REXX Summary

for VM/ESA

by Sylvie Migneault

REXX Summary

Functions

bit = **ABBREV(***information*,*info*[,*minlength*]**)**	Abbreviation match
num = **ABS(***number***)**	Absolute value
str = **ADDRESS()**	Query current environment
num = **ARG()**	Number of arguments
val = **ARG(***n***)**	Nth argument
bit = **ARG(***n*,**'Exists')**	Does nth argument exist?
bit = **ARG(***n*,**'Omitted')**	Was nth argument omitted?
str = **BITAND(***str1*[,[*str2*][,*pad*]]**)**	Logically and strings
str = **BITOR(***str1*[,[*str2*][,*pad*]]**)**	Logically or strings
str = **BITXOR(***str1*[,[*str2*][,*pad*]]**)**	Logically xor strings
str = **B2X(***str***)**	Binary to hex (0F=00001111)
str = **CENTER(***str*,*length*[,*pad*]**)**	Center string
str = **CENTRE(***str*,*length*[,*pad*]**)**	CENTER for Brits
str = **CHARIN(**[*name*][,[*start*][,*length*]]**)**	Read chars from input stream
num = **CHAROUT(**[*name*][,[*start*][,*length*]]**)**	Write chars to output stream
bit = **CHARS(**[*name*]**)**	Any chars in input stream?
bit = **CMSFLAG(***flag***)**	Query SET ... (see page 3)
cix = **COMPARE(***str1*,*str2*[,*pad*]**)**	0 or first mismatch
str = **CONDITION('Condition')**	Name of trapped condition
str = **CONDITION(['Instruction'])**	CALL or SIGNAL
str = **CONDITION('Description')**	Description or null
str = **CONDITION('Status')**	ON, OFF, or DELAY
str = **COPIES(***str*,*n***)**	n copies of entire string
rc = **CSL(***'rtnname retcode parms'***)**	Callable services library
num = **C2D(***str*[,*n*]**)**	Character to decimal
str = **C2X(***str***)**	Character to hexadecimal
str = **DATATYPE(***str***)**	NUM or CHAR
bit = **DATATYPE(***str*,*type***)**	Matches type? (see page 3)
str = **DATE()**	Current date (dd Mmm yyyy)
str = **DATE(***dopt***)**	Date info (see page 3)
... = **DB...(***str*,*...***)**	13 DBCS support functions
str = **DELSTR(***str*,*n***)**	Delete cix from n to end
str = **DELSTR(***str*,*n*,*length***)**	Delete cix n for length
str = **DELWORD(***str*,*n***)**	Delete wix from n to end
str = **DELWORD(***str*,*n*,*length***)**	Delete wix n for length
str = **DIAG(***hex*[**?**][,*data*][,*data*]...**)**	? displays diagnostics
str = **DIAGRC(***hex*[**?**][,*data*][,*data*]...**)**	Also returns CP codes
str = **D2C(***wholenumber*[,*n*]**)**	Decimal to character
str = **D2X(***wholenumber*[,*n*]**)**	Decimal to hexadecimal
str = **ERRORTEXT(***n***)**	Error message text (0-99)
num = **EXTERNALS()**	See PARSE EXTERNAL
wix = **FIND(***haystack*,*needle***)**	0=not found; prefer WORDPOS
str = **FORM()**	Query NUMERIC FORM
str = **FORMAT(***num*[,[*before*][,[*after*]]]**)**	Around decimal place
str = **FUZZ()**	Query NUMERIC FUZZ
cix = **INDEX(***haystack*,*needle*[,*start*]**)**	Default start=1; prefer POS
str = **INSERT(***new*,*str*[,[*n*][,[*length*][,*pad*]]]**)**	Insert after cix n
str = **JUSTIFY(***str*,*length*[,*pad*]**)**	Right-left justify
cix = **LASTPOS(***needle*,*haystack*[,*start*]**)**	POS from right to left
str = **LEFT(***str*,*length*[,*pad*]**)**	Take chars from left
num = **LENGTH(***str***)**	Shape (in chars)
str = **LINEIN(**[*name*][,[*line*][,*count*]]**)**	Read line from input stream
bit = **LINEOUT(**[*name*][,[*string*][,*line*]]**)**	Write line to output stream

REXX Summary

Functions

num = **LINES(**[*name*]**)**	Input stream lines remaining
num = **LINESIZE()**	From CP TERM LINESIZE
num = **MAX(***num*[,*num...*]**)**	Maximum (up to 10 numbers)
num = **MIN(***num*[,*num...*]**)**	Minimum (up to 10 numbers)
str = **OVERLAY(***new*,*str*[,[*n*][,[*length*][,*pad*]]]**)**	Overlay after cix n
cix = **POS(***needle*,*haystack*[,*startcix*]**)**	0=not found
num = **QUEUED()**	Lines in stack
num = **RANDOM()**	Random whole number 0-999
num = **RANDOM(**[*min*][,[*max*][,*seed*]]**)**	Random whole number in range
str = **REVERSE(***str***)**	Rotate string
str = **RIGHT(***str*,*length*[,*pad*]**)**	Take chars from right
num = **SIGN(***num***)**	Signum: -1, 0, or 1
num = **SOURCELINE()**	Lines in source file
str = **SOURCELINE(***n***)**	Nth line from file
str = **SPACE(***str*[,*n*][,*pad*]]**)**	Normalize spaces; def n=1
hex = **STORAGE()**	Virtual storage size in hex
hex = **STORAGE(***address*,*length***)**	Read storage
hex = **STORAGE(***address*,*length*,*data***)**	Write storage
str = **STREAM(***name*[,**'State'**]**)**	State of stream
str = **STREAM(***name*,**'Description'**)**	State of stream, more detail
str = **STREAM(***name*,**'Command'**,*cmd***)**	Apply command to stream
str = **STRIP(***str*[,[*option*][,*char*]]**)**	L, T, or default=Both
str = **SUBSTR(***str*,*firstcix*[,[*length*][,*pad*]]**)**	Substring
str = **SUBWORD(***str*,*firstwix*[,*length*]**)**	Def length=rest of string
str = **SYMBOL(***name***)**	State: BAD, VAR, or LIT
str = **TIME()**	Current time (hh:mm:ss)
str = **TIME(***topt***)**	Time info (see page 3)
str = **TRACE()**	Query trace actions
str = **TRACE(***option***)**	Alter trace, return prev
str = **TRANSLATE(***str*[,[*new*][,[*old*][,*pad*]]]**)**	Map old to new
num = **TRUNC(***num*[,*n*]**)**	Truncate to n decimals
str = **USERID()**	Query logon userid
val = **VALUE(***name***)**	Query value of name
val = **VALUE(***name*,*val***)**	Change value of name
val = **VALUE(***name*[,*val*],*selector* [*groupname*]**)**	GLOBAL/LASTING/SESSION var
cix = **VERIFY(***str*,*okchars*[,[**'Nomatch'**],*start*]**)**	First bad cix; 0=all ok
cix = **VERIFY(***str*,*okchars*,**'Match'**[,*start*]**)**	First good cix; 0=none
str = **WORD(***str*,*wix***)**	Extract nth word
cix = **WORDINDEX(***str*,*wix***)**	Char pos of nth word
num = **WORDLENGTH(**str,wix**)**	Chars in nth word
wix = **WORDPOS(***needle*,*haystack*[,*start*]**)**	Find word(s)
num = **WORDS(***str***)**	Count number of words
str = **XRANGE(**[*start*][,*end*]**)**	One byte codes between
str = **X2B(***str***)**	Hexadecimal to binary
str = **X2C(***hex***)**	Hexadecimal to character
num = **X2D(***hex*[,*n*]**)**	Hexadecimal to decimal

num	Number
str	String
bit	0 or 1
cix	Character index
wix	Word index
val	Value (num or str)

REXX Summary

CMSFLAG(*flag*)

flag:	**ABBREV**	1=SET ABBREV ON, 0=SET ABBREV OFF
	AUTOREAD	1=SET AUTOREAD ON, 0=SET AUTOREAD OFF
	CMSTYPE	1=RT or SET CMSTYPE RT, 0=HT or SET CMSTYPE HT
	DOS	1=SET DOS ON, 0=SET DOS OFF
	EXECTRAC	1=TS or SET EXECTRAC ON, 0=TE or SET EXECTRAC OFF
	IMPCP	1=SET IMPCP ON, 0=SET IMPCP OFF
	IMPEX	1=SET IMPEX ON, 0=SET IMPEX OFF
	PROTECT	1=SET PROTECT ON, 0=SET PROTECT OFF
	RELPAGE	1=SET RELPAGE ON, 0=SET RELPAGE OFF
	SUBSET	1=SUBSET, 0=RETURNed
	XA	1=Running in XA or XC virtual machine, 0=370
	XC	1=Running in XC virtual machine, 0=XA or 370
	370	1=Running in 370 virtual machine, 0=XA or XC

DATATYPE(*str*,*type*)

type:	**A**lphanumeric	Alphanumeric (a-z, A-Z, 0-9)
	Binary	Binary (0-1)
	C	Mixed SBCS/DBCS string
	Dbcs	DBCS-only string enclosed by SO and SI bytes
	Lowercase	Lowercase (a-z)
	Mixed case	Mixed case (a-z, A-Z)
	Number	Number
	Symbol	Symbol (valid REXX name)
	Uppercase	Uppercase (A-Z)
	Whole number	Whole number
	he**X**adecimal	Hexadecimal (a-f, A-F, 0-9)

DATE(*dopt*)

dopt:	**B**ase	Whole days since 1 Jan 0001 (//7 for 0=Monday, 6=Sunday)		
	Century	Days so far in this century		
	Days	Days so far in this year		
	European	dd/mm/yy		
	Julian	yyddd		
	Month	Month	**S**tandard	yyyymmdd
	Normal	dd Mon yyyy	**U**sa	mm/dd/yy
	Ordered	yy/mm/dd	**W**eekday	Tuesday

TIME(*topt*)

topt:	**C**ivil	hh:mmxx (1-12, 00-59, am/pm)
	Elapsed	ssssssss.uuuuuu (seconds, microseconds)
	Hours	Hours since midnight
	Long	hh:mm:ss.uuuuuu
	Minutes	Minutes since midnight
	Normal	hh:mm:ss
	Reset	Returns elapsed time, restarts timer
	Seconds	Seconds since midnight

Error reports appreciated. Permission granted to distribute unmodified copies. Copyright © 1994-1996 by:
Rex Swain • Independent Consultant • 8 South Street • Washington, CT 06793 • 860-868-0131 • rex@rexswain.com

REXX Summary

Syntax

/* ... */	Comment (may span lines, may be nested)
,	Line continuation
;	Statement separator
v.i	Compound variable
'0f'x	Hexadecimal notation
'0010'b	Binary notation

Symbols

May use characters: A-Z, a-z, 0-9, and @ # $ \ . ! ? _
Special variables: RC, RESULT, SIGL

Operators

Operators are grouped by precedence (highest to lowest) below
Operators of equal precedence are evaluated from left to right

\ ¬	Logical NOT (prefix)
+	Numeric (prefix); same as 0+num
-	Negate (prefix); same as 0-num
**	Raise to (whole) power
*	Multiply
/	Divide
%	Integer divide: divide and return integer part
//	Remainder: divide and return remainder (not modulo; result may be negative)
+	Add
-	Subtract
(abuttal) ||	Concatenate without blank
(blank)	Concatenate with blank
==	Strictly equal (identical)
¬== /== \==	Not strictly equal
>>	Strictly greater than
<<	Strictly less than
>>= ¬<< \<<	Strictly greater than or equal to; not less than
<<= ¬>> \>>	Strictly less than or equal to; not greater than
=	Equal (numerically or when padded, etc.)
¬= /= \= >< <>	Not equal; greater than or less than
>	Greater than
<	Less than
>= ¬< \<	Greater than or equal to; not less than
<= ¬> \>	Less than or equal to; not greater than
&	AND
|	Inclusive OR (either, or both)
&&	Exclusive OR (either, but not both)

REXX Summary

Instructions

ADDRESS
Permanently toggle destination of commands to last environment
ADDRESS *environment*
CMS Full command resolution; default for EXECs
CMSMIXED Same as CMS but no uppercase translation
COMMAND Must use EXEC and CP prefixes; most efficient
'' Same as COMMAND
XEDIT Route commands to system editor
Permanently change destination of commands
ADDRESS VALUE *envexpression*
Permanently change destination of commands
ADDRESS *environment cmdexpression*
Specify destination for just this one command
ARG [*template*]
Same as PARSE UPPER ARG [*template*]
CALL *name* [*expression* [,*expression*]]... (up to 20 expressions)
If name in quotes, only built-in or external function called
Optional result returned in special variable RESULT
CALL ON *condition* [**NAME** *trapname*]
ERROR Host command returns non-zero rc (or just positive rc if ON FAILURE too)
FAILURE Host command returns negative return code
HALT Attempt to interrupt (such as HI)
NOTREADY Error during input or output operation
Call on specified condition; variable SIGL contains line number
CALL OFF *condition*
Cancel CALL ON trap
DO [*name* **=** *expr* [**TO** *expr*] [**BY** *expr*] [**FOR** *expr*]] [**WHILE** *expr*]
[**FOREVER**] [**UNTIL** *expr*]
[*expr*]
...instructions...
END [*symbol*]
Repeated execution of a group of instructions
DROP *name*|**(***namelist***)** [*name*|**(***namelist***)** *...*]
Unassign named variables and/or list of vars contained in variable
EXIT [*expression*]
Unconditionally leave; optionally return data
IF *expression* **THEN** *instruction* [**ELSE** *instruction*]
Conditional execution
INTERPRET *expression*
Execute expression as though it were a line in input file
ITERATE [*name*]
Steps current or named do loop
LEAVE [*name*]
Ends current or named do loop
NOP
Dummy instruction
NUMERIC DIGITS [*expression*]
Number of significant digits (default=9; can be arbitrarily high!)
NUMERIC FORM SCIENTIFIC|**ENGINEERING**|**VALUE** *expression*
Exponential notation format; default=SCIENTIFIC
NUMERIC FUZZ *expression*
Digits, at full precision, to ignore during comparisons (default=0)

Error reports appreciated. Permission granted to distribute unmodified copies. Copyright © 1994-1996 by:
Rex Swain • Independent Consultant • 8 South Street • Washington, CT 06793 • 860-868-0131 • rex@rexswain.com

REXX Summary

Instructions

```
OPTIONS 'ETMODE'    Check for valid DBCS strings
        'NOETMODE'  Don't check for valid DBCS strings (default)
        'EXMODE'    Handle DBCS data on a logical character basis
        'NOEXMODE'  All string data handled on a byte basis (default)
       Check for valid DBCS strings; how to process DBCS string data
PARSE [UPPER] ARG [template]      Program/subroutine/function parameters
              EXTERNAL            Next string from terminal input buffer
              LINEIN              Next line from default input stream
              NUMERIC             Current DIGITS FUZZ FORM settings
              PULL                Next string from program stack
              SOURCE              Source of program being executed
              VALUE [expr] WITH   Parse result of expression
              VAR name            Parse named variable
              VERSION             Query REXX version
       Assign data to one or more variables
PROCEDURE [EXPOSE name|(namelist) [name|(namelist) ...]]
       Protect (localize) variables in internal function or subroutine
       By default, all variables are global
PULL [template]
       Same as PARSE UPPER PULL
PUSH [expression]
       Stack resulting string LIFO in program stack
QUEUE [expression]
       Stack resulting string FIFO in program stack
RETURN [expression]
       Return control; like EXIT if no internal routine active
       Returns result from function; sets RESULT variable from subroutine
SAY [expression]
       Displays (TYPEs) a line to default output stream (the terminal)
SELECT
  WHEN expression THEN instruction
  WHEN expression THEN instruction
  ...etc...
  OTHERWISE instruction
  END
       Case-conditional execution
       Only the first matching case is executed
SIGNAL label
       Pass control to label
SIGNAL [VALUE] expression
       Pass control to label evaluated from expression
SIGNAL ON condition [NAME trapname]
         ERROR     Host command returns non-zero rc
                     (or just positive rc if ON FAILURE too)
         FAILURE   Host command returns negative return code
         HALT      Attempt to interrupt (such as HI)
         NOTREADY  Error during input or output operation
         NOVALUE   Uninitialized variable used
         SYNTAX    Interpretation syntax error
       Signal on specified condition; variable SIGL contains line number
SIGNAL OFF condition
       Cancel SIGNAL ON trap
```


REXX Summary

Instructions

```
TRACE [prefix] [letter]
   prefix:
      ?=Toggle pause for interactive debug input after trace occurs
      !=Toggle inhibit host command execution (try TRACE !C)
   letter:
      A=All: all clauses traced (displayed) before execution
      C=Commands: all host commands traced, non-zero rc's displayed
      E=Error: host commands with non-zero rc traced after execution
      F=Failure: host commands with negative rc traced after execution
      I=Intermediates: All + intermediate results
      L=Labels: display labels as passed during execution
      N=Normal=Negative: host cmds with negative rc traced after execution
      O=Off, or no argument: nothing traced; prefix actions off
      R=Results: final (not intermediate) results, plus PULL, ARG, PARSE
      S=Scan: trace remaining clauses without executing
TRACE [number]
   n=skip next n interactive debug pauses
  -n=skip next n tracing displays
UPPER variable [variable ...]
   Translate contents of named variables to uppercase
```

```
PARSE SOURCE s.1 s.2 s.3 s.4 s.5 s.6 s.7
s.1 = 'CMS'
s.2 = 'COMMAND'|'FUNCTION'|'SUBROUTINE'
s.3 = Filename
s.4 = Filetype
s.5 = Filemode
s.6 = Name as invoked (possibly different than s.3 if synonym used)
s.7 = Default ADDRESS for commands
```

```
TRACE Messages
   *-*  Source data in program
   +++  Trace message
   >>>  Result of expression, parsed value, or returned from subroutine
   >.>  Value assigned to placeholder during parsing
during TRACE I:
   >C>  Name of compound variable
   >F>  Result of function call
   >L>  Literal (string or uninitialize variable)
   >O>  Result of operation on two terms
   >P>  Result of prefix operation
   >V>  Contents of a variable
```

Error reports appreciated. Permission granted to distribute unmodified copies. Copyright © 1994-1996 by:
Rex Swain • Independent Consultant • 8 South Street • Washington, CT 06793 • 860-868-0131 • rex@rexswain.com