

# Subroutines and Functions

## Section 8



The Software That Manages eBusiness™

## What are they?

- Sections of a program they perform specific tasks.
- Can be branch to from anywhere in the program
- Can be inside or outside the program
- Created as a routine or a function



8 - 2

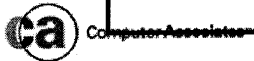
The Software That Manages eBusiness™

N.B.

Functions should always return a value.

### CALL and Commas

CALL with	PARSE With	Result
CALL subrt 1 2 3	ARG first second third	First = "1" Second = "2" Third = "3"
CALL subrt 1 2 3	ARG first, second, third	First = "1 2 3" Second = "" Third = ""
CALL subrt 1, 2, 3	ARG first second third	First = "1" Second = "" Third = ""
CALL subrt 1, 2, 3	ARG first, second, third	First = "1" Second = "2" Third = "3"



8 - 3

The Software That Manages eBusiness™

## Internal Sample Program

```
CALL check_name "FRED FLINTSTONE"  
EXIT ←
```

```
check_name:  
  PARSE ARG test_name  
  SAY "Your name is : "LENGTH(test_name)" letters long."  
  RETURN
```

```
Your name is : 15 letters long.  
***
```



## External Sample Program

```
CALL chckname "FRED FLINTSTONE"
```

```
/****** REXX ******/  
/* Program */  
/* chckname */  
/* Arguments */  
/* one argument which is the string to be checked */  
/* Description */  
/* REXX routine to return the length of a string */  
/* Author : Michaelangelo DeParma */  
/* Date : 3rd February 2000 */  
/*----- Amendment History -----*/  
/*******/  
PARSE ARG test_name, rubbish  
SAY "Your name is : "LENGTH(test_name)" letters long."  
RETURN
```



The executing program calls the member (in this example) in the same dataset called chckname.

## Internal Sample Function

```
ARG nm1 nm2 nm3 unused
check_name = name_check(nm1 nm2 nm3)
IF check_name = 1 THEN DO
    SAY "All the names are in this department."
END
ELSE DO
    SAY "NOT all the names are in this department."
END
EXIT

name_check:
    ARG name1 name2 name3
    IF name1 = "FRED" & name2 = "BOB" & name3 = "JANE" THEN DO
        name_result = 1
    END
    ELSE DO
        name_result = 0
    END
    RETURN name_result
```



Computer Associates™

8-6

The Software That Manages eBusiness™

## Internal Sample Function

```
ex 'crone90.crone.rx101(sample9)' 'fred bob jane'
```

```
All the names are in this department.  
***
```



## External Sample Function

```
ARG nm1 nm2 nm3 unused
check_name = namechek(nm1 nm2 nm3)
IF check_name = 1 THEN DO
    SAY "All the names are in this department."
END
ELSE DO
    SAY "NOT all the names are in this department."
END
```





## Work Section 8.1

- Write a REXX program which add a series of numbers that are passed to the subroutine

```
CALL addup 1 2 3 4 5 6 7 8
```

```
The total of : 1 2 3 4 5 6 7 8  
is : 36  
***
```



## Work Section 8.2

- Re-write work section 8.1 as an external function.

```
total = addup(1 2 3 4 5 6 7 8)
SAY "The total is : "total
```

```
The total is : 36
***
```



## Additional Program

- Write an external REXX range function to check if a number is within a given range.

```
low = 1
high = 100
number_check = RANGECHK(low high number)
```

```
Please enter the number you wish to check :
123
The number is out of range 1 to 100
***
```



8 - 11

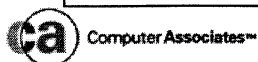
The Software That Manages eBusiness™

## Additional Program

- Create a reverse word function which will reverse the order of words passed to it.

```
SAY "Please enter your list of words : "  
PARSE PULL list_of_words  
list_of_words = STRIP(list_of_words)  
SAY "The list is : "||REVWORD(list_of_words)
```

```
    Please enter your list of words :  
one two three  
The list is :  THREE TWO ONE  
***
```



8 - 12

The Software That Manages eBusiness™

# ADDRESSing

## Section 9



The Software That Manages eBusiness™

## ADDRESS

```
/****** REXX *****/
/* Program */
/* address1 */
/* Description */
/* REXX program to test statements */
/* Author : Michaelangelo DeParma */
/* Date : 2nd February 2000. */
/*----- Amendment History -----*/
/*******/
ADDRESS "TSO"
"CLEAR"
"LISTC LEVEL(IULC39)"
```



9-2

The Software That Manages eBusiness™

ADDRESS temporarily or permanently changes the destination of commands.

Here the destination for commands is set to TSO.

## ADDRESS

```
/****** REXX *****/
/* Program */
/* address2 */
/* Description */
/* REXX program to test statements */
/* Author : Michaelangelo DeParma */
/* Date : 2nd February 2000. */
/*----- Amendment History -----*/
/*******/
ADDRESS "MVS"
"CLEAR"
ADDRESS "TSO" "LISTC LEVEL(IULC39) "
```



9 - 3

The Software That Manages eBusiness™

ADDRESS "MVS" - Makes the change permanent from the program or until another ADDRESS "ENV" is detected.

ADDRESS "TSO" "COMMAND" will change the environment for the one command only.

## OUTTRAP

```
/****** REXX *****/
/* Program */
/* outtrap1 */
/* Description */
/* REXX program to test statements */
/* Author : Michaelangelo DeParma */
/* Date : 2nd February 2000. */
/*----- Amendment History -----*/
/*******/
ADDRESS "TSO"
"CLEAR"
un_used = OUTTRAP("line.", "*", "NOCONCAT")
"LISTC LEVEL(IULC39)"
un_used = OUTTRAP("OFF")
```



Computer Associates™

9 - 4

The Software That Manages eBusiness™

OUTTRAP is used to retrieve the output from the TSO commands.

As a function a value is returned and is set to a variable (un\_used).

The data from the command (LISTC) is stored in an array with a stem variable.

"\*" indicates how many lines of data to be retrieved.

"NOCONCAT" ensures that anything in an OUTTRAP before the command is over written.

"OFF" switches off the trap and ensures that no more commands will go to it.



## Retrieving the OUTTRAP

```
/* ***** REXX ***** */
/* Program */
/* outtrap1 */
/* Description */
/* REXX program to test statements */
/* Author : Michaelangelo DeParma */
/* Date : 2nd February 2000. */
/*----- Amendment History -----*/
/* ***** */
ADDRESS "TSO"
"CLEAR"
un_used = OUTTRAP("line.", "*", "NOCONCAT")
"LISTC LEVEL(IULC39)"
un_used = OUTTRAP("OFF")
DO line_counter = 1 TO line.0
SAY "Line " || line_counter || " contains : " || line.line_counter
END
```



Computer Associates™

9-5

The Software That Manages eBusiness™

One of the features of the array is that STEM.0 contains the number of elements.

e.g.

line.0 contains how lines are return from the "LISTC" command.

The DO loop is used to retrieve the result.

The output is shown on the next page.

## Retrieving the OUTTRAP

```
Line 1 contains : NONVSAM ----- IULC39.DEMOED.ISPPROF
Line 2 contains :          IN-CAT --- ICF.USERID.USERCAT4
Line 3 contains : NONVSAM ----- IULC39.OPSTCMLS
Line 4 contains :          IN-CAT --- ICF.USERID.USERCAT4
***
```



## Work Section 9.1

- Write a REXX program to show the result of the TSO "LU" command.

```
The result of the LU command is :  
RACF PRODUCT DISABLED: COMMAND ENDED.  
***
```



9-7

The Software That Manages eBusiness™

Do not use outtrap for this.

Use your dataset for all the REXX programs.

CLCS.IULCnn.REXX

Replace nn with your user number.

E.g. for id IULC35

CLCS.IULC35.REXX

## Work Section 9.2

- Write a REXX program to send a message to a TSO user, using the TSO send command.

```
Please enter the message to send:
hello bob
Please enter the TSO ID :
IULC23
hello bob CRONE90
***
```



9 - 8

The Software That Manages eBusiness™

### TSO Command

```
SE 'message' USER(userid) LOGON
```

## Work Section 9.3

- Write a REXX program to display a userid's datasets name only. (do not include the other entries)

```
Please enter the TSO ID :
iulc23
IULC23.CA7.BTIOUT1
IULC23.CA7.TEXT
IULC23.CA7.TEXT1
IULC23.DB2.WORK
IULC23.DEMOED.ISPPROF
IULC23.IULC.RULES
IULC23.JCL.CNTL
IULC23.OUTPUT
IULC23.REXX.CLIST
IULC23.SAREXP.DELCARDS
IULC23.TEST.PDS
***
```



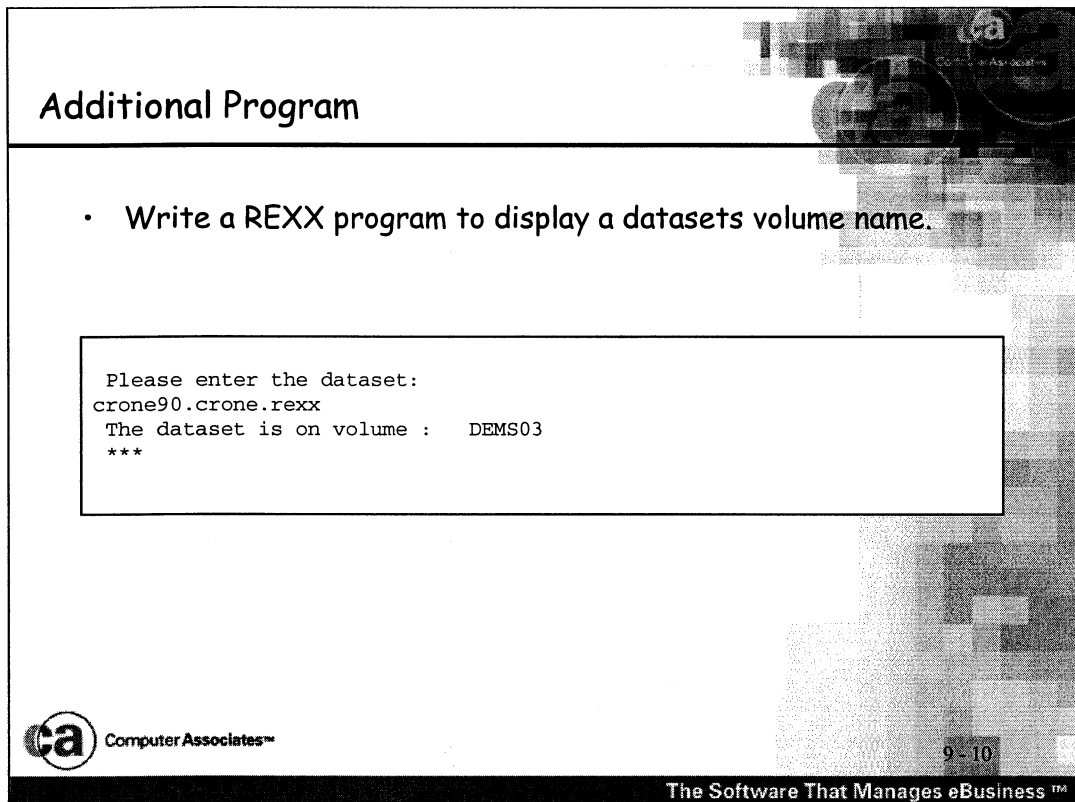
Computer Associates™

9-9

The Software That Manages eBusiness™

TSO Command

```
LISTC LEVEL(userid)
```



**Additional Program**

- Write a REXX program to display a datasets volume name.

```
Please enter the dataset:
crone90.crone.rexx
The dataset is on volume :  DEMS03
***
```

Computer Associates™

9 - 10

The Software That Manages eBusiness™

TSO Command

LISTDS

# EXECIO

## Section 10



The Software That Manages eBusiness™

## QSAM file Access

- TSO REXX has Access to QSAM files.
- Using EXECIO you can read and write to both a PDS and a Sequential dataset.



The Software That Manages eBusiness™



## Checking a DSN

```
ADDRESS "TSO"  
"CLEAR"  
dsn_name = "crone90.crone.rexx(rubbish)"  
dsn_check = SYSDSN("'"dsn_name"'")  
SAY dsn_check
```

```
MEMBER NOT FOUND  
***
```



10-3

The Software That Manages eBusiness™

## Checking a DSN

```
ADDRESS "TSO"  
"CLEAR"  
dsn_name = "crone90.crone.rexx(test)"  
dsn_check = SYSDSN("'"dsn_name"')  
SAY dsn_check
```

```
OK  
***
```



10 - 4

The Software That Manages eBusiness™

## EXECIO read

```


dsn_name = "crone90.crone.rexx(test)"
dsn_check = SYSDSN("'"dsn_name"'")
IF dsn_check = "OK" THEN DO
  "ALLOC DD(ddname) DSN("'"dsn_name"') SHR REU"
  "EXECIO 1 DISKR ddname (STEM row. FINIS)"
  SAY "The first line of the dataset is : "
  SAY row.1
END
ELSE DO
  SAY dsn_check
END

```

```

The first line of the dataset is :
WRITE "Enter a name"
***

```


Computer Associates™

10 - 5  
The Software That Manages eBusiness™

EXECIO always reads from DDNAME.

The TSO ALLOC command allocates a ddname.

EXECIO - Command

1 - Number of lines to read

DISKR - Read

ddname - The DD-Name

STEM - to indicate that a stem will be used.


row. - The STEM

FINIS - close the dataset.

## EXECIO read

```
dsn_name = "crone90.crone.rexx(test)"
dsn_check = SYSDSN("'"dsn_name"'")
IF dsn_check = "OK" THEN DO
  "ALLOC DD(ddname) DSN("'"dsn_name"'') SHR REU"
  "EXECIO * DISKR ddname (STEM row. FINIS)"
END
ELSE DO
  SAY dsn_check
END
DO line_counter = 1 TO row.0
  SAY row.line_counter
END
```

```
WRITE "Enter a name"
READ &name
WRITE &name
***
```

 Computer Associates™

10-6

The Software That Manages eBusiness™

\* - Number of lines to read

row.0 - the total number of lines read into the array. (the same as OUTTRAP)

## EXECIO write

```
dsn_name = "crone90.crone.rexx(test)"
dsn_check = SYSDSN("'"dsn_name'"")
IF dsn_check = "OK" THEN DO
  row.1 = "/* REXX */"
  "ALLOC DD(ddname) DSN(''"dsn_name'"') SHR REU"
  "EXECIO 1 DISKW ddname (STEM row. FINIS)"
END
ELSE DO
  SAY dsn_check
END
```



10-7

The Software That Manages eBusiness™

DISKW - This will write the rows to the member. This will overwrite the current member.

The member now contains

```
/* REXX */
```

## Work Section 10.1

- Write a REXX program to check that work section 9 member is a valid REXX program

```
/*-----REXX-----*/  
SAY "Hello"
```



10-8

The Software That Manages eBusiness™

## Work Section 10.2

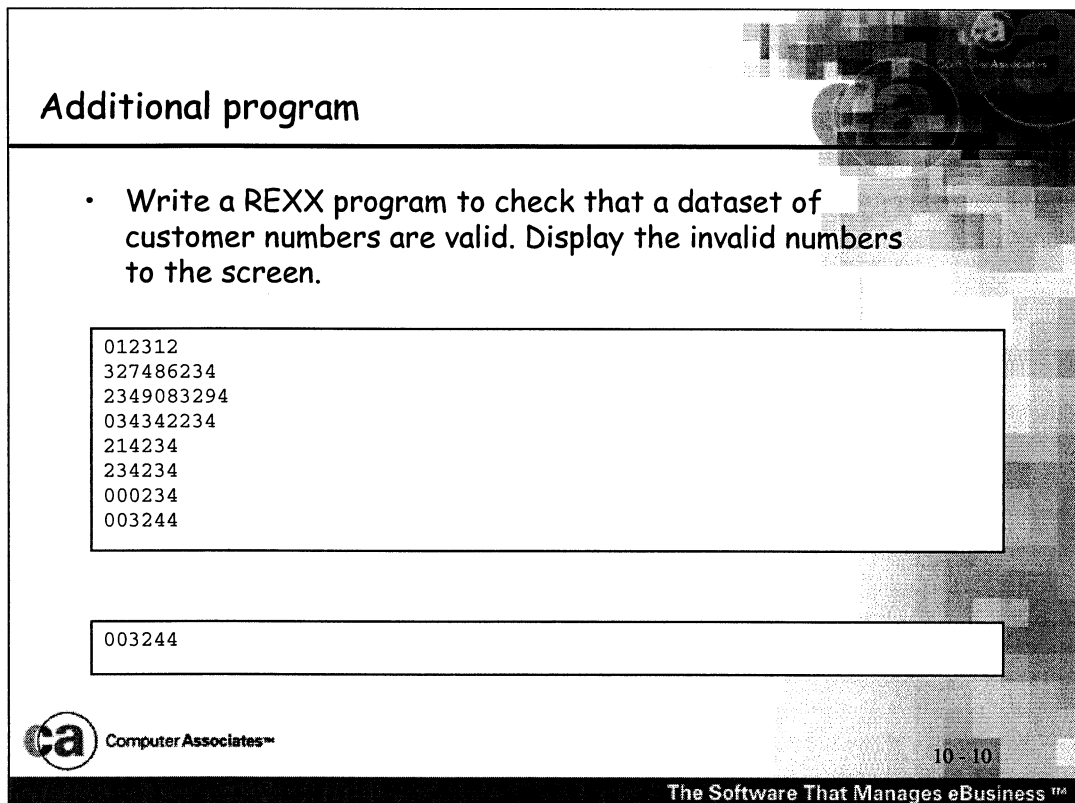
- Write a REXX program to create a new member called "generate".
- This new member will be a new executable REXX program.

```
/* REXX */  
SAY 'Enter your name'  
PARSE PULL name  
SAY 'Hello '||name
```



10-9

The Software That Manages eBusiness™



**Additional program**

- Write a REXX program to check that a dataset of customer numbers are valid. Display the invalid numbers to the screen.

```
012312
327486234
2349083294
034342234
214234
234234
000234
003244
```

```
003244
```

ca Computer Associates™

10 - 10

The Software That Manages eBusiness™

Valid Numbers are in the range 0 to 999999 and must be 6 characters long.

Create a test member called customer and manually put these customer numbers into that member.



# Data stacks

## Section 11



The Software That Manages eBusiness™

## Stacks

- A method to store data in memory without using variables.

Data 1
Data 2
Data 3
Data 4
Data 5



11-2

The Software That Manages eBusiness™

**QUEUE**

Data 1


Data 2

Data 3

Data 4

Data 5

Data 1
Data 2
Data 3
Data 4
Data 5



Computer Associates™


11 - 3

The Software That Manages eBusiness™

## QUEUE example

```
ADDRESS "TSO"
"CLEAR"
SAY "Please enter your name : "
PARSE EXTERNAL full_name un_used
SAY "Please enter another name : "
PARSE PULL second_name un_used
QUEUE full_name
QUEUE second_name
PARSE PULL name
SAY "The first line off the stack was : "
SAY name
PARSE PULL name
SAY "The second line off the stack was : "
SAY name
```

```
Please enter your name :
bob
Please enter another name :
jane
The first line off the stack was :
bob
The second line off the stack was :
jane
***
```



Computer Associates™

11-4

The Software That Manages eBusiness™


QUEUE will put the data in the stack in FIFO order.

PARSE PULL will remove the data from the stack one line at a time. The data will be pulled into a variable. (any variable can be used.)

It can be useful under TSO to only use EXTERNAL instead of pull when working with stacks, PARSE PULL will go to a stack for data before the screen.

**PUSH**

Data 1	Data 5
Data 2	Data 4
Data 3	Data 3
Data 4	Data 2
Data 5	Data 1

 Computer Associates™

11 - 5

The Software That Manages eBusiness™

## PUSH example

```
SAY "Please enter your name :"  
PARSE EXTERNAL full_name un_used  
SAY "Please enter another name :"  
PARSE PULL second_name un_used  
PUSH full_name  
PUSH second_name  
PARSE PULL name  
SAY "The first line off the stack was : "  
SAY name  
PARSE PULL name  
SAY "The second line off the stack was : "  
SAY name
```

```
Please enter your name :  
bob  
Please enter another name :  
jane  
The first line off the stack was :  
jane  
The second line off the stack was :  
bob  
***
```



PUSH will put the data in the stack in LIFO order.

## QUEUED() example

```
DO FOREVER
  SAY "Please enter your name : "
  PARSE UPPER EXTERNAL full_name un_used
  IF full_name = "" THEN DO
    LEAVE
  END
  ELSE DO
    QUEUE full_name
  END
END
no_in_stack = QUEUED()
DO no_in_stack
  PARSE PULL name
  SAY name
END
```



11 - 7

The Software That Manages eBusiness™

The QUEUED() function will return how many items are in the stack, and can be used to loop until the stack is empty.

## Output

```
Please enter your name :  
bob  
Please enter your name :  
jane  
Please enter your name :  
jim  
Please enter your name :  
  
BOB  
JANE  
JIM  
***
```



11-8

The Software That Manages eBusiness™



### Multiple Stacks

The diagram illustrates three overlapping stacks of data. The leftmost stack contains five items labeled 'Data 1' through 'Data 5' from top to bottom. The middle stack is shifted to the right and contains three items labeled 'Data 1', 'Data 2', and 'Data 3' from top to bottom. The rightmost stack is shifted further to the right and contains three items labeled 'Data 1', 'Data 2', and 'Data 3' from top to bottom. The overlapping nature of the stacks shows that the top of the rightmost stack is positioned above the middle of the middle stack, which is above the bottom of the leftmost stack.

Computer Associates™


11 - 9

The Software That Manages eBusiness™

More than one stack can be in use at any time, but you can only access the current stack. The current stack has to be deleted to access the stacks behind it.

## Multiple Stacks

<pre> SAY "Please enter your first name : " PARSE UPPER EXTERNAL fore_name un_used QUEUE fore_name "NEWSTACK" SAY "Please enter your surname : " PARSE UPPER EXTERNAL sur_name un_used QUEUE sur_name SAY "Please enter your surname : " PARSE UPPER EXTERNAL sur_name un_used QUEUE sur_name "QSTACK" SAY "You have "  RC  " of stacks." /*---- Empty stacks ----*/ PARSE PULL stuff SAY stuff PARSE PULL stuff SAY stuff "DELSTACK" PARSE PULL stuff SAY stuff </pre>	<pre> Please enter your first name : bob Please enter your surname : smith Please enter your surname : jones You have 2 of stacks.  SMITH JONES  BOB *** </pre>
---	---

 Computer Associates™

11 - 10

The Software That Manages eBusiness™

"NEWSTACK" creates an additional stack in front of the current stack. The newstack has to be removed before the data on the original stack can be retrieved.

"DELSTACK" deletes the current stack.

"QSTACK" returns the number of stacks into RC

## Unused stack data

```
tso_cmd = "LISTC LEVEL(IULC20) "  
PUSH tso_cmd  
EXIT
```



The Software That Manages eBusiness™

Putting TSO commands on the stack allows them to be executed on completion of the REXX program.

## EXECIO and stacks

```
dsn_name = "crone90.crone.rexx(test)"
dsn_check = SYSDSN("'"dsn_name"'")
IF dsn_check = "OK" THEN DO
  "ALLOC DD(ddname) DSN("'"dsn_name"' ) SHR REU"
  "EXECIO 1 DISKR ddname (FINIS)"
  PARSE PULL line
  SAY "The first line of the dataset is : "
  SAY line
END
ELSE DO
  SAY dsn_check
END
```



11 - 12

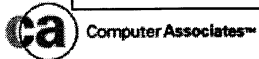
The Software That Manages eBusiness™

The data stack can be used with EXECIO instead of using STEM variables by removing the STEM option and variable.

## Work Section 11.1

- Write a REXX program to accept any number of names to the screen and when they type "STOP", display all the names.
- Store the names in a stack.

```
Please enter your name :  
jane  
Please enter your name :  
sue  
Please enter your name :  
stop  
JANE  
SUE  
***
```



## Work Section 11.2

- Write a REXX program to accept a list of names and store in a stack, then create a new stack and accept a number of Date of births and store them in the new stack.
- Display the contents of each stack showing the names in reverse order.

```
Please enter your name :  
bob  
Please enter your name :  
fred  
Please enter your name :  
stop  
Please enter your DOB :  
051276  
Please enter your DOB :  
040303  
Please enter your DOB :  
stop  
051276  
040303  
FRED  
BOB  
***
```



Computer Associates™

11 - 14

The Software That Manages eBusiness™

## Additional Program

- Using EXECIO read one of your members into a data stack and then display the contents one line at a time with the option to stop displaying the data.

```
Do you wish to see a line from the stack (Yes/end) ?
Y
/* REXX */
Do you wish to see a line from the stack (Yes/end) ?
Y
SAY 'Enter your name'
Do you wish to see a line from the stack (Yes/end) ?
end
***
```



11 - 15

The Software That Manages eBusiness™

Section end

ca Computer Associates™

11 - 16

The Software That Manages eBusiness™



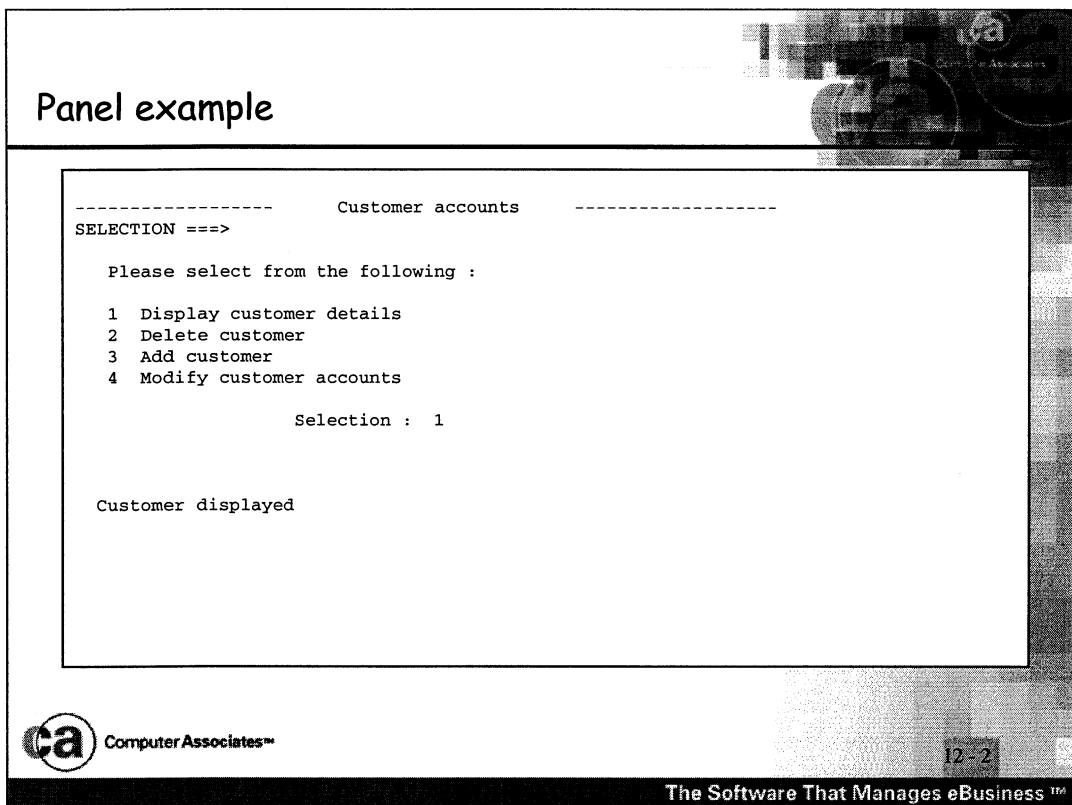
**Panels**

**Section 12**

12-1

ca Computer Associates™

The Software That Manages eBusiness™



Here is the ISPF panel. This is stored in the ISPPLIB dataset.

)BODY - Identifies the displayable section.

% - indicates highlighted protected field

+ - protected field

\_name - input field with variable

& - variable protected.

)INIT - section to indicate set-up details.

.CURSOR - set cursor location on panel start-up

)PROC - Executable section

VER - Verification options

LIST - Check a list

NB - Check value is None blank

)END - End of panel.

**Code**

```
CALL initialise

DO FOREVER
  CALL display_panel
  msg1 = ""
  return_code = RC
  IF return_code = 8 THEN DO
    EXIT
  END
  ELSE DO
    CALL options
  END
END
EXIT

initialise:
  ADDRESS ISPEXEC "LIBDEF ISPLLIB DATASET ID('crone90.rx201.isplib')"
```

12-4

Computer Associates™

The Software That Manages eBusiness™

```
ADDRESS ISPEXEC "LIBDEF ISPLLIB DATASET
ID('crone90.rx201.isplib')"
```

Address the ISPEXEC environment to define the library. (The display will work without this but will not display the latest version.)

```
ADDRESS ISPEXEC "DISPLAY PANEL(demo1)"
```

DISPLAY PANEL will display the member called demo1

Control will be returned when ever enter is pressed the DO FOREVER keeps the panel on the screen until PF3 is pressed and sends a return code of 8 to the program.

## Code - Continued

```
options:
  IF msg1 <> "" THEN DO
    RETURN
  END
  SELECT
    WHEN choice = 1 THEN DO
      CALL discust
      msg1 = RESULT
    END
    WHEN choice = 2 THEN DO
      CALL delcust
      msg1 = RESULT
    END
    WHEN choice = 3 THEN DO
      CALL addcust
      msg1 = RESULT
    END
    WHEN choice = 4 THEN DO
      CALL modcust
      msg1 = RESULT
    END
  OTHERWISE DO
    msg1 = "You must enter one of the options listed."
  END
END
RETURN
```



Computer Associates™

12-5

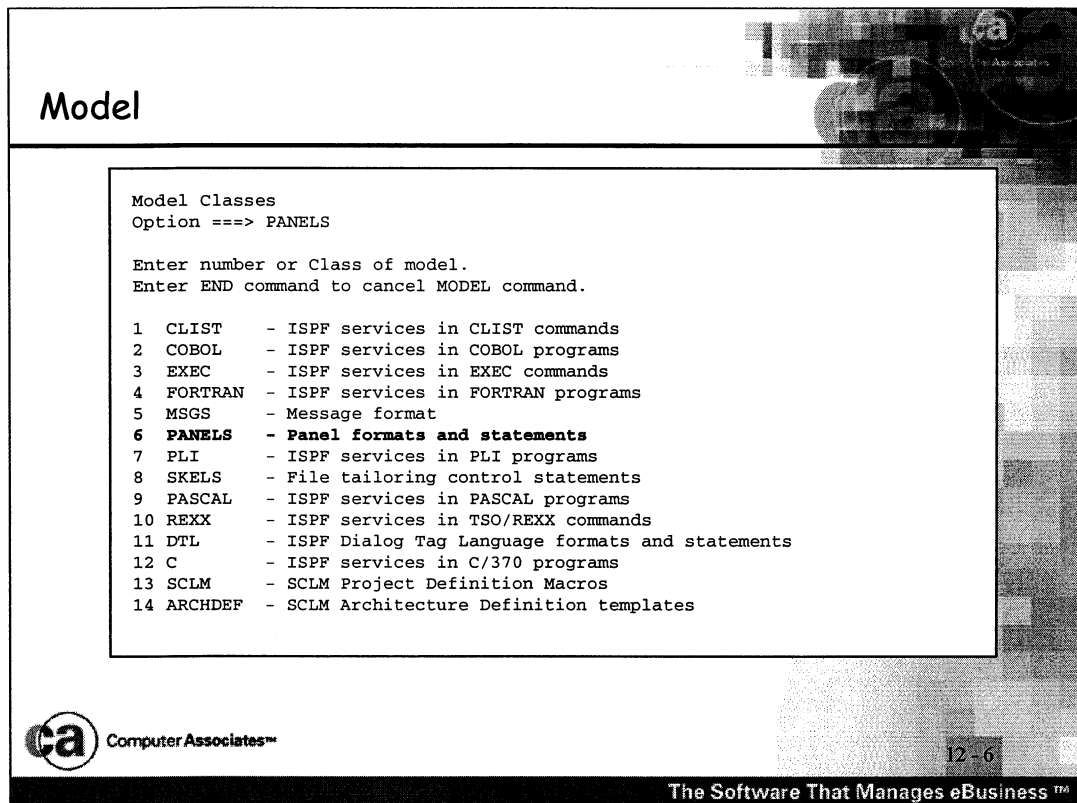
The Software That Manages eBusiness™

Calling external subroutines.

The returned value is stored in a system variable called RESULT.

The REXX program and the panel share the same variable pool and therefore variable assigned to the panel can be set in the program e.g.

```
msg1 = RESULT
```



Enter "MODEL" and a help screen for the system is displayed.

Select option PANELS for panels.


## Model

```
Panel Models
Option ==> ver

Enter number or statement name.
Enter END command to cancel MODEL command.

More:      +

S1 ASSIGN - Assignment statement
S2 ATTR  - )ATTR section header
S3 ATTRIB - Attribute character definition
S4 BODY  - )BODY section header
S5 CONTROL - Control variables
S6 IF    - If statement
S7 MODEL - )MODEL section header
S8 VER  - Verify statement
S9 VPUT  - Variable put statement
S10 REFRESH - Refetch variables prior to redisplay
S11 ATTRIBA - Other attribute types
S12 VGET  - Variable get statement
S13 PANEXIT - Panel Language Exit
S14 ABC   - Action bars
S15 KEYLIST - Keylist specification
S16 PDC   - Action bar pull-down
S17 VEDIT - Validate a variable
```



Computer Associates™

12-7

The Software That Manages eBusiness™

Select option VER for Verification section.

## Model


Verification Statements

Option ==> num

Enter number or statement name.  
Enter END command to cancel MODEL command.

More: +

V1	ALPHA	- Alphabetic or special characters
V2	BIT	- Binary characters
V3	DSNAME	- TSO data set name
V4	FILEID	- CMS file identification
V5	HEX	- Hexadecimal characters
V6	LIST	- List of valid values
V7	MEMNAME	- Data set member name
V8	NONBLANK	- Field must not be blank
V9	<b>NUM</b>	<b>- Numeric characters</b>
V10	PICT	- Mixed characters matching picture
V11	RANGE	- Numeric value within specified limits
V12	LENGTH	- Length of data stored in variable
V13	ENUM	- Extended Numeric
V14	ALPHAB	- Alphabetic characters
V15	INCLUDE	- Specify list of types
V16	LISTV	- Verify against one or more values
V17	LISTX	- Verify against a list of invalid values

 Computer Associates™

12-8

The Software That Manages eBusiness™

Select option NUM for Verification of numbers.



## Model

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          CRONE90.RX201.ISPPLIB(TEST1) - 01.00          Columns 00001 00072
Command ==>                                     Scroll ==> HALF
***** ***** Top of Data *****
000001      VER (VARNAME,NUM,MSG=MSGID)
=NOTE=
=NOTE=          VARNAME - NAME OF THE VARIABLE TO BE CHECKED FOR ALL NUMERIC
=NOTE=          CHARACTERS (0-9).
=NOTE=
=NOTE=          MSGID   - OPTIONAL, THE ERROR MESSAGE TO BE DISPLAYED IF THE
=NOTE=          VERIFY FAILS.
=NOTE=
=NOTE=          EXAMPLE:  VER (&SALES,NUM)
***** ***** Bottom of Data *****
```

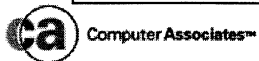


The help is displayed and the example code is entered into the member.

## Work Section 12.1

- Write a REXX program to display a selection panel.
- The panel should do nothing more than display the selection in a message section.

```
----- Customer accounts -----  
SELECTION ==>  
  
Please select from the following :  
  
1 Option 1  
2 Option 2  
3 Option 3  
4 Option 4  
5 Option 5  
  
Selection : 5  
  
You chose 5
```



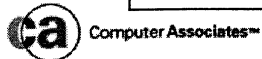
12 - 10

The Software That Manages eBusiness™

## Work Section 12.2

- Write a REXX program to display a name collection panel.
- When PF3 is pressed leave the panel and display the list of names.

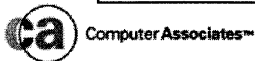
```
----- Name Collection Panel -----  
SELECTION ==>  
  
Please Enter a list of names :  
  
Names : JANE  
  
JANE stored.  
  
JANE  
***
```



## Additional Program

- Write a REXX program to add new customers to a dataset.
- Before writing the data to the dataset ask if the user is OK to save the data..
- The data should be customer name and postcode.

```
----- New Customer Entry Screen -----  
SELECTION ==>  
  
Please enter a customer name and postcode press enter for  
next customer.  
<PF3> to save the new customers.  
  
Customer Name : harrods  
Postcode      : W1  
  
HEINZ ASD213 accepted.
```



## Additional Program continued

```
Do you want to save the data (YES/NO) ?  
yes  
Data stored successfully.  
***
```

```
***** Top of Data *****  
000001 HEINZ ASD213  
000002 HARRODS W1  
***** Bottom of Data *****
```



Section end

ca Computer Associates™

12 - 14

The Software That Manages eBusiness™

## TSO REXX Keywords and functions

_ ABBREV	_ ADDRESS	_ ADDRESS()	_ ARG
_ ABS	_ BITAND	_ BITOR	_ BITXOR
_ ARG()	_ CALL	_ CENTER	_ CENTRE
_ B2X	_ CONDITION	_ COPIES	_ C2D
_ COMPARE	_ DATATYPE	_ DATE	_ DELSTACK
_ C2X	_ DELWORD	_ DIGITS	_ DO
_ DELSTR	_ DROPBUF	_ D2C	_ D2X
_ DROP	_ EXECIO	_ EXECUTIL	_ EXIT
_ ERRORTXT	_ FIND	_ FORM	_ FORMAT
_ EXTERNALS	_ GETMSG	_ HE	_ HI
_ FUZZ	_ IF	_ INDEX	_ INSERT
_ HT	_ ITERATE	_ JUSTIFY	_ LASTPOS
_ INTERPRET	_ LEFT	_ LENGTH	_ LINESIZE
_ LEAVE	_ MAKEBUF	_ MAX	_ MIN
_ LISTDSI	_ MVSVAR	_ NEWSTACK	_ NOP
_ MSG	_ OPTIONS	_ OTHERWISE	_ OUTTRAP
_ NUMERIC	_ PARSE	_ POS	_ PROCEDURE
_ OVERLAY	_ PULL	_ PUSH	_ QBUF
_ PROMPT	_ QSTACK	_ QUEUE	_ QUEUED
_ QELEM	_ RETURN	_ REVERSE	_ REXX
_ RANDOM	_ RT	_ SAY	_ SELECT
_ RIGHT	_ SIGN	_ SIGNAL	_ SOURCELI
_ SETLANG	_ SPACE	_ STORAGE	_ STRIP
_ SOURCELINE	_ SUBSTR	_ SUBWORD	_ SYMBOL
_ SUBCOM	_ SYSDSN	_ SYSVAR	_ TE
_ SYSCPUS	_ TRACE	_ TRACE()	_ TRANSLATE
_ TIME	_ TS	_ UPPER	_ USERID
_ TRUNC	_ VERIFY	_ WHEN	_ WORD
_ VALUE	_ WORDLENG	_ WORDLENGTH	_ WORDPOS
_ WORDINDEX	_ XRANGE	_ X2B	_ X2C
_ WORDS			
_ X2D			

### ABBREV

The ABBREV function is used to determine whether one character string is a prefix for another; or, in other words, whether one character string can be used as an abbreviation for another string.

### ABS

The ABS function returns the absolute value of a number.

### ADDRESS

The ADDRESS instruction is used to indicate what the destination of non-REXX commands is to be.

### ADDRESS ( )

The ADDRESS function returns the name of the destination to which commands are currently being routed

### ARG

ARG is used to parse out the arguments passed to a program or internal subroutine and store the arguments in variables.

### ARG ( )

The ARG function is used to return or test the presence of one of the arguments passed to a called subroutine or function.

### BITAND

The BITAND function logically AND's two strings bit-by-bit and returns the result of the AND operation

### BITOR

The BITOR function logically OR's two strings bit-by-bit and returns the result of the OR operation

### BITXOR

The BITXOR function logically EXCLUSIVE OR's two strings bit-by-bit and returns the result of the EXCLUSIVE OR operation.

### B2X

The B2X function converts a string of binary digits (i.e., zeros and ones) to a string of one or more upper-case hexadecimal digits in character format.

### CALL

The CALL statement is used to invoke a subroutine, a program or external routine, or a built-in function.

### CENTER or CENTRE

The CENTER function (also spelled CENTRE) is used to center one string within a certain length area, and pad on the left and right of the centered string with an optional padding character.

### COMPARE

The CENTER function (also spelled CENTRE) is used to center one string within a certain length area, and pad on the left and right of the centered string with an optional padding character.

### CONDITION

The CONDITION function is used to retrieve the setting information for the currently trapped REXX condition.

### COPIES

The COPIES function is used to concatenate or append a string to itself a certain number of times.

### C2D

The C2D function converts a string value into a decimal number.



### C2X

The C2X function converts a character string to its EBCDIC representation in hexadecimal.

### DATATYPE

The DATATYPE function is used to determine the data type of a string.

### DATE

The DATE function returns the current date.

### DELSTACK

The DELSTACK REXX command is used to delete the data stack that was created last in that REXX environment.

### DELSTR

The DELSTR function is used to delete or remove one string from within another string.

### DELWORD

The DELWORD function is used to delete a word or group of words from a character string.

### DIGITS

The DIGITS function is used to extract the current setting of the NUMERIC DIGITS option.

### DO

The DO instruction is used to execute a group of REXX statements under the control of an expression that determines how many times the DO statement set is to be executed.

### DROP

You can use the DROP statement to return one or more REXX variables to their initial or uninitialized state.

### DROPBUF

The DROPBUF REXX command is used to delete a data stack buffer from the data stack in that REXX environment.

### D2C

The D2C function is used to convert a decimal number into its internal hexadecimal format.

### D2X

The D2X function is used to convert a decimal number into the character representation of its internal hexadecimal format.

### ERRORTXT

The ERRORTXT function is used to extract from REXX the error message that is associated with a particular REXX error number.

### EXECIO

The EXECIO REXX command is used to perform read and write operations against a sequential data set or a PDS member.

### EXECUTIL

The EXECUTIL REXX command is used to control REXX processing options for the current REXX environment.

### EXIT

The EXIT instruction is used to unconditionally leave a program and (optionally) pass back a string to the caller being returned to.

### EXTERNALS

The EXTERNALS function is used to extract the number of terminal buffer or command stack elements that have been logically typed ahead by the terminal user.

### FIND

The FIND function is used to determine the position of a phrase of words within another string of words.

### FORM

The FORM function is used to extract the current setting of the NUMERIC FORM option.

### FORMAT

The FORMAT function is used to round off and format a number using REXX rules.

### FUZZ

The FUZZ function is used to extract the current setting of the NUMERIC FUZZ option.

### GETMSG

The GETMSG TSO/E external function retrieves in variables a message issued during an extended MCS console session that was established under TSO/E using the CONSOLE command.

### HE

The HE REXX immediate command is used to halt execution of a REXX exec.

### HI

The HI REXX command is used to immediately halt interpretation and execution of all active REXX execs.

### HT

The HT REXX command is used to immediately halt typing at the terminal, that is, all terminal output from REXX execs is prevented.

### IF

The IF instruction is used to conditionally execute a single REXX statement or a group of REXX statements.

### INDEX

The INDEX function is used to find the position of one character string within another character string.

### INSERT

The INSERT function is used to insert one string into another at a designated point.

### INTERPRET

The INTERPRET instruction is used to execute instructions that have been built dynamically

### ITERATE

The ITERATE instruction is used to restart the flow of execution in a repetitive DO loop.

### JUSTIFY

The JUSTIFY function justifies text (words with blanks between them) so that the text is evenly aligned on the left and right boundaries of the area whose width is given by the positive number 'length' in the function call.

### LASTPOS

The LASTPOS function returns the last occurrence position, relative to 1, of one character string within another.

### LEAVE

The LEAVE instruction causes REXX to stop executing the current DO-END loop just as if the DO loop termination condition had occurred.

### LEFT

The LEFT function is used to extract the leftmost characters of a string.

### LENGTH

The LENGTH function is used to obtain the length of the string passed to the function.

### LINESIZE

The LINESIZE function is used to determine the value that is the current setting for the line length of the TSO terminal.

### LISTDSI

The LISTDSI function will retrieve information about an existing data set and store the information in special REXX variables.

### MAKEBUF

The MAKEBUF REXX command is used to add a buffer to the data stack in the REXX environment.

### MAX

The MAX function returns the maximum numeric value from a list of numeric values.

### MIN

The MIN function returns the minimum numeric value from a list of numeric values.

### MSG

The MSG function is a TSO/E function, not a REXX function. It returns the prior setting of the message issuing feature, either ON OR OFF.

### MVSVAR

MVSVAR returns information about MVS, TSO/E, and the current session, such as the symbolic name of the MVS system, or the security label of the TSO/E session.

### NEWSTACK

The NEWSTACK REXX command is used to create a new data stack in the current REXX environment.

### NOP

The NOP instruction causes REXX to perform a 'no-operation' condition; that is, to do nothing.

### NUMERIC

The NUMERIC instruction sets controlling limits that govern how REXX evaluates and reports the results of arithmetic operations it performs.

### OPTIONS

OPTIONS is used pass processing options to REXX.

### OUTTRAP

The OUTTRAP function is used to trap the output from TSO/E commands.

### OVERLAY

The OVERLAY function is used to overlay a target string with another string, starting at a particular location within the target string.

### PARSE

The PARSE instruction tells REXX how to assign data to one or more variables.

### POS

The POS function returns the position, relative to 1, of one string within another.

### PROCEDURE

The PROCEDURE instruction is used within a called subroutine or function to protect (save) the variables in existence when the subroutine or function is called.

### PROMPT

The PROMPT function is used to turn on and off prompting of the TSO user when a TSO/E command in an exec needs input from the user.

### PULL

PULL is used to obtain the topmost element from the REXX data stack.

### PUSH

PUSH is used to place a new element on the top of the REXX data stack.

### QBUF

The QBUF REXX command is used to determine the number of data stack buffers that have been explicitly created by the MAKEBUF REXX command.

### QELEM

The QELEM REXX command is used to determine the number of data stack elements that are contained in the buffer that was most recently created by the MAKEBUF REXX command.

### QSTACK

The QSTACK REXX command is used to determine the total number of data stacks currently in existence.

### QUEUE

QUEUE is used to place a new element on the bottom of the REXX data stack.

### QUEUED

The QUEUED function returns the number of elements that remain on the REXX data stack.

### RANDOM

The RANDOM function generates a "pseudo-random" number that will lie somewhere between a supplied (or defaulted) upper and lower bound.

### RETURN

The RETURN instruction is used to pass control back to the caller of a subroutine or function. An optional result value can also be passed

### REVERSE

The REVERSE function reverses the order of all of the characters in a string.

### RIGHT

The RIGHT function returns the rightmost characters from a given string.

### RT

The RT REXX command is used to resume typing at the terminal after it has been previously stopped with the HT (Halt Typing) command.

### SAY

SAY is used to write a line of output to the TSO terminal or to the SYSTSPRT DD statement in batch.

### SELECT

The SELECT instruction causes REXX to execute one of several different instructions.

### SETLANG

The SETLANG function is used to query or query and set the language code used to control the language in which REXX messages are issued.

### SIGN

The SIGN function is used to determine the sign of a number.

### SIGNAL

The SIGNAL instruction is used to cause a change in the flow of execution within a REXX exec.

### SOURCELINE

The SOURCELINE function is used to extract the relative line number of a line within the source for current REXX exec.

### SPACE

The SPACE function is used to force padding space to be inserted or deleted between the words in a string.

### STORAGE

The STORAGE function is a TSO/E external function that is available within REXX in any MVS address space. It is used to retrieve some number of bytes from a specified main storage address, and optionally to store data at the specified main storage address.

### STRIP

The STRIP function is used to remove the leading and/or trailing characters from a character string.

### SUBCOM

The SUBCOM REXX command is used to determine whether or not a particular host command environment exists.

### SUBSTR

The SUBSTR function is used to extract a portion of a string.

### SUBWORD

The SUBWORD function is used to extract a substring of one or more words from a string.

### SYMBOL

The SYMBOL function is a TSO/E external function that is available from REXX in a TSO/E address space. SYMBOL is used to determine whether a particular symbol is a valid REXX symbol, a variable with a value assigned to it, or a literal.

### SYSCPUS

SYSCPUS places, in a stem variable, information about those CPUs that are online.

### SYSDSN

The SYSDSN function is a TSO/E function that is used to determine whether or not a particular data set exists, and to obtain some descriptive information about the data set.

### SYSVAR

The SYSVAR function is a TSO/E function that is used to extract information about the TSO/E and MVS environment.

### TE

The TE REXX command is used to stop interactive tracing of a REXX exec.

### TIME

The TIME function is a TSO/E external function that is available within REXX in a TSO/E address space.

### TRACE

The TRACE instruction is used to enable or disable selective tracing of the interpretation and execution of a REXX exec.

### TRACE ( )

The TRACE function is used to alter or return the current REXX trace options that are in effect.

### TRANSLATE

The TRANSLATE function is used to translate or reorder the characters in a string, based upon the contents of an input and output translate table.

### TRUNC

The TRUNC function call is used to truncate a number to an integer portion and a decimal fraction portion.

### TS

The TS REXX command is used to start interactive tracing of a REXX exec.

### UPPER

The UPPER function is used to translate the contents of a single variable or a list of variables to uppercase.

### USERID

The USERID function returns the TSO/E user ID of the TSO/E address space in which the USERID function is called.

### VALUE

The VALUE function is used to return the current value of a REXX symbol.

### VERIFY

The VERIFY function is used to check whether or not a string contains only certain characters.

### WORD

The WORD function is used to extract a specific word from within a string of words.

### WORDINDEX

The WORDINDEX function is used to extract the position, relative to 1, of the first character in a specific word within a string of words.

### WORDLENGTH

The WORDLENGTH function is used to extract the length in characters of a specific word contained within a string of words.

### WORDPOS

The WORDPOS function is used to extract position, relative to 1, of a specific word string within another word string.

### WORDS

The WORDS function is used to determine the number of words contained within a string of words.

### XRANGE

The XRANGE function is used to develop a contiguous EBCDIC characters in the range bounded by a supplied starting and ending character.

### X2B

The X2B function converts a string of hexadecimal digits (i.e., 0-9, A-F) to a string of four or more binary digits (i.e., zeros and ones) in character format.

### X2C

The X2C function is used to convert a string containing only hexadecimal characters into a printable EBCDIC character string.

### X2D

The X2D function is used to convert a string containing only hexadecimal characters into a numeric decimal string.

