SHARE Session 2819
Introduction to OS/390 REXX

By Lionel B. Dyck
Kaiser Permanente Information Technology
e-mail: lionel.b.dyck@kp.org

Assumptions: a basic programming knowledge and how to use TSO and ISPF. The focus of this session and handout is on the OS/390 unique features of REXX.

Getting Started

1) Logon to TSO
2) Get into ISPF
3) Get into option 3.2 and allocate a Library for your REXX programs
   a) Userid.REXX.EXEC
   b) RECFM=FB LRECL=80 or RECFM=VB LRECL=255
4) Get to ISPF Option 6
5) Then issue:
   ALTLIB ACTIVATE APPLICATION(exec) DATASET(userid.REXX.EXEC)
6) Get into ISPF Edit using this data set
7) Note: This library will only work in the ISPF Session (split) in which the ALTLIB is issued).

Exercises

These exercises will build upon the information later in this handout that you can use for reference. Also provided by **Computer Associates** is a copy of **REXX with OS/2, TSO, & CMS Features Quick Reference Guide** by **Gabriel F. Garguilo** which is available for reference. Solutions can be found at the end of this handout.

Exercise 1

For this exercise create member **EX1** in your REXX library.
Goal: Display the name of the current system name and level of OS/390.
Hint: Use the Say command and the MVSVAR function.
Save the REXX program and then run it by issuing from the ISPF Command Prompt **TSO %EX1** or split the screen and option ISPF option 6 and issue **%EX1**.

Exercise 2

For this exercise create member **EX2** in your REXX Library.
Goal: Issue any TSO Command from within a REXX Program.
e.g. LISTD 'SYS1.PARMLIB'
Hint: Use ARG or PARSE ARG to get the desired TSO command and options from the command line.
Note: A variable will be processed before it is executed.
Save the REXX program and run it.

Exercise 3

For this exercise create member **EX3** in your REXX Library.
Goal: The same as Exercise 2 but this time capture the output and display it inside a loop.
Hint: Use OUTTRAP and a DO END loop.
Save the REXX program and run it.

Exercise 4

For this exercise create member **EX4** in your REXX Library.
Goal: The same as Exercise 3 but this time instead of displaying the results using a loop we will use the ISPF Browse service.

Hint: Use TSO Allocate and Free. Use EXECIO to write the trapped results. Use Address ISPEXEC Browse to view it.
Save the REXX program and run it.

Exercise 5

For this exercise create member EX5 in your REXX Library.
Goal: Test the existence of a data set passed to the program and display information about it.
Hint: Save the status (from SYSDSN) in a variable to test and then use LISTDSI for more information.
Save the REXX program and run it.

Basics

Comments: /* comment text */

Start Every REXX program with /* … REXX … */

REXX Programs may be placed in a library concatenated to **SYSPROC** or **SYSEXEC**. Note SYSEXEC is searched first and then SYSPROC.

The **ALTLIB** command may be used to dynamically access your REXX program library. Note that ALTLIB will be searched before SYSEXEC or SYSPROC.

TSO Commands are enclosed in quotes (either ‘ or “)

Continuation is a comma (,) at the end of a line.

REXX Variables are case insensitive. Thus abc is the same as ABC and the same as aBc. An uninitialized variable has the value of its name.

Useful Commands/Functions:

ARG *var* or **PARSE ARG *var*** Read input from the command line
Parse Arg will result in the variable being in its original case

PARSE – this is a powerful function and worth reading up on.

Say *xxxx* Display text on the terminal (or SYSTSPRT in batch)

Looping:

One pass:
```
Do
End
```

Multi-pass:
```
Do I = 1 to n
End

Do while x > 0
End

Do until x > 0
End

Do Forever
End
```

Tests:

If a = b **Then ….**
Else …

Select
When x = y **Then ….**
When x = z **Then ….**
Otherwise …
End

Tracing:

Trace "?i"	Interactive Trace
Trace "i"	Instruction Trace

Date and Time:

date()7	Feb 2001
date('b')	730522
date('c')	404
date('e')	07/02/01
date('j')	01038
date('m')	February
date('n')	7 Feb 2001
date('o')	01/02/07
date('s')	20010207
date('u')	02/07/01
date('w')	Wednesday

time()	14:12:28
time('c')	2:12pm
time('h')	14
time('l')	14:12:28.864201
time('m')	852
time('n')	14:12:28
time('s')	51148

Here is an example showing a date calculation:

```
/* rexx */
/* ---------------------------------------------------------- *
 * REXX Code to calculate the number of days until SHARE.     *
 * ---------------------------------------------------------- */
 share = date("b","02/25/01","u")
 days = share-date('b')
 say days "days until SHARE"
```

String Operations:

POS(phrase, string, start)
WORDPOS(phrase, string)
WORDS(string)
SUBSTR(string,start,len,pad)
SUBWORD(string,start-word,-words)
LEFT(string,length,pad)
RIGHT(string,length(pad)
CENTER or **CENTRE**(string,width,pad)

Testing if ISPF is Active:

If SYSVAR('sysipf') = "ACTIVE" Then …..

Testing a Data Set:

SYSDSN(*dsname*)
Returns **OK** or a reason (e.g. **DATASET NOT FOUND**)

TSO Message Suppresion:

Call Msg 'off'
Call Msg 'on'

ISPF Commands:

Address ISPEXEC …	ISPF Commands
Address ISREDIT …	ISPF Edit Macro

REXXTRY

This tool, provided by IBM (and available on my web site), can be used to experiment with any REXX command. Here are a few examples:

```
%rexxtry
REXXTRY lets you interactively try REXX statements.
 Each string is executed when you hit Enter.
 Enter 'call tell' for a description of the features.
 Go on - try a few... Enter 'exit' to end.
say sysvar('sysuid')
 SYSLBD
 ------------------------------------------------ REXXTRY on TSO
do i = 1 to 3;say "Count"i;end
 Count1
 Count2
 Count3
 ------------------------------------------------ REXXTRY on TSO
exit
```

LISTDSI (List Data Set Information):

The LISTDSI function is very powerful and can provide a LOT of useful information about an allocated DD or a Data Set.

Syntax:

> **LISTDSI**(data-set-name location directory recall smsinfo)
> or
> **LISTDSI**(filename type)

Example: **call listdsi "'syslbd.lionel.exec'"**
Returns these variables:

```
Variable      Value
SYSDSNAME   : SYSLBD.LIONEL.EXEC
SYSVOLUME   : DV2008
SYSUNIT     : 3390
SYSDSORG    : PO
SYSRECFM    : VB
SYSLRECL    : 255
SYSBLKSIZE  : 32760
SYSKEYLEN   : 0
SYSALLOC    : 186
SYSUSED     : N/A
SYSUSEDPAGES: 2171
SYSPRIMARY  : 184
SYSSECONDS  : 190
SYSUNITS    : TRACK
SYSEXTENTS  : 3
SYSCREATE   : 1999/320
SYSREFDATE  : 2001/038
SYSEXDATE   : 0
SYSPASSWORD : NONE
SYSRACFA    : GENERIC
SYSUPDATED  : YES
SYSTRKSCYL  : 15
SYSBLKSTRK  : N/A
SYSADIRBLK  :
SYSUDIRBLK  :
SYSMEMBERS  :
SYSREASON   : 0000
SYSMSGLVL1  :
SYSMSGLVL2  :
SYSDSSMS    : DATA_LIBRARY
SYSDATACLASS:
SYSSTORCLASS:
SYSMGMTCLASS:
```

Trapping Output of TSO Commands

Call OUTTRAP "***var***." var. is a stem variable (.0 = number of stems)
….
Call OUTTRAP "off"

Example:

```
/* rexx */
call outtrap 'stem.'
"Listd 'sys1.parmlib' mem"
call outtrap 'off'
do i = 1 to stem.0
say stem.i
end
```

System Variables (SYSVAR and MVSVAR)

MVSVAR: Information on MVS, TSO/E, etc.

var = MVSVAR("mvsvar-name")
Say var

Variable-Name	Description
SYSAPPCLU	The APPC/MVS LU Name
SYSDFP	Level of MVS/DFP
SYSMVS	Level of OS/390 BCP
SYSNAME	The name of the active system
SYSSECLAB	Security Label of active session
SYSSMFID	Current SMF ID
SYSSMS	Status of SMS on current system
SYSCLONE	System Symbol of active system
SYSPLEX	Active Sysplex name

```
mvsvar('SYSAPPCLU'):
mvsvar('SYSCLONE') : AS
mvsvar('SYSDFP') : 01.01.05.00
mvsvar('SYSMVS') : SP6.0.7
mvsvar('SYSNAME'): ASYS
mvsvar('SYSPLEX'): CDCPROD
mvsvar('SYSSECLAB'):
mvsvar('SYSSMFID') : ASYS
mvsvar('SYSSMS') : ACTIVE
```

SYSVAR: More system information:

var =SYSVAR("sysvar-name")
Say var

User Information

Variable-Name	Description
SYSPREF	Users Prefix
SYSPROC	Logon Procedure used
SYSUID	Userid

```
sysvar('SYSPREF') : SYSLBD
sysvar('SYSPROC') : TSONULL
sysvar('SYSUID'): SYSLBD
```

Terminal Information

Variable-Name	Description
SYSLTERM	Number of lines on the screen
SYSWTERM	Width of the screen

```
sysvar('SYSLTERM'): 27
sysvar('SYSWTERM'):132
```

Language Information

Variable-Name	Description
SYSPLANG	Primary Language
SYSSLANG	Secondary Language
SYSDTERM	DBCS Support status
SYSKTERM	Katakana support status

```
sysvar('SYSPLANG'): ENU
sysvar('SYSSLANG'): ENU
sysvar('SYSDTERM'): NO
sysvar('SYSKTERM'): NO
```

Exec Information

Variable-Name	Description
SYSENV	Foreground or Background
SYSICMD	Implicit command name
SYSISPF	ISPF Status
SYSNEST	Was the current command invoked from a CLIST or EXEC
SYSPCMD	More recently executed command name
SYSSCMD	More recently executed Sub-command name

```
sysvar('SYSENV'): FORE
sysvar('SYSICMD') : TSYSVAR
sysvar('SYSISPF') : ACTIVE
sysvar('SYSNEST') : YES
sysvar('SYSPCMD') : EXEC
```

System Information

Variable-Name	Description
SYSCPU	Number of CPU seconds in the session
SYSHSM	Level of DFHSM
SYSJES	Name and level of JES
SYSLRACF	Level of RACF
SYSRACF	RACF Status
SYSNODE	NJE Node of active JES
SYSSRV	Number of Service Units for session
SYSTERMID	Active Terminal ID
SYSTSOE	Level of TSO/E

```
sysvar('SYSCPU'): 38.15
sysvar('SYSHSM'): 1050
sysvar('SYSJES'): JES2 OS 2.7.0
```

```
sysvar('SYSLRACF'): 2060
sysvar('SYSRACF') : AVAILABLE
sysvar('SYSNODE') : NKAISERA
sysvar('SYSSRV'): 45133366
sysvar('SYSTERMID') : TCPN0051
sysvar('SYSTSOE') : 2060
```

Console Information

Variable-Name	Description
SOLDISP	Whether Solicited Messages should be displayed
UNSDISP	Whether Unsolicited Messages should be displayed
SOLNUM	Number of Solicited Messages in message table
UNSNUM	Number of Unsolicited Messages in message table
MFTIME	Whether the time stamp should be displayed
MFOSNM	Whether the originating system name should be displayed
MFJOB	Whether the originating job name should be displayed
MFSNMJBX	Whether the originating job and system name should be excluded from the display

```
sysvar('SOLDISP') : NO
sysvar('UNSDISP') : NO
sysvar('SOLNUM'): 1000
sysvar('UNSNUM'): 1000
sysvar('MFTIME'): NO
sysvar('MFOSNM'): NO
sysvar('MFJOB') : NO
sysvar('MFSNMJBX'): NO
```

MVSVAR SYMDEF: System Defined Symbols

Variable-Name	Description
JOBNAME	Current active Jobname
LYYMMDD	Local Date
LHHMMSS	Local Time
HHMMSS	GMT Time
LYR4	4 digit Year
SYSR1	Primary IPL Volume
User-defined	Any symbol you want in IEASYMxx.

```
MVSVAR('SYMDEF',LYYMMDD): 010202
```

```
MVSVAR('SYMDEF',LHHMMSS): 103114
MVSVAR('SYMDEF',HHMMSS) : 183114
MVSVAR('SYMDEF',LYR4) : 2001
MVSVAR('SYMDEF',JOBNAME): SYSLBD
MVSVAR('SYMDEF',SYSR1): P22SR2
```

REXX I/O

EXECIO is the command to perform input/output operations.

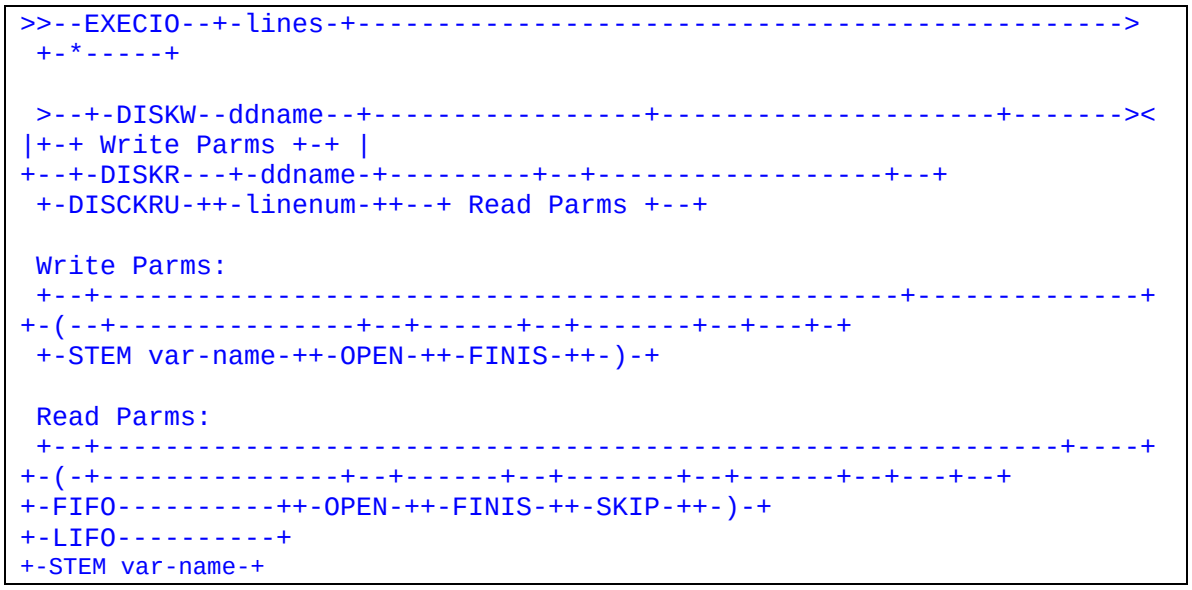

Figure borrowed from IBM's OS/390 TSO REXX Reference (SC28-1975-05)

Examples:

Displaying a file:

```
/* rexx */
"Alloc f(ddname) shr dsn('sys1.parmlib(ieasys00)')"
"Execio * diskr ddname (finis stem in."
"Freef(ddname)"
do i = 1 to in.0
say in.i
end
```

Outtrap Project: the TSOTRAP Program

This project shown below is an example of a REXX program that runs under ISPF. Its purpose is to issue a TSO command, trap the results, and then place the user in ISPF Browse to view the results. This example demonstrates how to:

1. Code comments
2. Read in an argument from the command
3. Use OUTTRAP
4. Issue a TSO command from a variable
5. Issue TSO commands (Allocate and Free)
6. Write information from a variable (Execio)
7. Invoke an ISPF service (Browse)

```
/* rexx */
/* Name:TSOTRAP*
 * Function: Use Outtrap to capture*
 * the results of a TSO*
 * command and then invoke *
 * ISPF Browse.*
 * Syntax: TSO %tsotrap tso-command*
 * */
/* read passed input */
 arg option
 /* setup the output trap */
 call outtrap "trap."
 /* invoke the tso command */
 option
 /* turn off output trapping */
 call outtrap "off"
 /* setup the variable dsname */
 trapds = "trapdd.temp.list"
 /* Allocate the output data set */
 "Alloc f(trapdd) ds("trapds") new" ,
 "spa(15,15) tr"
 /* write the captured output to the
output data set */
 "Execio * diskw trapdd (finis stem trap."
 /* Invoke the ISPF Browse Function */
 Address ISPExec,
 "Browse dataset("trapds")"
 /* free and delete the data set */
 "Free f(trapdd) Delete"
```

Access Storage via REXX

The REXX **STORAGE** function is very powerful, especially when combined with the Decimal to Hex (d2x) and the Character to Decimal (c2d) functions.

```
/* ------------------- REXX --------------------- *
 * code to display job information from:*
 * Robert Zenuk, robzenuk@aol.com *
 * ---------------------------------------------- */
tcb= storage(21c,4)
tiot = storage(d2x(c2d(tcb)+12),4)
jscb = storage(d2x(c2d(tcb)+180),4)
ssib = storage(d2x(c2d(jscb)+316),4)
jobname= strip(storage(d2x(c2d(tiot)),8))
jobtype= storage(d2x(c2d(ssib)+12),3)
jobnum = strip(storage(d2x(c2d(ssib)+15),5),l,0)
stepname = strip(storage(d2x(c2d(tiot)+8),8),l,0)
procstep = strip(storage(d2x(c2d(tiot)+16),8),l,0)
say "Jobname:" jobname"("jobtype""jobnum")"
say "Step/Procstep:" stepname procstep
```

Let's look at one of the statements: **tcb = storage(21c,4)**. This statement instructs the storage function to look at offset 21c and take 4 bytes from that location and insert them into the variable tcb.

A bit more complicated is the next statement: **tiot = storage(d2x(c2d(tcb)+12),4).** In this statement the value of **tcb** must first be converted from character to decimal using the **c2d** function, then add 12 to that decimal number, and then from decimal to hexadecimal using the **d2x** function. Then the storage function can work with the computed address.

This example is to whet your appetite for the **storage** function. It obviously will help for you to have the layout of the control blocks you want to work with.

SYSVAR/MVSVAR/Storage Example

Here is an example using all of these functions. It can be used during your TSO LOGON process to 'remind' you where you are:

```
/* rexx */
/* */
CVT= C2d(Storage(10,4))/* point to CVT */
/* */
SMCA = Storage(D2x(CVT + 196),4) /* point to SMCA*/
SMCA = Bitand(SMCA,'7FFFFFFF'x)/* zero high order bit*/
SMCA = C2d(SMCA) /* convert to decimal */
/* */
/*****************************************************************/
/*The IPL date is stored in packed decimal format - so to make */
/*the date printable, it needs to be converted back to hex and */
/*the packed sign needs to be removed. */
/*****************************************************************/
IPLTIME= C2d(Storage(D2x(SMCA + 336),4)) /* IPL Time - binary*/
IPLDATE= C2d(Storage(D2x(SMCA + 340),4)) /* IPL Date - 0CYYDDDF*/
IPLDATE= D2x(IPLDATE)/* convert back to hex*/
IPLDATE= Substr(IPLDATE,2,5) /* keep YYDDD */
iplday = date('w',ipldate,'j')
ipldate= date('u',ipldate,'j')
IPLTIME= IPLTIME / 100 /* remove hundreths */
HH = IPLTIME % 3600/* IPL hour */
MM = (IPLTIME - (3600 * HH)) % 60/* IPL minute */
SS = (IPLTIME - (3600 * HH)- (60 * MM)) % 1/* IPL seconds*/
HH = Right(HH,2,'0') /* ensure 2 digit HH*/
MM = Right(MM,2,'0') /* ensure 2 digit MM*/
SS = Right(SS,2,'0') /* ensure 2 digit SS*/
IPLTIME= HH':'MM':'SS/* time in HH:MM format */
say "OS/390 Level:" left(mvsvar('sysmvs'),10) ,
"JES2:" sysvar('sysjes') ,
"System ID:" mvsvar('sysname')
say "IPL Volume:" left(mvsvar('symdef','sysr1'),10) ,
"NJE Node:" left(sysvar('sysnode'),10) ,
" Proc: " sysvar('sysproc')
say "IPL Date/Time: " iplday "the" ipldate "at" ipltime
```

Resulting in something like this:

```
OS/390 Level: SP6.0.7JES2: JES2 OS 2.7.0 System ID: ASYS
IPL Volume: P22SR2 NJE Node: NKAISERA Proc: TSONULL
IPL Date/Time: Sunday the 01/21/01 at 03:53:43
```

Web Resources

- IBM TSO REXX and ISPF publications
  http://www.s390.ibm.com/os390/bkserv/
- My web site (for lots of examples):
  http://www.geocities.com/lbdyck
- The CBT Tape web site
  http://www.cbttape.org
- Dave Alcock's Planet MVS site
  http://planetmvs.com/
- The REXX Language Association
  http://www.rexxla.org

Other useful/interesting sites:

- http://www.uberfish.freeserve.co.uk/Computers/rexxfaq.html#C2Environs
- ftp://ftp.ziplink.net/users/mirvin/RexOnMVS/
- http://www.estreetjournal.com/cgi-local/dir/dir.cgi/Computers/Programming/Languages/Rexx/S390/Scripts/
- http://members.aol.com/rexxauthor/rexxref.htm
- http://vm.marist.edu/htbin/wlvindex?TSO-REXX

- news:bit.listserv.tsorexx
- news:comp.lang.rexx

Exercise Solutions

Exercise 1:

```
/* --------------------REXX------------------------------- *
 * Exercise 1 Solution                                     *
 * ------------------------------------------------------- */
 say "Current System Name: " mvsvar('sysname')
 say "Current System Level:" mvsvar('sysmvs')
```

Exercise 2:

```
/* --------------------REXX------------------------------- *
 * Exercise 2 Solution                                     *
 * ------------------------------------------------------- */
 parse arg command options
 Say "Command:" command
 Say "Options:" options
 command options
```

Exercise 3:

```
/* --------------------REXX------------------------------- *
 * Exercise 3 Solution                                     *
 * ------------------------------------------------------- */
 parse arg command options
 Say "Command:" command
 Say "Options:" options
 call outtrap 'trap.'
 command options
 call outtrap 'off'
 do i = 1 to trap.0
    say trap.i
    end
```

Exercise 4:

```
/* ------------------------REXX---------------------------- *
 * Exercise 4 Solution                                     *
 * ------------------------------------------------------- */
 parse arg command options
 call outtrap 'trap.'
 command options
 call outtrap 'off'
 "Alloc f(ex4dd) ds(ex4.temp) new spa(30,30) tr",
    "recfm(f b) lrecl(133) blksize(0)"
 "Execio * diskw ex4dd (finis stem trap."
 Address ISPExec "Browse Dataset(ex4.temp)"
 "Free f(ex4dd) delete"
```

Exercise 5

```
/* ------------------------REXX---------------------------- *
 * Exercise 5 Solution                                     *
 * ------------------------------------------------------- */
 arg dataset
 stat = sysdsn(dataset)
 Say "Dataset: " dataset "status:" stat
 if stat <> "OK" then Exit
 call listdsi dataset
 say "DSName:  " sysdsname
 say "Volume:  " sysvolume
```