

4 The REXX functions

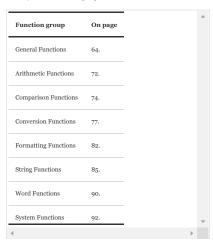
Functions play an important role in REXX. In particular, the functions for word and string processing. When you have worked through the next chapter, you might be amazed about what you can do with the REXX functions. So, have fun reading about the REXX functions! The REXX language we can use on z/OS TSO/E consists of two blocks of language elements:

- $-\,$ The REXX commands, as we learned in chapter 3 The REXX commands on page 25.
- $-\,$ $\,$ The large stock of standard functions that belong to the REXX language and which we will discuss in this chapter.

We now turn to the second part, the REXX functions. I will not discuss any function that is described in the brochure TSO/E REXX reference, but only those that are in my opinion often needed in practice. How functions are called, I have explained already in the section3.3 CALL – Call other programs on page 35. There we already talked about the topics parameter input and return of the results.

The REXX functions can be divided into groups by application area. I have made the following division into groups for the presentation of the REXX functions in this book.

Table 4.1: REXX function groups



The number of available standard functions is larger than the number of instructions in the REXX language. This lets you know how important the knowledge of the functions in REXX is. Through internal and external programs, you can easily create your own, new features and thereby extend your pool of available functions arbitrarily.

General rules for the use of functions

 The REXX interpreter recognizes a function call when parentheses follow the function name

```
    Some functions return a value even if no parameters have been entered. In
this case, the parentheses must still be set. Then they remain empty.
```

 The left parenthesis must always follow immediately the function name without any blanks, otherwise the function name will be assumed as REXX command. If this rule is not adhered, then a program abort occurs.

In the headings of the following functional descriptions, I will omit the parentheses respectively.

4.1 GENERAL FUNCTIONS

4.1.1 ADDRESS - Get the active host command environment

This function returns the name of the current host command environment. No parameters can be entered.

Function:

Returns the currently active host command environment.

Format:

name = address()

A host command environment, whose name was written by the ADDRESS function to a variable, can only be restored by using the following commands.

- By setting the variable in surrounding parentheses.
- By using the REXX VALUE function.

The following example shows the two variants:

```
envir = address() /* Save the current active HCE in envir */
/* processing */.....
address (envir) /* These both commands restores the HCE */
address value envir /* saved in envir */
```

HCE → Host Command Environment

See also the section 3.1 ADDRESS – Connection to the subsystems on page 25.

4.1.2 ARG - Input parameter test or take

The ARG function has three application forms:

- Determining the number of input parameters.
- Testing for the presence of certain input parameters.
- Taking single or all the parameters.

Format:

ARG (n,option)

```
n Means the nth parameter. If specifying this option, the contents of the nth parameter will returned. If this parameter is not set, a null string will be returned.

option

e Returns 1 if the nth parameter is available.
Otherwise, 0 is returned.

o Returns 1 if the nth parameter is not available. Otherwise, 0 is returned.
```

```
/* Call name (no arguments) */
ARG() -> 0
ARG(1) -> ''
ARG(2) -> ''
ARG(1,'e') -> 0
ARG(1,'e') -> 1
/* Call name 'a',,'b'; /* (three arguments) */
na = ARG() -> 'a'
a2 = ARG(2) -> ''
a3 = ARG(3) -> 'b'
ARG(1,'e') -> 1
ARG(1,'e') -> 1
ARG(2,'e') -> 0
ARG(2,'e') -> 0
ARG(2,'e') -> 0
ARG(4,'o') -> 0
```

4.1.3 DATE - Date functions

The DATE function returns the current date in many different forms. Moreover, you can perform with the DATE function also date calculations.

- Return of the current date in different forms Calculation of date values. On the following pages, I will explain these two variants in detail. 1. Return the current date Returns today's date back in different forms. today_date = DATE(format) It returns the current date in the notation defined in the format. The possible entries for format and the resulting return values are summarized in the following table. Only the first letter is entered to define the format. Table 4.2: Parameters of the DATE function Returned value Days since January 1, 0001 without the present day. The format wd = date("B") // 7can be used to determine the current day of the week. o is Monday and 6 is Sunday. Days since January 1 of the current century, Century including today. Days since January 1 of the current year, Days including today. Date in European format: DD/MM/YY Julian Date in the form: YYDDD this means Julian Day. Month English name of the month in upper and lower case. For example, January. Date in the format: DD Mon YYYY. Mon is the short form of month: Jan, Feb, Mar, etc. This is the $\mathbf{default}$ setting, if no parameter is entered. Ordered Date in the format: YY/MM/DD. Standard Date in the format: YYYYMMDD. Date in the American format: MM/DD/YY. Weekday English weekday. Monday, Tuesday, etc. Example DATE output: DATE() --> 29 Apr 2004 DATE('B') --> 731699 DATE('C') --> 1581 DATE('D') --> 120 DATE('E') --> 29/04/04 DATE('J') --> 04120 DATE('M') --> April
DATE('N') --> 29 Apr 2004
DATE('O') --> 04/04/29 DATE('S') --> 20040429 DATE('U') --> 04/29/04 DATE('W') --> Thursday 2. Date calculations using the DATE function With the DATE function, you can perform some date conversion to another format and date calculations.

Find answers on the fly, or master something new. Subscribe today. See pricing options.

 $result = DATE(format1,input,format2) \ Rule:$

To demonstrate these capabilities of the DATE function, I wrote and performed a

Note:

For conversions and calculations, formats C and J cannot be used as a result. This is a shame, because conversions into the J format are often necessary in practice.

4.1.4 TIME - Time functions

The TIME function returns the current date in many different forms. Moreover, you can perform with the TIME function also date calculations.

Function

The TIME function provides two ways to use it:

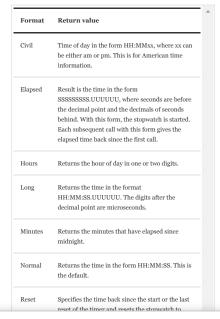
- Returns the current time of day.
- Starting and stopping the difference timing measurement

Format:

Time = TIME (format)

The following table shows which options can be set with format and which results TIME then provides.

Table 4.3: Options of the TIME function





```
can occur. The SMART ISPE utilities program SPROFVAR stores this
               information into the ISPF user profiles. The organization of the variable is as
               The profile variables SYSEXEC and STEPLIB each contains the number of
               existing DSNs for this type that are to be included in the batch job. The
               corresponding DSNs are then in the variables SYSEXECn and STEPLIBn, where
               n is from 1 to content of SYSEXEC and STEPLIB respectively
               The following commands define the variables. That is their content in the profile
               sysexec = 4
               steplib = 3
               sysexec1 = "PROX.LANZ.REXX"
               sysexec2 = "PROX.LOGON.REXX"
sysexec3 = "SMQP.USER.CEXEC"
               sysexec4 = "SMQP.PROD.CEXEC"
               steplib1 = "PROX.LOGON.LOAD"
               steplib2 = "DB2P.ALIAS.DSNLOAD"
steplib3 = "SMQP.LOAD"
                Subsequently, the program section from the edit macro #TSOB that reads the
                variables from the profile and includes them in the JCL:
              /* Reading the environment variables //

*ISPERC CONTROL RERORS RETURN

*ISPERC VOLT (WYSERCE SEPELIN*,

if rc > 0 then do
    red.lmg - "ACCOUNT JORCLASS MEGCLASS) PROFIL*

if rc > 10 then do
    red.lmg - "FORCI VENT A BROCKASS) PROFIL*

if rc > 10 then do
    red.lmg - "ACCOUNT JORCLASS MEGCLASS) PROFIL*

if rc rl nc > "EXEMINT When Declarag - seeling strip(zerrln)
    address "ISPERC" "SETHER MEG (ISRZ001) COMD"

exit
                 address "ISPEREC" "SETHER MSG(ISRERUU1) COND"
ext:
end
// Assemble the batch job //
// Sether MSG(ISRERUU1) COND"
              end queue *//SYSEXEC DD DISP-SHR,DSN-*den if sysexec > 0 then do 1 = 1 to sysexec = "ispexe Variations for the system of the sys
                The rest of the procedure is suppressed.
               4.2 ARITHMETIC FUNCTIONS
               In the following sections, I will discuss functions relating to various ways of
               treatment and control of arithmetic operations.
4.2.1 ABS - absolute value of a number
               Function:
                Returns the absolute value of a number.
               Format:
               result = ABS(number)
               This function returns the unsigned absolute value of a number. The format of the
               returned number is obtained from the currently valid NUMERIC options.
                ABS(-2/3) 0.666666667
                ABS('-0.307') 0.307
4.2.2 DIGITS, FORM, FUZZ - Query options for arithmetic
               These three functions query the values that apply just for performing arithmetic
                expressions. This information can also be obtained by the PARSE NUMERIC var
               command.
               Example:
               numeric digits 12
               numeric fuzz
               parse numeric nums
dig = digits()
                fuz = fuzz()
```

Here the results when calling this procedure:

Values from PARSE = 12 3 SCIENTIFIC Values from functions = 12 3 SCIENTIFIC

4.2.3 MIN, MAX - Minimum and maximum value

These functions return the minimum or maximum value of a set of numbers that are entered as a single parameter. In a parameter list, a maximum of 20 values can be entered. However, since these functions could be interleaved, a maximum of $20 \times 20 = 400$ numbers are possible in one function call.

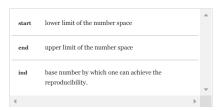
4.2.4 RANDOM - Generate random numbers

Function:

This function generates a random number within a lower and an upper limit.

Format:

number = random(start.end[.ind])



Example:

In this example, it is shown that with repeated calls always the same random numbers are received when using the same number for option ind.

Program 4.1: RAMDOM - Generate random numbers

```
/* DOC. BEXX DARDON
/* DOC. EXEMPLE for the generation of random numbers*/
/* DOC. EXEMPLE for the generation of random numbers*/
/* DOC. EXEMPLE for the generation of random numbers*/
/* DOC. EXEMPLE for the generation of random numbers*/
/* DOC. EXEMPLE for the generation of the
```

This is the printout of the above program:

```
311 886 265 230 967 115 390 676 577 161
976 509 127 16 332 637 364 595 134 852 333
3 8 2 6 5 11 2 4 12 5 9
976 509 127 16 332 637 364 595 134 852 333
3 8 2 6 5 11 2 4 12 5 9
```

As you can see, in rows 2 and 4 plus 3 and 5 respectively appear the same numbers, because in the corresponding RANDOM calls a base number was specified. It is irrelevant what base number you choose.

4.2.5 SIGN - Return of the sign

Function:

Returns the sign of a number.

Format:

value = sign(number)

SIGN returns the sign of the number specified by the following rules:

```
number > 0 \rightarrow value= 1
number = 0 \rightarrow value= 0
number < 0 \rightarrow value= -1
```

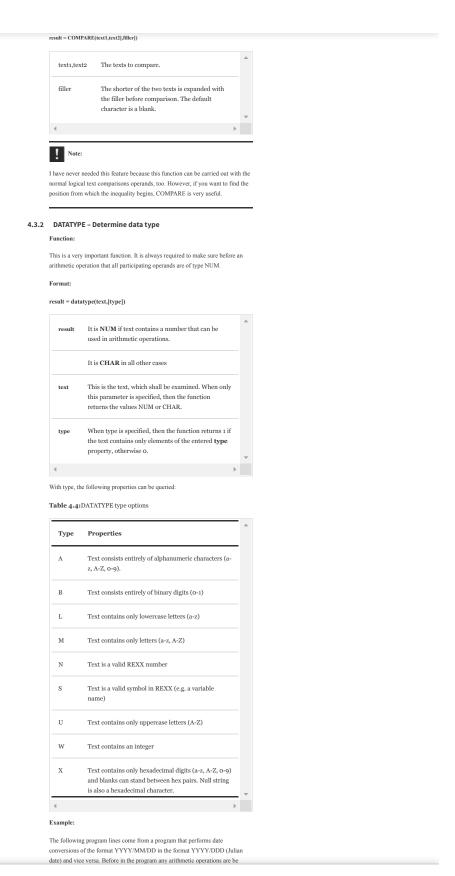
4.3 COMPARISON FUNCTIONS

Besides the possibility of comparisons using the IF command, there are some functions that you can use to perform comparisons in REXX. The most important function is DATATYPE, because without this function plausibility checks of numerical data could not be performed.

4.3.1 COMPARE - Compare texts

Function:

This function returns a zero if the two texts are equal. If the texts are not equal



The PARSE command in line 24 parses the number entered in the variables YYYY, MM and DD. The IF statement in line 25 checks whether all three values are numeric. It works like this:

The DATATYPE function returns for all tests in which a correct number is found 1, otherwise 0. If only one of the tests provides 0, then the multiplication of all three tests is 0. This means that at least one value is not numeric



Caution trap:

If anyone thinks now "You can do this much better by contracting the two IF statements in lines 25 and 27 into one IF statement." Then I can only say, "You have traded with lemons!" Why does this not work properly? This does not lead until then to an error if the program is actually called with wrong input values. Because then the following happens: All arithmetic operations of IF statements are executed to determine the result. This means in our case that the multiply operation yyyy*mm*dd is executed to complete the IF statement. Therefore, if one of the values contains a nonnumeric character, the program will cancel. Therefore, if someone should resist the temptation to contract these two IFs in one thing, he has built a classic time bomb here. The issue works until an

So, remember the following rule:



Before you perform any arithmetic operations in a statement, you must previously perform tests in which no arithmetic operations occur, to ensure that

By the way, what is the most embarrassing error in the programming?



The most embarrassing programming error:

Suppose you installed an extensive plausibility test in your program and this breaks in the reasonableness test without reporting the user's error. Then it is a bad thing for the user, because he does not know what he did wrong.

4.4 CONVERSION FUNCTIONS

With these functions, you can convert texts and numbers in other internal representations. Normally, you do not really need these features since in the TSO/REXX environment everything is very simple: All internal data representations are essentially stored in EBCDIC format. This means that

- The number 1234567 is internally stored in seven bytes, with the following hexadecimal representation: "F1 F2 F3 F4 F5 F6 F7" X. In general, this representation is called the unpacked format.
- The text "I am happy!" has internal a length of 11 bytes and consists of the following hexadecimal characters: " C9 40 81 94 40 88 81 97 97 A8 5A" X

The conversion functions are always very useful when values have to be converted into other forms of representation

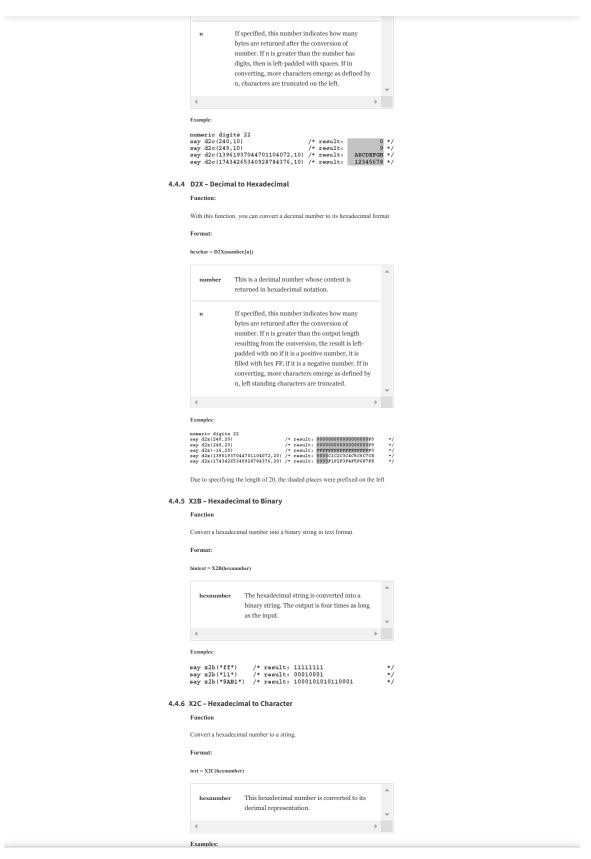
This small REXX program determines the job name directly from the system tables of z/OS

Screen 4.1: Read the job name from CVT

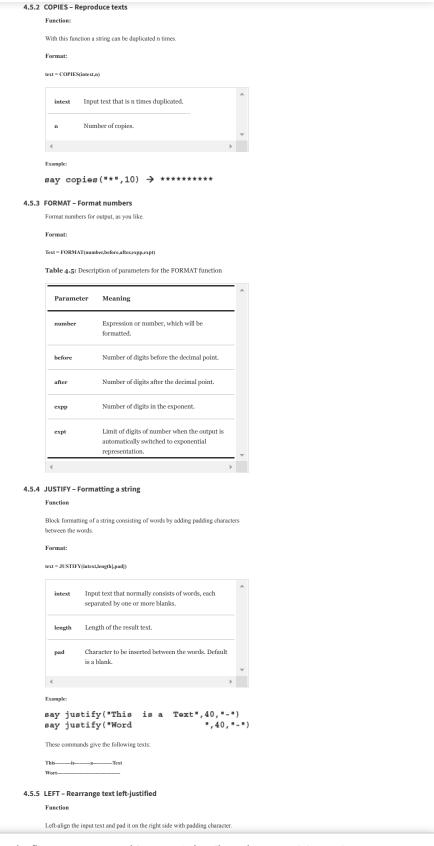
If you have the necessary knowledge of the z/OS operating system, try to decipher the REXX commands. Find the necessary knowledge about the conversion instructions in the following subsections. Of course, here all people who have ever programmed in assembler have an advantage. This applies to all conversion functions

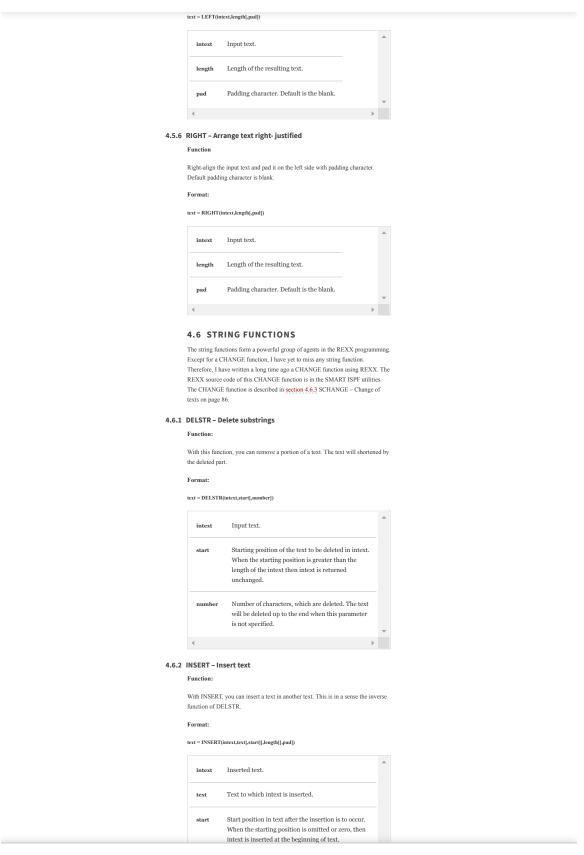
4 4 1 C2D - Character to decimal

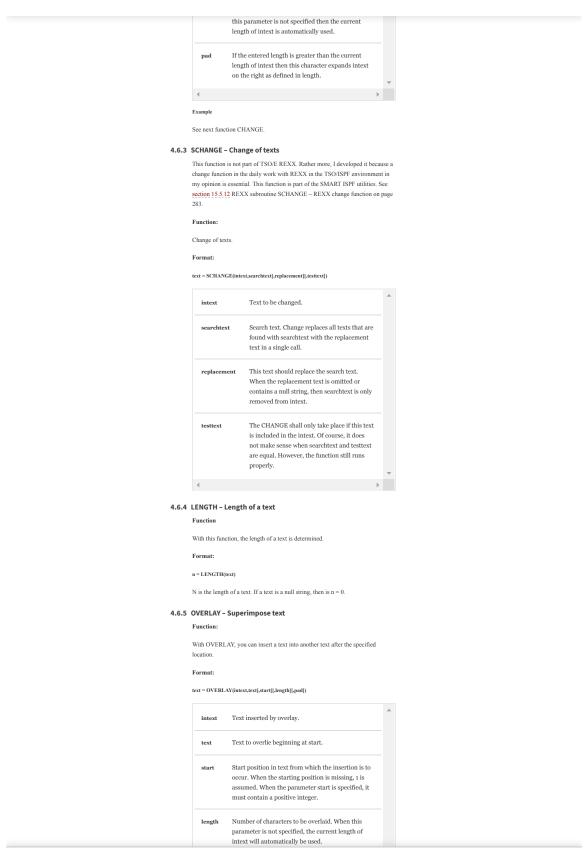




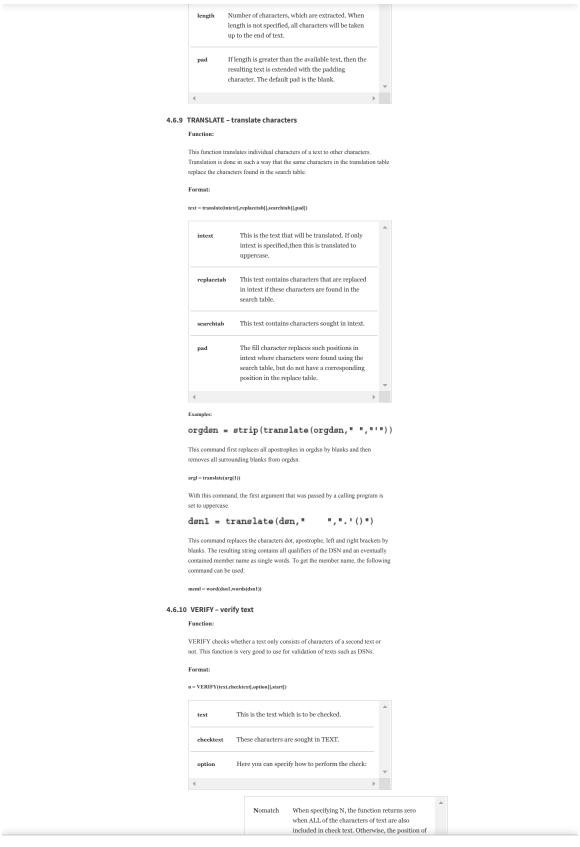


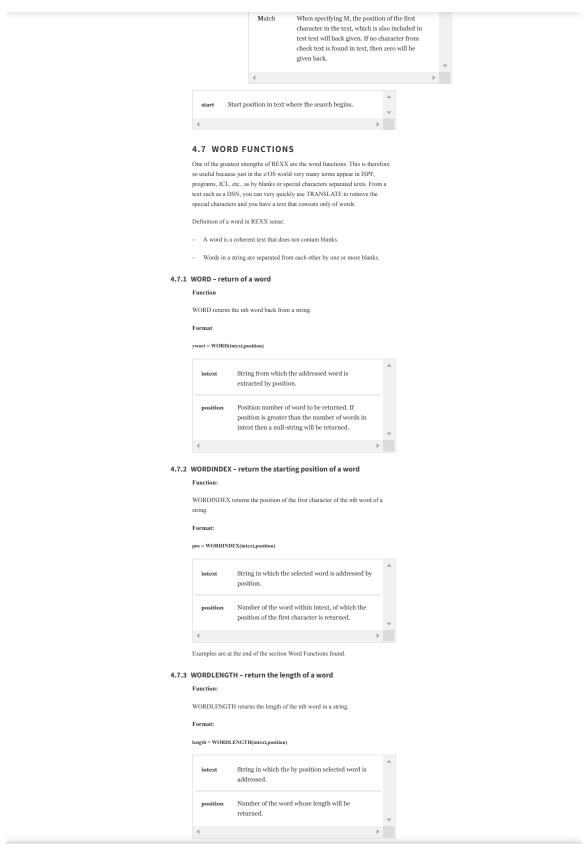


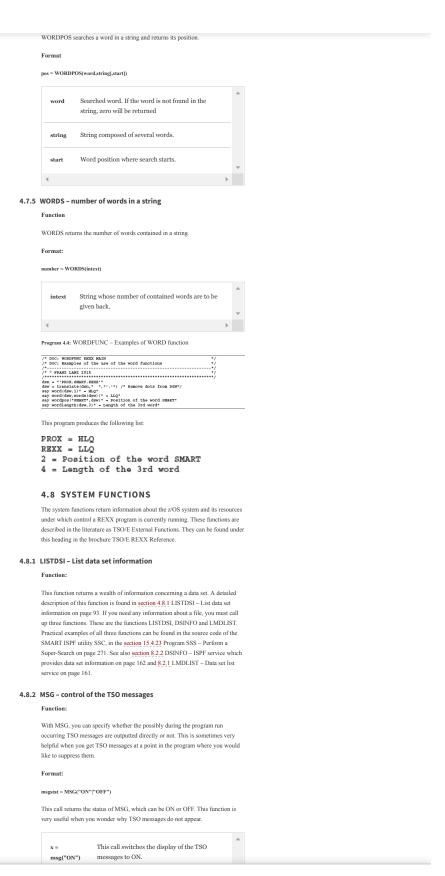


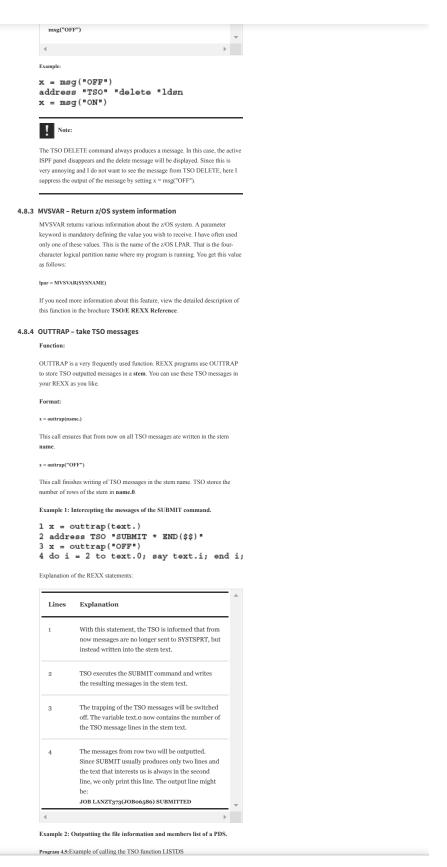












```
DOC: REXX LISTDS
DOC: Example of calling the TSO function LISTDS
FRANZ LANZ 2015
        Printed output:
       PRTT. REXX
        -- RECFM-LRECL-BLKSIZE-DSORG
                       80
                                   27920
           FB
        --VOLUMES--
           D1D115
        --MEMBERS--
           SCONCAT
            SLPARS
           BNDUMSPP
           EINSATZS
           EXNAME
           TT
        I Remark:
       I will not describe the LISTDS function here in detail. The above example may
        suffice as information. The description of the TSO function LISTDS is contained
       in the brochure TSO-E Command Reference.
4.8.5 SYSDSN - check data set status
       Function
       Use SYSDSN to determine whether a file exists and that you can access it.
       Format:
       text = SYSDSN(DSN)
       DSN must appoint a cataloged file.
       The SYSDSN function returns ONE of the following texts, which are self-
       explanatory:
       OK /* data set is available and can be used */
MEMBER NOT FOUND
MEMBER SPECIFIED, BUT DATASET IS NOT PARTITIONED
        DATASET NOT FOUND
ERROR PROCESSING REQUESTED DATASET
        PROTECTED DATASET
        VOLUME NOT ON SYSTEM
INVALID DATASET NAME, dsname
MISSING DATASET NAME
        UNAVAILABLE DATASET
       Example:
       The following program part allocates a data set. If the data set already exists, it is
       only allocated. If it does not exist, it will be created.
       "dsn("ldsn") old r
else "alloc dd("ddname")",
    "dsn("ldsn") new reuse space(15) tracks",
    "recfm(v b a) lrecl(133)"
    "secoio "zl" diskw "ddname" (stem out. finis"
"free dd("ddname")"
4.8.6 SYSVAR - get system Information
       Function
       With SYSVAR, similar to MVSVAR, some system information can be retrieved
```

Here only two information points at run around procedures are of interest:

Example 1: Does a program run in foreground or in background.

foreground means: The program runs in the ISPF online

background means: The program runs under ISPF that is started as a batch job.



It is important to know this operating mode, because maybe files are be assigned in a batch job via DD statement in the JCL, while in foreground mode the TSO alloc command must be used. On the other hand, any error messages in background mode differently than in foreground mode are outputted. For the query where the program runs, you can use the following statement:

FORE \rightarrow The REXX program runs in foreground. BACK → The program runs in background in a batch job Example: Program 4.6: Control messages display ONLINE and in BATCH If the procedure is running in the foreground, the error message is displayed in an

online window using SETMSG. Otherwise, the error message is sent via SAY to the SYSTSPRT DD in the batch job.

Example 2: Does the program run in the ISPF environment runs

If you execute REXX procedures, which need ISPF services in batch jobs, then you should definitely check whether the ISPF is also available in the batch job. If not, you have to abort the procedure with an appropriate error message. If the procedure continues, there may be inexplicable crashes

For this query, you can use SYSVAR(SYSISPF). This call returns ACTIVE when the procedure is running under ISPF. It returns NOT ACTIVE when the procedure only runs under TSO.

Example:

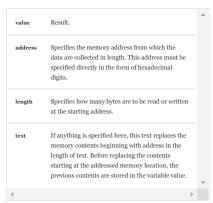
```
If sysvar(sysispf) <> "ACTIVE" then do
  rc = 20
  say "ISPF is not active, procedure ends"
      exit rc
```

4.8.7 STORAGE - read and write memory contents

Function:

With this function, you can read out memory contents and overwrite them. The possibility of overwriting should be handled with caution. This function is normally to read memory contents, which contain interesting information used.

value = STORAGE(address.length.text)





The replacement of memory locations should be handled very carefully. You can shoot down your own address space. However, you do not need to worry, because with the security rights that you have as a normal user, you cannot override

The Common Vector Table (CVT) contains pointers to other tables in the z/OS system. Therefore, if you want to read data from these tables, such as the job name, then you first have to read the contents of the CVT. This address is common in all z/OS systems at the memory location 16. The number 16 expressed as a hexadecimal number is 10. The pointer to the CVT is always four

Find answers on the fly, or master something new. Subscribe today. See pricing options.

The result of the command car of eletarage (10.41) is in my our

Starting from this address, you can now point through all control blocks until you have found the information you need. You can find an example in Screen 4.1

Read the job name from CVT on page 77.

Example 2

I found another example of the application of the STORAGE function on the Internet. It gets the name of the program that is currently executing.

Program 4.7: PROGNAME examples for using STORAGE function

/* REXEX PROGNAME

/* RUDGEL: NOW to find program executed under NOS

/* RUDGEL: NOW to find program executed under NOS

/* RUDGEL: NOW to find program executed under NOS

/* RUDGEL: ACC SCREEN CONTROLL CONTRO

Recommended / Playlists / History / Topics / Settings / Get the App / Sign Out



5 The TSO/E REXX commands