BREXX/370 Release Announcement V2R1

Document Version 1.1
Authors: Peter Jacob (pej), Mike Großmann (mig)

After good progress in the last months, we are just about ready to ship the first MVS release of BREXX/370. Thanks to all who have helped in testing the previous alpha version. We entered the code freeze zone and will perform testing and plan to ship the new software by May 1, 2019.
This document will cover important changes and improvements on a high level. Details of the described functions and commands will be documented in detail in the BREXX/370 User's Guide, which will be delivered with the software launch.

I. Changes and Improvements

A. Calling external REXX Scripts or Functions

Due to the extended calling functionality in the new version, an import of required REXX scripts is no longer required. You can now call any external REXX script directly.

1. Main REXX Script location via fully qualified DSN

If you call a REXX script using a fully qualified partitioned dataset (PDS) name, it must be present in the specified PDS. You can also use a fully qualified sequential dataset name that holds your script. If it is not available an error message will terminate the call. In TSO you can invoke your script using the REXX or RX commands.

Example: **RX 'MY.EXEC(MYREX)'** if the script resides in a PDS, alternatively:
RX 'MY.SAMPLE.REXX' if it is a sequential dataset

2. Location of the Main REXX script via PDS search (TSO environments)

In TSO environments the main script can be called with the RX or REXX command. The search path for finding your script is SYSUEXEC, SYSUPROC, SYSEXEC, SYSPROC. At least one of these need to be pre-allocated during the TSO logon. It is not mandatory to have all of them allocated, it really depends on your planned REXX development environment. The allocations may consist of concatenated datasets.

3. Running scripts in batch

In batch, you can use the delivered RXTSO or RXBATCH JCL procedure and specify the REXX script and its location to execute it. There is no additional search path used to locate it.

4. Calling external REXX scripts

It is now possible to call external REXX scripts, either by:

CALL your-script parm1,parm2… or by function call:
value=your-script(parm1,parm2,…)

The call might take place from within your main REXX, or from a called subroutine. The search of the called script is performed in the following sequence:

- Internal sub-procedure or label (contained in the running REXX script)
- current PDS (where the calling REXX is originated) *[1]
- from the delivered BREXX.RXLIB library, which needs to be allocated with the DD-name RXLIB

*[1]only from the 1st library within a concatenation (this limitation may be lifted in a forthcoming release)

B. Important Notice

The variable scope slightly differs from IBM's REXX implementation. In IBM's REXX, a call to an external REXX script is always treated as a procedure without access to the caller's variable pool. BREXX can access and update the caller's pool. If the called external script is defined with a PROCEDURE statement, variables can be made available by the EXPOSE statement.

II. Software behaviour modifications

The behaviour of some BREXX functions has been changed or extended compared to the original.

STORAGE(memory-address,length)

The STORAGE function now expects hex addresses. For consistency reasons, a new **BSTORAGE** function has been added which acts as the original BREXX STORAGE function.

If you have created REXX scripts using the STORAGE function and dislike to manually update all of them, you can replace the STORAGE function by BSTORAGE, which still works with decimal addresses. BSTORAGE is a REXX function being part of the BREXX RXLIB library.

OPEN(file-name, mode, [DSN | DD])

The open function now has an optional third parameter, which allows opening a Dataset directly without a preceding file allocation.

Example: **ftoken=OPEN('BREXX.RXLIB(RXDATE)','RT','DSN')**

If no third parameter is coded the opening mode defaults to DD, which means the file-name is a DD Name.

RX and REXX command in TSO support parameter

The easiest way to start a REXX script in TSO is using the RX or REXX command call in TSO. These 2 functions are now able to pick up input parameters during the call.

Example:

TSO RX my-script 'parm1','parm2', … alternatively
TSO REXX my-script 'parm1','parm2', …

III. New Functionality

A. Added BREXX functions

ABEND(abend-code)

Terminate the program with Abend-Code, produces an SYSUDUMP if an SYSUDUMP allocation is present.

USERID()

Signed in UserId (available in Batch and Online)

WAIT(wait-time)

Stops REXX script for some time, wait-time is in hundreds of a second

WTO(console-message)

Write a message to the operator's console

TSO REXX functions

SYSDSN(dataset-name) or
SYSDSN(dataset-name(member-name))

Check the availability of a given dataset, or a member within a dataset.

SYSVAR(request-type)

a TSO-only function to retrieve certain TSO runtime information, as running in foreground/background, SYSUID, etc.

LISTDSI(dataset-name) or LISTDSI('dd-name FILE')

Returns information of the dataset in REXX variables, as SYSDSNAME, SYSVOLUME, SYSDSORG, SYSRECFM, SYSLRECL, SYSBLKSIZE

B. RXLIB functions

BREXX has the capability to implement new functions or commands in REXX. They are transparent and will be called in the same way as basic BREXX functions. They are stored in the library BREXX.RXLIB and are automatically allocated (via DD RXLIB) in RXBATCH and RXTSO (Batch). In this release we deliver the following:

BSTORAGE(…)

Storage command in the original BREXX decimal implementation

BRXMSG(…)

Standard message module to display a message in a formatted way

B2C(…)

Converts bit string to Character

C2B(…)

Converts Character to bit string

DEFINED('variable-name')

Tests if variable or STEM already exists

DAYSBETW(date1,date-2,[format-date1],[format-date2])

Return days between 2 dates of a given format.

JOBINFO(request-type)

Return information about currently running job or TSO session in REXX variables, as JOBNAME, JOBNUMBER, STEPNAME, PROGRAMNAME

LINKMVS(load-module, parms)

Starts a load module. Parameters (if any) will send in the format JCL would pass it.

LISTALC()

Lists all allocated Datasets in this session or region

LOWER(string)

Returns translated lower case string

MVSCBS()

Allows addressing of some MVS control blocks. This function must be imported, then the following functions can be used:

Cvt(), Tcb(), Ascb(), Tiot(), Jscb(), Rmct(), Asxb(), Acee(), Ecvt(), Smca()

IMPORT command is described in Vassilis N. Vlachoudis BREXX documentation.

QUOTE(string,qtype)

Enclose string in quotes, double quotes, or parenthesis

PDSDIR(pds-name)

Return all member names from the given PDS in a stem variable

PDSRESET(pds-name)

Removes all members of a PDS and runs a compress.

RXDATE(…)

Return and optionally convert dates in various formats

RXSORT(…)

Sorts a stem variable with different sort algorithms

SEC2TIME(seconds)

Converts a number of seconds into the format [days]hh:mm:ss

SORTCOPY(stem-variable)

Copies any stem variable into the stem SORTIN, which can be used by RXSORT

STEMCOPY(source-stem-variable,target-stem-variable)

Copies any stem variable into another stem variable

TODAY()
Returns today's date in various formats

UNQUOTE(string)
Remove from string leading and trailing quotes, double quotes, or parenthesis

UPPER(string)
Returns translated upper case string

VERSION()
Returns BREXX/370 version information

For further details of these functions refer to the BREXX/370 User's Guide.

Credits: we would like to thank Gerard Wassink for proofreading and making improvement suggestions to this document.