

IBM / Technical Newsletter

This Newsletter No. SN28-1293
Date February 10, 1989
Base Publication No. SC28-1883-0
File No. S370-39

Prerequisite Newsletters/
Supplements None

TSO Extensions Version 2 REXX Reference

©Copyright IBM Corp. 1988

TSO Extensions Version 2, Program Number 5685-025

This newsletter contains replacement pages for *TSO Extensions Version 2 REXX Reference* in support of TSO Extensions Version 2, Program Number 5685-025.

Before inserting any of the attached pages into *TSO Extensions Version 2 REXX Reference* read *carefully* the instructions on this cover. They indicate when and how you should insert pages.

Pages to be Removed

Cover - Edition Notice
iii - xiv
51 - 52
247 - 248
271 - 272
425 - 426
431 - 448

Attached Pages to be Inserted*

Cover - Edition Notice
iii - xiv
51 - 52
247 - 248
271 - 272.2
425 - 426
431 - 448

*If you are inserting pages from different Newsletters/Supplements and *identical* page numbers are involved, always use the page with the latest date (shown in the slug at the top of the page). The page with the latest date contains the most complete information.

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

Summary of Amendments

This newsletter documents the following new and changed information for TSO/E Version 2 support of the REXX programming language:

- New information about how to initialize a language processor environment if you use a user-written terminal monitor program (TMP)
- New values returned by the PARSE VERSION instruction, for example, the language level description and language processor release date. The new values support APAR OY17590 and are returned *only* if you install the PTF that supports the APAR.

This newsletter also includes minor technical changes.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

IBM Corporation, Information Development, Dept. D58, Building 921-2,
P.O. Box 950, Poughkeepsie, New York 12602



100



100





TSO Extensions Version 2

REXX Reference

SC28-1883-0



First Edition (December 1988)

This edition with Technical Newsletter SN28-1293 applies to the TSO Extensions (TSO/E) Version 2 Licensed Program, Program Number 5685-025, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product in this publication is not intended to state or imply that only IBM's product may be used. Any functionally equivalent product may be used instead. This statement does not expressly or implicitly waive any intellectual property right IBM may hold in any product mentioned herein.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921, PO Box 950, Poughkeepsie, New York 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Before Using the Information in This Book

Before you use the information in this book, please read "Summary of Changes" on page 425. This topic lists the instructions, functions, and services described in this book that support various APARs.

Contents

| | |
|--|----|
| Chapter 1. Introduction | 1 |
| Who This Book Is For | 1 |
| What Systems Application Architecture Is Supported Environments | 2 |
| Supported Environments | 2 |
| Common Programming Interface | 3 |
| How to Use This Book | 4 |
| How to Read the Syntax Diagrams | 5 |
| For Further REXX Information | 6 |
| | |
| Chapter 2. General Concepts | 7 |
| Brief Description of the Restructured Extended Executor Language | 7 |
| Where to Find More Information | 8 |
| Structure and General Syntax | 8 |
| Tokens | 9 |
| Implied Semicolons | 12 |
| Continuations | 12 |
| Expressions and Operators | 13 |
| Expressions | 13 |
| Operators | 13 |
| String Concatenation | 13 |
| Arithmetic | 14 |
| Comparison | 14 |
| Logical (Boolean) | 15 |
| Parentheses and Operator Precedence | 16 |
| Examples | 17 |
| Clauses and Instructions | 17 |
| Null Clauses | 17 |
| Labels | 17 |
| Assignments | 18 |
| Keyword Instructions | 18 |
| Commands | 18 |
| Assignments and Symbols | 18 |
| Constant Symbols | 19 |
| Simple Symbols | 19 |
| Compound Symbols | 19 |
| Stems | 20 |
| Notes | 21 |
| Commands to External Environments | 22 |
| Environment | 22 |
| Commands | 22 |
| Host Commands and Host Command Environments | 23 |
| The TSO Environment | 24 |
| The ISPEXEC and ISREDIT Environments | 24 |
| The MVS Environment | 24 |
| The LINK and ATTACH Environments | 25 |
| | |
| Chapter 3. Keyword Instructions | 27 |
| ADDRESS | 28 |
| ARG | 30 |
| CALL | 32 |
| DO | 35 |
| Simple DO Group | 35 |

| | |
|---------------------------------------|-----------|
| Simple Repetitive Loops | 36 |
| Controlled Repetitive Loops | 36 |
| Conditional Phrases (WHILE and UNTIL) | 38 |
| DROP | 39 |
| EXIT | 40 |
| IF | 41 |
| INTERPRET | 42 |
| ITERATE | 44 |
| LEAVE | 45 |
| NOP | 46 |
| NUMERIC | 47 |
| OPTIONS | 49 |
| PARSE | 50 |
| PROCEDURE | 53 |
| PULL | 55 |
| PUSH | 56 |
| QUEUE | 57 |
| RETURN | 58 |
| SAY | 59 |
| SELECT | 60 |
| SIGNAL | 62 |
| TRACE | 64 |
| Alphabetic Character (Word) Options | 65 |
| Prefix Options | 65 |
| Numeric Options | 66 |
| Tracing Tips | 66 |
| A Typical Example | 67 |
| Format of TRACE Output | 67 |
| UPPER | 69 |
| Chapter 4. Functions | 71 |
| Syntax | 71 |
| Calls to Functions and Subroutines | 72 |
| Search Order | 73 |
| Errors during Execution | 76 |
| Built-in Functions | 77 |
| ABBREV | 78 |
| ABS | 78 |
| ADDRESS | 78 |
| ARG | 79 |
| BITAND | 80 |
| BITOR | 80 |
| BITXOR | 81 |
| CENTRE/CENTER | 81 |
| COMPARE | 82 |
| CONDITION | 82 |
| COPIES | 83 |
| C2D | 83 |
| C2X | 84 |
| DATATYPE | 84 |
| DATE | 85 |
| DBCS | 86 |
| DELSTR | 87 |
| DELWORD | 87 |
| DIGITS | 87 |
| D2C | 88 |

| | |
|---------------------------|-----|
| D2X | 88 |
| ERRORTXT | 89 |
| EXTERNALS | 89 |
| FIND | 90 |
| FORM | 90 |
| FORMAT | 90 |
| FUZZ | 91 |
| INDEX | 92 |
| INSERT | 92 |
| JUSTIFY | 93 |
| LASTPOS | 93 |
| LEFT | 94 |
| LENGTH | 94 |
| LINESIZE | 94 |
| LISTDSI | 95 |
| MAX | 95 |
| MIN | 95 |
| MSG | 95 |
| OUTTRAP | 95 |
| OVERLAY | 96 |
| POS | 96 |
| PROMPT | 96 |
| QUEUED | 97 |
| RANDOM | 97 |
| REVERSE | 98 |
| RIGHT | 98 |
| SIGN | 98 |
| SOURCELINE | 99 |
| SPACE | 99 |
| STORAGE | 99 |
| STRIP | 100 |
| SUBSTR | 100 |
| SUBWORD | 101 |
| SYMBOL | 101 |
| SYSDSN | 101 |
| SYSVAR | 102 |
| TIME | 102 |
| TRACE | 103 |
| TRANSLATE | 104 |
| TRUNC | 104 |
| USERID | 105 |
| VALUE | 105 |
| VERIFY | 106 |
| WORD | 106 |
| WORDINDEX | 107 |
| WORDLENGTH | 107 |
| WORDPOS | 107 |
| WORDS | 108 |
| XRANGE | 108 |
| X2C | 108 |
| X2D | 109 |
| TSO/E Functions | 110 |
| LISTDSI | 110 |
| Specifying Data Set Names | 112 |
| Variables Set by LISTDSI | 113 |
| Messages | 115 |

| | |
|--|------------|
| Function Codes | 115 |
| Reason Codes | 116 |
| Error Codes | 117 |
| Examples | 117 |
| MSG | 118 |
| Example | 119 |
| OUTTRAP | 119 |
| Additional Variables Available | 121 |
| Examples | 122 |
| PROMPT | 123 |
| Interaction of Three Ways to Affect Prompting | 124 |
| Examples | 125 |
| STORAGE | 126 |
| Examples | 126 |
| SYSDSN | 127 |
| Examples | 128 |
| SYSVAR | 128 |
| Control Variables Not Supported by SYSVAR | 130 |
| Examples | 130 |
| Chapter 5. Parsing for PARSE, ARG, and PULL | 131 |
| Introduction | 131 |
| Parsing Words | 131 |
| Parsing Using String Patterns | 132 |
| Parsing Using Numeric Patterns | 132 |
| Parsing Arguments | 133 |
| Definition | 133 |
| Parsing with Literal Patterns | 134 |
| Parsing with Variable Patterns | 135 |
| Use of the Period as a Placeholder | 136 |
| Parsing with Positional Patterns and Relative Patterns | 136 |
| Parsing Multiple Strings | 138 |
| Chapter 6. Numerics and Arithmetic | 139 |
| Introduction | 139 |
| Definition | 140 |
| Numbers | 140 |
| Precision | 140 |
| Arithmetic Operators | 141 |
| Arithmetic Operation Rules — Basic Operators | 141 |
| Addition and Subtraction | 142 |
| Multiplication | 142 |
| Division | 142 |
| Arithmetic Operators — Additional Operators | 143 |
| Power | 143 |
| Integer Division | 144 |
| Remainder | 144 |
| Comparison Operators | 145 |
| Exponential Notation | 146 |
| Numeric Information | 147 |
| Whole Numbers | 147 |
| Numbers Used Directly by REXX | 147 |
| Errors | 148 |
| Chapter 7. Conditions and Condition Traps | 149 |
| Action Taken When a Condition is Trapped | 150 |

| | |
|---|-----|
| Condition Information | 152 |
| Chapter 8. Using REXX in Different Address Spaces | 155 |
| Additional TSO/E REXX Support | 155 |
| TSO/E REXX Programming Services | 155 |
| TSO/E REXX Customizing Services | 156 |
| Writing Execs That Execute in Non-TSO/E Address Spaces | 157 |
| Executing an Exec in a Non-TSO/E Address Space | 158 |
| Writing Execs That Execute in the TSO/E Address Space | 159 |
| Executing an Exec in the TSO/E Address Space | 161 |
| | |
| Chapter 9. Reserved Keywords, Special Variables, and Command Names | 163 |
| Reserved Keywords | 163 |
| Special Variables | 164 |
| Reserved Command Names | 165 |
| | |
| Chapter 10. TSO/E REXX Commands | 167 |
| DELSTACK | 168 |
| DROPBUF | 169 |
| EXECIO | 171 |
| EXECUTIL | 178 |
| HI | 185 |
| HT | 186 |
| Immediate Commands | 187 |
| MAKEBUF | 188 |
| NEWSTACK | 190 |
| QBUF | 192 |
| QELEM | 194 |
| QSTACK | 196 |
| RT | 198 |
| SUBCOM | 199 |
| TE | 201 |
| TS | 202 |
| | |
| Chapter 11. Debug Aids | 203 |
| Interactive Debugging of Programs | 203 |
| Interrupting Execution and Controlling Tracing | 206 |
| | |
| Chapter 12. TSO/E REXX Programming Services | 209 |
| General Considerations for Calling TSO/E REXX Routines | 212 |
| IRXJCL and IRXEXEC Routines | 214 |
| The IRXJCL Routine | 214 |
| Using IRXJCL to Execute a REXX Exec in MVS Batch | 214 |
| Invoking IRXJCL From a REXX Exec or a Program | 215 |
| Return Codes | 217 |
| The IRXEXEC Routine | 217 |
| Entry Specifications | 218 |
| Parameters | 218 |
| The Exec Block (EXECBLK) | 220 |
| Format of Argument List | 222 |
| The In-Storage Control Block (INSTBLK) | 222 |
| The Evaluation Block (EVALBLOCK) | 225 |
| Return Specifications | 227 |
| Return Codes | 227 |
| Function Packages | 229 |
| Interface for Writing Function and Subroutine Code | 231 |

| | |
|--|-------|
| Entry Specifications | 231 |
| Parameters | 231 |
| Argument List | 232 |
| Evaluation Block | 232 |
| Directory for Function Packages | 234 |
| Format of Entries in the Directory | 235 |
| Example of a Function Package Directory | 236 |
| Specifying Directory Names in the Function Package Table | 238 |
| Variable Access (IRXEXCOM) | 240 |
| Entry Specifications | 241 |
| Parameters | 241 |
| The Shared Variable (Request) Block - SHVBLOCK | 241 |
| Function Codes (SHVCODE) | 243 |
| Return Specifications | 245 |
| Return Codes | 246 |
| Maintain Entries in the Host Command Environment Table (IRXSUBCM) | 247 |
| Entry Specifications | 248 |
| Parameters | 248 |
| Functions | 248 |
| Format of a Host Command Environment Table Entry | 249 |
| Return Specifications | 249 |
| Return Codes | 250 |
| Trace and Execution Control Routine (IRXIC) | 251 |
| Entry Specifications | 251 |
| Parameters | 251 |
| Return Specifications | 252 |
| Return Codes | 252 |
| The IRXRLT (Get Result) Routine | 253 |
| Entry Specifications | 253 |
| Parameters | 254 |
| Return Specifications | 256 |
| Return Codes | 256 |
| Chapter 13. TSO/E REXX Customizing Services | 259 |
| Flow of REXX Exec Processing | 260 |
| Initialization and Termination of a Language Processor Environment | 260 |
| Types Of Language Processor Environments | 263 |
| Loading and Freeing a REXX Exec | 263 |
| Processing of the REXX Exec | 263 |
| Overview of Replaceable Routines | 264 |
| Exit Routines | 265 |
| Chapter 14. Language Processor Environments | 267 |
| Overview of Language Processor Environments | 268 |
| Using the Environment Block | 271 |
| When Environments are Automatically Initialized in TSO/E | 272 |
| Initializing Environments for User-Written TMPs | 272.1 |
| When Environments are Automatically Initialized in MVS | 273 |
| Types of Environments - Integrated and Not Integrated Into TSO/E | 274 |
| Characteristics of a Language Processor Environment | 275 |
| Flags and Corresponding Masks | 281 |
| Module Name Table | 286 |
| Host Command Environment Table | 291 |
| Function Package Table | 295 |
| Values Provided in the Three Default Parameters Modules | 299 |
| How IRXINIT Determines What Values to Use for the Environment | 302 |

| | |
|---|------------|
| Values IRXINIT Uses to Initialize Environments | 302 |
| Chains of Environments and How Environments Are Located | 304 |
| Locating a Language Processor Environment | 307 |
| Changing the Default Values for Initializing an Environment | 310 |
| Providing Your Own Parameters Modules | 311 |
| Changing Values for ISPF | 311 |
| Changing Values for TSO/E | 311 |
| Changing Values for TSO/E and ISPF | 312 |
| Changing Values for Non-TSO/E | 313 |
| Considerations for Providing Parameters Modules | 314 |
| Specifying Values for Different Environments | 315 |
| Parameters You Cannot Change | 315 |
| Parameters You Can Use in Any Language Processor Environment | 315 |
| Parameters You Can Use for Environments That Are Integrated Into TSO/E | 318 |
| Parameters You Can Use in Environments That Are Not Integrated Into TSO/E | 318 |
| Flag Settings for Environments Initialized for TSO/E and ISPF | 320 |
| Using SYSPROC and SYSEXEC for REXX Execs | 321 |
| Control Blocks Created for a Language Processor Environment | 323 |
| Format of the Environment Block (ENVBLOCK) | 323 |
| Format of the Parameter Block (PARMBLOCK) | 324 |
| Format of the Work Block Extension | 326 |
| Format of the REXX Vector of External Entry Points | 328 |
| Changing the Maximum Number of Environments in an Address Space | 332 |
| Using the Data Stack in Different Environments | 334 |
| Chapter 15. Initialization and Termination Routines | 339 |
| Initialization Routine - IRXINIT | 340 |
| Entry Specifications | 340 |
| Parameters | 341 |
| How IRXINIT Determines What Values to Use for the Environment | 342 |
| Parameters Module and In-Storage Parameter List | 343 |
| Specifying Values for the New Environment | 345 |
| Return Specifications | 346 |
| Output Parameters | 347 |
| Return Codes | 350 |
| Termination Routine - IRXTERM | 352 |
| Entry Specifications | 353 |
| Parameters | 353 |
| Return Specifications | 353 |
| Return Codes | 354 |
| Chapter 16. Replaceable Routines and Exits | 355 |
| Replaceable Routines | 356 |
| General Considerations | 356 |
| Installing Replaceable Routines | 357 |
| Exec Load Routine | 358 |
| Entry Specifications | 359 |
| Parameters | 359 |
| Format of the Exec Block | 361 |
| Format of the In-Storage Control Block | 363 |
| Return Specifications | 365 |
| Return Codes | 365 |
| Input/Output Routine | 366 |
| Entry Specifications | 367 |

| | |
|---|------------|
| Parameters | 367 |
| Functions Supported for the I/O Routine | 368 |
| Buffer and Buffer Length Parameters | 370 |
| Line Number Parameter | 372 |
| Data Set Information Block | 372 |
| Return Specifications | 375 |
| Return Codes | 375 |
| Host Command Environment Routine | 377 |
| Entry Specifications | 377 |
| Parameters | 377 |
| Error Recovery | 379 |
| Return Specifications | 379 |
| Return Codes | 380 |
| Data Stack Routine | 381 |
| Entry Specifications | 382 |
| Parameters | 382 |
| Functions Supported for the Data Stack Routine | 383 |
| Return Specifications | 385 |
| Return Codes | 385 |
| Storage Management Routine | 386 |
| Entry Specifications | 386 |
| Parameters | 386 |
| Return Specifications | 388 |
| Return Codes | 388 |
| User ID Routine | 389 |
| Entry Specifications | 389 |
| Parameters | 389 |
| Return Specifications | 390 |
| Return Codes | 390 |
| Message Identifier Routine | 391 |
| Entry Specifications | 391 |
| Parameters | 391 |
| Return Specifications | 391 |
| Return Codes | 391 |
| REXX Exit Routines | 392 |
| Exits for Language Processor Environment Initialization and Termination | 392 |
| Exec Initialization and Termination Exits | 393 |
| IRXEXEC Exit Routine | 393 |
| Attention Handling Exit Routine | 394 |
| Appendix A. Error Numbers and Messages | 395 |
| Appendix B. Double Byte Character Set (DBCS) | 405 |
| General Description | 405 |
| DBCS Enabling Data | 406 |
| Mixed String Validation | 406 |
| Instruction Examples | 407 |
| PARSE | 407 |
| PUSH and QUEUE | 408 |
| SAY and TRACE | 408 |
| DBCS Function Handling | 408 |
| Built-in Function Examples | 410 |
| ABBREV | 410 |
| COMPARE | 410 |
| COPIES | 410 |
| DATATYPE | 411 |

| | |
|--|------------|
| FIND | 411 |
| INDEX, POS, and LASTPOS | 411 |
| INSERT and OVERLAY | 411 |
| JUSTIFY | 411 |
| LEFT, RIGHT, and CENTER | 412 |
| LENGTH | 412 |
| REVERSE | 412 |
| SPACE | 412 |
| STRIP | 412 |
| SUBSTR and DELSTR | 412 |
| SUBWORD and DELWORD | 413 |
| TRANSLATE | 413 |
| VERIFY | 413 |
| WORD, WORDINDEX, and WORDLENGTH | 413 |
| WORDS | 413 |
| WORDPOS | 414 |
| External Functions | 414 |
| Counting Option | 414 |
| Function Descriptions | 414 |
| DBADJUST | 414 |
| DBBRACKET | 415 |
| DBCENTER | 415 |
| DBCJUSTIFY | 416 |
| DBLEFT | 416 |
| DBRIGHT | 417 |
| DBRLEFT | 417 |
| DBRRIGHT | 418 |
| DBTODBCS | 418 |
| DBTOSBCS | 419 |
| DBUNBRACKET | 419 |
| DBVALIDATE | 419 |
| DBWIDTH | 420 |
| Appendix C. IRXTERMA and RXSECT | 421 |
| RXSECT Environment Control Macro | 421 |
| IRXTERMA Routine | 422 |
| Parameters | 423 |
| Return Specifications | 423 |
| Return Codes | 424 |
| Summary of Changes | 425 |
| Bibliography | 427 |
| Related Publications | 427 |
| Index | 431 |



PARSE NUMERIC

The current numeric controls (as set by the NUMERIC instruction, see page 47) are made available. These controls are in the order DIGITS FUZZ FORM.

Example:

After: Parse Numeric Var1
Var1 would be equal to: 9 0 SCIENTIFIC

See Numeric instruction on page 47. Also refer to the built-in functions DIGITS, FORM, and FUZZ found on pages 87, 90, 91, respectively.

PARSE PULL

The next string from the queue is parsed. If the queue is empty, lines will be read from the default input (typically the user's terminal). Data can be added to the head or tail of the queue by using the PUSH and QUEUE instructions respectively. The number of lines currently in the queue can be found by using the QUEUED built-in function, described on page 97. The queue will remain active as long as the language processor is active. The queue can be altered by other programs in the system and can be used as a means of communication between these programs and programs written in REXX.

Note: PULL and PARSE PULL read from the data stack. If that is empty, they read from the terminal (TSO/E address space) or from the data set that represents the input stream (non-TSO/E address space). See the PULL instruction on page 55 for further details.

PARSE SOURCE

The data parsed describes the source of the program being executed.

The source string contains the following tokens:

1. The characters TSO
2. The string COMMAND, FUNCTION, or SUBROUTINE depending on whether the program was invoked as some kind of host command (for example, as an exec from TSO/E READY mode), or from a function call in an expression, or via the CALL instruction.
3. Name of the exec in uppercase. If the name is not known, this token is a question mark (?).
4. Name of the DD from which the exec was loaded. If the name is not known, this token is a question mark (?).
5. Name of the data set from which the exec was loaded. If the name is not known, this token is a question mark (?).
6. Name of the exec as it was invoked, that is, the name is not folded to uppercase. If the name is not known, this token is a question mark (?).
7. Initial (default) host command environment in uppercase. For example, this token may be TSO, MVS, or ISPEXEC.
8. Name of the address space in uppercase. For example, the value may be MVS (non-TSO/E) or TSO/E or ISPF. If the exec was invoked from ISPF, the address space name is ISPF.

The value is taken from the parameter block (see page 280). Note that the initialization exit routines may change the name specified in the parameters module. If the name of the address space is not known, this token is a question mark (?).

9. Eight character user token. This is the token that is specified in the PARSETOK field in the parameters module (see page 277).

For example, the string parsed might look like one of the following:

TSO COMMAND PROGA SYSXR07 EGGERS.ECE.EXEC ? TSO TSO/E ?

TSO SUBROUTINE PROGSUB SYSEXEC ? ? TSO ISPF ?

PARSE VALUE

expression is evaluated, and the result is the data that is parsed. Note that WITH is a subkeyword in this context and so cannot be used as a symbol within *expression*.

Thus, for example:

PARSE VALUE time() WITH hours ':' mins ':' secs

will get the current time and split it up into its constituent parts.

PARSE VAR *name*

The value of the variable specified by *name* is parsed. *name* must be a symbol that is valid as a variable name (that is, it can not start with a period or a digit). Note that the variable name may be included in the template, so that for example:

PARSE VAR string word1 string

will remove the first word from *string* and put it in the variable *word1*, and

PARSE UPPER VAR string word1 string

will also translate the data from *string* to uppercase before it is parsed.

PARSE VERSION

Information describing the language level and the date of the language processor is parsed. This consists of five words:

- A word describing the language, which is the string "REXX370"
- The language level description, for example, "3.45" or "3.46"
- Three tokens describing the language processor release date in the format as the default for the DATE() function (see page 85), for example, "20 Oct 1987" or "30 Jun 1988".

The values returned for the language level description and the language processor release date depend on whether or not your installation has installed the PTF for APAR OY17590. If the PTF is installed, the values returned are "3.46" and "30 Jun 1988". If the PTF is not installed, the values returned are "3.45" and "20 Oct 1987".

Note: PARSE VERSION information should be parsed on a word basis rather than on an absolute column position.

Maintain Entries in the Host Command Environment Table (IRXSUBCM)

Use the IRXSUBCM routine to maintain entries in the host command environment table. The table contains the names of the valid host command environments that REXX execs can use to execute host commands. In an exec, you can use the ADDRESS instruction to direct a host command to a specific environment for execution. The host command environment table also contains the name of the routine that is invoked to handle the execution of commands for each specific environment. "Host Command Environment Table" on page 291 describes the table in more detail.

Note: To permit FORTRAN programs to call IRXSUBCM, TSO/E provides an alternate entry point for the IRXSUBCM routine. The alternate entry point name is IRXSUB.

Using IRXSUBCM, you can add, delete, update, or query entries in the table. You can also use IRXSUBCM to dynamically update the host command environment table while a REXX exec is executing.

A program can access IRXSUBCM using either the CALL or LINK macro instructions, specifying IRXSUBCM as the entry point name. You can obtain the address of the IRXSUBCM routine from the REXX vector of external entry points. "Format of the REXX Vector of External Entry Points" on page 328 describes the vector.

If a program uses IRXSUBCM, it must create a parameter list and pass the address of the parameter list in register 1.

IRXSUBCM changes or queries the host command environment table for the current language processor environment, that is, for the environment in which it executes (see "General Considerations for Calling TSO/E REXX Routines" on page 212 for information). IRXSUBCM affects only the environment in which it executes. Changes to the table take effect immediately and remain in effect until the language processor environment is terminated.

Environment Customization Considerations

If you use the initialization routine to initialize environments, on the call to IRXSUBCM, you can optionally pass the address of an environment block in register 0. If the environment block is valid, IRXSUBCM will execute in the environment represented by that environment block. If register 0 does not point to a valid environment block, IRXSUBCM will locate the current environment.

If the environment in which IRXSUBCM executes is part of a chain of environments and you use IRXSUBCM to change the host command environment table, the following applies:

- The changes do not affect the environments that are higher in the chain or existing environments that are lower in the chain.
- The changes are propagated to any language processor environment that is created on the chain after IRXSUBCM updates the table.

Entry Specifications

For the IRXSUBCM routine, the contents of the registers on entry are:

| | |
|-----------------------|--|
| Register 0 | Address of an environment block (optional) |
| Register 1 | Address of the parameter list passed by the caller |
| Registers 2-12 | Unpredictable |
| Register 13 | Address of a register save area |
| Register 14 | Return address |
| Register 15 | Entry point address |

Parameters

In register 1, you pass the address of a parameter list, which consists of a list of addresses. Each address in the parameter list points to a parameter. You must pass all parameters on the call. The high order bit of the last address in the parameter list must be set to 1. Figure 29 describes the parameters for IRXSUBCM.

| Parameter | Number of Bytes | Description |
|-------------|-----------------|---|
| Parameter 1 | 8 | The function to be performed. The name of the function must be left justified and padded to the right with blanks. The valid functions are: <ul style="list-style-type: none"> • ADD • DELETE • UPDATE • QUERY Each function is described after the table in "Functions." |
| Parameter 2 | 4 | The address of a string. On both input and output, the string has the same format as an entry in the host command environment table. "Format of a Host Command Environment Table Entry" on page 249 describes the entry in more detail. |
| Parameter 3 | 4 | The length of the string (entry) that is pointed to by parameter 2. |
| Parameter 4 | 8 | The name of the host command environment. The name must be left justified and padded to the right with blanks. |

Functions

Parameter 1 contains the name of the function IRXSUBCM is to perform. The functions are:

ADD

Adds an entry to the table using the values specified on the call. IRXSUBCM does not check for duplicate entries. If a duplicate entry is added and then IRXSUBCM is called to delete the entry, IRXSUBCM will delete the duplicate entry and leave the original one.

Using the Environment Block

The main control block that is created for a language processor environment is the environment block. The environment block represents the language processor environment and points to other control blocks that contain information about the environment.

The environment block is known as the *anchor* that is used by all callable interfaces to REXX. All REXX routines, except for the IRXINIT initialization routine, cannot execute unless an environment block exists, that is, a language processor environment must exist. When IRXINIT initializes a new language processor environment, it always returns the address of the environment block in register 0. (If you explicitly call IRXINIT, it also returns the address of the environment block in the parameter list.) You can also use the IRXINIT routine to obtain the address of the environment block for the current non-reentrant environment (see page 340). IRXINIT returns the address in register 0 and also in a parameter in the parameter list.

The address of the environment block is useful for calling a REXX routine or for obtaining information from the control blocks that were created for the environment. If you call any of the REXX routines (for example, IRXEXEC to execute an exec or the variable access routine IRXEXCOM), you can optionally pass the address of an environment block to the routine in register 0. By passing the address of an environment block, you can specify in which specific environment you want either the exec or the service to execute. This is particularly useful if you use the IRXINIT routine to initialize several environments on a chain and then want to execute a REXX routine in a specific environment. When you call the routine, you can pass the address of the environment block in register 0.

If you call a REXX routine and do not pass the address of an environment block in register 0, the routine will execute:

- In the last environment on the chain under the current task (non-TSO/E address space)
- In the last environment on the chain under the current task or a parent task (TSO/E address space).

If you call IRXEXEC or IRXJCL and an environment does not exist, IRXINIT is invoked to initialize an environment in which the exec will execute. When the exec completes processing, the newly created environment is terminated.

The environment block points to several other control blocks that contain the parameters used to define the environment and the addresses of REXX routines, such as IRXINIT, IRXEXEC, and IRXTERM, and replaceable routines. You can access these control blocks to obtain this information. The control blocks are described in "Control Blocks Created for a Language Processor Environment" on page 323.

Note About Changing Any Control Blocks

You can obtain information from the control blocks. However, you **must not change** any of the control blocks. If you do, unpredictable results may occur.

When Environments are Automatically Initialized in TSO/E

The initialization routine IRXINIT initializes a language processor environment. The system calls IRXINIT to automatically initialize a default environment when a user logs on to TSO/E and when ISPF is invoked.

When a user logs on to TSO/E, IRXINIT is called as part of the logon process to automatically initialize a language processor environment for the TSO/E session. The initialization of a language processor environment is transparent to the user. After users log on to TSO/E, they can simply invoke a REXX exec without performing any other tasks.

Note: If your installation uses a user-written terminal monitor program (TMP) instead of the TMP provided by TSO/E, a language processor environment is not automatically initialized. See "Initializing Environments for User-Written TMPs" on page 272.1 for information about the tasks you must perform to initialize a language processor environment in order to execute REXX execs.

Similarly, when a user invokes ISPF from TSO/E, the IRXINIT routine is called and automatically initializes a language processor environment for ISPF, that is, for the ISPF screen. The second language processor environment is separate from the environment that was initialized for the TSO/E session. If the user enters split screen in ISPF, IRXINIT initializes a third language processor environment for the second ISPF screen. At this point, three separate language processor environments exist. If the user executes a REXX exec from the second ISPF screen, the exec executes under the third language processor environment, that is, the environment IRXINIT initialized for the second ISPF screen. If the user executes the exec from the first ISPF screen, it runs under the second language processor environment.

The termination routine, IRXTERM, terminates a language processor environment. Continuing the above example, when the user returns to one screen in ISPF, the IRXTERM routine is called. IRXTERM terminates the third language processor environment that was initialized for the second ISPF screen. Similarly, when the user exits from ISPF and returns to TSO/E READY mode, IRXTERM terminates the language processor environment for the first ISPF screen. In TSO/E READY mode, the first language processor environment still exists. At this point, if the user executes a REXX exec from READY mode, the exec executes under the environment that was initialized at TSO/E logon. When the user logs off, IRXTERM terminates the language processor environment for the TSO/E session.

To summarize, the IRXINIT routine automatically initializes a language processor environment when a user logs on to TSO/E and whenever an ISPF screen is initialized. Each environment that is initialized is separate from another environment. The IRXTERM routine automatically terminates the language processor environment for an ISPF screen when the screen session ends and terminates the environment created at TSO/E logon when the user logs off.

You can also call the IRXINIT routine to initialize a language processor environment. On the call to IRXINIT, you specify values you want defined for the new environment. Using IRXINIT gives you the ability to define a language processor environment and *customize* how REXX execs execute and how system services are accessed and used. This is particularly important in non-TSO/E address spaces where you may want to provide replaceable routines to handle system services.

However, you may want to use IRXINIT in TSO/E in order to create an environment that is similar to a non-TSO/E address space to test any replaceable routines or REXX execs you have developed for non-TSO/E.

If you explicitly call IRXINIT to initialize a language processor environment, you must call the IRXTERM routine to terminate the environment. The system does not terminate language processor environments that you initialized by calling IRXINIT. Information about IRXINIT and IRXTERM is described later in this chapter. Chapter 15, "Initialization and Termination Routines" provides reference information about the parameters and return codes for IRXINIT and IRXTERM.

Initializing Environments for User-Written TMPs

If your installation uses a user-written terminal monitor program (TMP) instead of the TMP provided by TSO/E, a language processor environment is not automatically initialized in the TSO/E address space when a user logs on to TSO/E. That is, a language processor environment is not initialized for TSO/E READY mode. A language processor environment is required for executing REXX execs. To allow users to execute REXX execs from TSO/E READY mode, your user-written TMP must invoke the initialization routine IRXINIT to initialize a language processor environment. To initialize the environment, the TMP must do the following:

- Invoke the initialization routine IRXINIT to initialize a language processor environment. The environment must be integrated into TSO/E, that is, the TSOFL flag must be on. On the call to IRXINIT, you can provide parameters that are equivalent to the default values that TSO/E provides in the IRXTSPRM default parameters module.
- The initialization routine IRXINIT returns the address of the environment block for the new language processor environment in register 0. You must store the address of the environment block in the ECTENVBK field of the environment control block (ECT).
- You must ensure that the ECTEXTPR field in the ECT is set to zeroes.
- When all user-written TMP processing is completed, you must invoke the termination routine IRXTERM to terminate the language processor environment that was initialized by IRXINIT. The system does not automatically terminate the environment.

The following topics in this chapter describe the characteristics of a language processor environment, the different types of environments, and the default parameters modules that TSO/E provides. Chapter 15, "Initialization and Termination Routines" describes the initialization and termination routines IRXINIT and IRXTERM.



Summary of Changes

**Summary of Changes
for SC28-1883-0
as Updated February 10, 1989
by Technical Newsletter SN28-1293**

This Technical Newsletter, which supports TSO Extensions (TSO/E) Version 2, contains the following changes for TSO/E support of the REXX programming language. The newsletter also contains minor technical changes.

- New information about how to initialize a language processor environment if you use a user-written terminal monitor program (TMP)
- New values returned by the PARSE VERSION instruction for the language level description (3.46) and the language processor release date (30 Jun 1988). The new values support APAR OY17590 and are returned if you install the PTF that supports the APAR. If the PTF is not installed, the values returned are "3.45" and "20 Oct 1987."

**Summary of Changes
for SC28-1883-0
TSO Extensions Version 2**

This book is a new book in the TSO/E Version 2 library. It contains reference information about TSO/E REXX.

APAR Information

The following APARs provide TSO/E REXX instructions, functions, and services that are described in this book. The instructions, functions, and services listed below can be used only if your installation installs the PTF that supports the particular APAR.

- APAR OY17498 provides the TSO/E function MSG, which is described on page 118.
- APAR OY17590 provides the:
 - Ability to enable and disable condition traps using the CALL instruction (CALL ON and CALL OFF). The CALL instruction is described on page 32. Chapter 7, "Conditions and Condition Traps" describes how to enable and disable condition traps.
 - Ability to specify NAME *trapname* using the SIGNAL ON instruction. The SIGNAL instruction is described on page 62. Chapter 7, "Conditions and Condition Traps" describes how to enable and disable condition traps.
 - CONDITION built-in function, which is described on page 82.
 - Ability to specify up to 20 expressions on the CALL instruction and on function calls, such as MAX and MIN. If the PTF for the APAR is not installed, the maximum number of expressions you can specify is 10.
 - Exit routines for exec initialization and exec termination. The exits are described in "REXX Exit Routines" on page 392.
- APAR OY17558 provides the SYS1.SAMPLIB members for coding the parameters modules IRXPARMs, IRXTSPRM, and IRXISPRM. The SAMPLIB members are:
 - TSOREXX1 (for IRXPARMs)
 - TSOREXX2 (for IRXTSPRM)
 - TSOREXX3 (for IRXISPRM)
- APAR OY17979 provides alternate entry point names for the TSO/E REXX external entry points. The alternate entry point names are less than six characters and allow FORTRAN programs to call the TSO/E REXX external entry points.



Index

A

ABBREV function
 description 78
 using to select a default 78
 abbreviations
 looking for one in a string 137
 testing with ABBREV function 78
 abnormal change in flow of control 149
 ABS function 78
 absolute value
 finding using ABS function 78
 used with power 143
 abuttal 13
 accessing REXX variables 240
 active loops 44
 addition
 definition 141
 operator 14
 ADDRESS
 function 78
 instruction 28
 settings saved during subroutine calls 34
 address of environment block, obtaining 340
 address of environment block, passing to REXX routines 213, 271, 307
 address spaces
 executing execs in non-TSO/E 158
 executing execs in TSO/E 161
 name of for language processor environment 280
 using REXX in different 155
 using REXX in non-TSO/E 157
 using REXX in TSO/E 159
 algebraic precedence 16
 allocation information
 about a data set 110
 retrieving with LISTDSI 110
 alphabets
 checking with DATATYPE 84
 used as symbols 10
 alphanumeric checking with DATATYPE 84
 altering
 flow within a repetitive DO loop 44
 REXX variables 22
 alternate entry point names 328
 alternate exec libraries 8
 alternate messages flag 284
 ALLLIB command 8
 ALTMSG flag 284
 AND operator 15
 AND'ing character strings together 80
 AND, logical 15
 ARG function 79
 ARG instruction 30
 ARG option of PARSE instruction 50
 argument list for function package 232

arguments
 checking with ARG function 79
 of functions 30, 71
 of subroutines 30, 32
 passing to functions 71
 retrieving with ARG function 79
 retrieving with ARG instruction 30
 retrieving with the PARSE ARG instruction 50
 arithmetic
 combination rules 141
 comparisons 144
 errors 147
 NUMERIC settings 47
 operators 14, 139, 141
 overflow 147
 precision 140
 underflow 147
 array
 initialization of 20
 setting up 19
 assigning data to variables 50
 assignment
 description of 18
 of compound variables 19, 20
 assignment indicator (=) 18
 associative storage 19
 ATTACH host command environment 25
 attaching programs 25
 ATTNROUT field (module name table) 289
 automatic initialization of language processor environments
 in non-TSO/E address space 273
 in TSO/E address space 272

B

backslash, use of 15
 BASEDATE option of DATE function 85
 BITAND function 80
 BITOR function 80
 bits checked using DATATYPE 84
 BITXOR function 81
 blank removal with STRIP function 100
 blanks
 adjacent to special character 8
 as concatenation operator 13
 boolean operations 15
 bottom of program reached during execution 40
 bracketed DBCS strings
 DBBRACKET function 415
 DBUNBRACKET function 419
 distinguishing from SBCS data 406
 built-in function invoking 32
 built-in functions
 ABBREV 78
 ABS 78
 ADDRESS 78
 ARG 79

built-in functions (*continued*)

BITAND 80
 BITOR 80
 BITXOR 81
 CENTER 81
 CENTRE 81
 COMPARE 82
 CONDITION 82
 COPIES 83
 C2D 83
 C2X 84
 DATATYPE 84
 DATE 85
 DELSTR 87
 DELWORD 87
 description of 72
 DIGITS 87
 D2C 88
 D2X 88
 ERRORTXT 89
 EXTERNALS 89
 FIND 90
 FORM 90
 FORMAT 90
 FUZZ 91
 INDEX 92
 INSERT 92
 JUSTIFY 93
 LASTPOS 93
 LEFT 94
 LENGTH 94
 LINESIZE 94
 MAX 95
 MIN 95
 OVERLAY 96
 POS 96
 QUEUED 97
 RANDOM 97
 REVERSE 98
 RIGHT 98
 SIGN 98
 SOURCELINE 99
 SPACE 99
 STRIP 100
 SUBSTR 100
 SUBWORD 101
 SYMBOL 101
 TIME 102
 TRACE 103
 TRANSLATE 104
 TRUNC 104
 USERID 105
 VALUE 105
 VERIFY 106
 WORD 106
 WORDINDEX 107
 WORDLENGTH 107
 WORDPOS 107

built-in functions (*continued*)

WORDS 108
 XRANGE 108
 X2C 108
 X2D 109
 BY phrase of DO instruction 35

C

CALL instruction 32
 calling REXX routines, general considerations 212
 CENTER function 81
 centering a string using CENTER function 81
 centering a string using CENTRE function 81
 CENTRE function 81
 CENTURY option of DATE function 85
 chains of environments 269, 304
 changing defaults for initializing language processor environments 310
 changing destination of commands 28
 changing maximum number of language processor environments 332
 changing value in specific storage address 126
 character position of a string 93
 character position using INDEX 92
 character removal with STRIP function 100
 character to decimal conversion 83
 character to hexadecimal conversion 84
 characteristics of language processor environment 259, 275
 check existence of a data set 127
 clause
 as labels 17
 assignment 18
 continuation of 12
 description of 8
 null 17
 close data set flag 283
 CLOSEXFL flag 283
 CMDSOFL flag 281
 collating sequence using XRANGE 108
 colon
 as a special character 11
 in a label 17
 colon as label terminators 17
 combination, arithmetic 141
 comma
 as continuation character 12
 in CALL instruction 33
 in function calls 71
 separator of arguments 33, 71
 within a parsing template 30, 132, 133, 138
 command errors, trapping 149
 command inhibition
 See TRACE instruction
 command processor parameter list
 See CPPL
 command search order flag 281
 commands
 alternative destinations 22
 destination of 28
 host, definition of 23
 inhibiting with TRACE instruction 66

- commands (*continued*)
 - issuing to host 22
 - obtaining name of last command executed 128
 - reserved names 165
 - set prompting on/off 123
 - trap lines of output 119
 - TSO/E REXX 167
 - comments
 - description of 9
 - REXX exec identifier 8
 - COMPARE function 82
 - comparisons
 - of numbers 14, 144
 - of strings 14
 - using COMPARE 82
 - compound symbols 19
 - compound variable
 - description of 19
 - setting new value 20
 - concatenation of strings 13
 - concatenation operator
 - abuttal 13
 - blank 13
 - || 13
 - CONDITION function 82
 - condition trap info using CONDITION 82
 - conditional loops 35
 - conditions
 - ERROR 149
 - FAILURE 149
 - HALT 149
 - NOVALUE 149
 - saved during subroutine calls 34
 - SYNTAX 149
 - conditions, trapping of 149
 - considerations for calling REXX routines 212
 - console
 - See* terminals
 - constant symbols 19
 - content addressable storage 19
 - continuation
 - character 12
 - of clauses 12
 - of data for display 59
 - control blocks
 - environment block (ENVBLOCK) 271, 323
 - evaluation (EVALBLOCK) 225, 232
 - exec block (EXECBLK) 220
 - for language processor environment 270, 323
 - in-storage (INSTBLK) 222
 - parameter block (PARMBLOCK) 275, 325
 - request (SHVBLOCK) 242
 - return result from exec 225
 - shared variable (SHVVBLOCK) 242
 - SHVBLOCK 242
 - vector of external entry points 328
 - work block extension 326
 - control variable 36
 - controlled loops 36
 - controlling display of TSO/E messages 118, 119
 - controlling prompting from interactive commands 123
 - controlling search order for REXX execs 284
 - conversion
 - character to decimal 83
 - character to hexadecimal 84
 - decimal to character 88
 - decimal to hexadecimal 88
 - formatting numbers 90
 - hexadecimal to character 108
 - hexadecimal to decimal 109
 - conversion functions 77-109
 - COPIES function 83
 - copying a string using COPIES 83
 - copying information to and from data sets 171
 - counting words in a string 108
 - CPPL
 - in work block extension 327
 - passing on call to IRXEXEC 220
 - creating
 - buffer on the data stack 188
 - new data stack 190, 337
 - non-reentrant environment 340
 - reentrant environment 340
 - current non-reentrant environment, locating 340
 - current terminal line width 94
 - customizing services
 - description 259
 - environment characteristics 259
 - exit routines 259
 - general considerations for calling routines 212
 - language processor environments 267
 - replaceable routines 259, 264, 265
 - summary of 156
 - customizing TSO/E REXX
 - See* customizing services
 - C2D function 83
 - C2X function 84
- ## D
- Data Facility Hierarchical Storage Manager (DFHSM), status of 128
 - data length 13
 - data set
 - check existence of 127
 - copying information to and from 171
 - obtain allocation, protection, directory information 110
 - data stack
 - counting lines in 97
 - creating 190, 337
 - creating a buffer 188
 - deleting 168
 - DELSTACK command 168
 - discarding a buffer 169
 - DROPBUF command 169
 - dropping a buffer 169
 - MAKEBUF command 188
 - NEWSTACK command 190, 337
 - number of buffers 192

data stack (*continued*)

- number of elements on 194
 - primary 337
 - QBUF command 192
 - QUELEM command 194
 - QSTACK command 196
 - querying number of elements on 194
 - querying the number of 196
 - querying the number of buffers 192
 - reading from with PULL 55
 - replaceable routine 380
 - secondary 337
 - sharing between environments 334
 - use in different environments 334
 - writing to with PUSH 56
 - writing to with QUEUE 57
- data stack flag 281
 - data terms 13
 - DATATYPE function 84
 - date and version of the language processor 52
 - DATE function 85
 - DBADJUST function 414
 - DBBRACKET function 415
 - DBCENTER function 415
 - DBCJUSTIFY function 416
 - DBCS functions
 - DBADJUST 414
 - DBBRACKET 415
 - DBCENTER 415
 - DBCJUSTIFY 416
 - DBLEFT 416
 - DBRIGHT 417
 - DBRLEFT 417
 - DBRRIGHT 418
 - DBTODBCS 418
 - DBTOSBCS 419
 - DBUNBRACKET 419
 - DBVALIDATE 419
 - DBWIDTH 420
 - DBCS handling 405
 - DBCS strings 49, 405
 - DBCS (Double-Byte Character Set) characters 405
 - DBLEFT function 416
 - DBRIGHT function 417
 - DBRLEFT function 417
 - DBRRIGHT function 418
 - DBTODBCS function 418
 - DBTOSBCS function 419
 - DBUNBRACKET function 419
 - DBVALIDATE function 419
 - DBWIDTH function 420
 - DD from which execs are loaded 287
 - debugging programs
 - See* interactive debug
 - See* TRACE instruction
 - debug, interactive 64, 203
 - decimal arithmetic 139–148
 - decimal to character conversion 88
 - decimal to hexadecimal conversion 88
 - default environment 22
 - See also* language processor environment
 - defaults for initializing language processor environments 299
 - defaults provided for parameters modules 299
 - deleting a data stack 168
 - deleting part of a string 87
 - deleting words from a string 87
 - delimiters in a clause
 - See* colon
 - See* semicolons
 - DELSTACK command 168
 - DELSTR function 87
 - DELWORD function 87
 - derived name 19
 - derived names of variables 19
 - DFHSM, status of 128
 - DIGITS function 87
 - DIGITS option of NUMERIC instruction 47, 140
 - direct interface to variables (IRXEXCOM) 240
 - directory names, function packages
 - IRXFLOC 230, 234
 - IRXFUSER 230, 234
 - directory, function package 234
 - example of 236
 - format 234
 - format of entries 235
 - specifying in function package table 238
 - discarding a buffer on the data stack 169
 - displaying data
 - See* SAY instruction
 - displaying message IDs 390
 - division
 - definition 141
 - operator 14
 - DO instruction 35–38
 - See also* loops
 - Double-Byte Character Set (DBCS) strings 49, 405
 - DROP instruction 39
 - DROPBUF command 169
 - dropping a buffer on the data stack 169
 - dummy instruction
 - See* NOP instruction
 - D2C function 88
 - D2X function 88

E

- EFPL (external function parameter list) 231
- elapsed time saved during subroutine calls 34
- elapsed-time calculator 102
- ELSE keyword
 - See* IF instruction
- enabled exec for variable access (IRXEXCOM) 240
- END clause
 - See also* DO instruction
 - See also* SELECT instruction
 - specifying control variable 36
- engineering notation 146
- entry point names 328
- ENVBLOCK
 - See* environment block
- environment block
 - description 271, 307, 323
 - format 323
 - obtaining address of 340

environment block (*continued*)

- overview for calling REXX routines 213
- passing on call to REXX routines 213, 271, 307
- environment table for number of language processor environments 332
- environments
 - See also* host command environment
 - See also* language processor environment
 - addressing of 28
 - default 29, 51
 - determining current using ADDRESS function 78
 - host command 22
 - language processor 260, 267
 - temporary change of 28
- equal operator 14
- equality, testing of 14
- error codes
 - set by LISTDSI 117
 - syntax errors 395
- ERROR condition of SIGNAL and CALL instructions 149
- error messages
 - and codes 395
 - control display of TSO/E messages 118, 119
 - displaying the message ID 390
 - replaceable routine for message ID 390
 - retrieving with ERRORTXT 89
 - syntax errors 395
- errors
 - during execution of functions 76
 - from host commands 22
 - messages 395
 - syntax 395
 - traceback after 68
- errors, trapping 149
- ERRORTXT function 89
- ESTAE, recovery 283
- EUROPEAN option of DATE function 85
- EVALBLOCK
 - See* evaluation block
- evaluation block
 - for function packages 231, 232
 - for IRXEXEC routine 225
 - obtaining a larger one 253
- evaluation of expressions 13
- exception conditions saved during subroutine calls 34
- exclusive OR operator 15
- exclusive ORing character strings together 81
- exec block (EXECBLK) 220
- exec identifier 8
- exec information, obtaining
 - availability of ISPF dialog manager services 128
- exec invocation 128
- last command executed 128
- last subcommand executed 128
- name used to invoke exec 128
- whether exec is running in
 - foreground/background 128
- exec initialization exit 392
- exec libraries
 - defining alternate using ALTLIB 7
 - storing REXX execs 7

- exec load replaceable routine 358
- exec processing routines
 - IRXEXEC 217
 - IRXJCL 214
- exec termination exit 392
- EXECINIT field (module name table) 289
- EXECIO command 171
- execs
 - description of 1
 - executing in MVS batch 158, 214
 - executing in non-TSO/E 158, 214
 - executing in TSO/E 161, 214
 - loading of 358
 - overview of writing 155
 - preloading 358
 - writing for non-TSO/E 157
 - writing for TSO/E 159
- EXECTERM field (module name table) 290
- EXECUTIL command 178
- executing a REXX exec
 - from MVS batch 214
 - in non-TSO/E 158
 - in TSO/E 161
 - using IRXEXEC routine 217
 - using IRXJCL routine 214
- execution by language processor 8
- execution of data 42
- EXIT instruction 40
- exit routines 265, 391
 - attention handling 393
 - exec initialization 392
 - exec termination 392
 - for exec processing 392
 - for IRXEXEC 392
 - IRXINITX 391
 - IRXITMV 391
 - IRXITTS 391
 - IRXTERMX 391
 - language processor environment initialization 391
 - language processor environment termination 391
- exponential notation
 - definition 146
 - description of 139
 - usage 10
- exponentiation
 - definition 143
 - operator 14
- EXPOSE option of PROCEDURE instruction 53
- expressions
 - evaluation 13
 - examples 16
 - parsing of 52
 - results of 13
 - tracing results of 64
- EXROUT field (module name table) 288
- external entry points
 - alternate names 328
 - IRXEXCOM 240
 - IRXEXEC 217
 - IRXIC 251

external entry points (*continued*)

IRXINIT 340
 IRXINOUT 366
 IRXJCL 214
 IRXLOAD 358
 IRXMSGID 390
 IRXRLT 253
 IRXSTK 380
 IRXSUBCM 247
 IRXTERM 352
 IRXUID 388

external function parameter list (EFPL) 231

external functions

description of 72
 LISTDSI 110
 MSG 118
 OUTTRAP 119
 PROMPT 123
 providing in function packages 229
 search order 73
 STORAGE 126
 SYSDSN 127
 SYSVAR 128
 writing 229

EXTERNAL option of PARSE instruction 50

external routine invoking 32

external subroutines

description of 72
 providing in function packages 229
 search order 73
 writing 229

EXTERNALS function 89

extracting a substring 100

extracting words from a string 101

F

FAILURE condition of SIGNAL and CALL instructions 149

FIFO (first-in/first-out) stacking 57

FIND function 90

finding a mismatch using COMPARE 82

finding a string in another string 92, 96

finding the length of a string 94

flags for language processor environment 277, 281

ALTMSGs 284

CLOSEXFL 283

CMDSOFL 281

defaults provided 299

FUNCSOFL 281

LOCPKFL 283

NEWSCFL 283

NEWSTKFL 282

NOESTAE 283

NOLOADDD 284

NOMSGIO 285

NOMSGWTO 285

NOPMSGs 284

NOREADFL 282

NOSTKFL 281

NOWRTFL 282

flags for language processor environment (*continued*)

RENTRANT 284

restrictions on settings 316, 320

SPSHARE 284

STORFL 284

SYSPKFL 283

TSOFL 274, 281

USERPKFL 282

flow control

abnormal, with CALL 149

abnormal, with SIGNAL 149

with CALL/RETURN 32

with DO construct 35

with IF construct 41

with SELECT construct 60

flow of REXX exec processing 260

FOR phrase of DO instruction 35

FOREVER repetitor on DO instruction 35

FORM function 90

FORM option of NUMERIC instruction 47, 146

FORMAT function 90

formatting

DBCS blank adjustments 414

DBCS bracket adding 415

DBCS bracket stripping 419

DBCS DBCS strings to SBCS 419

DBCS EBCDIC to DBCS 418

DBCS string width 420

DBCS text justification 416

numbers for display 90

numbers with TRUNC 104

of output during tracing 67

text centering 81

text justification 93

text left justification 94, 416

text left remainder justification 417

text right justification 98, 415, 417

text right remainder justification 418

text spacing 99

text validation function 419

FORTRAN programs, alternate entry points for external entry points 328

FUNCSOFL flag 281

function codes

set by LISTDSI 115

function package flags 282

function package table 238, 275, 295

defaults provided 299

function packages

add entries in directory 178, 182

change entries in directory 178, 182

description 229

directory 234

directory names 230, 234

IRXFLOC 230, 234

IRXFUSER 230, 234

specifying in function package table 238

system-supplied 230, 234

example of directory 236

external function parameter list 231

function packages (*continued*)

- format of entries in directory 235
- function package table 238
- getting larger area to store result 253
- getting larger evaluation block 253
- interface for writing code 231
- IRXFLOC 230, 234
- IRXFUSER 230, 234
- link editing the code 235
- overview 209
- parameters code receives 231
- rename entries in directory 178, 182
- summary of 156
- system-supplied directory names 230, 234
- types of
 - local 229
 - system 229
 - user 229
- writing 229

function search order flag 281

functions

- built-in 72, 78
- description of 71
- external 72
- forcing built-in or external reference 73
- internal 72
- invocation of 71
- numeric arguments of 147
- providing in function packages 229
- return from 58
- search order 73
- TSO/E external 110
- variables in 53
- writing external 229

function, built-in

See built-in functions

FUZZ

- controlling numeric comparison 145
- option of NUMERIC instruction 47, 145

FUZZ function 91

G

- general considerations for calling REXX routines 212
- get result routine (IRXRLT) 253
- GETFREER field (module name table) 288
- getting a larger evaluation block 253
- GOTO, abnormal 149
- greater than operator 14
- greater than or equal operator 14
- greater than or less than operator (> <) 14
- grouping instructions to execute repetitively 35
- group, DO 35

H

- HALT condition of SIGNAL and CALL instructions 149
- Halt Interpretation (HI) immediate command 185, 203, 251
- Halt Typing (HT) immediate command 186, 251

- halting a looping program 206
 - from a program 251
 - HI immediate command 185
 - using the IRXIC routine 251
 - with EXECUTIL command 178

halt, trapping 149

hexadecimal

See also conversion

checking with DATATYPE 84

hexadecimal digits 9

hexadecimal strings 9

HI (Halt Interpretation) immediate command 185, 206, 251

host command environment

ATTACH 25

change entries in SUBCOMTB table 247

check existence of 199

description 22

IRXSUBCM routine 247

ISPEXEC 24, 160

ISREDIT 24, 160

LINK 25

MVS 24

replaceable routine 376

TSO 24

host command environment table 275, 291

defaults provided 299

host commands 22

definition of 23

TSO/E REXX 167

using in non-TSO/E 157

using in TSO/E 159, 160

hours calculated from midnight 102

HT (Halt Typing) immediate command 186, 251

I

identifier, exec 8

identifier, REXX exec 8

identifying users 87, 90, 91, 105

IDROUT field (module name table) 289

IF instruction 41

IKJCT441 240

immediate commands 187

HI (Halt Interpretation) 185, 206, 251

HT (Halt Typing) 186, 251

issuing from program 251

RT (Resume Typing) 198, 251

TE (Trace End) 201, 206, 251

TS (Trace Start) 202, 206, 251

implied semicolons 12

imprecise numeric comparison 145

in-storage control block (INSTBLK) 222

in-storage parameter list 343

inclusive OR operator 15

INDD field (module name table) 287

indefinite loops 35

See also looping program

indentation during tracing 67

INDEX function 92

indirect evaluation of data 42

- inequality, testing of 14
- infinite loops 35
 - See also* looping program
- inhibition of commands with TRACE instruction 66
- initialization
 - of arrays 20
 - of compound variables 20
 - of language processor environments 269, 340
 - for user-written TMP 272.1
 - in non-TSO/E address space 273
 - in TSO/E address space 272
 - routine (IRXINIT) 272, 340
- initialization routine (IRXINIT)
 - description 340
 - how environment values are determined 302
 - how values are determined 342
 - in-storage parameter list 343
 - output parameters 347
 - overview 272
 - parameters module 343
 - reason codes 347
 - restrictions on values 345
 - specifying values 345
 - to initialize an environment 340
 - to locate an environment 340
 - user-written TMP 272.1
 - values used to initialize environment 302
- input/output
 - replaceable routine 366
 - to and from data sets 171
- INSERT function 92
- inserting a string into another 92
- INSTBLK 222
- instructions
 - ADDRESS 28
 - ARG 30
 - CALL 32
 - DO 35
 - DROP 39
 - EXIT 40
 - IF 41
 - INTERPRET 42
 - ITERATE 44
 - LEAVE 45
 - NOP 46
 - NUMERIC 47
 - OPTIONS 49
 - PARSE 50
 - PROCEDURE 53
 - PULL 55
 - PUSH 56
 - QUEUE 57
 - RETURN 58
 - SAY 59
 - SELECT 60
 - SIGNAL 62
 - TRACE 64
 - UPPER 69
- integer arithmetic 139–148
- integer division
 - definition 143
 - description of 139
 - operator 14
- integrated language processor environments (into TSO/E) 263, 274
- interactive debug 64, 203
 - See also* TRACE instruction
- Interactive System Productivity Facility
 - See* ISPF
- interface for writing functions and subroutines 231
- interface to variables (IRXEXCOM) 240
- internal functions
 - description of 72
 - return from 58
 - variables in 53
- internal routine invoking 32
- INTERPRET instruction 42
- interpretive execution of data 42
- interrupting program execution 181, 185, 206, 251
- invoking
 - built-in functions 32
 - REXX execs 158, 161
 - routines 32
- IOROUT field (module name table) 288
- IRXANCHR 332
- IRXARGTB mapping macro 222, 232
- IRXDSIB mapping macro 366, 371
- IRXEFMVS 230
- IRXEFPCK 230
- IRXEFPL mapping macro 231
- IRXENVB mapping macro 323
- IRXENVT mapping macro 332
- IRXEVALB mapping macro 226, 232
- IRXEXCOM 240
- IRXEXEC
 - argument list 222
 - description 214, 217
 - evaluation block 225
 - exec block 220
 - getting larger area to store result 253
 - getting larger evaluation block 253
 - in-storage control block 222
 - overview 209
 - parameters 218
 - return codes 227
 - returning result from exec 225
- IRXEXECB mapping macro 220, 361
- IRXEXECX field (module name table) 289
- IRXEXTE mapping macro 328
- IRXFLOC 230, 234
- IRXFPDIR mapping macro 234
- IRXFUSER 230, 234
- IRXIC 251
- IRXINIT 272, 340
- IRXINITX 391
- IRXINOUT 366
- IRXINSTB mapping macro 223, 363
- IRXISPRM parameters module 275, 299
- IRXITMV 391

- IRXITTS 391
- IRXJCL
 - description 214
 - invoking 215
 - overview 209
 - parameters 215
 - return codes 217
- IRXLOAD 358
- IRXMODNT mapping macro 286
- IRXMSGID 390
- IRXPACKT mapping macro 295
- IRXPARMB mapping macro 278, 325
- IRXPARMS parameters module 275, 299
- IRXRILT 253
- IRXSHVB mapping macro 242
- IRXSTK 380
- IRXSUBCM 247
- IRXSUBCT mapping macro 249, 291
- IRXTERM 272, 352
- IRXTERMX 391
- IRXTSPRM parameters module 275, 299
- IRXUID 388
- IRXWORKB mapping macro 326
- ISPEXEC host command environment 24
- ISPF
 - determining availability of dialog manager services 128
 - host command environments 24
 - using ISPF services 24, 160
- ISREDIT host command environment 24
- issuing host commands 22
- ITERATE instruction
 - See also* DO instruction
 - description 44
 - use of variable on 44
- I/O
 - replaceable routine 366
 - to and from data sets 171
- J**
- JULIAN option of DATE function 86
- JUSTIFY function 93
- K**
- keyword instructions 27
 - See also* instructions
- keywords
 - conflict with commands 163
 - mixed case 27
 - reservation of 163
- L**
- label
 - as targets of CALL 32
 - as targets of SIGNAL 62
 - description of 17
 - duplicate 62
 - in INTERPRET instruction 42
 - search algorithm 62
- language code for REXX messages 276
- language processor date and version 52
- language processor environment
 - automatic initialization in non-TSO/E 273
 - automatic initialization in TSO/E 272
 - chains of 269, 304
 - changing the defaults for initializing 310
 - characteristics 275
 - considerations for calling REXX routines 213
 - control blocks for 270, 323
 - data stack in 334
 - description 260, 267
 - flags and masks 281
 - how environments are located 307
 - initializing for user-written TMP 272.1
 - integrated into TSO/E 274
 - maximum number of 269, 332
 - non-reentrant 340
 - not integrated into TSO/E 274
 - obtaining address of environment block 340
 - overview for calling REXX routines 213
 - reentrant 340
 - restrictions on values for 315
 - sharing data stack 334
 - terminating 352
 - types of 263, 274
 - user-written TMP 272.1
- language structure and syntax 8
- LASTPOS function 93
- leading blank removal with STRIP function 100
- leading zeros
 - adding with the RIGHT function 98
 - removal with STRIP function 100
- LEAVE instruction
 - See also* DO instruction
 - description of 45
 - use of variable on 45
- leaving your program 40
- LEFT function 94
- LENGTH function 94
- less than operator 14
- less than or equal operator 14
- less than or greater than operator (< >) 14
- level of RACF installed 128
- level of TSO/E installed 128
- LIFO (last-in/first-out) stacking 56
- line length of terminal 94
- line width of terminal 94
- lines from a program retrieved with SOURCELINE 99
- LINESIZE function 94
- LINK host command environment 25
- linking to programs 25
- list 19
- LISTDSI function 110
 - error codes 117
 - function codes 115
 - messages 115
 - reason codes 116
 - variables set by 113
- literal patterns, parsing with 134

literal strings 9
 LOADDD field (module name table) 287
 loading a REXX exec 358
 local function packages 229
 locating a phrase in a string 90
 locating a string in another string 92, 96
 locating current non-reentrant environment 340
 LOCPKFL flag 283
 logical bit operations
 BITAND 80
 BITOR 80
 BITXOR 81
 logical operations 15
 logon procedure
 obtain name of for current session 128
 looping program
 halting 206, 251
 tracing 179, 181, 206, 251
 loops
 See also DO instruction
 See also looping program
 active 44
 execution model 38
 modification of 44
 repetitive 35
 termination of 45
 lower case symbols 10

M

macros
 See mapping macros
 MAKEBUF command 188
 managing storage 385
 mapping macros
 IRXARGTB (argument list for function packages) 232
 IRXARGTB (argument list for IRXEXEC) 222
 IRXDSIB (data set information block) 366, 371
 IRXEFPL (external function parameter list) 231
 IRXENVB (environment block) 323
 IRXENVT (environment table) 332
 IRXEVALB (evaluation block) 226, 232
 IRXEXECB (exec block) 220, 361
 IRXEXTE (vector of external entry points) 328
 IRXFPDIR (function package directory) 234
 IRXINSTB (in-storage control block) 223, 363
 IRXMODNT (module name table) 286
 IRXPACKT (function package table) 295
 IRXPARMB (parameter block) 278, 325
 IRXSHVB (SHVBLOCK) 242
 IRXSUBCT (host command environment table) 249, 291
 IRXWORKB (work block extension) 326
 mask settings 279
 masks for language processor environment 279, 281
 MAX function 95
 maximum number of language processor environments 269, 332
 message identifier replaceable routine 390
 message IDs, displaying 390

messages
 control display of TSO/E messages 118, 119
 language code for 276
 set by LISTDSI function 115
 syntax errors 395
 MIN function 95
 minutes calculated from midnight 102
 mixed DBCS string 85, 406
 module name table
 ATTNROUT field 289
 defaults provided 299
 description 286
 EXECINIT field 289
 EXECTERM field 290
 EXROUT field 288
 format 286
 GETFREER field 288
 IDROUT field 289
 in parameter block 275
 INDD field 287
 IOROUT field 288
 IRXEXECX field 289
 LOADDD field 287
 MSGIDRT field 289
 OUTDD field 287
 part of parameters module 275
 STACKRT field 289
 MONTH option of DATE function 85
 MSG function 118
 MSGIDRT field (module name table) 289
 multiple
 string parsing 138
 multiplication
 definition 141
 operator 14
 MVS batch
 executing exec in 214
 MVS host command environment 24

N

names
 of functions 72
 of programs 51
 of subroutines 32
 of TSO/E REXX external entry points 328
 of variables 10
 reserved command names 165
 negation
 of logical values 15
 of numbers 14
 nesting of control structures 34
 new data stack flag 282
 new data stack, creating 190
 new host command environment flag 283
 NEWSCFL flag 283
 NEWSTACK command 190, 337
 NEWSTKFL flag 282
 NOESTAE flag 283
 NOLOADDD flag 284

NOMSGIO flag 285
 NOMSGWTO flag 285
 non-reentrant environment 284, 340
 non-TSO/E address spaces
 host command environments 23
 initialization of language processor environment 273
 overview of executing an exec 158
 writing execs for 157
 NOP instruction 46
 NOPMSG flag 284
 NOREADFL flag 282
 Normal option of DATE function 86
 NOSTKFL flag 281
 not equal operator 14
 not greater than operator 14
 not less than operator 14
 NOT operator 15
 notation
 engineering 146
 scientific 146
 NOVALUE condition
 on SIGNAL instruction 149
 use of 163
 NOVALUE condition of SIGNAL instruction 149
 NOWRTFL flag 282
 null clauses 17
 null instruction
 See NOP instruction
 null strings 9, 13
 number of language processor environments, changing maximum 332
 numbers
 arithmetic on 14, 139, 141
 checking with DATATYPE 84
 comparison of 14, 144
 definition 140
 description of 10, 139
 formatting for display 90
 in DO instruction 35
 truncating 104
 use in the language 147
 NUMERIC
 DIGITS option 47
 FORM option 47
 FUZZ option 47
 instruction 47
 option of PARSE instruction 50, 147
 settings saved during subroutine calls 34
 numeric patterns, parsing with 132

O

obtaining a larger evaluation block 253
 operation tracing results 64
 operator
 arithmetic 14, 139, 141
 as special characters 11
 comparison 14, 144
 concatenation 13
 logical 15

operator (*continued*)

 precedence (priorities) of 16
 OPTIONS instruction 49
 ORDERED option of DATE function 85
 ORing character strings together 80
 OR, logical
 exclusive 15
 inclusive 15
 OTHERWISE clause
 See SELECT instruction
 OUTDD field (module name table) 287
 output trapping 119
 OUTTRAP function 119
 overflow, arithmetic 147
 OVERLAY function 96
 overlaying a string onto another 96
 overview of REXX processing in different address spaces 155

P

packages, function
 See function packages
 packing a string with X2C 108
 parameter block 275
 format 275, 325
 relationship to parameters modules 275
 parameters modules
 changing the defaults 310
 default values for 299
 defaults 269, 275, 299
 IRXISPRM 275, 299
 IRXPARM 275, 299
 IRXTSPRM 275, 299
 for IRXINIT 343
 format of 275
 providing you own 310
 relationship to parameter block 275
 restrictions on values for 315
 parentheses
 adjacent to blanks 11
 in expressions 13
 in function calls 71
 in parsing templates 135
 PARMBLOCK
 See parameter block
 PARSE instruction 50
 PARSE SOURCE token 277
 parsing 131-138
 definition 133
 general rules 131, 133
 introduction 131
 literal patterns 134
 multiple strings 138
 patterns 134
 positional patterns 136
 selecting words 134
 variable patterns 135
 parsing templates
 in ARG instruction 30
 in PARSE instruction 50
 in PULL instruction 55

passing address of environment block to REXX routines 213, 307
 patterns in parsing 134
 period
 causing substitution in variable names 19
 in numbers 140
 period as placeholder in parsing 136
 permanent command destination change 28
 POS function 96
 position
 last occurrence of a string 93
 of character using INDEX 92
 positional patterns, parsing with 136
 powers of ten in numbers 10
 precedence of operators 16
 precision of arithmetic 140
 prefix
 as used in examples in book 4, 110, 167
 defined in user profile, obtaining 128
 prefix operators 14, 15
 preloading a REXX exec 358
 primary data stack 337
 primary messages flag 284
 PROCEDURE instruction 53
 profile
 See user profile
 programming restrictions 7
 programming services
 description 209
 function packages 229
 general considerations for calling routines 212
 IKJCT441 (variable access) 240
 IRXEXCOM (variable access) 240
 IRXIC (trace and execution control) 251
 IRXRLT (get result) 253
 IRXSUBCM (host command environment table) 247
 passing address of environment block to routines 213
 summary of 155
 writing external functions and subroutines 229
 programs
 attaching 25
 linking to 25
 retrieving lines with SOURCELINE 99
 PROMPT function 123
 protecting variables 53
 pseudo random number function of RANDOM 97
 PULL instruction 55
 PULL option of PARSE instruction 51
 pure DBCS string 85, 406
 PUSH instruction 56

Q

QBUF command 192
 QELEM command 194
 QSTACK command 196
 query
 data set information 110
 existence of host command environment 199
 number of buffers on data stack 192

query (*continued*)
 number of data stacks 196
 number of elements on data stack 194
 queue
 See also data stack
 counting lines in 97
 reading from with PULL 55
 writing to with PUSH 56
 writing to with QUEUE 57
 QUEUE instruction 57
 QUEUED function 97

R

RACF
 level installed 128
 status of 128
 RANDOM function 97
 random number function of RANDOM 97
 RC (return code)
 not set during interactive debug 204
 set by host commands 22
 set to 0 if commands inhibited 66
 special variable 164
 reading from the data stack 55
 reads from input file 282
 reason codes
 for IRXINIT routine 347
 set by LISTDSI 116
 recovery ESTAE 283
 reentrant environment 284, 340
 remainder
 definition 143
 description of 139
 operator 14
 RENTRANT flag 284
 reordering data with TRANSLATE function 104
 repeating a string with COPIES 83
 repetitive loops
 altering flow 45
 controlled repetitive loops 36
 exiting 45
 simple do group 36
 simple repetitive loops 36
 replaceable routines 259, 264, 355
 data stack 380
 exec load 358
 host command environment 376
 input/output (I/O) 366
 message identifier 390
 storage management 385
 user ID 388
 request (shared variable) block (SHVBLOCK) 242
 reservation of keywords 163
 reserved command names 165
 restoring variables 39
 restrictions
 embedded blanks in numbers 11
 first character of variable name 18
 maximum length of results 13

restrictions in programming 7
 restrictions on values for language processor environments 315
 Restructured Extended Executor language (REXX)
 built-in functions 71
 description 1
 keyword instructions 27
 RESULT
 set by RETURN instruction 33, 58
 special variable 164
 results
 length of 13
 Resume Typing (RT) immediate command 198, 251
 retrieving argument strings with ARG 30
 return codes
 as set by host commands 22
 setting on exit 40
 RETURN instruction 58
 return string
 setting on exit 40
 returning control from REXX program 58
 REVERSE function 98
 REXX built-in functions
 See built-in functions
 REXX commands
 See TSO/E REXX commands
 REXX customizing services
 See customizing services
 REXX exec identifier 8
 REXX exit routines
 See exit routines
 REXX external entry points 328
 IRXEXCOM 240
 IRXEXEC 217
 IRXIC 251
 IRXINIT 340
 IRXINOUT 366
 IRXJCL 214
 IRXLOAD 358
 IRXMSGID 390
 IRXRLT 253
 IRXSTK 380
 IRXSUBCM 247
 IRXTERM 352
 IRXUID 388
 REXX instructions
 See instructions
 REXX processing in different address spaces 155
 REXX programming services
 See programming services
 REXX replaceable routines
 See replaceable routines
 REXX vector of external entry points 328
 REXX (Restructured Extended Executor) language 1
 REXX, using in different address spaces 155
 RIGHT function 98
 rounding
 definition 141
 using a character string as a number 10
 routines
 See also functions
 See also subroutines

routines (*continued*)
 exit 391
 for customizing services 259
 for programming services 209
 general considerations for TSO/E REXX 212
 replaceable 355
 RT (Resume Typing) immediate command 198, 251
 running off the end of a program 40

S

SAMPLIB
 samples for parameters modules 310
 SAY instruction 59
 scientific notation 146
 search order
 controlling for REXX execs 284
 for external functions 73
 for external subroutines 73
 for functions 73
 for subroutines 33
 searching a string for a phrase 90
 secondary data stack 337
 seconds calculated from midnight 102
 seconds of CPU time used 128
 SELECT instruction 60
 semicolons
 implied 12
 omission of 27
 within a clause 8
 service units used (system resource manager) 128
 shared variable (request) block (SHVBLOCK) 242
 sharing of data stack between environments 334
 sharing subpools 284
 Shift-in (SI) characters 405, 410
 Shift-out (SO) characters 405, 410
 SHVBLOCK 242
 SIGL
 set by CALL instruction 33
 special variable 164
 SIGNAL function 98
 SIGNAL
 execution of in subroutines 34
 in INTERPRET instruction 42
 SIGNAL instruction 62
 significant digits in arithmetic 140
 simple number
 See numbers
 simple symbols 19
 single stepping
 See interactive debug
 SORTED option of DATE function 85
 source of the program and retrieval of information 51
 SOURCE option of PARSE instruction 51
 SOURCELINE function 99
 SPACE function 99
 special characters 11
 special variables
 RC 164
 RESULT 164
 SIGL 164

- SPSHARE flag 284
- stack
 - See data stack
- STACKRT field (module name table) 289
- status of Data Facility Hierarchical Storage Manager (DFHSM) 128
- status of RACF 128
- stem of a variable
 - assignment to 20
 - description of 19
 - used in DROP instruction 39
 - used in PROCEDURE instruction 53
- stepping through programs
 - See interactive debug
- storage
 - change value in specific storage address 126
 - management replaceable routine 385
 - managing 385
 - obtain value in specific storage address 126
- STORAGE function 126
 - restricting use of 284
- storage management replaceable routine 385
- STORFL flag 284
- storing REXX execs 7, 321
- strictly equal operator 14
- strictly greater than operator 14, 15
- strictly greater than or equal operator 15
- strictly less than operator 14, 15
- strictly less than or equal operator 15
- strictly not equal operator 14
- strictly not greater than operator 15
- strictly not less than operator 15
- string
 - as literal constants 9
 - as names of functions 9
 - as names of subroutines 34
 - comparison of 14
 - concatenation of 13
 - description of 9
 - hexadecimal specification of 9
 - interpretation of 42
 - length of 13
 - null 9, 13
 - quotes in 9
 - verifying contents of 106
- string patterns, parsing with 132
- STRIP function 100
- structure and syntax 8
- SUBCOM command 199
- subpool number 279
- subpools, sharing 284
- subroutines
 - calling of 32
 - external, search order 73
 - forcing built-in or external reference 33
 - naming of 34
 - passing back values from 58
 - providing in function packages 229
 - return from 58
 - use of labels 32
 - variables in 53
- subroutines (*continued*)
 - writing external 229
- substitution
 - in expressions 13
 - in variable names 19
- SUBSTR function 100
- subtraction
 - definition 141
 - operator 14
- SUBWORD function 101
- symbol
 - assigning values to 18
 - classifying 19
 - compound 19
 - constant 19
 - description of 10
 - simple 19
 - uppercase translation 10
 - use of 18
 - valid names 10
- SYMBOL function 101
- syntax checking
 - See TRACE instruction
- SYNTAX condition of SIGNAL instruction 149
- syntax diagrams 5
- syntax error
 - messages 395
 - traceback after 68
 - trapping with SIGNAL instruction 149
- syntax, general 8
- SYSDSN function 127
- SYSEXEC 321
 - controlling search of 284
 - overview of storing REXX execs 7
- SYSPKFL flag 283
- SYSPROC 321
 - controlling search of 284
 - overview of storing REXX execs 7
- system files
 - overview of SYSPROC and SYSEXEC 7
 - storing REXX execs 7
 - SYSEXEC 321
 - SYSPROC 321
- system function packages 229
 - IRXEFMVS 230
 - IRXEFPCK 230
 - TSO/E-supplied 230
- system information, obtaining
 - CPU time used 128
 - RACF level installed 128
 - RACF status 128
 - SRM service units used 128
 - status of DFHSM 128
 - TSO/E level installed 128
- system resource manager (SRM), number of service units used 128
- system-supplied routines
 - IKJCT441 240
 - IRXEXCOM 240
 - IRXEXEC 214

system-supplied routines (*continued*)

IRXIC 251
 IRXINIT 340
 IRXINOUT 366
 IRXJCL 214
 IRXLOAD 358
 IRXMSGID 390
 IRXRLT 253
 IRXSTK 380
 IRXSUBCM 247
 IRXTERM 352
 IRXUID 388

Systems Application Architecture (SAA) 6
 SYSTSIN 287
 SYSTSPRT 287
 SYSVAR function 128

T

TE (Trace End) immediate command 201, 206, 251

templates, parsing

general rules 131
 in ARG instruction 30
 in PARSE instruction 50
 in PULL instruction 55

temporary command destination change 28

ten, powers of 146

terminal information, obtaining

lines available on terminal screen 128
 width of terminal screen 128

terminal monitor program

See TMP

terminals

finding number of lines with SYSVAR 128
 finding width with LINESIZE 94
 finding width with SYSVAR 128
 reading from with PULL 55
 writing to with SAY 59

terminating a language processor environment 352

termination routine (IRXTERM) 272, 352

user-written TMP 272.1

terms and data 13

text formatting

See formatting

See word

THEN

as free standing clause 27
 following IF clause 41
 following WHEN clause 60

TIME function 102

TMP

language processor environments for
 user-written 272.1
 user-written 272.1

TO phrase of DO instruction 35

token for PARSE SOURCE 277

tokens 9

trace and execution control (IRXIC routine) 251

Trace End (TE) immediate command 201, 203, 251

TRACE function 103

TRACE instruction 64

See also interactive debug

TRACE setting

altering with TRACE function 103
 altering with TRACE instruction 64
 querying 103

Trace Start (TS) immediate command 202, 203, 251

trace tags 67

traceback, on syntax error 68

tracing

action saved during subroutine calls 34
 by interactive debug 203
 data identifiers 67
 execution of programs 64
 external control of 206
 looping programs 206

tracing flags

+++ 67

. 67

>C> 67

>F> 67

>L> 67

>O> 68

>P> 68

>V> 68

>.> 67

>>> 67

trailing blank removed using STRIP function 100

trailing zeros 141

TRANSLATE function 104

translation

See also uppercase translation

with TRANSLATE function 104

with UPPER instruction 69

trap command output 119

trap conditions 82

trapping of conditions 149

TRUNC function 104

truncating numbers 104

TS (Trace Start) immediate command 202, 206, 251

TSO host command environment 24

TSOFL flag 274, 281

TSOREXX1 (sample for IRXPARMS) 310

TSOREXX2 (sample for IRXTSPRM) 310

TSOREXX3 (sample for IRXISPRM) 310

TSO/E address space

host command environments 23

initialization of language processor
 environment 272

overview of executing an exec 161

writing execs for 159

TSO/E external functions

LISTDSI 110

MSG 118

OUTTRAP 119

PROMPT 123

STORAGE 126

SYSDSN 127

SYSVAR 128

TSO/E level installed, obtaining 128

TSO/E profile
 See user profile

TSO/E REXX commands 167

- DELSTACK 168
- DROPBUF 169
- EXECIO 171
- EXECUTIL 178

immediate commands

- HI 185
- HT 186
- RT 198
- TE 201
- TS 202

- MAKEBUF 188
- NEWSTACK 190
- QBUF 192
- QELEM 194
- QSTACK 196
- SUBCOM 199

 valid in non-TSO/E 157

 valid in TSO/E 159

TSO/E REXX customizing services
 See customizing services

TSO/E REXX programming services
 See programming services

TSO/E REXX replaceable routines
 See replaceable routines

type of data checking with DATATYPE 84

types of function packages 229

types of language processor environments 263, 274

typing data
 See SAY instruction

U

unassigning variables 39

unconditionally leaving your program 40

underflow, arithmetic 147

unpacking a string with C2X 84

UNTIL phrase of DO instruction 35

UPPER instruction 69

UPPER option of PARSE instruction 50

uppercase translation

- during ARG instruction 30
- during PULL instruction 55
- of symbols 10
- with PARSE UPPER 50
- with TRANSLATE function 104
- with UPPER instruction 69

USA option of DATE function 85

user function packages 229

user ID

- as used in examples in book 4, 110, 167
- for current session 128
- replaceable routine 388

user information, obtaining

- logon procedure for session 128
- prefix defined in user profile 128
- user ID for session 128

user profile

- obtain prefix defined in 128
- prompting considerations 123
- prompting from interactive commands 123

user-written TMP

- executing REXX execs 272.1
- language processor environments for 272.1

USERID function 105

USERPKFL flag 282

V

VALUE function 105

VALUE option of PARSE instruction 52

values used to initialize language processor environment 302

VAR option of PARSE instruction 52

variable access (IRXEXCOM) 240

variable names 10

variable patterns, parsing with 135

variables

- compound 19
- controlling loops 36
- description of 18
- direct interface to 240
- dropping of 39
- exposing to caller 53
- getting value with VALUE 105
- in internal functions 53
- in subroutines 53
- new level of 53
- parsing of 52
- resetting of 39
- set by LISTDSI 113
- setting new value 18
- simple 19
- special
 - RC 164
 - RESULT 164
 - SIGL 164
- testing for initialization 101
- translation to uppercase 69
- valid names 18
- with the LISTDSI function 113

vector of external entry points 328

VERIFY function 106

VERSION option of PARSE instruction 52

W

WEEKDAY option of DATE function 85

WHEN clause
 See SELECT instruction

WHILE phrase of DO instruction 35

whole numbers

- checking with DATATYPE 84
- description of 11

word

- counting in a string 108
- deleting from a string 87
- extracting from a string 101, 106
- finding in a string 90

word (*continued*)

- finding length of 107
- in parsing 134
- locating in a string 107
- WORD function 106
- word processing
 - See formatting
 - See word
- WORDINDEX function 107
- WORDLENGTH function 107
- WORDPOS function 107
- WORDS function 108
- work block extension 326
- writes to output file 282
- writing external functions and subroutines 229
- writing REXX execs
 - for non-TSO/E 157
 - for TSO/E 159
- writing to the stack
 - with PUSH 56
 - with QUEUE 57

X

- XORing character string together 81
- XOR, logical 15
- XRANGE function 108
- X2C function 108
- X2D function 109

Z

- zeros added on the left 98
- zeros removal with STRIP function 100

Special Characters

(period)

- as placeholder in parsing 136
- causing substitution in variable names 19
- in numbers 140
- < (less than operator) 14
- << (strictly less than operator) 14, 15
- <= (strictly less than or equal operator) 15
- <> (less than or greater than operator) 14
- <= (less than or equal operator) 14
- + (addition operator) 14, 141
- +++ tracing flag 67
- | (inclusive OR operator) 15
- || (concatenation operator) 13
- && (exclusive OR operator) 15
- & (AND operator) 15
- ! prefix on TRACE option 66
- * (multiplication operator) 14, 141
- *-* tracing flag 67
- ** (power operator) 14, 143
- / (division operator) 14, 141
- // (remainder operator) 14, 143
- /= (not equal operator) 14
- /= (not strictly equal operator) 14
- , (comma)
 - as continuation character 12
 - in CALL instruction 33

, (comma) (*continued*)

- in function calls 71
- separator of arguments 33, 71
- within a parsing template 30, 132, 133, 138
- % (integer division operator) 14, 143
- > (greater than operator) 14
- >C> tracing flag 67
- >F> tracing flag 67
- >L> tracing flag 67
- >O> tracing flag 68
- >P> tracing flag 68
- >V> tracing flag 68
- >> tracing flag 67
- >< (greater than or less than operator) 14
- >> (strictly greater than operator) 14, 15
- >>> tracing flag 67
- >>= (strictly greater than or equal operator) 15
- >= (greater than or equal operator) 14
- ? prefix on TRACE option 66
- : (colon)
 - as a special character 11
 - in a label 17
- = (equal sign)
 - assignment indicator 18
 - equal operator 14
 - immediate debug command 203
 - in DO instruction 35
- = (strictly equal operator) 14
- (subtraction operator) 14, 141
- \ (NOT operator) 15
- \< (not less than operator) 15
- \<< (strictly not less than operator) 15
- \> (not greater than operator) 15
- \>> (strictly not greater than operator) 15
- \= (not equal operator) 14
- \= (strictly not equal operator) 14









SN28-1293-00

