8) Subroutines and functions

Functions: Internal, External, Built-in.

Instructions:
- CALL, SIGNAL,
- PROCEDURE, EXPOSE,
- RETURN, EXIT,
- INTERPRET

Special Variables RC, RESULT

Addressing:
- ADDRESS,
- OUTTRAP.

Resources: TSO/E REXX Reference
Chapter 4. Functions
Chapter 9. Reserved Keywords, Special Variables, and Command Names

This course has been prepared by Milos Forman for MCoE needs only!

PROPRIETARY AND CONFIDENTIAL INFORMATION

These education materials and related computer software program (hereinafter referred to as the "Education Materials") is for the end user's informational purposes only and is subject to change or withdrawal by CA, Inc. at any time.

These Education Materials may not be copied, transferred, reproduced, disclosed or distributed, in whole or in part, without the prior written consent of CA. These Education Materials are proprietary information and a trade secret of CA. Title to these Education Materials remains with CA, and these Education Materials are protected by the copyright laws of the United States and international treaties. All authorized reproductions must be marked with this legend.

RESTRICTED RIGHTS LEGEND

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is CA, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

Subroutines and functions
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

Function should always return a value

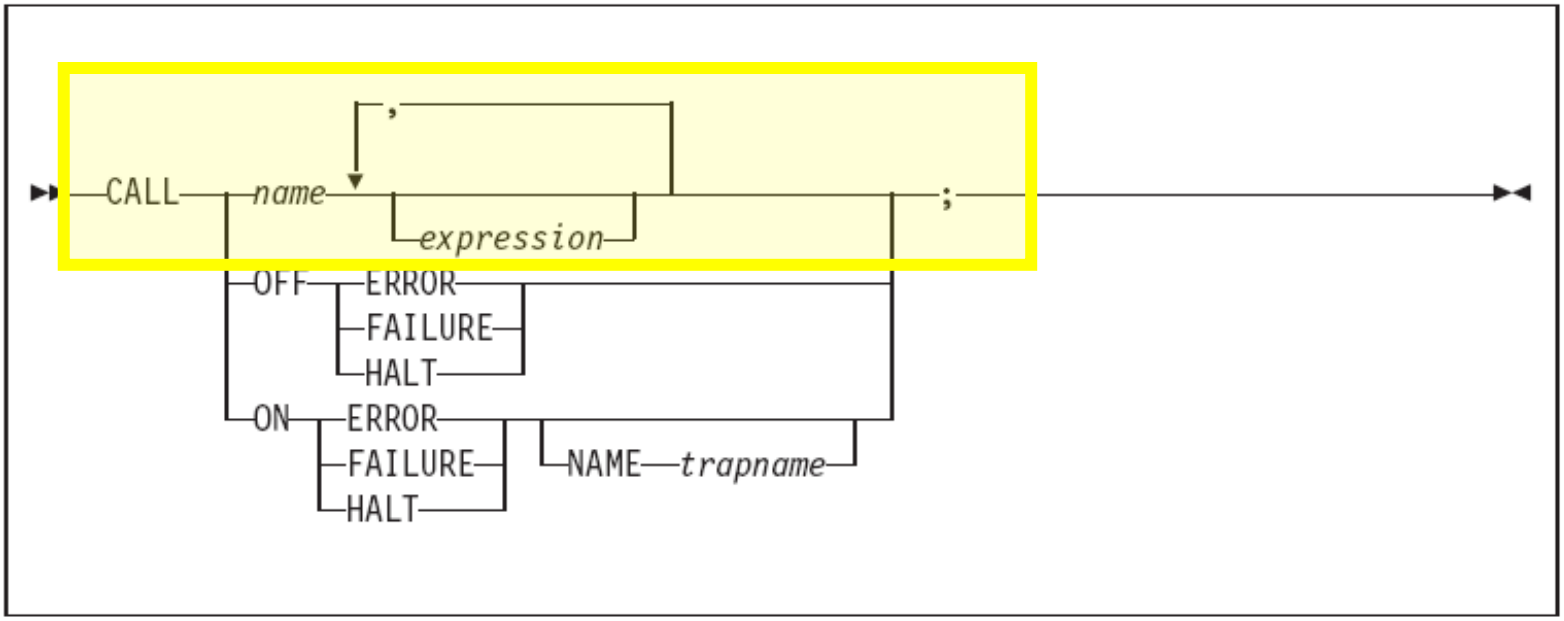
Subroutines and functions

Internal If the routine name exists as a label in the program, the current processing status is saved, so that it is later possible to return to the point of invocation to resume execution. Control is then passed to the first label in the program that matches the name. As with a

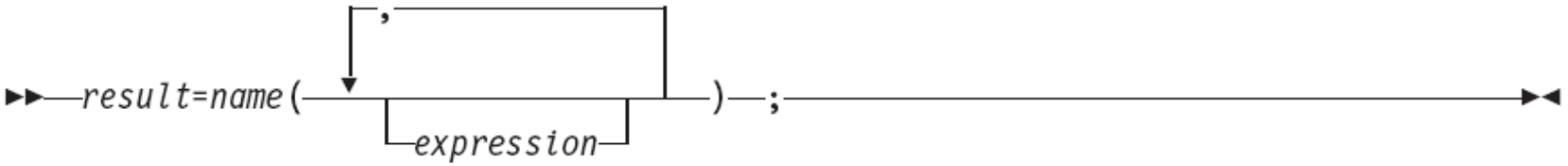
Built-in These functions are always available and are defined in the next section of this manual. (See pages 89 to 124.)

External You can write or use functions that are external to your program and to the language processor. An external routine can be written in any language (including REXX) that supports the system-dependent interfaces the language processor uses to call it. You can call a REXX program as a function and, in this case, pass more than one argument string. The ARG or PARSE ARG instructions or the ARG built-in function can retrieve these argument strings. When called as a function, a program must return data to the caller. For information

CALL a Subroutine

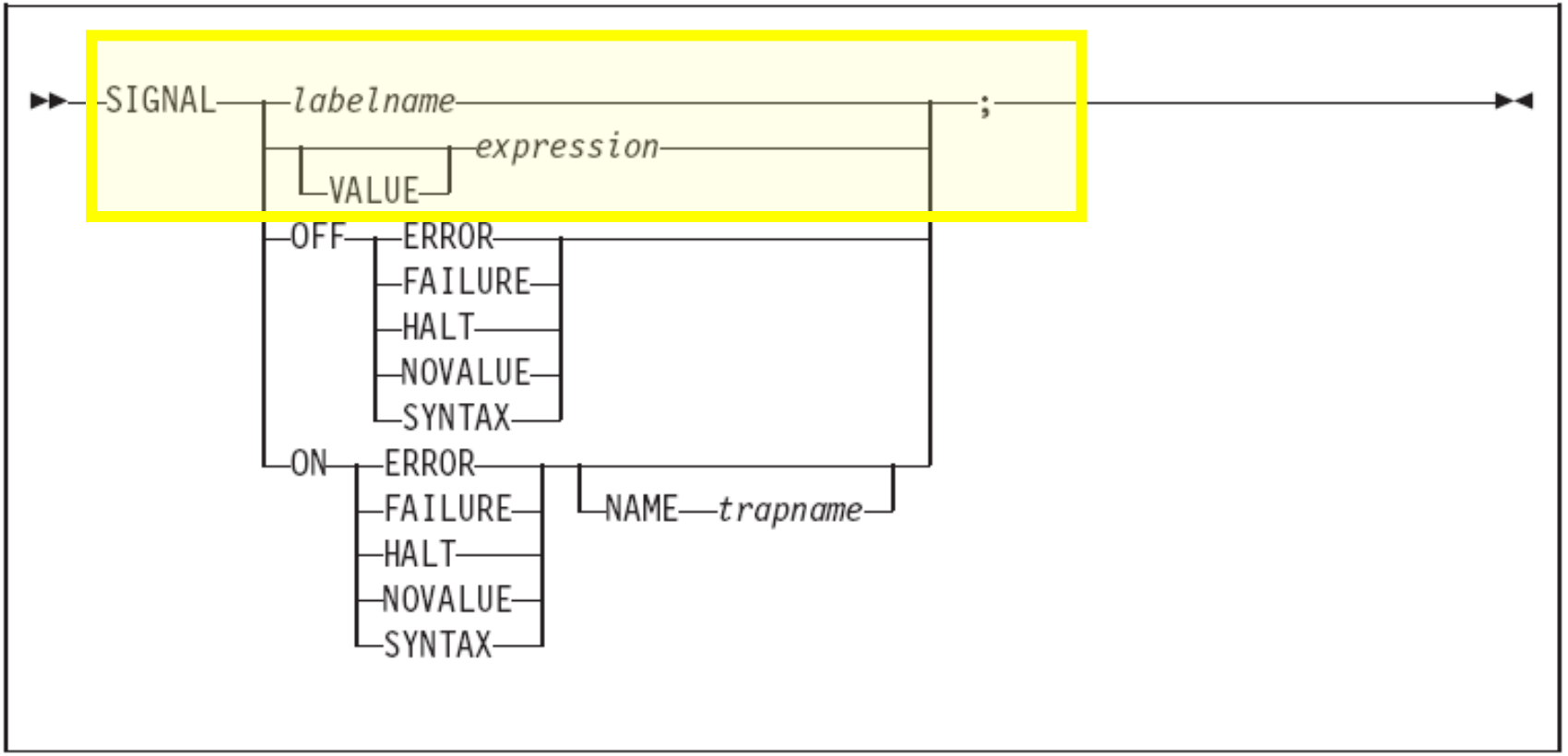


CALL a Function



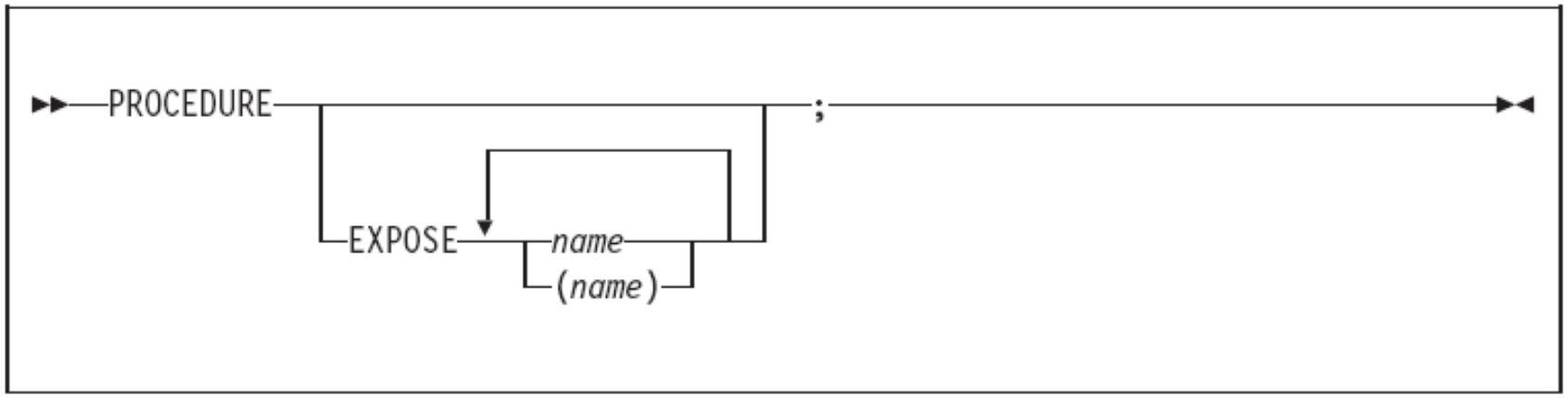
If *name* is a string (that is, you specify *name* in quotation marks), the search for internal routines is bypassed, and only a built-in function or an external routine is called.

SIGNAL

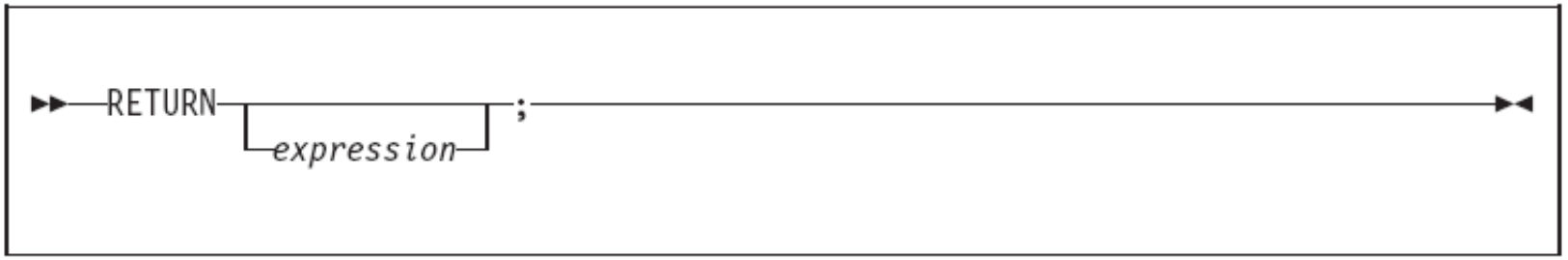


This course has been prepared by Milos Forman for MCoE needs only!

PROCEDURE



RETURN



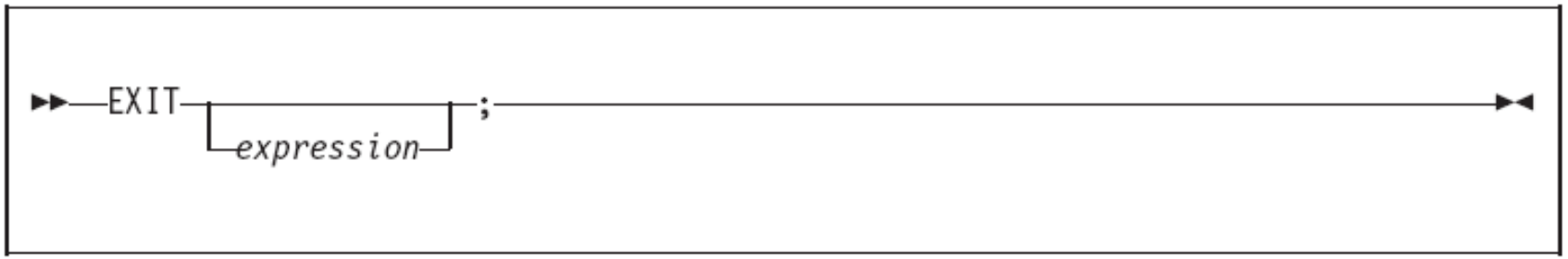
RETURN returns control (and possibly a result) from a REXX program or internal routine to the point of its invocation.

If no internal routine (subroutine or function) is active, RETURN and EXIT are identical in their effect on the program that is being run. (See 58.)

If a *subroutine* is being run (see the CALL instruction), *expression* (if any) is evaluated, control passes back to the caller, and the REXX special variable RESULT is set to the value of *expression*. If *expression* is omitted, the special variable RESULT is dropped (becomes uninitialized). The various settings saved at the time of the CALL (tracing, addresses, and so forth) are also restored. (See 48.)

If a *function* is being processed, the action taken is identical, except that *expression* **must** be specified on the RETURN instruction. The result of *expression* is then used in the original expression at the point where the function was called. See 83 for more details.

EXIT



EXIT leaves a program unconditionally. Optionally EXIT returns a character string to the caller. The program is stopped immediately, even if an internal routine is currently being run. If no internal routine is active, RETURN (see page 72) and EXIT are identical in their effect on the program that is being run.

If you specify *expression*, it is evaluated and the string resulting from the evaluation is passed back to the caller when the program stops.

Special Variables

RC is set to the return code from any executed host command (or subcommand). Following the SIGNAL events SYNTAX, ERROR, and FAILURE, RC is set to the code appropriate to the event: the syntax error number or the command return code. RC is unchanged following a NOVALUE or HALT event.

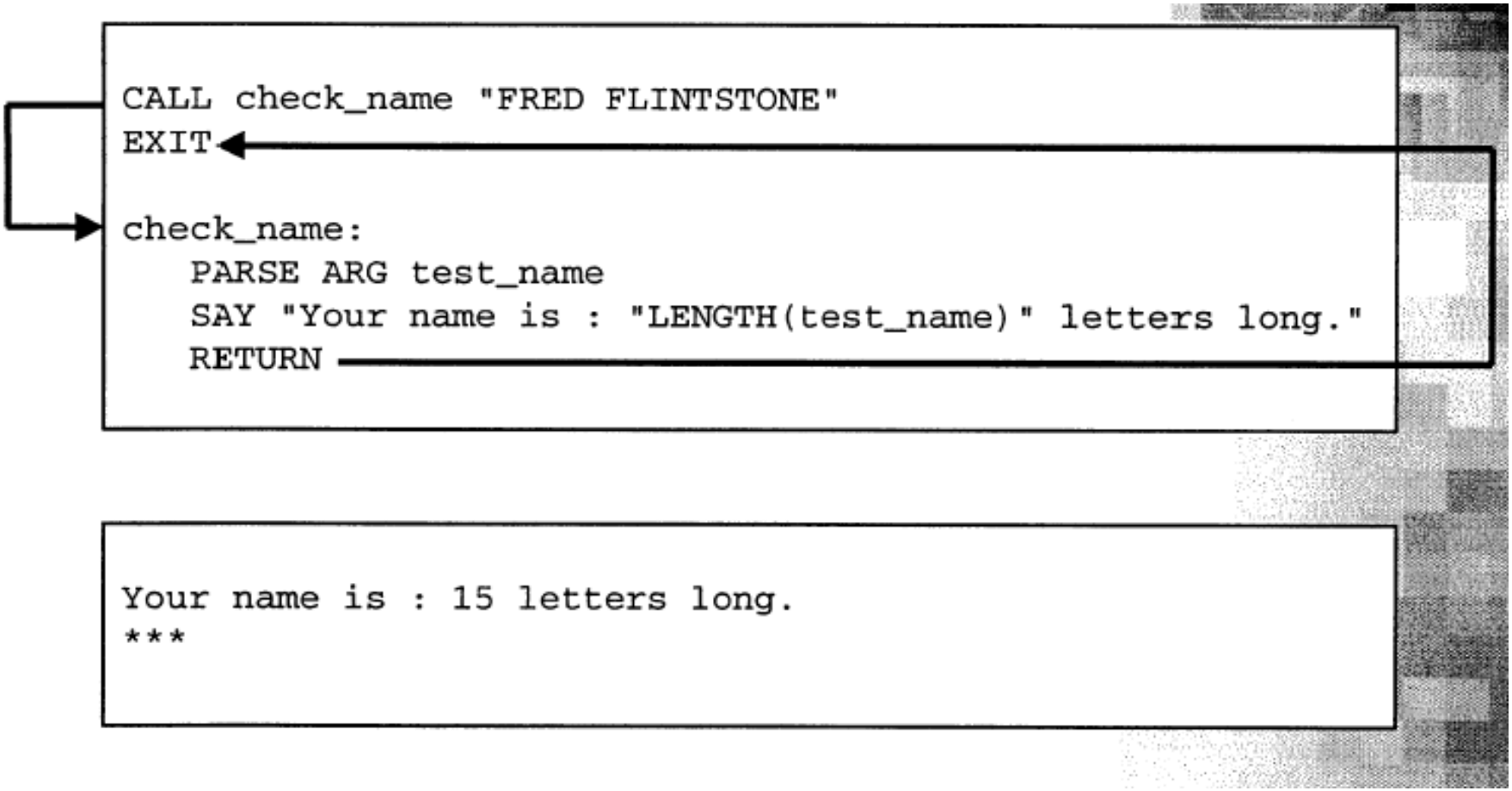
Note: Host commands issued manually from debug mode do not cause the value of RC to change.

RESULT is set by a RETURN instruction in a subroutine that has been called, if the RETURN instruction specifies an expression. If the RETURN instruction has no expression, RESULT is dropped (becomes uninitialized.)

CALL Parameters 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"

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
```

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
```

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
 ***
```

This course has been prepared by Milos Forman for MCoE needs only!

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) Subroutines and functions

Functions: Internal, External, Built-in.

Instructions:
- CALL, SIGNAL,
- PROCEDURE, EXPOSE,
- RETURN, EXIT,
- INTERPRET

Special Variables RC, RESULT

Addressing:
- ADDRESS,
- OUTTRAP.

Resources: TSO/E REXX Reference
Chapter 4. Functions
Chapter 9. Reserved Keywords, Special Variables, and Command Names

PROPRIETARY AND CONFIDENTIAL INFORMATION

These education materials and related computer software program (hereinafter referred to as the "Education Materials") is for the end user's informational purposes only and is subject to change or withdrawal by CA, Inc. at any time.
These Education Materials may not be copied, transferred, reproduced, disclosed or distributed, in whole or in part, without the prior written consent of CA. These Education Materials are proprietary information and a trade secret of CA. Title to these Education Materials remains with CA, and these Education Materials are protected by the copyright laws of the United States and international treaties. All authorized reproductions must be marked with this legend.

RESTRICTED RIGHTS LEGEND

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.
THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.
The manufacturer of this documentation is CA, Inc.
Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

Subroutines and functions
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

Function should always return a value

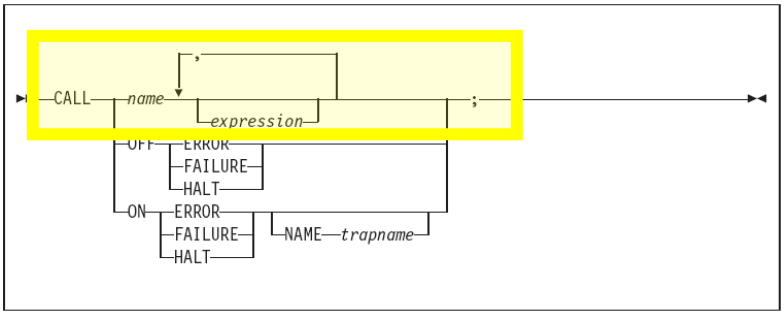
Subroutines and functions

Internal	If the routine name exists as a label in the program, the current processing status is saved, so that it is later possible to return to the point of invocation to resume execution. Control is then passed to the first label in the program that matches the name. As with a
Built-in	These functions are always available and are defined in the next section of this manual. (See pages 89 to 124.)
External	You can write or use functions that are external to your program and to the language processor. An external routine can be written in any language (including REXX) that supports the system-dependent interfaces the language processor uses to call it. You can call a REXX program as a function and, in this case, pass more than one argument string. The ARG or PARSE ARG instructions or the ARG built-in function can retrieve these argument strings. When called as a function, a program must return data to the caller. For information

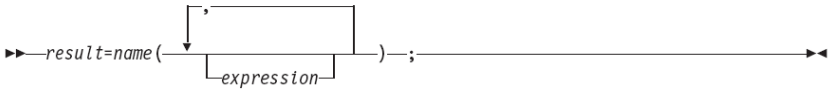
Built in functions were covered in previous presentation.

See ‘MCOE.REXA.REXX(NEWPDS)’ as an example of REXX which calls external subroutine ‘MCOE.REXA.REXX(NEWDATA)’.

CALL a Subroutine



CALL a Function



If *name* is a string (that is, you specify *name* in quotation marks), the search for internal routines is bypassed, and only a built-in function or an external routine is called.

5

CALL calls a routine (if you specify *name*) or controls the trapping of certain conditions (if you specify ON or OFF).

The routine called can be:

An internal routine

A function or subroutine that is in the same program as the CALL instruction that calls it. These are sequences of instructions, starting at the label that matches *name* in the CALL instruction.

A built-in routine

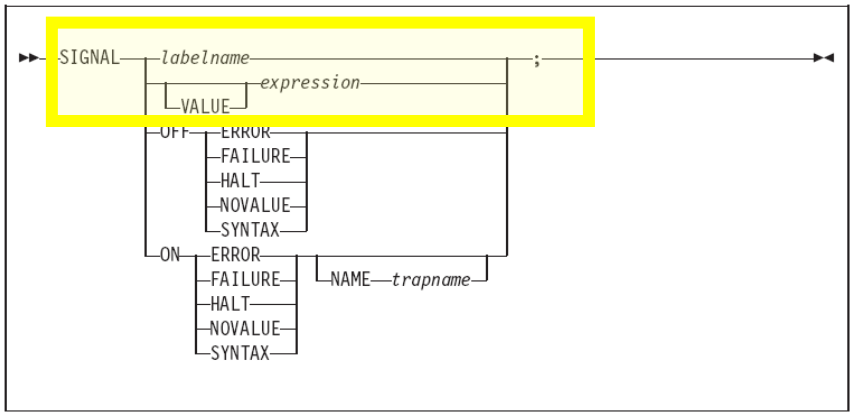
A function that is defined as part of the REXX language. These are routines built into the language processor for providing various functions. They always return a string that is the result of the routine.

An external routine

A function or subroutine that is neither built-in nor in the same program as the CALL instruction that calls it. Users can write or use routines that are external to the language processor and the calling program. You can code an external routine in REXX or in any language that supports the system-dependent interfaces. If the CALL instruction calls an external routine written in REXX as a subroutine, you can retrieve any argument strings with the ARG or PARSE ARG instructions or the ARG built-in function.

If *name* is a string (that is, you specify *name* in quotation marks), the search for internal routines is bypassed, and only a built-in function or an external routine is called. Note that the names of built-in functions

SIGNAL

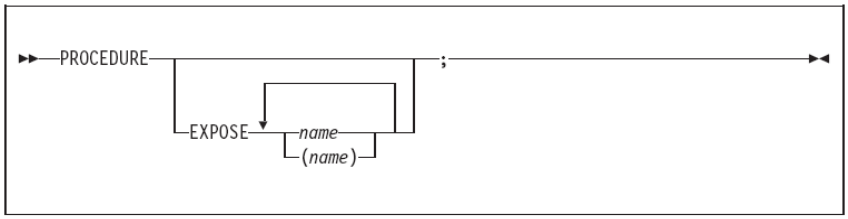


This course has been prepared by Milos Forman for MCoE needs only!

SIGNAL causes an *unusual* change in the flow of control (if you specify *labelname* or VALUE *expression*), or controls the trapping of certain conditions (if you specify ON or OFF). Simply: it works as a GO TO.

The *labelname* you specify must be a literal string or symbol that is taken as a constant. If you use a symbol for *labelname*, the search is independent of alphabetic case. If you use a literal string, the characters should be in uppercase.

PROCEDURE

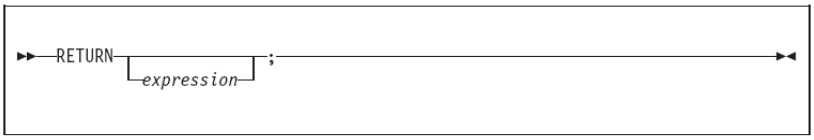


7

PROCEDURE within an internal routine, protects variables by making them unknown to the instructions that follow it. After a RETURN instruction is processed, the original variables environment is restored and any variables used in the routine (that were not exposed) are dropped.

EXPOSE keyword is used to unhide the callers varibles. It has meaning only for internal functions/procedures.

RETURN



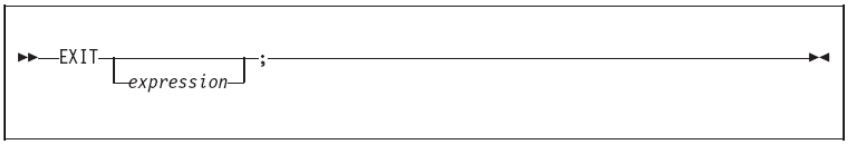
RETURN returns control (and possibly a result) from a REXX program or internal routine to the point of its invocation.

If no internal routine (subroutine or function) is active, RETURN and EXIT are identical in their effect on the program that is being run. (See 58.)

If a *subroutine* is being run (see the CALL instruction), *expression* (if any) is evaluated, control passes back to the caller, and the REXX special variable RESULT is set to the value of *expression*. If *expression* is omitted, the special variable RESULT is dropped (becomes uninitialized). The various settings saved at the time of the CALL (tracing, addresses, and so forth) are also restored. (See 48.)

If a *function* is being processed, the action taken is identical, except that *expression* **must** be specified on the RETURN instruction. The result of *expression* is then used in the original expression at the point where the function was called. See 83 for more details.

EXIT



EXIT leaves a program unconditionally. Optionally EXIT returns a character string to the caller. The program is stopped immediately, even if an internal routine is currently being run. If no internal routine is active, RETURN (see page 72) and EXIT are identical in their effect on the program that is being run.

If you specify *expression*, it is evaluated and the string resulting from the evaluation is passed back to the caller when the program stops.

Special Variables

RC is set to the return code from any executed host command (or subcommand). Following the SIGNAL events SYNTAX, ERROR, and FAILURE, RC is set to the code appropriate to the event: the syntax error number or the command return code. RC is unchanged following a NOVALUE or HALT event.

Note: Host commands issued manually from debug mode do not cause the value of RC to change.

RESULT is set by a RETURN instruction in a subroutine that has been called, if the RETURN instruction specifies an expression. If the RETURN instruction has no expression, RESULT is dropped (becomes uninitialized.)

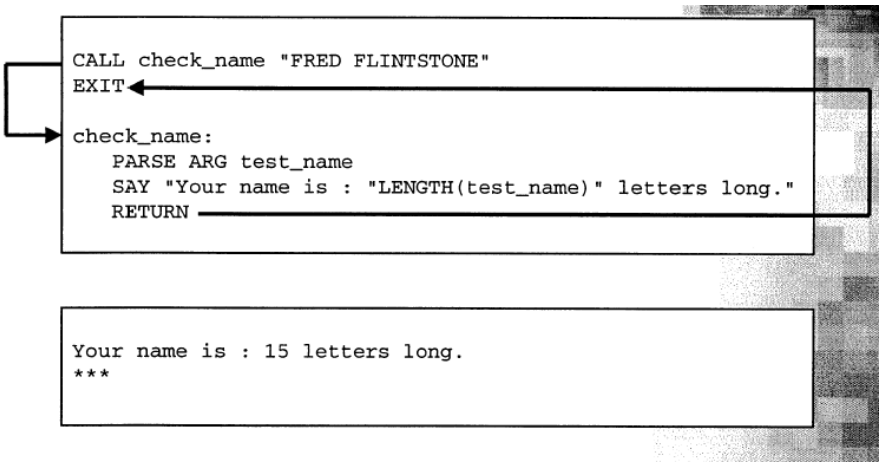
CALL Parameters and Commas

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"

Notice the differences in commas use and the results.

My recommendation: do not use any comas.

Internal Sample Program



12

Write it and test it.

See ‘MCOE.REXA.REXX(RX20184)’

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
```

13

Previous example as an external routine.

The executing program CALLs the member CHCKNAME (last three lines on the slide) from the same library.

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
```

14

Write it and test it.

Invoke it from 6 COMMAND: exec ,mcoe.rexa.rexx(rx20186)' 'Fred Bob Jane'

See ‘MCOE.REXA.REXX(RX20186)’

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
```

15

Write it and test it as an external routine.

The executing program CALLs this member NAME_CHECK from the same library.

See ‘MCOE.REXA.REXX(NEWPDS)’ as an example of REXX which calls external subroutine ‘MCOE.REXA.REXX(NEWDATA)’.

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
***
```

16

Write it and test it.

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
***
```


Write it and test it.

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
 ***
```


Write it and test it.

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
 ***
```

19

Write it and test it.