

# **ARexxGuide**

**COLLABORATORS**

|               |                              |                  |                  |
|---------------|------------------------------|------------------|------------------|
|               | <i>TITLE :</i><br>ARexxGuide |                  |                  |
| <i>ACTION</i> | <i>NAME</i>                  | <i>DATE</i>      | <i>SIGNATURE</i> |
| WRITTEN BY    |                              | October 17, 2022 |                  |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|--------|------|-------------|------|
|        |      |             |      |

# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>ARexxGuide</b>  | <b>1</b> |
| 1.1      | ARexxGuide Copyright © 1993,1994 Robin Evans . . . . .                       | 1        |
| 1.2      | ARexxGuide   Registration   THANK YOU FOR YOUR SUPPORT! . . . . .            | 1        |
| 1.3      | ARexxGuide   CONTENTS . . . . .  | 2        |
| 1.4      | ARexxGuide   GLOSSARY (Press -Retrace- to return to previous node) . . . . . | 9        |
| 1.5      | ARexxGuide   Interactive examples -- Requirements . . . . .                  | 14       |

---

# Chapter 1

## ARexxGuide

### 1.1 ARexxGuide Copyright © 1993,1994 Robin Evans

AN AMIGAGUIDE® TO ARexx Second edition (v2.0a)  
by Robin Evans

Registration

Comprehensive contents

Introduction

Tutorials

Techniques

Reference

Basic elements

Guide to the powders  
& potions in the ARexx  
chemistry set.

Operators

Glue for arithmetic,  
comparison, & logical  
expressions.

Instructions

Syntax & explanation  
of keywords and  
instructions

Commands

Utility programs.

Functions

Syntax & explanation  
of built-in & support  
functions.

Error codes

Error codes/messages.  
Possible causes and  
solutions.

INDEX

Copyright © 1993,1994 Robin Evans. All rights reserved.

### 1.2 ARexxGuide | Registration | THANK YOU FOR YOUR SUPPORT!

This guide is shareware. Please use it and pass it on (in its ↔  
original

archive) to your friends and acquaintances. If you learn something from  
the guide or find it helpful in writing ARexx programs, then please take a  
moment to fill out the registration form and send in the fee.

The requested fee for this guide is \$15.00. For that you get the complete  
reference you see before you now -- information you would pay \$25 to \$40  
for if it were presented in traditional book form.

Sorry, folks, we don't have operators standing by to take your order, but we do have -- yes, standing by -- representatives of the worlds' postal services who will deliver your registration fee to the address below.

To make things easier, and to provide information helpful in making future revisions to this guide, the button below will guide you through an on-line registration form that can be printed immediately or saved to disk.

Fill out registration form  
\*  
Send registration fee (and optional form) to:  
Robin Evans  
1020 Seneca #405  
Seattle WA 98101-2720

And thank you, very much.

Please send comments or requests to any of the following on-line addresses:

robin@halcyon.com            on Internet  
R.EVANS6                    on GENie  
r.evans6@genie.geis.com on Internet

Next: ARexxGuide contents | Prev: Registration intro | Contents: main

## 1.3 ARexxGuide | CONTENTS

Foreword:

Preface

- Acknowledgements
- References
- About the author
- Compatibility issues

About this guide

- Navigating hints

Intro to ARexx

- Hello World!
- Why ARexx?
- Getting it started
- Writing programs
- Running a script

Tutorials:

- Using ARexx with macros: Extending keyboard macros
  - Simple macro: recording keystrokes
  - Adding ARexx control to the macro
  - A closer look at ARexx IF instruction
  - Repeating macro with an ARexx loop

- Growing a macro
- ADDRESS and the macro
- Debugging a macro
- Using the ARexxGuide online help system
  - Setting up a help key macro
  - Environmental variables
  - Information about a non-matches
  - Building a new help-key macro
    - GetCLine: Get current line from editor
    - GetCPos: Get cursor position from editor
    - GetCWord: Calculate (or get) current word
    - DisplayStatus: Display a message in status bar
    - BoolReq: Post a Boolean requester
    - GetWinInfo: Get information about editor's window
    - EditorExit: Send special editor commands on exit
    - SetExecStr: Set command issued by info window
    - SetAddress: Set the address used by info window
    - DisplayAG: Display the node in AmigaGuide viewer
- A complete program example: Uncrunch.rexx

#### Interactive examples:

- \*
  - Test for valid symbols
  - Comparison demonstration
  - NUMERIC demonstration
  - TRACE demonstration
  - Standard I/O demonstration
  - Break-key demonstration

#### Techniques:

- Strings
  - CountChar(): count characters with COMPRESS()
  - CountWords(): count words in a file
  - Format data into table form
  - Format(): round and format a number
  - AddComma(): add commas to an integer
    - Alternative: add commas within a loop
  - WordWrap(): wordwrap text to a defined length
  - ParseFileName(): split name of file from path
- Input/Output
  - Open console windows for I/O
  - Output text to a printer
  - Read data from one file, write to another
  - Retrieve result of AmigaDOS command
  - Getting and sending message packets
- Data storage and retrieval
  - Store global variables on the clip list
  - Get and set environmental variables
  - Retrieve data from source code
  - Create a data scratchpad with PUSH, QUEUE, and PULL
  - SeekToRecord(): pull single record from data file
  - Use VALUE() to create interpreted variable names
  - Check unique datatypes with VERIFY()
  - Determine version number of any library

#### Basic Elements:

---

---

## Structure of an ARexx program

### Tokens

- Comment tokens

- String tokens

  - Hex and binary strings

- Symbol tokens

  - Fixed symbols

  - Variable symbols

- Operator tokens

  - Concatenation || <blank> <abuttal>

  - Arithmetic + - | / // %

  - Comparative < > = == >= <=

  - Logical & | && ~

- Reserved characters

  - The comma: Continuation character

  - The semicolon: Clause end symbol

    - Using semicolons for in-line scripts

  - The colon: Label identifier

  - Parenthesis: Grouping / Function argument list

### Expressions

- Numbers

  - Numeric precision

- Strings

  - Treating numbers as strings

- Variables

  - Using variables

  - Compound variables

    - Overview: Using compound variables

    - Stem variables

    - Extending stem variables

    - Substituting values in compound variables

    - Using strings as the derived name of a branch

    - Setting default value of a compound variable

    - Finding values in a compound variable

  - Special variables

    - RC

    - RESULT

    - SIGL

  - 'Natural' data typing in ARexx

- Function calls

  - Internal functions

  - Built-in functions

  - Library/Host functions

  - External functions

  - Function arguments

- Operations

  - Concatenation

  - Arithmetic

  - Comparative

  - Logical

  - Conditional expressions

- Avoiding accidental commands from expressions

### Clauses

- Assignment clauses

- Instructions

- Command clauses

  - Command host: what is it?

- The default host
- Determining the initial host
- Entering commands in a script
- Example script
- Label clauses
- Null clauses

Instruction keywords:

- ADDRESS
- ARG
- BREAK
  - Breaking structure
- CALL
- DO
  - <number>
  - Index variable/TO/BY
- FOR
- WHILE/UNTIL
- FOREVER
- END
- DROP
- ECHO
- EXIT
- IF
  - ELSE
- INTERPRET
- ITERATE
- LEAVE
- NOP
- NUMERIC
- OPTIONS
- PARSE
  - ARG
  - EXTERNAL
  - NUMERIC
  - PULL
  - SOURCE
  - VALUE <expression> WITH
  - VAR
  - VERSION
  - Templates
    - Tokenization
      - The period: Placeholder token
    - Pattern markers
    - Positional markers
    - Using variables as template markers
    - Combining different types of markers
    - Using multiple templates
  - ARexx departures from REXX-standard PARSE
- PROCEDURE
  - EXPOSE
- PULL
- PUSH
- QUEUE
  - PUSH, QUEUE and REXX data-stream I/O
- RETURN
- SAY

---



```
SELECT
  WHEN
  OTHERWISE
SIGNAL
  ON | OFF <interrupt>
  BREAK_C
  | BREAK_D
  | BREAK_E
  | BREAK_F
  ERROR
  FAILURE
  HALT
  IOERR
  NOVALUE
  SYNTAX
  <label name>
TRACE
  Trace options
  Interactive tracing: ?
  Command inhibition: !
UPPER
```

#### ARexx functions:

##### Comparison functions

```
ABBREV
COMPARE
FIND
INDEX
LASTPOS
POS
VERIFY
```

##### String manipulation

```
CENTER
COMPRESS
COPIES
DELSTR
INSERT
LEFT
LENGTH
OVERLAY
REVERSE
RIGHT
STRIP
SUBSTR
TRANSLATE
TRIM
UPPER
XRANGE
```

##### Word manipulation

```
DELWORD
SPACE
SUBWORD
WORD
WORDINDEX
WORDLENGTH
WORDS
```

##### Char/Num translation

---

---

- B2C
- C2B
- C2D
- C2X
- D2C
- D2X
- X2C
- X2D
- Number manipulation
  - ABS
  - HASH
  - MAX
  - MIN
  - RANDOM
  - RANDU
  - SIGN
  - TRUNC
- Informational
  - DATE
    - DATE() Options
    - Persistence of DATE() & TIME() settings
  - SHOW
  - SHOWDIR
  - SHOWLIST
    - SHOWLIST() Options
  - TIME
    - TIME() Options
    - The elapsed time counter
    - Persistence of DATE() & TIME() settings
- File input/output
  - Overview of I/O functions
  - Setting the logical file name
  - Using I/O functions other devices
  - Standard I/O files
  - CLOSE
  - EOF
  - LINES
  - OPEN
  - READCH
  - READLN
  - SEEK
  - WRITECH
  - WRITELN
- File management
  - DELETE
  - EXISTS
  - MAKEDIR
  - RENAME
  - STATEF
- ARexx control
  - ADDRESS
  - ADDLIB
  - ARG
  - DATATYPE
    - DATATYPE() Options
  - DELAY
  - DIGITS

---

ERRORTXT  
FORM  
FUZZ  
GETCLIP  
PRAGMA  
REMLIB  
SETCLIP  
SOURCELINE  
SYMBOL  
TRACE  
VALUE

Message ports

Using ports in ARexx programs

CLOSEPORT  
GETARG  
GETPKT  
OPENPORT  
REPLY  
TYPEPKT  
WAITPKT

Memory management

ALLOCMEM  
BADDR  
EXPORT  
FORBID  
FREEMEM  
FREESPACE  
GETSPACE  
IMPORT  
NEXT  
NULL  
OFFSET  
PERMIT  
STORAGE

Bit-wise operations

BITAND  
BITCHG  
BITCLR  
BITCOMP  
BITOR  
BITSET  
BITTST  
BITXOR

ARexx operators:

Concatenation

Arithmetic

Table of arithmetic operators

Comparison

Table of comparison operators

Logical

Table of logical operators

Operator priority

Parentheses: Change priority

AmigaDOS command programs:

RexxMast

---

RXC  
 RX  
 HI  
 RXLIB  
 RXSET  
 TCO  
 TCC  
 TS  
 TE  
 WaitForPort

Useful tools:

WShell  
 ExecIO

Error codes and messages

GLOSSARY  
 INDEX

## 1.4 ARexxGuide | GLOSSARY (Press -Retrace- to return to previous node)

ARexxGuide glossary of terms

~~~~~

ADDRESS STRING A four-character (4-byte) string that represents a machine address. The character string will be meaningless in itself, but can be translated to meaningful form with the `c2d()` or `c2x()` functions.'

ASSIGNMENT A process that gives (assigns) a value of some kind to a variable. An assignment clause takes this form:

`<symbol> = <expression>;`

The `<symbol>` -- a variable -- becomes a placeholder for the value of `<expression>`.

There are also other less common, ways that an assignment can be made, notably the `PARSE` and `DO` instructions.

BOOLEAN Either true or false, which -- in ARexx -- is considered to be 1 for true and 0 for false. Named after the mathematician George Boole.

CLAUSE A collection of tokens forming a program statement that can be executed by ARexx, usually contained on a single line. A clause is the smallest language unit that can be executed as a statement.

COMMAND A program statement ( a clause ) that is sent to an external environment ( the host ) to be run. The host

determines the syntax and other requirements for a command. Although it is often overlooked, commands should be enclosed in quotation marks.

CON: Or: Console Window. A logical device that opens a text window on the Workbench or other public screen. This device can be used as the <filename> with the file I/O functions to direct output to a window opened by the script.

CONCATENATE To combine one part with another to form a new whole. When two strings are concatenated, they are joined together to form a new string.

A space between two expressions acts as a concatenation operator in ARexx as do the characters `||'`.

CONSTANT In ARexx, a symbol that cannot be used as the target of a variable assignment. The most common constants in ARexx are numbers.

CONTINUATION When a comma `,` is used as the last significant token in a line, it indicates that the current line should be combined with the next line to form a single clause. Comments and other null values may be included after the continuation token.

CONTROL STRUCTURE A programming construct that allows a series of statements to be executed as part of a block. The instructions DO, SELECT, and IF create control structures in ARexx.

DEBUG To search for and eliminate (eventually) problems or bugs in a program. The TRACE instruction aids debugging in ARexx.

DYADIC Having two parts. In ARexx, the term refers to operations that have two operands (2 + 2, for instance). Some operations have only one operator (-1, for instance) and are referred to here as prefix operations. The more technical name for the opposite of a dyadic operation is unary operation.

EGREGIOUS It means "very bad," but use of this word shows that the writer has spent too much time in the company of lawyers. (Which may be the same thing, come to think of it.)

EXPONENTIAL A way of writing a number in which one value -- the exponent -- is a power of ten by which the other value will be multiplied before use.

In ARexx, an `e` in a number indicates exponential notation. 7.34e6 is the same number as 7340000.

EXPRESSION One or more tokens that can be evaluated to produce a

a single value. An expression can be anything from a single number to a mixture of numbers, strings, variables, sub-expressions, and function calls.

- FUNCTION** A subprogram that returns a single value to the calling environment. A function might be defined in any of several ways. Some are a built-in feature of the language, some are available in external libraries, and some are written by the user either as a subroutine in the executing script or as an external program.
- GUI** Graphic User Interface. It's the acronym used to refer to things like windows, icons, mouse pointers, menus, and requesters that are common on the current generations of computer systems.
- HOST** A program that can accept and act on commands issued from an ARexx script. The ADDRESS instruction is used to set up communication with a host.
- INSTRUCTION** The basic program statement in ARexx scripts. An instruction may include several clauses, but always begins with a REXX keyword which must be the first token in the clause.
- Instruction include IF, CALL, DO and similar statements.
- INTERPRETER** A program that translates source code (the program lines you write) into machine instructions. It does that each time the script is run. RexxMast is the ARexx interpreter program.
- IO** Input/Output. The term refers to the various ways of obtaining data and displaying or saving it. The I/O system on the Amiga includes disk drives, windows, and requesters.
- ITERATION** A program-ese synonym for 'repetition'. To a human the instruction to "Do forever" would be a Sisyphean punishment. To a computer, it is just another task. In ARexx, iteration is performed by a single instruction, DO, which has a wide range of options to give the programmer control over when the iteration stops.
- KEYWORD** The word that identifies an ARexx instruction or the option to an instruction. Keywords and instructions are detailed in the Instruction reference.
- LOGICAL DEVICE** A part of the computer system defined through software. In AmigaDOS, logical devices intervene between the application program (including ARexx) and such hardware devices as disk drives, printers, and the monitor screen.
-

- LOOP** A section of program code that is repeated (or iterated). Looping in ARexx is controlled by the `DO` instruction.
- MANTRA** In Hinduism, a sacred formula, repeated over and over again, that is believed to possess special power. (Looking up this word demonstrates one of two things: either the user wasn't around for the 60''s or wasn't paying attention. <insert smiley chars> )
- NESTED** To place one thing within another just as an egg is placed in a bird's nest. A nested function is one function used as an argument to another function as in `RIGHT(TRUNC(Amount, 2), 6)`. Here the `TRUNC()` function, which truncates the decimal points on a number, is nested within the `RIGHT()` function, which right-justifies the resulting number.
- NIL:** A logical device recognized by AmigaDOS that will throw away input or output directed to it.
- NUMBER** A string or symbol made up only of digits (0 - 9) with an optional decimal point `'.'` that may be placed anywhere within the string -- at the beginning, at the end, or anywhere in between.
- Another option allows for exponential notation when the letter `'e'` is included within the string: The number to the right of the `'e'` is interpreted as the exponent to the value on the left.
- OPERATION** An expression that includes an operator and usually two terms that are combined in a way specified by the operator to produce a new single value. `'3+5'` is an arithmetic operation.
- Some operators (like negation) act on a single term
- OPERATOR** Any of a variety of tokens that represent an operation that is to be performed on the adjoining expressions. Operators include these characters (sometimes used in combination):
- `+ - * / % | & = ~ > <`
- A space between two strings is also an operator.
- PREFERENCES** A series of programs that are part of the Amiga OS. They allow the user to customize most aspects of the system.
- PROTOTYPING** The process of developing an initial version of a software application in one language to test the logic of the code and the usefulness of contemplated options.
- PRT:** A logical device recognized by AmigaDOS that directs
-

output to the printer defined in Preferences. This device can be used as the <filename> with the file I/O functions to print data from an ARexx script.

**RESERVED** A token that serves a specialized purpose in the language and cannot be used for any other purpose.

REXX has a limited set of reserved tokens: The single characters representing operators and special characters are reserved in all situations. Instruction keywords and sub-keywords are reserved only within the limited range of the instruction itself. The variables [x] and [b] -- although they are not technically reserved -- should be avoided because of possible conflicts with hex and binary strings.

**STDERR** Standard error device. This is the logical name assigned to a device to which ARexx will send error messages and the output of tracing. If the trace console is open, that will become STDERR. The **PARSE EXTERNAL** instruction retrieves input from this device.

**STDIN** Standard input device. This is the logical name assigned to a device from which ARexx will retrieve input then the **PARSE PULL** instruction is used. It is usually the shell from which a program was launched, although a script started from another environment will often have **STDIN** assigned to **NIL:** .

**STDOUT** Standard output device. This is the logical name of the device to which ARexx will output the expression used in a **SAY** instruction. It is usually the shell from which a program was launched, although a script started from another environment will often have **STDOUT** assigned to **NIL:**.

**STRING** A character or group of characters that are stored and referenced as a unit. A 'literal string' or string token is surrounded by quotation marks -- either single { ' ' } or double { " }, but the word 'string' may also refer to the value of a variable, or the result of an expression.

A string can comprise up to 65535 characters.

**SUBROUTINE** A section of code separated from the main body of a program. In ARexx, subroutines are identified by labels and usually serve as internal functions .

**SYMBOL** A token made up of any of the following alphabetic characters, digits, or special characters:

A to Z a to z 0 to 9 . ! \$ \_ @ #

The following are symbols: Names for variables or functions , numbers , and instruction keywords .



A symbol may be entered in a mixture of upper- and lowercase alphabetic characters, but all symbols are translated to uppercase during evaluation.

Symbols can have up to 65535 characters.

|          |                                                                                                                                                                                              |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TOKEN    | The simplest (smallest) item in the language, from which more complex elements are formed. A token might be a single character like '+' or '/', a number, or a word like 'FOO' or 'CALL'.    |
| TRUNCATE | To shorten by chopping off the trailing end. If a decimal number like 1.9 is truncated to one digit, it would become 1, rather than the number 2 that would result from rounding the number. |
| VARIABLE | A symbol that becomes a placeholder for another value and can, in most cases, be used in place of the literal value it represents. A variable name follows general symbol-naming rules.      |

## 1.5 ARexxGuide | Interactive examples -- Requirements

The registration form and several interactive examples scattered throughout ARexxGuide use ARexx scripts to provide the interactive environment. Because they must try to run a script, the buttons impose some extra requirements:

1. The RX command must be located in a directory that is included in the Workbench command search path.

The RX command is included in a special directory, "rexxc," on Workbench disks distributed by Commodore. If the contents of that directory are not

---

included in your search path, the links will fail. Either add "sys:rexxc" to the search path or move RX to a directory like "C:" that is already in the path.

(AmigaGuide has a built-in "RX" link command. It is not used here because scripts launched with the command exhibit some inconsistent behavior.)

2. The #?.rexx files distributed with the ARexxGuide archive must be stored either in the REXX: directory or in ARexxGuide's current directory.

The most versatile place to store any .rexx file is in the REXX: directory since it can then be found and launched by RX no matter what the current directory is. That directory can become crowded, however. Since the interpreter looks for files first in the current directory, it can be a useful alternative for task-specific ARexx scripts like those included with this guide.

If you decide to keep the .rexx files in another directory and if the guide is launched from a shell or directory utility, the 'CD' command should be used before launching the guide to change the working directory to the location of the .rexx files. If the guide is launched with an icon, the .rexx files should be stored in the same directory as the icon's .info file.

3. AmigaGuide should be launched as a command rather than through a call to the ShowNode() function of amigaguide.library.

Scripts that use the library function to launch AmigaGuide files have circulated on the nets. Use of the function limits AmigaGuide's ability to call ARexx scripts.

---