by H. Fosdick © v 1.1
Open source – CC - BY ND
Corrections? webmasterA
at-sign RexxInfo dot org. Thx!

REXX Reference Card for z/OS

Based on IBM's *TSO/E REXX Reference (V2R3)*.

Operators

Operators are listed in precedence order, from highest to lowest. Operators from the same grouping are evaluated left to right.

\ ¬	Logical NOT (as a prefix)
+	Indicates a positive number (as a prefix)
-	Indicates a negative (as a prefix)
**	Raise to a whole number power
*	Multiply
/	Divide
%	Integer divide (return integer part)
//	Remainder divide (return remainder)
+	Add
-	Subtract
|| (abuttal)	Concatenate (with no space)
(blank)	Concatenate with blank in-between
==	Strictly equal
¬== \== /==	Not strictly equal
>>	Strictly greater than
<<	Strictly lesser than
>>= ¬<< \<<	Strictly greater than or equal to (strictly not less than)
<<= ¬>> \>>	Strictly less than or equal to (strictly not greater than)
=	Equal to
¬= /= \= <> ><	Not equal to
>	Greater than
<	Less than
>= ¬< \<	Greater than or equal to (not less than)
<= ¬> \>	Less than or equal to (not greater than)
&	And (both are true)
|	Or (either is true)
&&	Exclusive Or (either but not both are true)

Notes: not all systems support all notations (for example: ¬). You can always ensure a higher precedence for a clause by enclosing it in parentheses: ( ) .

File Operations

```
ADDRESS TSO          - send commands to TSO
ADDRESS ISPEXEC   - send commands to ISPF
ADDRESS ISREDIT    - for edit macros

Allocate a new dataset –
  “TSO ALLOC F(ddname) DA(dataset) LIKE(dataset) “
  "TSO ALLOC F(ddname) DA(dataset) NEW DIR(size)
      SPACE(size,size) DSORG(org) RECFM(format)
      LRECL(size) BLKSIZE(size) "

Allocate existing datasets --
  "TSO ALLOC F(ddname) DA(dataset)  SHR or OLD [REUSE] "

Free dataset(s) --
  "TSO FREE F(ddname)"  or  "TSO FREE ALL"

Process datasets –
  “EXECIO  lines or *  DISKR or DISKRU or DISKW   ddname
      Write parms: ( [OPEN]  [FINIS]  [STEM stemvar.]
      Read parms: ( [OPEN]  [FINIS]  [SKIP]
                        [FIFO]  |  [LIFO]  |  [STEM stemvar.]
  "EXECIO 0 DISKR my_dd (OPEN "   -- opens a dataset
  “EXECIO 0 DISKR my_dd (FINIS “    – closes a dataset
  "EXECIO 100 DISKR my_dd (STEM my_data. "
        -- reads first 100 records into my_data
  “EXECIO * DISKR my_dd (FINIS STEM my_data. “
        – reads entire dataset into my_data and closes file
  “EXECIO * DISKW my_dd (FINIS STEM my_data.”
        – write all lines from my_data and closes file

  DISKR-- read     DISKW-- write    DISKRU-- update
  lines    -- number of records to read or write
     *       -- process entire dataset
  FINIS      -- optionally close file after the operation
  STEM     -- literal prior to compound variable name
  stemvar. -- compound variable to read or write data

Common TSO commands --
   "TSO COPY   source_dataset  destination_dataset "
   "TSO RENAME source_dataset  new_dataset_name "
   "TSO DELETE dataset "
   "TSO LISTDS pds_dataset_name MEMBERS "

Common ISPF commands --
   "ISPEXEC VIEW dataset "        "ISPEXEC SAVE "
   "ISPEXEC EDIT  dataset "        "ISPEXEC END  "
```

Condition Traps

ERROR	Command to external environment indicates error
FAILURE	Command to external environment failed
HALT	External interrupt to the program
NOVALUE	Reference to an unitialized variable
SYNTAX	Syntax or runtime error in the script

Instructions

```
ADDRESS   |  environment [ expression ] |
          |  [ VALUE ] expression       |

 ARG  [ template_list ]

        |  name  [ expression ]  [, [ expression ] ] . . . |
CALL    |  ON condition [ NAME trapname ]                   |
        |  OFF condition                                    |

    where condition can be ERROR, FAILURE, or HALT

DO    [ repetitor ]  [ conditional ]
        [ instruction_list ]
END  [ symbol ]

where  repetitor is:   [  TO expression_t  ]
                       [  BY expression_b  ]
                       [  FOR expression_f  ]
                       expression_r
                       FOREVER

and condition is:  WHILE expression_w
                   UNTIL  expression_u

DROP   symbol  [ symbol ... ]

EXIT   [  expression  ]

IF expression [;]  THEN [;]  instruction  [  ELSE [;] instruction  ]

INTERPRET  expression

ITERATE  [ symbol ]

LEAVE  [ symbol ]

NOP

NUMERIC DIGITS  [  expression  ]
            FORM   [  SCIENTIFIC  |  ENGINEERING  |
                      [  VALUE  ]   expression  ]
            FUZZ     [ expression ]

OPTIONS  expression

PARSE  [ UPPER ]  type  [ template_list ]

type is:  [  ARG  |  EXTERNAL  | NUMERIC  |  PULL  |
          SOURCE  |  VERSION  ]
          VALUE  [ expression ]  WITH
          VAR  symbol

PROCEDURE   [  EXPOSE  variable_list  ]

PULL   [  template_list  ]

PUSH       [  expression  ]

QUEUE    [ expression   ]
```

```
RETURN  [  expression  ]

SAY         [  expression  ]

SELECT  ;  when_part  [ when_part ... ]  [ OTHERWISE [;]
             [ statement ... ]  ]  END  ;

  when_part is:  WHEN expression [;]  THEN [;] statement

               |  label_name                          |
SIGNAL         |  [ VALUE ] expression                 |
               |  ON condition  [ NAME trapname ]      |
               |  OFF condition                        |

  where condition is one of: ERROR, FAILURE, HALT,
                             NOVALUE, or SYNTAX

TRACE  number   |   trace_setting

  where trace_setting can be:
    A – All
    C-- Commands
    E-- Errors
    F-- Failure
    I--  Intermediates
    L-- Labels
    N-- Normal
    O-- Off
    R-- Results
    S-- Scan
    ?-- Toggles interactive trace On or Off
    ! – Inhibits host command execution
    +n -- Skips number of pauses specified by whole number
    -n –  Inhibits trace for number of clauses specified

UPPER  [ variable_list ]
```

Syntax

Characters include: A-Z, a-z, 0-9, and @ # $. ! ? _ ¢

/* REXX */	1st line of every Rexx program
; (semi-colon)	Statement separator
, (comma)	Continues a line
/* */	Encloses comments
( )	Enclose function arguments, grouping for precedence
var_1.var_2 [...]	Compound variable
stem.index […]	Array notation (same as compound variable)
'abc' or "abc"	Character string notation
'0C'X or "0c"x	Hexadecimal notation
'0101'b or "0101"B	Binary string notation

Functions

ABBREV(information, info [,length])

ABS(number)

ADDRESS()

ARG([argnum [,option]])

BITAND(string1 [,[string2] [,pad]])

BITOR(string1 [,[string2] [,pad]])

BITXOR(string1 [,[string2] [,pad]])

B2X(binary_string)

CENTER(string, length [,pad])
--or--
CENTRE(string, length [,pad])

COMPARE(string1, string2 [,pad])

CONDITION([option])

COPIES(string, times)

C2D(string [,length])

C2X(string)

DATATYPE(string [,type])

DATE([option_out
[,date [,option_in]]])

DELSTR(string, start [,length])

DELWORD(string, start [,length])

DIGITS()

D2C(integer [,length])

D2X(integer [,length])

ERRORTEXT(error_no)

EXTERNALS()

FIND(string,phrase)

FORM()

FORMAT(number [,[before]
[,[after]]])

FORMAT(number [,[before] [,[after]
[,[expp] [,expt]]]])

FUZZ()

INDEX(haystack,needle,start)

INSERT(string, target [,[position]
[,length] [,pad]])

JUSTIFY(string, length [,pad]))

LASTPOS(needle, haystack [,start])

LEFT(string, length [,pad])

LENGTH(string)

LINESIZE()

MAX(number1 [,number2]…)

MIN(number1 [,number2]...)

OVERLAY(string1, string2 [,[start]
[,[length] [,pad]]])

POS(needle, haystack [,start])

QUEUED()

RANDOM([min] [,max] [,seed])

REVERSE(string)

RIGHT(string, length [,pad])

SIGN(number)

SOURCELINE([line_number])

SPACE(string [, [length] [,pad]])

STRIP(string [,option] [,char]])

SUBSTR(string, start [,[length] [,pad]])

SUBWORD(string, start [,length])

SYMBOL(name)

TIME([option])

TRACE([setting])

TRANSLATE(string [,[tableout]
[,[tablein] [,pad]]])

TRUNC(number [,length])

USERID()

VALUE(symbol [,[newvalue] [,pool]])

VERIFY(string, reference [,[option]
[,start]])

WORD(string, wordno)

WORDINDEX(string, wordno)

WORDLENGTH(string, wordno)

WORDPOS(phrase, string [,start])

WORDS(string)

XRANGE([start] [,end])

X2B(hexstring)

X2C(hexstring)

X2D(hexstring [,length])

Options for Functions

Date	Time	Datatype
B Base	C Civil	A Alphanumeric
D Days	E Elapsed	B Binary
E European	H Hours	L Lowercase
M Month	L Long	M Mixed case
N Normal	M Minutes	N Number
O Ordered	N Normal	S Symbol
S Standard	R Reset	U Uppercase
U USA	S Seconds	W Whole #
W Weekly		X Hex
C Century		C, Dbcs DBCS

Special Variables

RC – Return code from a command, or a SYNTAX error code
SIGL – Line number of last instruction that caused a jump to a label
RESULT – Set by RETURN in subrtn

TSO/E External Functions

```
GETMSG(msgstem [,msgtype]  [,cart]  [,mask]  [,time] )

LISTDSI(  | dataset_name  [,location]  |
          | filename  file              |
          [,directory] [,multivol]  [,racf]  [,recall]  [,smsinfo] )

MSG( [option] )

MVSVAR(   |   arg_name    |
          |   arg_names  |  )

OUTTRAP(   |   [ off ]                                    |
           |  varname [,max]  [,concat]  [,skipamt]  |  )

PROMPT(  [option]  )

SETLANG(  [langcode]  )

STORAGE( address  [,length]  [,data]  )

SYSCPUS( cpus_stem )

SYSDSN( dsname )

SYSVAR( arg_name )

TRAPMSG(  [option]  )
```

TSO/E Rexx Commands

```
DELSTACK

DROPBUF [ n ]

EXECIO  lines  |  *
    DISKW  |  DISKR  |  DISKRU  ddname
    read_parms  |  write_parms

where write_parms: (  [STEM varname]  [OPEN]  [FINIS] )
and read_parms:    (  FIFO  |  LIFO  |  STEM varname
                      [OPEN]  [FINIS]  [SKIP]  )

EXECUTIL  |  EXECDD  ( CLOSE  |  NOCLOSE )   |
          |  TS   |   TE   |   HT   |  RT   |  HI          |
          |  RENAME  NAME (function_name)
             [SYSNAME(sysname]  [DD(sysdd)]   |
          |  SEARCHDD(  YES  |  NO )                |

MAKEBUF
NEWSTACK
QBUF
QELEM
QSTACK
SUBCOM envname
```

TSO/E Stream Package Functions

```
CHARIN([name] [,[start] [,length]])        LINEIN([name] [,[line] [,count]])

CHAROUT([name] [,[string] [,start]])       LINEOUT([name] [,[string] [,line]])

CHARS( [name]  )                           LINES( [name] )

                                           STREAM(name, operation, command)
```

Immediate Commands

```
HE – Halt Execution
HI  – Halt Interpretation
HT – Half Typing
RT – Resume Typing
TE – Trace End
TS – Trace Start
```

DBCS Processing Functions

```
DBAJUST( string [,operation] )             DBRRIGHT( string, length  [,option] )

DBBRACKET( string )                        DBTODBCS( string )

DBCENTER( string, length [,pad]  [,option] )   DBTOSBCS( string )

DBJUSTIFY( string, length [,pad]  [,option] )  DBUNBRACKET( string )

DBLEFT( string, length [,pad]  [,option] )     DBVALIDATE( string  [,'C'] )

DBRIGHT( string, length [,pad]  [,option] )    DBWIDTH( string [,option] )

DBRLEFT( string, length  [,option] )
```