

Description

The TSWORD() function is used to extract a word from a text string or to create the key for a Text Search Index. The <expC> is the character string from which to extract the word or on which the index key is built. The <expN> is the number of the word to be extracted or indexed, starting from 1. Words are defined as a series of characters separated by one or more spaces.

Memo fields can be included in Text Search Index keys by using the MTOS() memo-to-string conversion function.

The UPPER() or LOWER() functions can be used when creating indexes and when conducting index searches to make them case-insensitive.

The DB_TSINDEX environment variable / symbol must be set to "ON" when building or using Text Search Indexes.

Example

```
? tsword("Hello World",1)
```

Hello

use example

```
index on tsword(street,1) to streetsearch
```

```
nRecno = tspos("Road",1)
```

```
if nRecno > 0
```

```
  do while nRecno > 0
```

```
    ? "Record found. Record #",nRecno
```

```
    ?
```

```
    nRecno = tspos("Road",0)
```

```
  enddo
```

```
endif
```

use example

```
index on tsword(first_name+last_name,1) to namesearch
```

```
? tspos("Siegel",0)
```

```
1
```

```
? tspos("Al Siegel",0)
```

```
1
```

use example

index on tsword(mtos(notes),1) to memosearch

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TTOC()

Class

Expressions and Type Conversion

Purpose

Function to convert a datetime expression to a string value in an optionally specified format

Syntax

TTOC(<expT> [, <expN>])

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The TTOC() function converts the datetime expression <expT> to a string value. By default, the date part of the string returned will conform to the current SET DATE, SET MARK and SET CENTURY settings, in the same format as DTOC(). The time will be returned in the format hh:mm:ss AM|PM. If the expression to be converted contains no time information, 12:00:00 AM will be assumed. If SET SECONDS is OFF (ON by default), no seconds will be displayed. The SET HOURS set command determines whether hours are shown in 24 hour format or in 12 hour format with AM | PM postfix.

<expN>

The optional <expN> can be used to specify the format of the return value:

<expN>	Format
0	As defaults
1	YYYYMMDDhhmmss
2	Time only: hh:mm:ss AM PM (SET SECONDS ON) or hh:mm AM PM (SET SECONDS OFF)

Example

set date american

? ttoc({^2004-03-29 10:15:43 AM})

03/29/2004 10:15:43 AM

? ttoc({^2004-03-29 10:15:43 AM},1)

20040329101543

? ttoc({^2004-03-29 10:15:43 AM},2)

10:15:43 AM

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TTOD()

Class

Expressions and Type Conversion

Purpose

Function to convert datetime to date

Syntax

TTOD(<expT>)

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The TTOD() function is the datetime to date conversion function. It converts the <expT> datetime expression specified to a date. The <expT> must be a valid datetime, or the TTOD() function will return an empty date. The date returned will conform to the current SET DATE, SET MARK and SET CENTURY settings. For example, the default settings, SET DATE AMERICAN and SET CENTURY ON, will return a date in the format “MM/DD/YYYY”.

Example

```
set date american
set century on
mdate = ttod({^2004-03-29 10:15:43 AM})
? mdate
03/29/2004
? type("mdate")
D
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TXNISOLATION()

Class

Transaction Processing

Purpose

Function to return the current Transaction Isolation Level setting

Syntax

TXNISOLATION()

See Also

BEGIN...END TRANSACTION, RESET IN, ROLLBACK, COMPLETED(), ROLLBACK(), TXNLEVEL(), SET ROLLBACK, SET TRANSACTION

Description

The TXNISOLATION() function returns the current Transaction Isolation setting. The Transaction Isolation Level is set using the SET TRANSACTION [ISOLATION LEVEL <level>] command.

The Transaction Isolation Level can be any of the following, please see the SET TRANSACTION Set Command for full details:

- **SERIALIZABLE**
- **REPEATABLE READ**
- **READ COMMITTED**
- **READ UNCOMMITTED**

Example

```
set transaction isolation level read uncommitted;  
cTrans = txnisolation()
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TXNLEVEL()

Class

Transaction Processing

Purpose

Function to return the current Transaction Level number

Syntax

TXNLEVEL()

See Also

BEGIN...END TRANSACTION, COMMIT, RESET IN, ROLLBACK, COMPLETED(), ROLLBACK(), TXNISOLATION(), SET ROLLBACK, SET TRANSACTION

Description

The TXNLEVEL() function returns the current Transaction level as a number. Transactions can be nested by issuing a further BEGIN TRANSACTION when a transaction is already active. If no transaction is active, the TXNLEVEL() function returns 0. A transaction and any transactions nested within it are closed when a COMMIT, ROLLBACK or END TRANSACTION is issued.

Example

```
// config.db
set sql to recital
set sql on
// end of config.db

// Nested Transactions
? txnlevel() && 0
BEGIN TRANSACTION trans1;
? txnlevel() && 1
INSERT INTO customer
    (TITLE, LAST_NAME, FIRST_NAME, INITIAL, STREET,
     CITY, STATE, ZIP,LIMIT, START_DATE)
VALUES
    ('Ms', 'Jones', 'Susan', 'B', '177 High Street','Beverly', 'MA', '01915', 2000, date());
INSERT INTO accounts (ORD_VALUE) VALUES (30);
BEGIN TRANSACTION trans2;
? txnlevel() && 2
INSERT INTO accounts (ORD_VALUE) VALUES (60);
// Commit the trans1 transaction and any transactions
// nested in trans1
COMMIT TRANSACTION trans1;
? txnlevel() && 0
END TRANSACTION;
? txnlevel() && 0
// End of program
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

TYPE()

Class

Expressions and Type Conversion

Purpose

Function to return a letter code that represents a data type

Syntax

TYPE(<expC>)

See Also

ERROR(), ERRNO(), MESSAGE(), ON ERROR

Description

The TYPE() function returns a letter code which represents the data type of the expression in <expC>. The return value of the letter code is a character string from the table below.

Data type	Return Value
ARRAY (STATIC)	A
ARRAY (DYNAMIC)	O
BIGINT	N
BIT	L
BYTE	N
CHARACTER	C
CURRENCY	Y
DATE	D
DATETIME	T
DECIMAL	N
DOUBLE	N
FLOAT	N
GENERAL	G
INTEGER	N
LOGICAL	L
LONG VARCHAR	M
LONG VARBINARY	G
MEDIUMINT	N
MEMO	M
NUMERIC	N
OBJECT	O
PACKED	N
QUAD	N
REAL	N
SHORT	N
SMALLINT	N
Syntax error	U
TEXT	M
TIME	C
TIMESTAMP	T
TINYINT	N
Undefined	U

VAXDATE	C
VARCHAR	C
ZONED NUMERIC	C

If <expC> contains a syntax error, or an undeclared variable, then TYPE() returns 'U'. TYPE() will also return a 'U' for an undefined variable if SET CLIPPER is ON. TYPE() is primarily used to check for the existence of a variable, or the syntax of an expression.

Example

```
i = 10
? type("i")
N
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

UNDERLINE()

Class

Input/Output

Purpose

Function to display underlined text

Syntax

UNDERLINE(<expC>)

See Also

BLINK(), BOLD(), REVERSE(), SET SCREENIO

Description

The UNDERLINE() function displays the specified character expression <expC> underlined on the screen. Key words in MESSAGE commands can be underlined to make them stand out. SET SCREENIO must be ON (default is off), for the UNDERLINE() function to operate.

Example

@3,3 say "Hit " + underline("ENTER") + " key."

Products

Recital Mirage Server, Recital Terminal Developer

UNIQUE()

Class

Indexing

Purpose

Function to test for unique indexes

Syntax

UNIQUE(<expN>)

See Also

INDEXKEY(), KEY(), INDEXORDER(), SET UNIQUE ON, INDEX

Description

The UNIQUE() function returns .T. if the specified index <expN> is unique, and .F. if it is not unique. <expN> is the index number from 0 to 20. If zero is specified then UNIQUE() tests the master index. For tagged multiple indexes, only the master index can be tested.

Example

```
use accounts
index on name to name unique
? unique(0)
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

UNIQUEROWID()

Class

Data Connectivity

Purpose

Function to return a unique row identifier for the current table or table in the specified workarea

Syntax

UNIQUEROWID([<expN>])

See Also

ALIAS(), DBF(), FCOUNT(), FIELD(), FILTER(), FMT(), INDEXKEY(), NDX (), READVAR(), SELECT(), WORKAREA()

Description

The UNIQUEROWID() function returns the name of a unique row identifier for the current table. The optional <expN> can be used to specify a workarea number. If there is a table open in the workarea number specified in <expN>, UNIQUEROWID() will return the name of a unique row identifier for that table.

Example

```
use demo
select state
list state.state, (uniquerowid(select("state")))
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

UPDATED()

Class

Screen Forms

Purpose

Function to check for data changes following reads

Syntax

UPDATED()

See Also

@...GET, READ, SET QUERYMODE, CHANGE, EDIT, CREATE SCREEN, DBF()

Description

The UPDATED() function will return .T. if the last READ changed any of the data in the associated GETs. It will also return .T. if any changes to the table are made through a user designed form in CHANGE or EDIT mode. This function can be used to check if changes were made to the table so that appropriate procedures may be called if necessary.

Example

```
if updated()
    do upd_master
endif
```

Products

Recital Mirage Server, Recital Terminal Developer

UPPER()

Class

String Data

Purpose

Function to convert character expression to upper case

Syntax

UPPER(<expC>)

See Also

LOWER(), ISUPPER(), ISLOWER(), DESCEND(), PROPER()

Description

The UPPER() function converts a character expression, <expC>, to upper case. This function is particularly useful with index expressions, as when the FIND key is pressed in forms, a search string typed in upper case will match both upper and lower case characters stored in the table.

Example

? upper("Recital")

RECITAL

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

USED()

Class

Table Basics

Purpose

Function to determine if a table is open

Syntax

USED([<expN> | <expC>])

See Also

USE, SET EXCLUSIVE, SET DCACHE, SET CACHELOAD, DBF(), NDX(), NETERR()

Description

The USED() function with no parameter specified, returns .T. if there is a table open in the current workarea and .F. if the current workarea is empty. <expN> is a numeric expression giving a workarea number: 1-DB_MAXWKA in which to check for an open table. <expC> is a character expression giving a workarea letter, (A-Z/a-z excluding M/m) or a table alias in which to check for an open table. The alias can be specified in the USE command. If not specified, the table basename is used.

Example

```
// The demo view bridge opens
// the tables customer, accounts,
// state and products in workareas 1 – 4
use demo
? select()
    1
? used()
.T.
? used("product")
.T.
? used(5)
.F.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

USER()

Class

Environment

Purpose

Function to return the login name of the current user

Syntax

USER()

See Also

DISPLAY USER, LIST USER, GETPID(), GETUID(), GETGID(), ID()

Description

The USER() function returns the login name of the current user. The login name is assigned by the operating system. This function is synonymous with the ID() function.

Example

```
?user()
```

william

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

USERLOG()

Class

Error Trapping and Debugging

Purpose

To log information to a user-specific log file for debugging or audit trail purposes.

Syntax

USERLOG(<expC>)

See Also

DB_USERLOG, ERRNO(), ERROR(), LINENO(), MESSAGE (), ON ERROR, RETRY

Description

The USERLOG() function is used to log information to a user-specific log file for debugging or audit trail purposes. The USERLOG() function writes the specified character string <expC> into the file specified by the DB_USERLOG environment variable / symbol.

Example

```
userlog("Entered Main Menu @ " + time())
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

VAL()

Class

Expressions and Type Conversion

Purpose

Function to perform character string to numeric conversion

Syntax

VAL(<expC>)

See Also

SET FIXED, SET DECIMAL, INT(), STR(), TRANSFORM(), ROUND()

Description

The VAL() function converts the character expression <expC> into a numeric value. If SET FIXED is on, then the value of the string will be rounded to the nearest decimal place according to the SET DECIMALS command.

Example

```
string = "1234"
```

```
? val(string)
```

1234

```
set decimals to 2
```

```
set fixed on
```

```
? val("678.7615")
```

678.76

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

VALIDTIME()

Class

Date and Time Data

Purpose

Function to check the validity of a time-string

Syntax

VALIDTIME(<time-string>)

See Also

TIME(), SECS(), TSTRING(), DAYS(), ELAPTIME(), HOURS(), MINUTES(), SECONDS(), AMPM()

Description

The VALIDTIME() function the validity of the specified <time-string>. A valid <time-string> must be in a 24 hour "HH:MM:SS" format. The VALIDTIME() function returns .T. for a valid time-string and .F. otherwise.

Example

```
store "00:00" to m_time
@5,5 get m_time picture "99:99";
    valid validtime(m->m_time+":00");
    error "Invalid time. Press any key.";
    help "Enter the start time."
read
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

VARREAD()

Class

Screen Forms

Purpose

Function to return name of the @...GET variable being processed by a READ

Syntax

VARREAD()

See Also

@...GET...VALIDATE, RECNO(), PCOUNT(), READVAR()

Description

The VARREAD() function is synonymous with the READVAR() function. The VARREAD() function returns the name of the current @...GET variable being processed by a READ command. The VARREAD() function is most useful if used in conjunction with the @...GET...VALIDATE command and a validation procedure used to validate more than one field/variable.

Example

```
// Pop-up choice list called from a field validation
procedure choices
parameters m_input
if the [HELP] key is pressed
    if m_input = "HELP"
        m_exp = iif(varread()="acc_no","account", "supplier")
        select codes
        save screen
        set message to "Select code. ^C to abandon."
        menu browse &m_exp at 5,30 to 15,40 quit clear
        restore screen
        // if a selection was made
        if not empty(menuitem(),3)
            set fieldval to left(menuitem(),3)
            set validate on
        endif
        select accounts
    else
        // validate as required
    endif
endif
return
```

Products

Recital Mirage Server, Recital Terminal Developer

VARTYPE()

Class

Expressions and Type Conversion

Purpose

Function to return a letter code that represents a data type

Syntax

VARTYPE(<exp>[,<expL>])

See Also

ERROR(), ERRNO(), MESSAGE(), TYPE(), ON ERROR

Description

The VARTYPE() function returns a letter code which represents the data type of the expression in <exp>. The return value of the letter code is a character string from the table below.

Data type	Return Value
ARRAY (STATIC)	A
ARRAY (DYNAMIC)	O
BIGINT	N
BIT	L
BYTE	N
CHARACTER	C
CURRENCY	Y
DATE	D
DATETIME	T
DECIMAL	N
DOUBLE	N
FLOAT	N
GENERAL	G
INTEGER	N
LOGICAL	L
LONG VARCHAR	M
LONG VARBINARY	G
MEDIUMINT	N
MEMO	M
NULL	X
NUMERIC	N
OBJECT	O
PACKED	N
QUAD	N
REAL	N
SHORT	N
SMALLINT	N
Syntax error	U
TEXT	M
TIME	C
TIMESTAMP	T
TINYINT	N

Undefined	U
VAXDATE	C
VARCHAR	C
ZONED NUMERIC	C

If <exp> contains a syntax error, or an undeclared variable, then VARTYPE() returns 'U'. VARTYPE() will also return a 'U' for an undefined variable if SET CLIPPER is ON. Unlike the TYPE() function, VARTYPE() does not require the expression for evaluation to be enclosed in quotes.

The optional <expL> is used to determine whether VARTYPE() returns the data type for expressions which evaluate to null (.NULL.). If <expL> is True (.T.) the data type is returned for <exp>. If <expL> is False (.F.), then VARTYPE() returns "X".

Example

i = 10

? vartype(i)

N

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

VERSION()

Class

Environment

Purpose

Function to return software version information

Syntax

VERSION([<expN>])

See Also

OS()

Description

The VERSION() function returns a character string containing the Recital product name and version number. If the optional <expN> is specified, further version and release information is returned.

<expN> Value	Return Value
1	Compilation date
2	Patch Release number
3	License error code. The string 'No License error code' is returned if the product is correctly licensed.

Example

? version()

Recital UD Enterprise Edition Version 8.0 for Unix

? version(1)

Compiled on Tue Oct 26 17:03:08 GMT 1999

? version(2)

8.0.8 Started 28-May-1999

? version(3)

No License error code

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

VTOD()

Class

Expressions and Type Conversion

Purpose

Function to convert OpenVMS dates to Recital dates

Syntax

VTOD(<expC>)

See Also

DTOV(), SET VAXTIME, DAY(), CDOW(), CTOD(), CMONTH(), DATE(), DOW(), DTOC(), MONTH(), TIME(), WEEK(), YEAR(), STOD(), DTOS(), DMY(), MDY()

Description

The VTOD() function converts VAXdates to Recital dates so that date arithmetic can be performed. The resulting date conforms to the current settings of SET DATE, SET CENTURY and SET MARK. <expC> must be an 11 character string containing a VAXdate in the format DAY-MONTH-YEAR. SET VAXTIME must be off, so that no time-string is included.

Example

```
? vtod("17-MAY-1990")+7  
05/24/1990
```

Products

All OpenVMS

WCOLS()

Class

Screen Windows

Purpose

Function to return the number of columns in a window

Syntax

WCOLS([<window-name>])

See Also

ACTIVATE WINDOW, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, WROWS(), WEXIST(), WVISIBLE(), WONTOP(), WOUTPUT()

Description

The WCOLS() function returns the number of columns in a window. A window is an area of the screen designated for output and input. There is no limit to the number of defined windows. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. With no parameter, the WCOLS() function operates on the active window. If no windows are active, the WCOLS() function returns a zero.

If a <window-name> is specified, the WCOLS() function returns the number of columns in that window. If the window specified does not exist, or is not active, the WCOLS() function returns a zero.

Example

```
define window browse;  
  from 2,2 to 12,43;  
  title "BROWSE Window";  
  color n/bg;  
  float;  
  shadow  
activate window browse  
?wcols("browse")
```

40

Products

Recital Mirage Server, Recital Terminal Developer

WEEK()

Class

Date And Time Data

Purpose

Function to return the week number for the specified date or datetime, from 1 to 53.

Syntax

WEEK(<expD> | <expT>[, <expN1>[, <expN2>]])

See Also

AMPM(), CDOW(), CMONTH(), CTOD(), CTOT(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), ELAPTIME(), EMPTY(), EPOCH(), GOMONTH(), HOUR(), HOURS(), LTOS(), MDY(), MINUTE(), MINUTES(), MONTH(), MTOD(), MTOS(), QUARTER(), SEC(), SECONDS(), SECS(), STOD(), STR(), TIME(), TIMESTAMP(), TSTRING(), TTOC(), TTOD(), TYPE(), VAL(), VALIDTIME(), VTOD(), WEEK(), YEAR(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK, SET SECONDS, SET VAXTIME

Description

The WEEK() function returns the week number for the specified date or datetime, from 1 to 53.

Parameter	Description
<expD>	Date expression for which to return the week number
<expT>	Time expression for which to return the week number
<expN1>	Optional first week setting
<expN2>	Optional week start date setting

The optional first week setting, <expN1>, can be any of the following:

Value	Setting
0	First week contains January 1 st
1	First week contains January 1 st (default if <expN1> not specified)
2	First week contains at least 4 days from current year.
3	First week is first full (7 day) week

The optional week start date setting, <expN2>, can be any of the following:

Value	Setting
0	Sunday
1	Sunday (default if <expN2> not specified)
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday

Example

```
? week(datetime())
```

```
? week(date(),2)
```

```
? week(date(),0,2)
```

```
// Week starts on Sunday (default), first week has four days in current year
```

```
> ? week({01/01/2004},2)
```

53

```
// Week starts on Monday, first week has four days in current year
```

```
? week({01/01/2004},2,2)
```

1

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

WEXIST()

Class

Screen Windows

Purpose

Function to check for the definition of the specified window

Syntax

WEXIST(<window-name>)

See Also

ACTIVATE SCREEN, ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, MODIFY MEMO, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, SET COMMANDWINDOW, SET ERRORWINDOW, SET STATUS, SET TRACEWINDOW, SET WINDOW OF EDIT, SET WINDOW OF MEMO, WCOLS(), WROWS(), WVISIBLE(), WONTOP(), WOUTPUT()

Description

The WEXIST() function returns .T. if the specified <window-name> has been previously defined, otherwise .F.. A window is an area of the screen designated for output and input. Windows are defined with the DEFINE WINDOW command and are activated with the ACTIVATE window command. There is no limit to the number of defined windows.

Example

```
define window browse;
    from 2,2 to 12,43;
    title "BROWSE Window";
    color n/bg;
    float;
    shadow
activate window browse
?wexist("browse")
.T.
```

Products

Recital Mirage Server, Recital Terminal Developer

WFONT()

Class

Fonts

Purpose

Function to return current font information

Syntax

WFONT(<expN> [,<window-name>])

See Also

AFONT(), FONTMETRIC(), GETFONT(), WOUTPUT()

Description

The WFONT() function returns information about either the current font or the active font in the specified window.

Parameters	Description
<expN>	<ul style="list-style-type: none">• 1 returns the font name as a string• 2 returns the font size as a number• 3 returns the font styles as a string
<window-name>	If the optional <window-name> is specified, the information returned will be based on the active font in that window.

Font Styles:

Code	Style
B	Bold
I	Italic
N	Normal
O	Outline
Q	Opaque
S	Shadow
-	Strikeout
T	Transparent
U	Underline

Example

```
dialog box "Font name is " + wfont(1)
dialog box "Font size is " + alltrim(str(wfont(2)))
dialog box "Font attributes are " + wfont(3)
define window window1 from 1,1 to 20,40
activate window window1
dialog box "Font name is " + wfont(1, "window1")
dialog box "Font size is " + alltrim(str(wfont(2,"window1")))
dialog box "Font attributes are " + wfont(3, "window1")
```

Products

Recital Mirage Server, Recital Terminal Developer

WINDOW()

Class

Screen Windows

Purpose

Function to check if the specified window is the active, or output window

Syntax

WINDOW([<window-name>])

See Also

ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, WCOLS(), WEXIST(), WONTOP(), WOUTPUT(), WROWS(), WVISIBLE()

Description

The WINDOW() function is used to determine which window is the currently active, or output window. A window is an area of the screen designated for output and input. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. There is no limit to the number of defined windows. With no parameter, the WINDOW() function will return the name of the active window as a character string in upper case. If no windows are active, the WINDOW() function returns a null string. WINDOW() returns .T. if the window specified by <window-name> is the active window, otherwise .F..

Example

```
define window one from 0,0 to 10,10
define window two from 0,40 to 10,50
activate window one
activate window two
?window()
TWO
?window("ONE")
.F.
```

Products

Recital Mirage Server, Recital Terminal Developer

WONTOP()

Class

Screen Windows

Purpose

Function to check if the specified window is on top of the others

Syntax

WONTOP([<window-name>])

See Also

ACTIVATE SCREEN, ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, MODIFY MEMO, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, SET COMMANDWINDOW, SET ERRORWINDOW, SET STATUS, SET TRACEWINDOW, SET WINDOW OF EDIT, SET WINDOW OF MEMO, WCOLS(), WEXIST(), WOUTPUT(), WROWS(), WVISIBLE()

Description

The WONTOP() function is used to determine which window is the topmost. A window is an area of the screen designated for output and input. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. There is no limit to the number of defined windows.

Windows may be displayed or moved on top of one another. The command ACTIVATE WINDOW activates a specified window, and displays it according to its coordinates, regardless of the number of windows already displayed on the screen. The MOVE WINDOW command may be used to move windows by degrees or to certain coordinates. If the moving window is the active window, it will overlay existing windows. The RESIZE WINDOW command, used to change the size of a window, will also overlay existing windows with the active window. With no parameter, the WONTOP() function returns the name of the top most window in upper case.

If the window specified by <window-name> has not been defined, or is covered by another window, the WONTOP() function returns .F.. If the specified <window-name> is the topmost window the WONTOP() function returns .T..

Example

```
activate window edit
activate window memo
activate window browse
```

```
?wontop()
```

BROWSE

```
?wontop("memo")
```

```
.F.
```

Products

Recital Mirage Server, Recital Terminal Developer

WORKAREA()

Class

Environment

Purpose

Function to return the lowest available workarea number

Syntax

WORKAREA()

See Also

SELECT, USE, SET VIEW, USED(), DBRELATION(), DBRSELECT(), DBFILTER(), ALIAS(), SELECT()

Description

The WORKAREA() function returns the number of the lowest available workarea which is not in use. If all workareas are in use, it returns -1. The WORKAREA() function returns the number of the free workarea but it does not select that workarea. To determine the first free workarea AND select it, use SELECT 0 or SELECT WORKAREA().

Example

```
close databases
```

```
select 1
```

```
use state.rdb
```

```
? workarea()
```

```
2
```

```
? select()
```

```
1
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

WOUTPUT()

Class

Screen Windows

Purpose

Function to check if the specified window is the active, or output window

Syntax

WOUTPUT([<window-name>])

See Also

ACTIVATE SCREEN, ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, MODIFY MEMO, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, SET COMMANDWINDOW, SET ERRORWINDOW, SET STATUS, SET TRACEWINDOW, SET WINDOW OF EDIT, SET WINDOW OF MEMO, WCOLS(), WEXIST(), WONTOP(), WROWS(), WVISIBLE()

Description

The WOUTPUT() function is used to determine which window is the currently active, or output window. A window is an area of the screen designated for output and input. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. There is no limit to the number of defined windows. With no parameter, the WOUTPUT() function returns the name of the currently active, or output window in upper case. If there are no windows active, the WOUTPUT() function returns a null string.

If the window specified by <window-name> has not been defined, or is not the currently active window, the WOUTPUT() function returns .F.. If the specified <window-name> is the current output window the WOUTPUT() function returns .T..

Example

activate window edit
activate window memo
activate window browse
? woutput()

BROWSE

? woutput("edit")
.F.

Products

Recital Mirage Server, Recital Terminal Developer

WROWS()

Class

Screen Windows

Purpose

Function to return the number of rows in a window

Syntax

WROWS([<window-name>])

See Also

ACTIVATE SCREEN, ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, MODIFY MEMO, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, SET COMMANDWINDOW, SET ERRORWINDOW, SET STATUS, SET TRACEWINDOW, SET WINDOW OF EDIT, SET WINDOW OF MEMO, WCOLS(), WEXIST(), WVISIBLE(), WONTOP(), WOUTPUT()

Description

The WROWS() function returns the number of rows in either the current window, or the window specified by <window-name>. A window is an area of the screen designated for output and input. There is no limit to the number of defined windows. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. With no parameter, the WROWS() function returns the number of rows in the currently active window. If no windows are active, the WROWS() function returns a zero. If the window <window-name> is specified and does not exist, or is not active, the WROWS() function returns a zero.

Example

```
define window browse;  
  from 2,2 to 12,43;  
  title "BROWSE Window";  
  color n/bg;  
  float;  
  shadow  
activate window browse  
? wrows("browse")
```

9

Products

Recital Mirage Server, Recital Terminal Developer

WTITLE()

Class

Screen Windows

Purpose

Function to return the title of the current or specified window

Syntax

WTITLE([<window-name>])

See Also

ACTIVATE SCREEN, ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, MODIFY MEMO, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, SET COMMANDWINDOW, SET ERRORWINDOW, SET STATUS, SET TRACEWINDOW, SET WINDOW OF EDIT, SET WINDOW OF MEMO, WCOLS(), WEXIST(), WONTOP(), WOUTPUT(), WROWS(), WVISIBLE(),

Description

The WTITLE() function returns the title of either the current window, or the window specified by <window-name>. A window is an area of the screen designated for output and input. There is no limit to the number of defined windows. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. With no parameter, the WTITLE() function returns the title of the currently active window. If no windows are active, the WTITLE() function returns an empty string. If the window <window-name> is specified and does not exist, or is not active, the WTITLE() function returns an empty string.

Example

```
define window browse;  
  from 2,2 to 12,43;  
  title "BROWSE Window";  
  color n/bg;  
  float;  
  shadow  
activate window browse  
? wtitle("browse")
```

BROWSE Window

Products

Recital Mirage Server, Recital Terminal Developer

WVISIBLE()

Class

Screen Windows

Purpose

Function to check if the specified window is active and not hidden

Syntax

WVISIBLE(<window-name>)

See Also

ACTIVATE SCREEN, ACTIVATE WINDOW, CLEAR WINDOWS, DEACTIVATE WINDOW, DEFINE WINDOW, HIDE WINDOW, MOVE WINDOW, MODIFY MEMO, RELEASE WINDOWS, RESIZE WINDOW, RESTORE WINDOW, SAVE WINDOW, SHOW WINDOW, SET COMMANDWINDOW, SET ERRORWINDOW, SET STATUS, SET TRACEWINDOW, SET WINDOW OF EDIT, SET WINDOW OF MEMO, WCOLS(), WEXIST(), WONTOP(), WOUTPUT(), WROWS()

Description

The WVISIBLE() function is used to determine if a window, <window-name>, is activated and not hidden. A window is an area of the screen designated for output and input. Windows are created with the DEFINE WINDOW command, and are activated with the ACTIVATE WINDOW command. There is no limit to the number of defined windows. The HIDE WINDOW command removes a window or group of windows from a screen. Hidden windows remain active in memory, and output may be directed to hidden windows. Hidden windows may be revealed with either the SHOW WINDOW or ACTIVATE WINDOW commands. If the window is active and not hidden, WVISIBLE() returns .T., otherwise .F..

Example

```
activate window edit
activate window memo
activate window browse
hide window browse
?woutput()
browse
?wvisible("browse")
.T.
```

Products

Recital Mirage Server, Recital Terminal Developer

XMLCOUNT()

Class

XML

Purpose

Function to return the number of records from an XML file

Syntax

XMLCOUNT(<XML filename>)

See Also

XMLCREATEDTD(), XMLVALIDATE(), XMLFIRST(), XMLNEXT(), COPY TO ... TYPE XML, SELECT ... SAVE AS XML, FETCH ... INTO XML, UPDATE ... FROM XML

Description

The XMLCOUNT() function will return the number of records from an XML file.

Parameters	Required	Default	Description
<XML filename>	Yes	None	The name of the XML file to validate.

If the XML file contains no records, then a value of zero will be returned.

Example

```
select company, street, town, state, zip from sales save as xml sales
```

```
? xmlcount("sales.xml")
```

25

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

XMLCREATEDTD()

Class

XML

Purpose

Function to create a Document Type definition file for a particular table

Syntax

XMLCREATEDTD([<workarea>])

See Also

XMLCOUNT(), XMLVALIDATE(), XMLFIRST(), XMLNEXT(), COPY TO ... TYPE XML, SELECT ... SAVE AS XML, FETCH ... INTO XML, UPDATE ... FROM XML

Description

The XMLCREATEDTD() function will create a Document Type Definition (DTD) file which matches the table open in the current workarea. This DTD file can then be used to validate Extensible Markup Language (XML) files created for this table. An optional workarea name or number may be specified to select a table in another workarea.

Parameters	Required	Default	Description
<workarea>	No	None	The workarea number or name to use.

The XMLCREATEDTD() function will return .T. for success and .F. if it fails.

Note: The XMLFORMAT setting determines whether Recital creates the DTD file. A DTD file will only be created when XMLFORMAT is set to Recital. If XMLFORMAT is set to ADO, the XMLCREATEDTD() will return .T., but no DTD file will be created.

Example

```
use prospect
set xmlformat to recital
? xmlcreatedtd()
.T.
```

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

XMLFIRST()

Class

XML

Purpose

Function to read the first record contained in the specified XML file and return the number of fields in the record

Syntax

XMLFIRST(<XML filename>, <memvar1>, <memvar2>, <memvar3>, <array1>, <array2>)

See Also

XMLCOUNT(), XMLCREATEDTD(), XMLVALIDATE(), XMLNEXT(), COPY TO ... TYPE XML, SELECT ... SAVE AS XML, FETCH ... INTO XML, UPDATE ... FROM XML, SET XMLFORMAT

Description

The XMLFIRST() function will read the first record contained in the specified XML file, returning the number of fields in the record.

Parameters	Required	Default	Description
<XML filename>	Yes	None	The name of the XML file to read.
<memvar1>	Yes	None	The name of a memory variable that will return the XML target table name. Ignored if XMLFORMAT is ADO.
<memvar2>	Yes	None	The name of a memory variable that will return the transaction type for the record. Valid types are INSERT, UPDATE and DELETE. Ignored if XMLFORMAT is ADO.
<memvar3>	Yes	None	The name of a memory variable that will return the where condition for the transaction if it is an UPDATE or DELETE type. Ignored if XMLFORMAT is ADO.
<array1>	Yes	None	The name of an array that will be created automatically and loaded with the field names for the record.
<array2>	Yes	None	The name of an array that will be created created automatically and loaded with the data for each field in the record.

The XMLFIRST() function will return -1 if it fails, as some transaction type can contain zero fields. The XML file format is set to ADO (Microsoft® ActiveX® Data Objects XML Format) by default. The SET XMLFORMAT command can be used to toggle the XML file format between ADO and Recital.

Example

```
number = xmlfirst("sales.xml", target, trans, where, names, data)
```

```
? number
```

30

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

XMLNEXT()

Class

XML

Purpose

Function to read the next record contained in the XML file specified with the XMLFIRST() function and return the number of fields in the record

Syntax

XMLNEXT(<memvar1>, <memvar2>, <array1>, <array2>)

See Also

XMLCOUNT(), XMLCREATEDTD(), XMLVALIDATE(), XMLFIRST(), COPY TO ... TYPE XML, SELECT ... SAVE AS XML, FETCH ... INTO XML, UPDATE ... FROM XML

Description

The XMLNEXT() function will read the next record contained in the XML file specified with the XMLFIRST() function, returning the number of fields in the record. The XMLFIRST() function must be called first before the XMLNEXT() function can be used.

Parameters	Required	Default	Description
<memvar1>	Yes	None	The name of a memory variable that will return the transaction type for the record. Valid types are INSERT, UPDATE and DELETE.
<memvar2>	Yes	None	The name of a memory variable that will return the where condition for the transaction if it is an UPDATE or DELETE type.
<array1>	Yes	None	The name of an array that will be created that contains all the field names for the record.
<array2>	Yes	None	The name of an array that will be created that contains all the data each field for the record.

The XMLNEXT() function will return -1 if it fails or where there are no more records left in the XML file, as some transaction type can contain zero fields.

Example

```
number = xmlnext(trans, where, names, data)
```

```
? number
```

30

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

XMLVALIDATE()

Class

XML

Purpose

Function to validate an XML file against its DTD file

Syntax

XMLVALIDATE(<XML filename>)

See Also

XMLCOUNT(), XMLCREATEDTD(), XMLFIRST(), XMLNEXT(), COPY TO ... TYPE XML, SELECT ... SAVE AS XML, FETCH ... INTO XML, UPDATE ... FROM XML

Description

The XMLVALIDATE() function will validate an Extensible Markup Language (XML) file against its Document Type Definition (DTD).

Parameters	Required	Default	Description
<XML filename>	Yes	None	The name of the XML file to validate.

The XMLVALIDATE() function will return .T. for success and .F. if it fails.

Note: The XMLFORMAT setting determines whether Recital creates an accompanying DTD file when creating XML files. A DTD file is only created when XMLFORMAT is set to Recital.

Example

```
? xmlvalidate("sales.xml")
```

.T.

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer

YEAR()

Class

Date and Time Data

Purpose

Function to extract year from date or datetime

Syntax

YEAR(<expD> | <expT>)

See Also

CROW(), CMONTH(), CTOD(), DATE(), DATETIME(), DAY(), DAYS(), DMY(), DOW(), DTOC(), DTOM(), DTOS(), DTOV(), EPOCH(), GOMONTH(), MDY(), MONTH(), MTOD(), QUARTER(), STOD(), VTOD(), SET CENTURY, SET DATE, SET EPOCH, SET HOURS, SET MARK

Description

The YEAR() function returns the numeric year value from a date expression, <expD> or a datetime expression <expT>. If SET CENTURY is ON (default), the century will be displayed with normal date displays.

Example

? date()

04/04/2004

? year(date())

2004

? year(datetime())

2004

Products

Recital Database Server, Recital Mirage Server, Recital Terminal Developer