5) Debugging and error trapping

Instructions:
SIGNAL/CALL,
TRACE,

TSO Immediate commands
HT, RT,
HE, HI,
TE, TS.

Resources: TSO/E REXX User's Guide
Chapter 9. Diagnosing Problems Within an Exec

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.

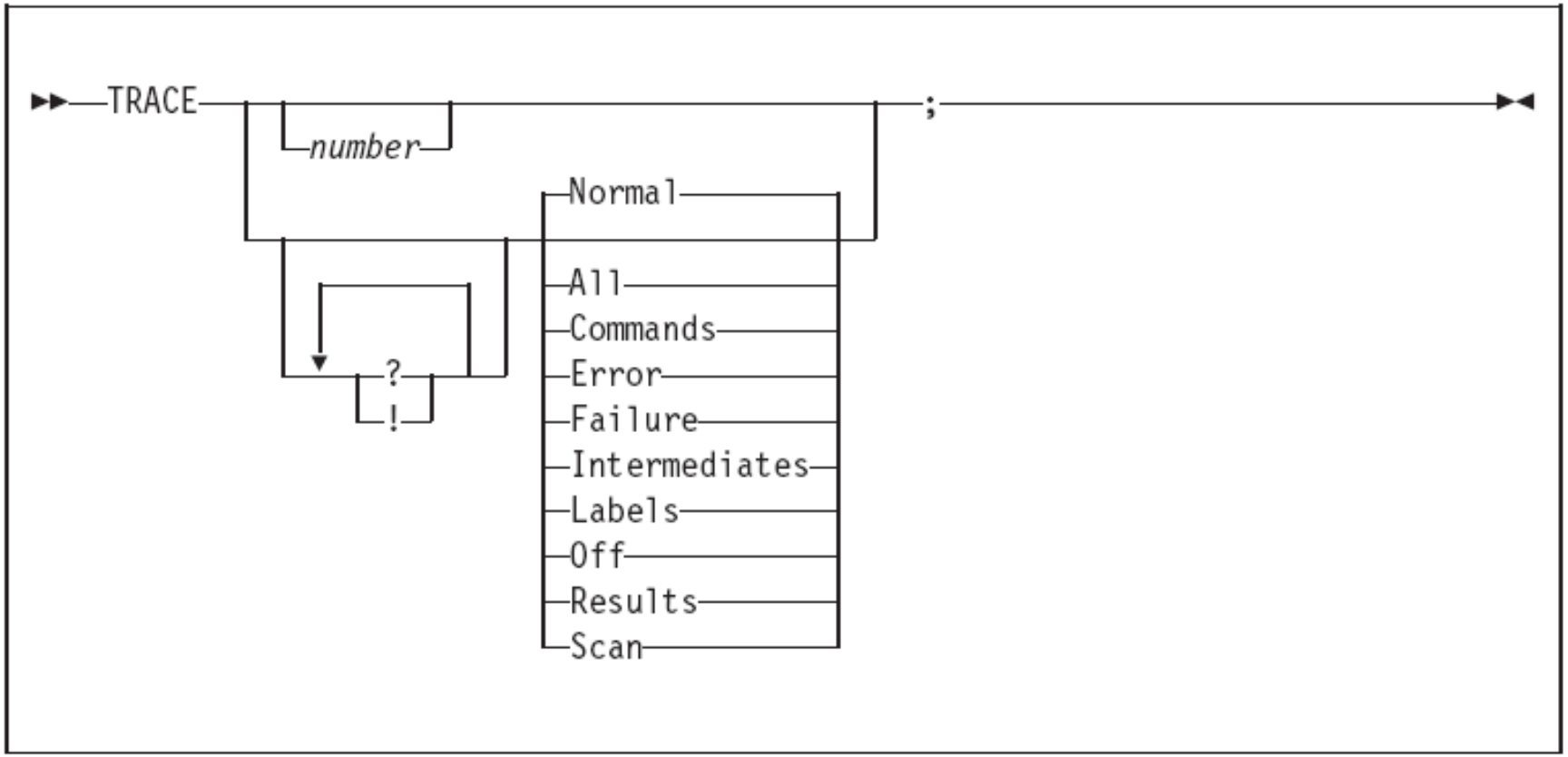
Debugging

- TRACE Instruction
  - Syntax errors
  - logic problems
  - often used when creating programs
- Error Trapping
  - abnormal situations
  - often used in production
- Dependent REXX platform
  - examples using TSO/E REXX

Interactive Tracing

- Before execution of a REXX program
  - EXECUTIL TS
- During execution of a REXX program
  - PA1 or ATTN
  - HI
- Commands in the REXX program
  - TRACE [?|!] option
  - EXECUTIL TS|TE|HT|RT|HI

TRACE



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

TRACE - output

`*-*`	Identifies the source of a single clause, that is, the data actually in the program.
`+++`	Identifies a trace message. This may be the nonzero return code from a command, the prompt message when interactive debug is entered, an indication of a syntax error when in interactive debug, or the traceback clauses after a syntax error in the program (see below).
`>>>`	Identifies the result of an expression (for `TRACE R`) or the value assigned to a variable during parsing, or the value returned from a subroutine call.
`>.>`	Identifies the value "assigned" to a placeholder during parsing (see 166).

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

TRACE A

TRACE Example - A

```
TRACE A
"test"
old_name = "Fred"
SAY "Please enter name : "
PARSE PULL tst_name
PARSE VAR tst_name new_name .
IF new_name = old_name THEN DO
   SAY "Hello Fred"
END
ELSE DO
   NOP
END
```

TRACE A (Cont.)

```
 12 *-* "test"
        >>>   "test"
 MISSING DSNAME
     13 *-* old_name = "Fred"
     14 *-* SAY "Please enter name : "
 Please enter name :
     15 *-* PARSE PULL tst_name
Fred
     16 *-* PARSE VAR tst_name new_name .
     17 *-* IF new_name = old_name
        *-*  THEN
        *-*  DO
     18 *-*   SAY "Hello Fred"
 Hello Fred
     19 *-*  END
 ***
```

```
 12 *-* : Source Code
 >>>    : Result of statement
 MISSING DSNAME : TSO Command
```

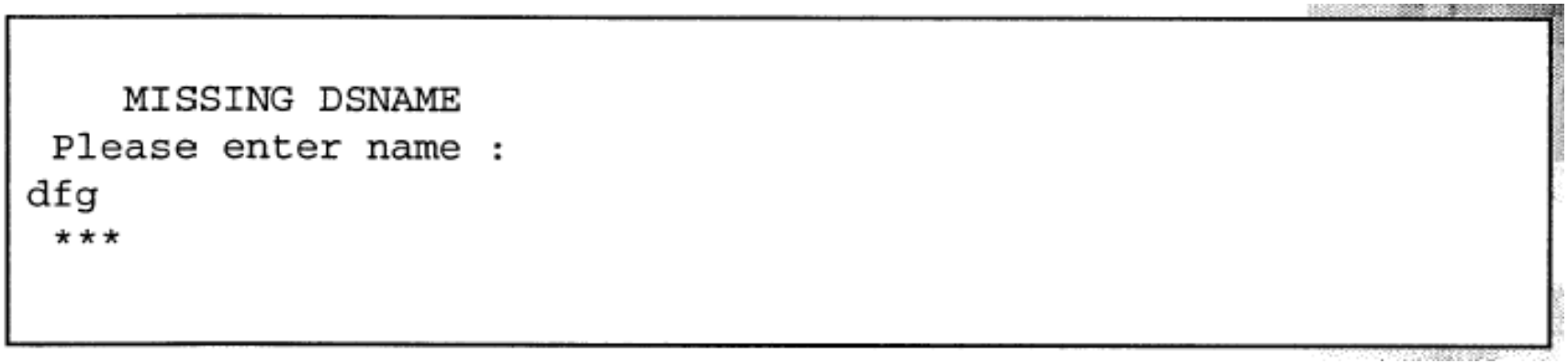
TRACE C

```
   12 *-* "test"
         >>>   "test"
 MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

TRACE E

```
 MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

TRACE F



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

TRACE I

```
 12 *-* "test"
        >L>   "test"
 MISSING DSNAME
     13 *-* old_name = "Fred"
        >L>   "Fred"
     14 *-* SAY "Please enter name : "
        >L>   "Please enter name : "
 Please enter name :
     15 *-* PARSE PULL tst_name
bob
        >>>   "                                                    bob"
     16 *-* PARSE VAR tst_name new_name .
        >>>   "bob"
        >.>   ""
     17 *-* IF new_name = old_name
        >V>   "bob"
        >V>   "Fred"
        >O>   "0"
     20 *-* ELSE
        *-*  DO
     21 *-*   NOP
     22 *-*  END
 ***
```

TRACE I - output

The following prefixes are used only if `TRACE Intermediates` is in effect:

>C>	The data traced is the name of a compound variable, traced after substitution and before use, provided that the name had the value of a variable substituted into it.
>F>	The data traced is the result of a function call.
>L>	The data traced is a literal (string, uninitialized variable, or constant symbol).
>O>	The data traced is the result of an operation on two terms.
>P>	The data traced is the result of a prefix operation.
>V>	The data traced is the contents of a variable.

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

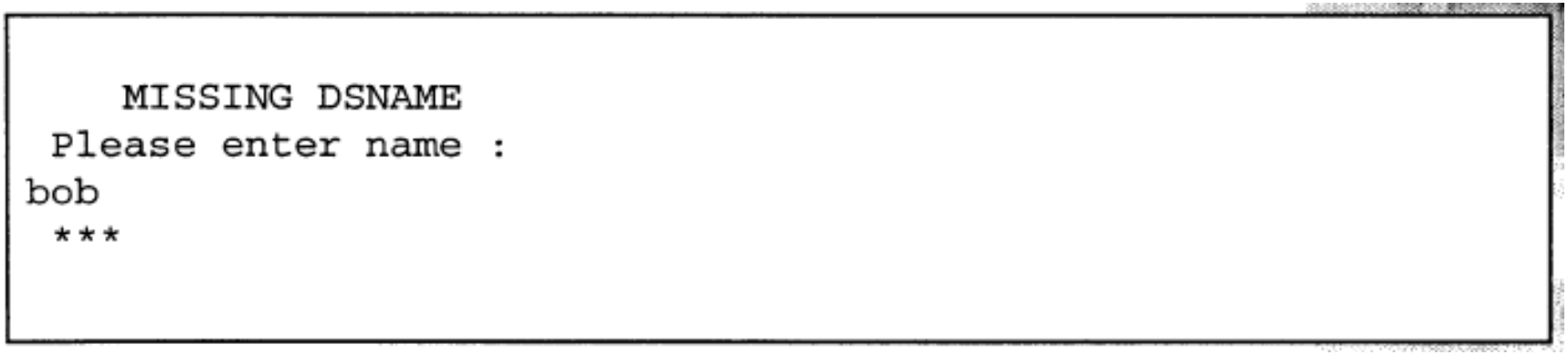
TRACE L

```
    MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

TRACE N

```
    MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

TRACE O



TRACE R

```
    12 *-* "test"
         >>>   "test"
 MISSING DSNAME
     13 *-* old_name = "Fred"
         >>>   "Fred"
     14 *-* SAY "Please enter name : "
         >>>   "Please enter name : "
 Please enter name :
     15 *-* PARSE PULL tst_name
bob
         >>>   "                                                    bob"
     16 *-* PARSE VAR tst_name new_name .
         >>>   "bob"
         >.>   ""
     17 *-* IF new_name = old_name
         >>>   "0"
     20 *-* ELSE
         *-*  DO
     21 *-*   NOP
     22 *-*  END
 ***
```

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

TRACE S

```
     11 *-* TRACE S
     12 *-* "test"
     13 *-* old_name = "Fred"
     14 *-* SAY "Please enter name : "
     15 *-* PARSE PULL tst_name
     16 *-* PARSE VAR tst_name new_name .
     17 *-* IF new_name = old_name
        *-*  THEN
        *-*  DO
     18 *-*   SAY "Hello Fred"
     20 +++   ELSE
Error running TRACE, line 20: Unexpected THEN or ELSE
***
```

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

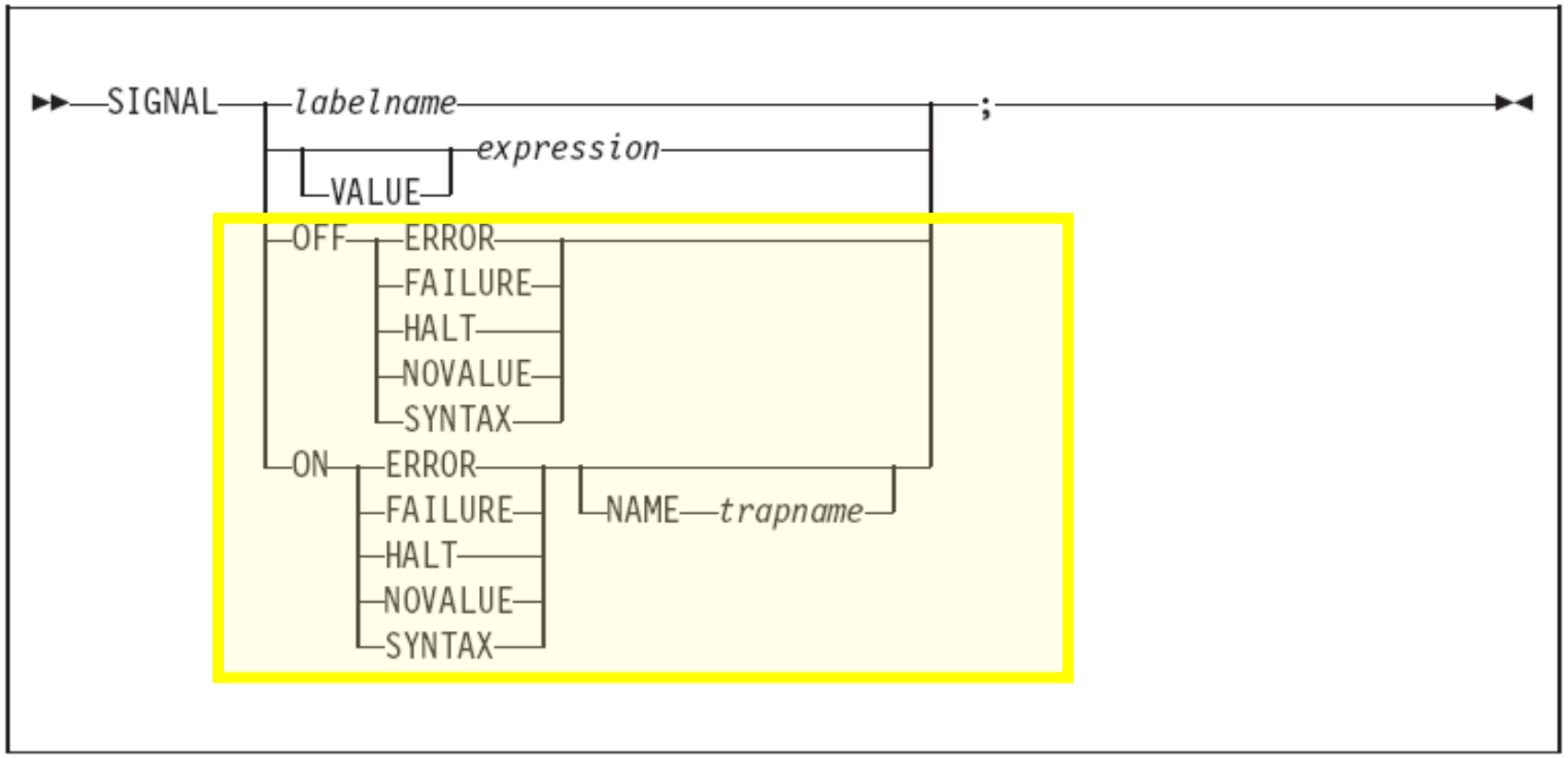
TRACE

? And ! In the TRACE Instruction

- ? - Toggles interactive debugging
  - display change variables
  - execute instructions
  - used with R and I e.g TRACE ?I

- ! - Toggles host command execution inhibit
  - will not execute TSO commands
  - use with C, R and I, or by itself.

SIGNAL



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

SIGNAL

The conditions and their corresponding events that can be trapped are:

ERROR

raised if a command indicates an error condition upon return. It is also raised if any command indicates failure and neither CALL ON FAILURE nor SIGNAL ON FAILURE is active. The condition is raised at the end of the clause that called the command but is ignored if the ERROR condition trap is already in the delayed state. The **delayed state** is the state of a condition trap when the condition has been raised but the trap has not yet been reset to the enabled (ON) or disabled (OFF) state.

In TSO/E, SIGNAL ON ERROR traps all positive return codes, and negative return codes only if CALL ON FAILURE and SIGNAL ON FAILURE are not set.

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

SIGNAL

FAILURE

raised if a command indicates a failure condition upon return. The condition is raised at the end of the clause that called the command but is ignored if the FAILURE condition trap is already in the delayed state.

In TSO/E, CALL ON FAILURE and SIGNAL ON FAILURE trap all negative return codes from commands.

HALT

raised if an external attempt is made to interrupt and end execution of the program. The condition is usually raised at the end of the clause that was being processed when the external interruption occurred.

For example, the TSO/E REXX immediate command HI (Halt Interpretation) or the EXECUTIL HI command raises a halt condition. The HE (Halt Execution) immediate command does not raise a halt condition. See “Interrupting Execution and Controlling Tracing” on page 247.

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

SIGNAL

NOVALUE

raised if an uninitialized variable is used:

- As a term in an expression
- As the *name* following the VAR subkeyword of a PARSE instruction
- As a variable reference in a parsing template, a PROCEDURE instruction, or a DROP instruction.

SYNTAX

raised if any language processing error is detected while the program is running. This includes all kinds of processing errors, including true syntax errors and “run-time” errors, such as attempting an arithmetic operation on nonnumeric terms. You can specify this condition only for SIGNAL ON.

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

SIGNAL ERROR Trapping

- Special variables assigned in TSO/E REXX.
  - SIGL
    - variable holds line number in error
  - SOURCELINE ()
    - used to extract the relative line number of a line within the source for current REXX exec.
  - ERRORTEXT(RC)
    - used to extract from REXX the error message that is associated with a particular REXX error number.
  - CONDITION
    - used to retrieve the setting information for the currently trapped REXX condition.

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

SIGNAL ERROR Trapping

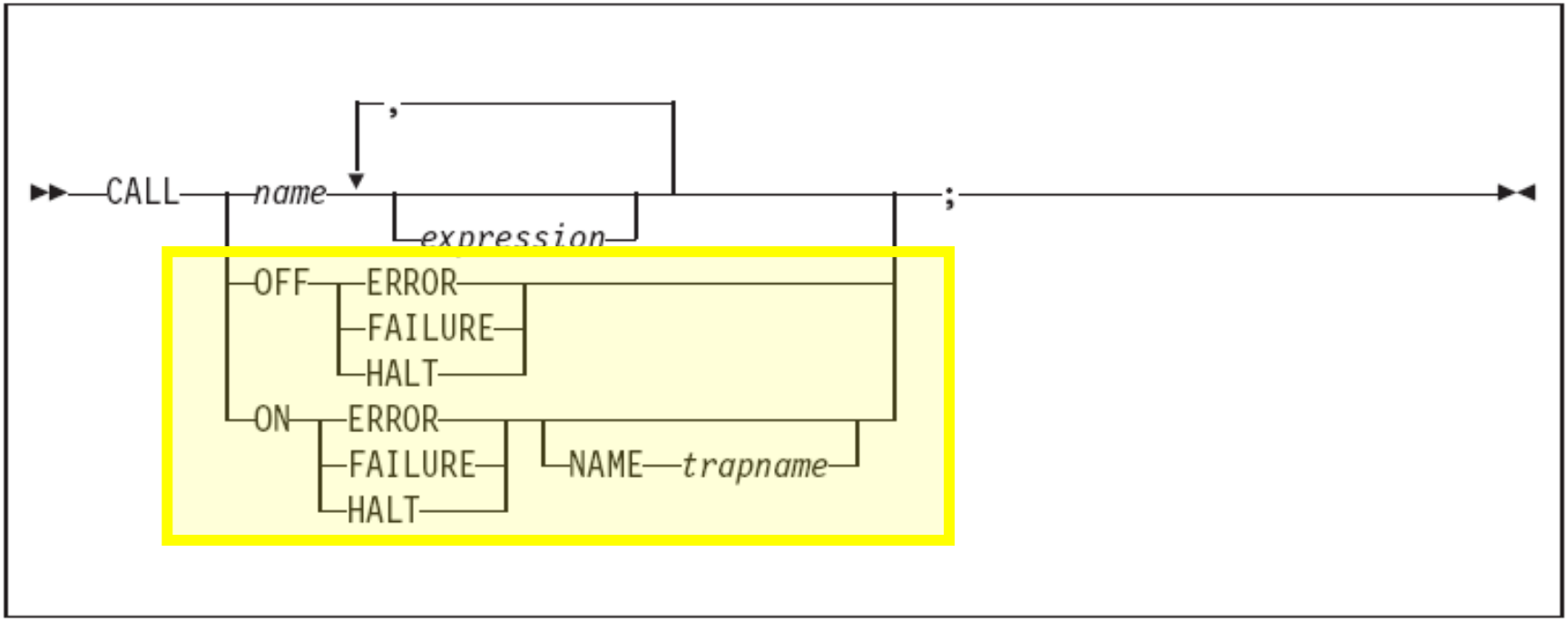
```
SIGNAL CN NOVALUE
IF new_name = old_name THEN DO
   SAY "Hello Fred"
END
EXIT

NOVALUE:
   SAY "Variable" CONDITION("D") has not been initialised on
   SAY "Line : "SIGL
   SAY "Source code : "SOURCELINE(SIGL)
```

```
Variable NEW_NAME HAS NOT BEEN INITIALISED ON
Line : 12
Source code : IF new_name = old_name THEN DO
***
```

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

CALL – ERROR Trapping



Immediate commands

Immediate commands are commands you can use if you are running a REXX exec in TSO/E and you press the attention interrupt key to enter attention mode. When you enter attention mode, the system displays the REXX attention prompting message, IRX0920I. In response to the message, you can enter an immediate command. The immediate commands are:

- HE – Halt Execution
- HI – Halt Interpretation
- HT – Halt Typing
- RT – Resume Typing
- TE – Trace End
- TS – Trace Start

TE and TS are also TSO/E REXX commands you can use in a REXX exec that runs in any address space. That is, TE and TS are available from the TSO and MVS host command environments.

Except for HE, when you enter an immediate command from attention mode in TSO/E, the system processes the command as soon as control returns to the exec but before the next statement in the exec is interpreted. For the HE immediate command, the system processes the command before control returns to the exec.

HE

HE (Halt Execution) is an immediate command you can use to halt the execution of a REXX exec. The HE immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter HE in response to the REXX attention prompting message, IRX0920I.

HE does not set the halt condition, which is set by the HI (Halt Interpretation) immediate command. If you need to halt the execution of an exec, it is recommended that you use the HI immediate command whenever possible. HE is useful if an exec is processing an external function or subroutine written in a programming language other than REXX and the function or subroutine goes into a loop.

HI

HI (Halt Interpretation) is an immediate command you can use to halt the interpretation of all currently executing execs. The HI immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter HI in response to the REXX attention prompting message, IRX0920I.

HT

HT (Halt Typing) is an immediate command you can use to suppress terminal output that an exec generates. The HT immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter HT in response to the REXX attention prompting message, IRX0920I.

RT

RT (Resume Typing) is an immediate command you can use to resume terminal output that was previously suppressed. The RT immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter RT in response to the REXX attention prompting message, IRX0920I. Terminal output that the exec generated after you issued the HT command and before you issued the RT command is lost.

TE

TE (Trace End) is an immediate command you can use to end tracing REXX execs.

TS

TS (Trace Start) is an immediate command you can use to start tracing REXX

Work section 5.2

- Execute the following Program using Trace ?i To see how REXX treats tails in compound variables.

```
TRACE ?I
day = "Tuesday"
month.day = "May"
tuesday = "Tuesday"
SAY month.tuesday
```

Additional Program

- Try using the TRACE() function instead
- e.g.
- TRACE(I)

- What is the difference ?

- Write a program using SIGNAL ON NOVALUE.
  - Use a SAY statement to display a variable which has not been initialised and use the NOVALUE label to give an appropriate message

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

5) Debugging and error trapping

Instructions:
SIGNAL/CALL,
TRACE,

TSO Immediate commands
HT, RT,
HE, HI,
TE, TS.

Resources: TSO/E REXX User's Guide
Chapter 9. Diagnosing Problems Within an Exec

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.

Debugging

- TRACE Instruction
  - Syntax errors
  - logic problems
  - often used when creating programs
- Error Trapping
  - abnormal situations
  - often used in production
- Dependent REXX platform
  - examples using TSO/E REXX

Interactive Tracing

- Before execution of a REXX program
  - EXECUTIL TS
- During execution of a REXX program
  - PA1 or ATTN
  - HI
- Commands in the REXX program
  - TRACE [?|!] option
  - EXECUTIL TS|TE|HT|RT|HI

4

EXECUTIL lets you change various characteristics that control how an exec processes in the TSO/E address space:

•**TS** Use TS (Trace Start) to start tracing execs.

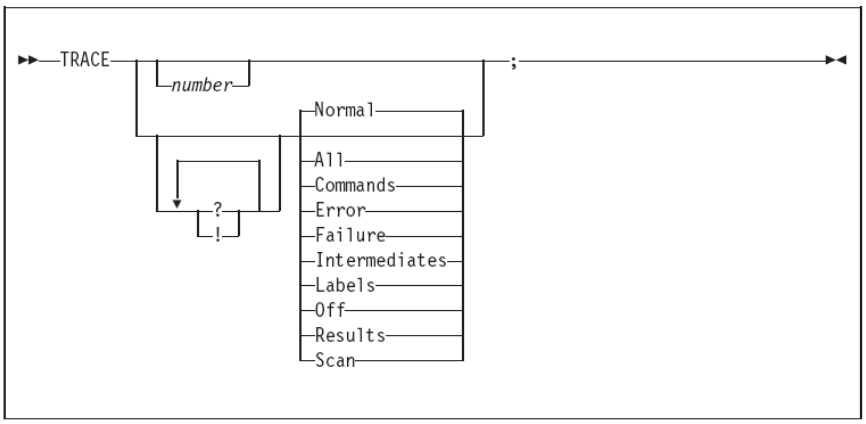
•**TE** Use TE (Trace End) to end tracing execs.

•**HT** Use HT (Halt Typing) to suppress terminal output generated by an exec.

•**RT** Use RT (Resume Typing) to resume terminal output that was previously suppressed.

•**HI** Use HI (Halt Interpretation) to halt the interpretation of all execs that are currently running in the language processor environment.

TRACE



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

5

•trace all - all REXX clauses are traced before they execute.

•trace commands - all TSO commands are traced before they are executed and any non-zero RC returned by a TSO command is also traced.

•trace error - any TSO command returning a non-zero RC is traced after it executes.

•trace intermediates - all clauses are traced before they execute, and the “intermediate” (working) results of evaluations of expressions and any substituted names are traced too.

•trace labels - labels on instructions that execute are traced.

•trace normal - a TSO command that returns a negative RC is traced after it executes – this trace option is the default if no other is specified.

•trace off - turns off all tracing.

•trace results - all REXX clauses are traced before they execute, and the final results of evaluating each expression are also traced. The values that are assigned via a PULL, and PARSE instructions are traced.

•trace scan - all clauses in the exec are traced, but not executed.

TRACE - output

`*-*`	Identifies the source of a single clause, that is, the data actually in the program.
`+++`	Identifies a trace message. This may be the nonzero return code from a command, the prompt message when interactive debug is entered, an indication of a syntax error when in interactive debug, or the traceback clauses after a syntax error in the program (see below).
`>>>`	Identifies the result of an expression (for `TRACE R`) or the value assigned to a variable during parsing, or the value returned from a subroutine call.
`>.>`	Identifies the value "assigned" to a placeholder during parsing (see 166).

TRACE A

TRACE Example - A

```
TRACE A
"test"
old_name = "Fred"
SAY "Please enter name : "
PARSE PULL tst_name
PARSE VAR tst_name new_name .
IF new_name = old_name THEN DO
   SAY "Hello Fred"
END
ELSE DO
   NOP
END
```

7

trace all - all REXX clauses are traced before they execute.

Following examples do not need necessarily work correctly. The **trace** instruction is important only. Change **trace** options and execute them.

See ‘MCOE.REXA.REXX(RX20156)’

TRACE A (Cont.)

```
 12 *-* "test"
       >>>   "test"
 MISSING DSNAME
     13 *-* old_name = "Fred"
     14 *-* SAY "Please enter name : "
 Please enter name :
     15 *-* PARSE PULL tst_name
Fred
     16 *-* PARSE VAR tst_name new_name .
     17 *-* IF new_name = old_name
        *-*  THEN
        *-*  DO
     18 *-*   SAY "Hello Fred"
 Hello Fred
     19 *-*  END
 ***
```

```
 12 *-* : Source Code
 >>>    : Result of statement
 MISSING DSNAME : TSO Command
```


Output from previous example execution.

TRACE C

```
  12 *-* "test"
       >>>   "test"
 MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

TRACE E

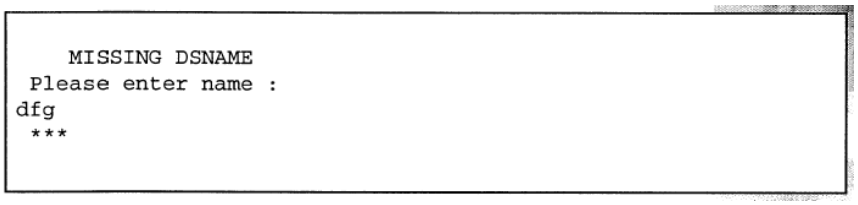
```
 MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

9

trace commands - all TSO commands are traced before they are executed and any non-zero RC returned by a TSO command is also traced.

trace error - any TSO command returning a non-zero RC is traced after it executes.

TRACE F



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

trace failure (same as **normal**) - a TSO command that returns a negative RC is traced after it executes – this trace option is the default if no other is specified.

TRACE I

```
 12 *-* "test"
        >L>   "test"
 MISSING DSNAME
     13 *-* old_name = "Fred"
        >L>   "Fred"
     14 *-* SAY "Please enter name : "
        >L>   "Please enter name : "
 Please enter name :
     15 *-* PARSE PULL tst_name
bob
        >>>   "                                                  bob"
     16 *-* PARSE VAR tst_name new_name .
        >>>   "bob"
        >.>   ""
     17 *-* IF new_name = old_name
        >V>   "bob"
        >V>   "Fred"
        >O>   "0"
     20 *-* ELSE
        *-*  DO
     21 *-*   NOP
     22 *-*  END
 ***
```

11

trace intermediates - all clauses are traced before they execute, and the “intermediate” (working) results of evaluations of expressions and any substituted names are traced too.

See ‘MCOE.REXA.REXX(RX201511)’

TRACE I - output

The following prefixes are used only if `TRACE Intermediates` is in effect:

>C>	The data traced is the name of a compound variable, traced after substitution and before use, provided that the name had the value of a variable substituted into it.
>F>	The data traced is the result of a function call.
>L>	The data traced is a literal (string, uninitialized variable, or constant symbol).
>O>	The data traced is the result of an operation on two terms.
>P>	The data traced is the result of a prefix operation.
>V>	The data traced is the contents of a variable.

TRACE L

```
    MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

TRACE N

```
    MISSING DSNAME
 Please enter name :
Fred
 Hello Fred
 ***
```

13

trace labels - labels on instructions that execute are traced.

trace normal - a TSO command that returns a negative RC is traced after it executes – this trace option is the default if no other is specified. See **trace failure**.

TRACE O

```
    MISSING DSNAME
 Please enter name :
bob
 ***
```


trace off - turns off all tracing.

TRACE R

```
    12 *-* "test"
          >>>   "test"
 MISSING DSNAME
     13 *-* old_name = "Fred"
          >>>   "Fred"
     14 *-* SAY "Please enter name : "
          >>>   "Please enter name : "
 Please enter name :
     15 *-* PARSE PULL tst_name
bob
          >>>   "                                                  bob"
     16 *-* PARSE VAR tst_name new_name .
          >>>   "bob"
          >.>   ""
     17 *-* IF new_name = old_name
          >>>   "0"
     20 *-* ELSE
          *-*  DO
     21 *-*   NOP
     22 *-*  END
 ***
```

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

trace results - all REXX clauses are traced before they execute, and the final results of evaluating each expression are also traced. The values that are assigned via a PULL, and PARSE instructions are traced.

TRACE S

```
    11 *-* TRACE S
    12 *-* "test"
    13 *-* old_name = "Fred"
    14 *-* SAY "Please enter name : "
    15 *-* PARSE PULL tst_name
    16 *-* PARSE VAR tst_name new_name .
    17 *-* IF new_name = old_name
       *-*  THEN
       *-*  DO
    18 *-*    SAY "Hello Fred"
    20 +++    ELSE
Error running TRACE, line 20: Unexpected THEN or ELSE
***
```

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

16

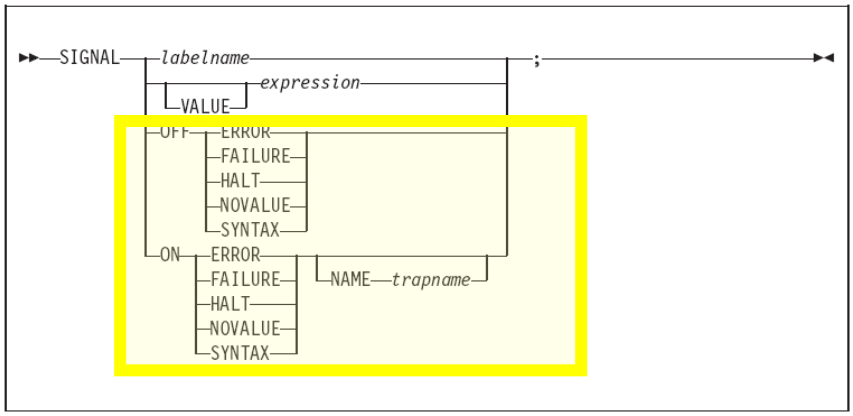
trace scan - all clauses in the exec are traced, but not executed.

TRACE

? And ! In the TRACE Instruction

- ? - Toggles interactive debugging
  - display change variables
  - execute instructions
  - used with R and I e.g TRACE ?I
- ! - Toggles host command execution inhibit
  - will not execute TSO commands
  - use with C, R and I, or by itself.

SIGNAL



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

18

SIGNAL controls the trapping of certain conditions (if you specify ON or OFF).

ERROR - A TSO command ends with a non-zero RC. The string that was processed and resulted in the error condition.

FAILURE - A TSO command ends with a negative RC. The string that was processed and resulted in the failure condition.

HALT - The terminal attention key is pressed and HI (Halt Interpretation) command is entered. Any string associated with the halt request. This can be the null string if no string was provided.

NOVALUE - An uninitialized variable is used in an expression or in a parse template. The derived name of the variable whose attempted reference caused the NOVALUE condition. The NOVALUE condition trap can be enabled only using SIGNAL ON.

SYNTAX - A REXX interpretation syntax error happens. Any string the language processor associated with the error. This can be the null string if you did not provide a specific string. Note that the special variables RC and SIGL provide information on the nature and position of the processing error.

SIGNAL

The conditions and their corresponding events that can be trapped are:

ERROR

raised if a command indicates an error condition upon return. It is also raised if any command indicates failure and neither CALL ON FAILURE nor SIGNAL ON FAILURE is active. The condition is raised at the end of the clause that called the command but is ignored if the ERROR condition trap is already in the delayed state. The **delayed state** is the state of a condition trap when the condition has been raised but the trap has not yet been reset to the enabled (ON) or disabled (OFF) state.

In TSO/E, SIGNAL ON ERROR traps all positive return codes, and negative return codes only if CALL ON FAILURE and SIGNAL ON FAILURE are not set.

SIGNAL

FAILURE

raised if a command indicates a failure condition upon return. The condition is raised at the end of the clause that called the command but is ignored if the FAILURE condition trap is already in the delayed state.

In TSO/E, CALL ON FAILURE and SIGNAL ON FAILURE trap all negative return codes from commands.

HALT

raised if an external attempt is made to interrupt and end execution of the program. The condition is usually raised at the end of the clause that was being processed when the external interruption occurred.

For example, the TSO/E REXX immediate command HI (Halt Interpretation) or the EXECUTIL HI command raises a halt condition. The HE (Halt Execution) immediate command does not raise a halt condition. See “Interrupting Execution and Controlling Tracing” on page 247.

SIGNAL

NOVALUE

raised if an uninitialized variable is used:

- As a term in an expression
- As the *name* following the VAR subkeyword of a PARSE instruction
- As a variable reference in a parsing template, a PROCEDURE instruction, or a DROP instruction.

SYNTAX

raised if any language processing error is detected while the program is running. This includes all kinds of processing errors, including true syntax errors and “run-time” errors, such as attempting an arithmetic operation on nonnumeric terms. You can specify this condition only for SIGNAL ON.

SIGNAL ERROR Trapping

- Special variables assigned in TSO/E REXX.
  - SIGL
    - variable holds line number in error
  - SOURCELINE ()
    - used to extract the relative line number of a line within the source for current REXX exec.
  - ERRORTEXT(RC)
    - used to extract from REXX the error message that is associated with a particular REXX error number.
  - CONDITION
    - used to retrieve the setting information for the currently trapped REXX condition.

Again: see ‘MCOE.REXA.REXX(RX201519)’

SIGNAL ERROR Trapping

```
SIGNAL ON NOVALUE
IF new_name = old_name THEN DO
   SAY "Hello Fred"
END
EXIT

NOVALUE:
   SAY "Variable" CONDITION("D") has not been initialised on
   SAY "Line : "SIGL
   SAY "Source code : "SOURCELINE(SIGL)
```

```
Variable NEW_NAME HAS NOT BEEN INITIALISED ON
Line : 12
Source code : IF new_name = old_name THEN DO
***
```

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

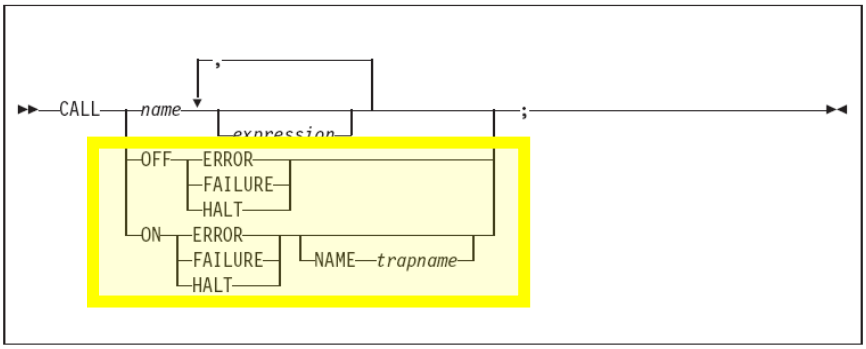
23

NOVALUE

An uninitialized variable is used in an expression or in a parse template. The derived name of the variable whose attempted reference caused the NOVALUE condition. The NOVALUE condition trap can be enabled only using SIGNAL ON.

See ‘MCOE.REXA.REXX(RX201521)’

CALL – ERROR Trapping



CALL

calls a routine (if you specify *name*) or controls the trapping of certain conditions (if you specify ON or OFF). To control trapping, you specify OFF or ON and the condition you want to trap. OFF turns off the specified condition trap. ON turns on the specified condition trap. It works similarly like SIGNAL ON instruction but If you use CALL, the *trapname* can be an internal label, a built-in function, or an external routine. If you use SIGNAL, the *trapname* can be only an internal label.

Immediate commands

Immediate commands are commands you can use if you are running a REXX exec in TSO/E and you press the attention interrupt key to enter attention mode. When you enter attention mode, the system displays the REXX attention prompting message, IRX0920I. In response to the message, you can enter an immediate command. The immediate commands are:

- HE – Halt Execution
- HI – Halt Interpretation
- HT – Halt Typing
- RT – Resume Typing
- TE – Trace End
- TS – Trace Start

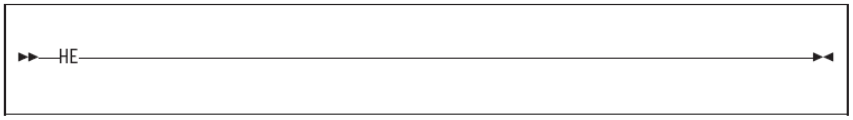
TE and TS are also TSO/E REXX commands you can use in a REXX exec that runs in any address space. That is, TE and TS are available from the TSO and MVS host command environments.

Except for HE, when you enter an immediate command from attention mode in TSO/E, the system processes the command as soon as control returns to the exec but before the next statement in the exec is interpreted. For the HE immediate command, the system processes the command before control returns to the exec.

25

See ‘MCOE.REXA.REXX(IMMEDIAT)’ – be careful to test it, immediate commands are strong.

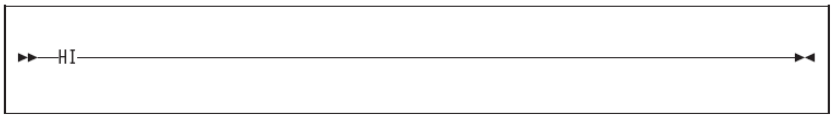
HE



HE (Halt Execution) is an immediate command you can use to halt the execution of a REXX exec. The HE immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter HE in response to the REXX attention prompting message, IRX0920I.

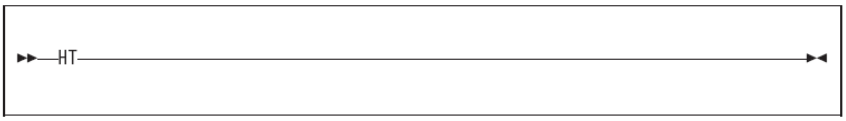
HE does not set the halt condition, which is set by the HI (Halt Interpretation) immediate command. If you need to halt the execution of an exec, it is recommended that you use the HI immediate command whenever possible. HE is useful if an exec is processing an external function or subroutine written in a programming language other than REXX and the function or subroutine goes into a loop.

HI



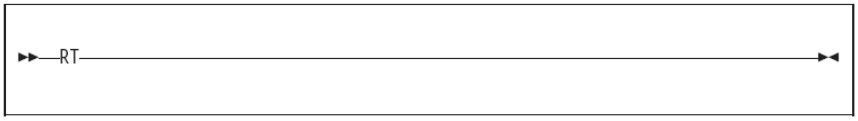
HI (Halt Interpretation) is an immediate command you can use to halt the interpretation of all currently executing execs. The HI immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter HI in response to the REXX attention prompting message, IRX0920I.

HT



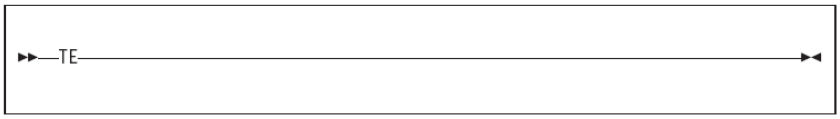
HT (Halt Typing) is an immediate command you can use to suppress terminal output that an exec generates. The HT immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter HT in response to the REXX attention prompting message, IRX0920I.

RT



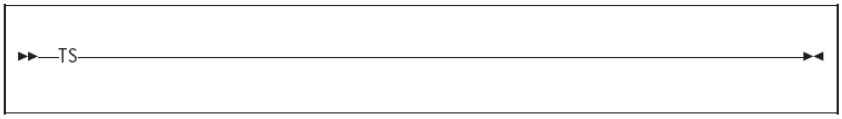
RT (Resume Typing) is an immediate command you can use to resume terminal output that was previously suppressed. The RT immediate command is available only if an exec is running in TSO/E and you press the attention interrupt key to enter attention mode. You can enter RT in response to the REXX attention prompting message, IRX0920I. Terminal output that the exec generated after you issued the HT command and before you issued the RT command is lost.

TE



TE (Trace End) is an immediate command you can use to end tracing REXX execs.

TS



TS (Trace Start) is an immediate command you can use to start tracing REXX

Work section 5.2

- Execute the following Program using Trace ?i To see how REXX treats tails in compound variables.

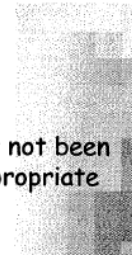
```
TRACE ?I
day = "Tuesday"
month.day = "May"
tuesday = "Tuesday"
SAY month.tuesday
```

Additional Program

- Try using the TRACE() function instead
- e.g.
- TRACE(I)

- What is the difference ?

- Write a program using SIGNAL ON NOVALUE.
  - Use a SAY statement to display a variable which has not been initialised and use the NOVALUE label to give an appropriate message



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

I cannot see any difference between **trace i** and **trace(i)** specification, you do not need to test it.

And see the 'MCOE.REXA.REXX(RX201521)' regarding the last assignment.