ARx_Func2.ag

COLLABORATORS			
	TITLE : ARx_Func2.ag		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 17, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ARx_Func2.ag

1.1 "

```
AN AMIGAGUIDE® TO ARexx                  Second edition (v 2.0)
by Robin Evans

   Note: This is a subsidiary file to ARexxGuide.guide. We recommend
   using that file as the entry point to this and other parts of the
   full guide.

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

1.2 ARexxGuide | Functions reference (3 of 12) | WORD MANIPULATION

```
 DELWORD
(<string>,<wordnum>, [<length>])

 SPACE
(<string>,<number>,[<padchar>])

 SUBWORD
(<string>,<wordnum>, [<length>])

 WORD
(<string>,<wordnum>)

 WORDINDEX
(<string>,<wordnum>)

 WORDLENGTH
(<string>,<wordnum>)

 WORDS
(<string>)
```

Related function:

```
    FIND

                    Also see  String manipulation functions
                              PARSE instruction

A 'word' is any collection of characters separated by one or more spaces
from other characters in a string. These functions allow the programmer to
manipulate such words in a quick and elegant manner.

Next: Translation func. | Prev: String functions | Contents: Function ref.
```

1.3 ARexxGuide | Functions reference | Words (1 of 7) | DELWORD

```
                rv = DELWORD(<string>, <wordnum>, [<length>])
     rv is a  string

Deletes a portion of <string> beginning at the word represented by
<wordnum> for <length> number of words. If <wordnum> is greater than the
number of words in <string> then <string> is returned unchanged.

If <length> is omitted, everything to the right of (and including) the
word at <wordnum> position is deleted.

   Examples:
         say delword('the indestructible chaos of timeless things',3,3);
                                        >>> the indestructible things

   Also see  DELSTR

                SUBWORD
               Next: SPACE() | Prev: Word functions | Contents: Word functions
```

1.4 ARexxGuide | Functions reference | Words (2 of 7) | SPACE

```
rv = SPACE(<string>, [<number>], [<padchar>])
     rv is a  string

Formats the original string by placing <number> of <padchar> characters
between each set of blank-delimited words. Leading and trailing blanks are
always removed. If <number> is omitted or is 0, then all spaces in the
string are removed.

The default <number> is 0. The default pad character is a blank.

   Example:
         say space('I don''t know what it is',3);
                                 >>> I   don't   know   what   it   is
         say space('In the end it was magic',1,'_');
                                 >>> In_the_end_it_was_magic
         say space('I    knew these  hills.')
```

```
                                >>> Iknewthesehills.

   Also see  CENTER
             COMPRESS

Next: SUBWORD() | Prev: DELWORD() | Contents: Word functions
```

1.5 ARexxGuide | Functions reference | Words (3 of 7) | SUBWORD

```
                rv = SUBWORD(<string>, <wordnum>, [<length>])
     rv is a  string

Returns a substring of the original, but the division is made by word
position rather than character position. The result will contain <length>
blank-delimited words made up the words in <string> beginning at the word
in position <wordnum>.

If <length> is omitted, then all of <string> starting at <wordnum> will be
returned.

<length> must be positive.

   Example:
         say subword('yet nothing is changed',2,2);    >>> nothing is

   Also see
                WORD
                             SUBSTR
            FIND

                DELWORD
               Next: WORD() | Prev: SPACE() | Contents: Word functions
```

1.6 ARexxGuide | Functions reference | Words (4 of 7) | WORD

```
                rv = WORD(<string>,<wordnum>)
     rv is a  string

The result is the blank-delimited word in <string> at position <wordnum>,
or a null string if there are fewer than <wordnum> words in <string>.

   Example:
         say word('the most you can hope',5);    >>> hope

   Also see
                SUBWORD

                WORDS
                             PARSE TOKENIZATION  instruction

Next: WORDINDEX() | Prev: SUBWORD() | Contents: Word functions
```

1.7 ARexxGuide | Functions reference | Words (5 of 7) | WORDINDEX

```
                rv = WORDINDEX(<string>,<wordnum>)
     rv is a  number

The result is the character position of the first character in the word at
position <wordnum> in <string> or 0 if there are less than <wordnum> words.

   Example:
         say wordindex('to be a little less the creature',4);  >>> 9

   Also see  FIND

                 WORDS

                 WORDLENGTH
                Next: WORDLENGTH() | Prev: WORD() | Contents: Word functions
```

1.8 ARexxGuide | Functions reference | Words (6 of 7) | WORDLENGTH

```
                rv = WORDLENGTH(<string>,<wordnum>)
     rv is a  number

The result is the length of the blank-delimited word at position <wordnum>
in <string>.

   Example:
         say wordlength('you were in the beginning',3);        >>> 2


   Also see  FIND

                 WORDINDEX

                 WORDS
                Next: WORDS() | Prev: WORDINDEX() | Contents: Word functions
```

1.9 ARexxGuide | Functions reference | Words (7 of 7) | WORDS

```
                rv = WORDS(<string>)
     rv is a  number

The result is the number of blank-delimited words in <string>.

   Example:
      say words('and the middle');      >>> 3
```

Also see FIND

WORDINDEX

WORDLENGTH

Technique note: CountWords() user function

Next: Word functions | Prev: WORDLENGTH() | Contents: Word functions

1.10 ARexxGuide | Functions reference (4 of 12) | TRANSLATION

```
B2C
(<binary-string>)

C2B
(<string>)

C2D
(<string>, [<numbytes>])

C2X
(<string>)

D2C
(<whole number>, [<length>])

D2X
(<whole number>, [<length>])

X2C
(<hex-string>)

X2D
(<hex-string>)
```

The function in this list translate values from one form of representation to another. D2C(), for instance, performs a task similar to the CHR$() function in BASIC -- translating a number to its ASCII character value.

Since ARexx stores all values as strings , it is difficult to decipher some of the values it returns -- such as the addresses returned by GETSPACE() or WAITPKT() . The C2D() function can translate those values into a more readily understood format.

Next: Number functions | Prev: Word functions | Contents: Function ref.

1.11 ARexxGuide | Functions reference | Translation (1 of 8) | B2C

```
               rv = B2C(<binary number>)
     rv is a  string
```

Translates a binary number into its ASCII character representation.

Spaces are allowed at the byte boundaries in the input <binary number>.

```
   Example:
         say b2c(01100001);                          >>> a
         say b2c(01000110 01000110 01010011);        >>> FFS

   Also see
                C2B

                C2X
                Compatibility issues:
   This function is an extension that is not defined in  TRL2 . Although a
   function of this name might be included in other REXX implementations,
   there is no assurance that it will be.

   TRL2 defines a b2x() function that is not supported in ARexx. It can be
   duplicated with the nested functions c2x(b2c(<binary>)).
```

Next: C2B() | Prev: Translation func. | Contents: Translation func.

1.12 ARexxGuide | Functions reference | Translation (2 of 8) | C2B

```
               rv = C2B(<string>)
     rv is a  string  of binary digits
```

Converts <string> into binary digits.

```
   Examples:
         say c2b('FFS');                 >>> 010001100100011001010011
         say c2b('F') c2b('F') c2b('S');>>> 01000110 01000110 01010011
         say c2b('a');                   >>> 01100001
```

Each character in the argument string is converted to its binary representation. The value returned is a concatenation of each of those binary numbers. This is the way <string> would be represented in the machine's memory.

```
   Also see
                B2C
                Compatibility issues:
   This function is an extension that is not defined in  TRL2 . Although a
   function of this name might be included in other REXX implementations,
   there is no assurance that it will be.
```

Next: C2D() | Prev: B2C() | Contents: Translation func.

1.13 ARexxGuide | Functions reference | Translation (3 of 8) | C2D

```
               rv = C2D(<string>, [<chars>])
     rv is a  number
```

In its simplest form, when <string> is one character, the function converts <string> to its ASCII value expressed as a decimal number.

```
   Examples:
         say c2d('b');           >>> 98
         say c2d(0)              >>> 49
```

The function will accept a <string> of up to four characters (four bytes) in length. When multiple characters are supplied, the function treats each character as a binary number, concatenates the result (see
c2b()
for an
example) and then returns the decimal equivalent of concatenated number.

```
   Example:
         say c2d('FFS')          >>> 4605523
```

The second argument, which must be a number from 1 to 4, allows a string shorter than that supplied by the first argument to be evaluated. The string is truncated from the right or padded with nulls to the number of characters specified.

```
   Examples:
   /**/
         say c2d('miga')         >>> 1835624289
         say c2d('Amiga', 4)     >>> 1835624289
         say c2d('a')            >>> 97
         say c2d('Amiga', 1)     >>> 97
```

```
   Also see
               C2B

               D2C
                               Technique note:  Determine library version number
```

Next: C2X() | Prev: C2B() | Contents: Translation func.

1.14 ARexxGuide | Functions reference | Translation (4 of 8) | C2X

```
               rv = C2X(<string>)
     rv is a  string  of  hex  digits and characters
```

In its simplest form, when <string> is one character, the function converts <string> to its ASCII value expressed as a hexadecimal number.

```
   Example:
         say c2x('b');           >>> 62
         say c2x('F') c2x('S');  >>> 46 53
```

```
         say c2x('FFS');          >>> 464653

When multiple characters are included in <string>, each character is
converted to its hexadecimal representation. The value returned is a
 concatenation  of each of those hex numbers. This is way the programmers
usually prefer to view values of a binary file.

   Also see
                X2C
               Next: D2C() | Prev: C2D() | Contents: Translation func.
```

1.15 ARexxGuide | Functions reference | Translation (5 of 8) | D2C

```
                rv = D2C(<whole number>, [<length>])
     rv is a  string

Converts a decimal <whole number> into a character string.

If <length> is supplied, the result will be truncated from the right or
padded with nulls to that size.

   Example:
         say d2c(98);               >>> b

   Also see
                C2D
               Next: D2X() | Prev: C2X() | Contents: Translation func.
```

1.16 ARexxGuide | Functions reference | Translation (6 of 8) | D2X

```
rv = D2X(<whole number>, [<length>])
     rv is a  string

Converts a decimal <whole number> into an equivalent hexadecimal string.

If <length> is supplied, the result will be truncated from the right or
padded with 0's to to that size. d2x(<number>, <trunc>) produces the same
result as  right(d2x(<number>), <trunc>, '0') .

   Example:
         say d2x(98);               >>> 62
         say d2x(464653);           >>> 7170D
         say d2x(464653,6);         >>> 07170D
         say d2x(464653,4);         >>> 170D

Next: X2C() | Prev: D2C() | Contents: Translation func.
```

1.17 ARexxGuide | Functions reference | Translation (7 of 8) | X2C

```
rv = X2C(<string>)
     rv is a  string

Converts a string of hexadecimal digits to their ASCII character
representation.

<string> must be an  expression  that evaluates to a valid hex number -- a
string of digits and/or the characters {a} through {f} or {A} through {F}.
It will be padded, if necessary, with a leading 0 to produce an even
number of characters.

   Examples:
         say x2c(416D696761);      >>> Amiga
         say x2c(464653);          >>> FFS


Next: X2D() | Prev: D2X() | Contents: Translation func.
```

1.18 ARexxGuide | Functions reference | Translation (8 of 8) | X2D

```
rv = X2D(<string>, [<length>])
     rv is a  string

Converts a string of hexadecimal digits to a whole decimal number. The
setting of  NUMERIC DIGITS  determines the size of number that can be
returned without generating an error.

<string> must be an  expression  that evaluates to a valid hex number -- a
string of digits and/or the characters 'a' through 'f' or 'A' through 'F'.
It will be padded, if necessary, with a leading 0 to produce an even
number of characters.

If <length> is specified, then <string> is either padded on the left with
0's or truncated to that length before the translation.

   Examples:
         say x2d(416D6967);        >>> 1097689447
         say x2d(464653);          >>> 4605523

Next: Translation func. | Prev: X2C() | Contents: Translation func.
```

1.19 ARexxGuide | Functions reference (5 of 12) | NUMBER MANIPULATION

```
 ABS
(<number>)

 HASH
(<string>)

 MAX
```

```
(<number>,<number> [,<number>,...])

 MIN
(<number>,<number>, [,<number>,...])

 RANDOM
([<min>],[<max>],[<seed>})

 RANDU
([<seed>])

 SIGN
(<number>)

 TRUNC
(<number>, [<places>])

     Also see  String manipulation functions
```

The functions take decimal digits as arguments. They perform some common alterations of the arguments, or return information about the number. The arguments sent to the function and the values returned are subject to alteration based on the setting of NUMERIC DIGITS . If a number is larger than the current DIGITS() setting, it will be converted to the current precision.

Next: Information func. | Prev: Translation func. | Contents: Function ref.

1.20 ARexxGuide | Functions reference | Number (1 of 9) | ABS

```
               rv = ABS(<number>)
     rv is a  number
```

Returns the absolute value of <number>. The result will not have a sign and will be formatted according to the current settings of NUMERIC .

```
   Examples:
         say abs(-100);               >>> 100
         say abs(10.5);               >>> 10.5
         say abs(-30);                >>> 30

   Also see
                SIGN
                           DATATYPE
```

Next: HASH() | Prev: Number functions | Contents: Number functions

1.21 ARexxGuide | Functions reference | Number (2 of 9) | HASH

```
rv = HASH(<string>)
     rv is a  number

Returns the hash attribute of a string as a decimal number.

A hash attribute is a number assigned to a string and is often used in
indexing schemes to provide a quick initial value for locating the string
within a range of data.

   Examples:
      say hash('AMIGA')        >>> 95
      say hash('Amiga')        >>> 223
      say hash('MAGIA')        >>> 95

The hash value is determined by adding the decimal ASCII values of each
character in the string and then performing a remainder-division (//) on
the result. Given a variable [Word], the following program would return
the same value as HASH(word):

   /**/
   Wtot = 0
   do i = 1 to length(Word)
      Wtot = Wtot + c2d(substr(Word,i,1))
   end
   return Wtot // 256

 Compatibility issues:
   This function is an extension that is not defined in  TRL2 . Although a
   function of this name might be included in other REXX implementations,
   there is no assurance that it will be.

Next: MAX() | Prev: ABS() | Contents: Number functions
```

1.22 ARexxGuide | Functions reference | Number (3 of 9) | MAX

```
                rv = MAX(<number>,<number> [,<number>,...])
     rv is a  number

The result is the largest of the <number>s in the supplied list. It is
returned in the format specified by the current  NUMERIC
settings.

   Examples:
      say max(3, 24/5, 2)                       >>> 4.8
      say max(length('pale'), length('gloom'))  >>> 5

   Also see
               MIN
                                 Technique note:  Extract file name from full spec

Next: MIN() | Prev: HASH() | Contents: Number functions
```

1.23 ARexxGuide | Functions reference | Number (4 of 9) | MIN

```
                rv = MIN(<number>,<number>, [,<number>,...])
     rv is a  number
```

The result is the smallest of the <number>s in the supplied list. It is returned in the format specified by the current NUMERIC settings.

```
   Examples:
   /**/
      say min(3, 24/5, 2)                        >>> 2
      say min(length('pale'), length('gloom'))  >>> 4
```

```
   Also see
                MAX
               Next: RANDOM() | Prev: MAX() | Contents: Number functions
```

1.24 ARexxGuide | Functions reference | Number (5 of 9) | RANDOM

```
                rv = RANDOM([<min>],[<max>],[<seed>})
  or
rv = RANDOM([<max>])
     rv is a  number
```

The result is a quasi-random non-negative whole number in the range <min> to <max> inclusive. The default for <min> is 0. The default for <max> is 999. If only one number is specified, it will be treated as the maximum value.

If a <seed> value (which must be an integer) is specified, it will begin a repeatable sequence of results.

```
   Examples:
         say random(10, 48)                            >>> 29  /* always */
         say random()                                  >>> 493 /* always */
         say random(10,48,506)                         >>> 31  /* always */
         say random(,, time(s))                        >>> 25  /* maybe  */
         call random(,, time(s));say random(10, 48)  >>> 34  /* maybe  */
```

```
   Also see
                RANDU
               Unless the function is seeded once within each script in which it ↩
                  is used,
```

it will always return the same values for a specified range of numbers. When a seed is specified, each call to random() will return a range of numbers that can be repeated exactly when the same seed value is used again. The

```
                TIME()
                function can provide a seed value that is itself
```

random enough to produce a more truly random set of numbers during subsequent calls to random() without a seed.

Next: RANDU() | Prev: MIN() | Contents: Number functions

1.25 ARexxGuide | Functions reference | Number (6 of 9) | RANDU

```
                rv = RANDU([<seed>])
     rv is a  number
```

The result is a quasi-random number between 0 and 1. The number of digits of precision is determined by the current setting of NUMERIC DIGITS.

If a <seed> value is specified, it will begin a repeatable sequence of results.

Examples:

```
      say randu(48)                       >>> 0.423783344  /* always */
      say randu()                         >>> 0.646834773  /* always */
      say randu(506)                      >>> 0.867625551  /* always */
      say randu(time(s))                  >>> 0.234561918  /* maybe  */
      call randu(time(s));say randu(48)   >>> 0.739330433  /* maybe  */
```

Also see

RANDOM

Compatibility issues:

This function is an extension that is not defined in TRL2 . Although a function of this name might be included in other REXX implementations, there is no assurance that it will be.

Next: SIGN() | Prev: RANDOM() | Contents: Number functions

1.26 ARexxGuide | Functions reference | Number (7 of 9) | SIGN

```
                rv = SIGN(<number>)
     rv is '-1', '0', or '1'
```

A result of '-1' indicates that the supplied number is less than 0. '1' indicates that it is greater than 0. A result of '0' is returned when <number> is 0.

Examples:

```
      say sign(45)      >>> 1
      say sign(-86);    >>> -1
```

Also see

ABS

DATATYPE

Next: TRUNC() | Prev: RANDU() | Contents: Number functions

1.27 ARexxGuide | Functions reference | Number (8 of 9) | TRUNC

```
rv = TRUNC(<number>, [<places>])
     rv is a  number

The result is the integer part of the supplied <number> formatted to
<places> decimal places. 0's are added if <number> did not have that many
decimal places.

If <places> is less than the number of decimals supplied, the fraction is
truncated without rounding.

The function truncates the number without rounding so trunc(6.19,2)
returns 6.1 rather than 6.2. The setting of  NUMERIC DIGITS  may be
manipulated to create a rounded number.

   Example:
         say trunc(10.5, 2)               >>> 10.50
         say trunc(6.7899, 3)             >>> 6.789
         say trunc(3, 4)                  >>> 3.0000
         say '$'right(trunc(25.7, 2),8)   >>> $   25.70
         say '$'right(trunc(125.4, 2),8)  >>> $  125.40


   Also see  SUBSTR
             RIGHT

                 Technique note:  Formatting tables

 Compatibility issues:
   Although this function follows the standard definition, there is a
   similar function defined in the standard that is not supported by
   ARexx: The format() function in standard REXX rounds and formats a
   number. Its simplest syntax is:

      format(<number>, <before>, <after>)

   A user function to provide those features for ARexx is described in the
   following note.

                 Technique note:  FORMAT(): A user function

Next: Number functions | Prev: SIGN() | Contents: Number functions
```

1.28 ARexxGuide | Functions reference (6 of 12) | INFORMATIONAL

```
                DATE
               ([<option>], [<date>, <format>])

                SHOW
               (<option>, [<name>], [<separator>])

                SHOWDIR
```

```
                (<directory>, ['ALL'|'FILE'|'DIR'], <separator>)

                 SHOWLIST
                (<option>, [<name>], [<separator>], ['A'])

                 TIME
                (<option>)

Related function:
    PRAGMA
    ADDRESS

                    Also see
                 File management functions
                Like the  SOURCE  and  VERSION  options to the  PARSE  instruction ↩
                   , these
functions give an ARexx script information about the system on which a
script is running and (since dates and times are important to most of us)
a bit of information about the world at large.

The DATE() function may also be used to translate dates from one format
into another.

Next: File I/O func. | Prev: Number functions | Contents: Function ref.
```

1.29 ARexxGuide | Functions reference | Informative (1 of 5) | DATE

```
                rv = DATE([<option>], [<date>, <format>])
     rv is a formatted  string
        or a  number

Without arguments, the result is the current system date.

The
                 <option>
                 argument (B|C|E|I|J|M|N|O|S|U|W) determines the
format of the result. It defaults to Normal format -- for example,
'20 Apr 1993'.

The second and third arguments provide information about other dates.
<date> must be entered in either  Sorted  or  Internal  format and
specified as the third argument (S|I). Unfortunately, ARexx will translate
only dates from January 1, 1978 onward. If an earlier date is passed as an
argument, the function will trigger  Error 18 .

   Examples:
         say date();                    >>> 20 Apr 1993
         say date(w);                   >>> Tuesday
         say date(i,'19930419',s);      >>> 5587
         say date(n,'5587',i);          >>> 19 Apr 1993
         say date(w,'19991231',s);      >>> Friday

   Also see
                 TIME
```

More information:
Persistence of DATE() value
Compatibility issues :
Although the definition may be extended in the future to recognize translation features similar (but probably not identical) to those now supported in ARexx, the current REXX standard recognizes only the first of the arguments supported by ARexx. Use of the other arguments would generate an error in other versions of REXX.

Next: SHOW() | Prev: Information func. | Contents: Information func.

1.30 ARexxGuide | Tutorials | Techniques (of) Persistence of DATE() and TIME() settings

The
DATE()
and
TIME()
settings are persistent within a single clause . A record is made of the initial value of both functions when either of them is first used in a clause. Thereafter, each call within the clause to one of the functions will return the initial value recorded at the first call.

The following, entered as three distinct clauses will return a different value for time() because of the delay() between the clauses:

```
say time();call delay(100);say time()
                                        >>> 11:45:29
                                        >>> 11:45:31
```

When the function calls are combined into a single clause, however, the value of the first call is returned on both calls to time():

```
say time() delay(100) time()
                                        >>> 11:45:43 0 11:45:43
```

A call to either date() or time() will freeze the values returned by both functions:

```
say time();say date() delay(100) time()
                                        >>> 11:54:03
                                        >>> 02 Nov 1993 0 11:54:03
```

This persistence guarantees that calls to the functions will return a consistent value within a single clause.

Next: TIME() | Prev: DATE() | Contents: Information func.

1.31 ARexxGuide | Functions reference | Informative | DATE (1 of 1) | OPTIONS

These are the <option>s recognized by the
date
function, they are:
All of these options can be shortened to the first character.

Option	Information returned
Normal	the date in the form dd MMM yy e.g. 09 Mar 1993
Ordered	the date in the form YY/MM/DD e.g. 93/09/03
Sorted	the date in the form YYYYMMDD e.g. 19930903
European	the date in the form DD/MM/YY e.g. 03/09/93
USA	the date in the form MM/DD/YY e.g. 09/03/93
Basedate	number of days since January 1, 0001
Julian	the date in the form YYDDD e.g. 91246
Century	number of days since January 1 of the current century
Days	number of days since January 1 of the current year
Internal	the date in internal system days e.g. 4993
Month	the current month in mixed case e.g. September
Weekday	the day of the week, mixed case e.g. Tuesday

Next, Prev. & Contents: DATE()

1.32 ARexxGuide | Functions reference | Informative (2 of 5) | SHOW

rv = SHOW(<option>, [<name>], [<separator>])
rv is a string
or a Boolean value

Returns a list of ARexx resources matching the specified <option>:

Option	Displays
Clips	The names of clips created by SETCLIP() or RXSET .
File	The logical names of files created with OPEN() , and the names of standard I/O files .

```
   Libraries     The names on the ARexx  Library List , added by  ADDLIB()
   Ports         The names of all system  message ports .

Only the first character of the <option> keyword need be used.

An optional <separator> can be used to divide the resource names, some of
which may have embedded blanks. (Be sure to use two commas before the
<separator>.)

If <name> is specified, the function will check for the existence of that
resource and return a Boolean success flag.

   Example:
         say show(p);            >>> REXX AREXX ConClip.rendezvous blanker
         say show(c);            >>> Molloy
         say show(c,'Molloy');   >>> 1
         say show(p,,'0a'x);     >>> REXX
                                     AREXX
                                     ConClip.rendezvous
                                     blanker

If 'L' is specified as the argument, a list of currently available ARexx
function libraries is returned. The support library function
                 SHOWLIST()
                ,
on the other hand, returns a list of all system libraries available when
the same argument is used.

SHOW('P') will return a list of all public message ports available on the
system. Some of those ports cannot be used as  hosts  for  commands  from
ARexx.

   Also see
                 SHOWLIST
                                  Technique note:  Output text to printer
                                   Using the clip list

 Compatibility issues:
   This function is an ARexx extension that is not supported and not
   duplicated in the standard language definition.

Next: SHOWDIR() | Prev: DATE() | Contents: Information func.
```

1.33 ARexxGuide | Functions reference | Informative (3 of 5) | SHOWDIR

```
                                                      a rexxsupport.library  ↩
                                                         function
rv = SHOWDIR(<directory>, ['ALL'|'FILE'|'DIR'], <separator>)
     rv is a  string

The result is a list of files matching the type specified by the second
argument and located in the <directory> specified.

The <separator> can be any character (including a null). It can be used to
```

```
separate filenames with a character (such as '0a'x) that cannot be used in
a filename.

   Examples:
         say showdir('sys:rexxc');
                  >>> HI RX RXC RXLIB RXSET TCC TCO TE TS WaitForPort

   Also see
                SHOWLIST

                STATEF
                           PRAGMA

The function library  rexxarplib.library , which is available on many
networks and bulletin boards, includes a function, FILELIST(), that is more
versatile since it will list only those files matching a specified pattern.

 Compatibility issues:
   All support functions are system specific.

Next: SHOWLIST() | Prev: SHOW() | Contents: Information func.
```

1.34 ARexxGuide | Functions reference | Informative (4 of 5) | SHOWLIST

```
                                                      a rexxsupport.library  ↩
                                                         function
rv = SHOWLIST(<option>, [<name>], [<separator>], ['A'])
     rv is a  string
        or a  Boolean value

Returns a list of system resources matching the specified
                 <option>
                and separated by the optional <separator> character.

If <name> is specified, the function will check for the existence of that
resource and return a Boolean success flag.

The <separator> can be any character, including '0a'x, which is a
convenient way to separate names in the list.

The optional fourth argument 'Address' or 'A' specifies that the function
is to return the base address of the named node, and is valid only if a
node name (second argument) has been supplied. The 'Address' option is
valid for both EXEC and DOS lists.

   Examples:
         say showlist(L);  >>> utility.library graphics.library keymap.library ...
         say showlist(L,'asl.library');      >>> 1
         say showlist(M);           >>> expansion memory chip memory
         say showlist(M,,'+');      >>> expansion memory+chip memory
         say showlist(D);  >>> gameport.device timer.device keyboard.device ...
         say showlist(V,,'|');      >>> XFER|WK|RAM DISK|HD1|HD0
         say showlist(R);  >>> potgo.resource ciaa.resource ciab.resource ...
         say showlist(A,'FONTS');   >>> 1
```

```
         say c2d(showlist(L ,'amigaguide.library',,A))
                                  >>> 5153220       /* for example */

   Also see
                SHOW

                SHOWDIR
                            PRAGMA

                Technique note:  Determine library version number

 Compatibility issues:
   All support functions are system specific.

Next: TIME() | Prev: SHOWDIR() | Contents: Information Func.
```

1.35 ARexxGuide | Functions reference | Informative | showlist (1 of 1) | OPTIONS

Any <option> to SHOWLIST() (some of them esoteric) may be specified by using only the single character capitalized in the list below:

Option	Information returned
Ports	Same information as SHOW('P'): all named message ports.
Libraries	All system libraries, not just ARexx libs.
Volumes	The volume names of all disks currently available.
Assign	The names of all assigned directories.
Handlers	The AmigaDOS interfaces to hardware devices. Includes names like DF0, PRT, CON.
Devices	The lower-level interface to hardware. Might include names like 'scsi.device', 'keyboard.device'.
Resources	The lowest-level software interface to some hardware elements of the machine. The resources cannot be accessed from ARexx, but this option returns names like 'potgo.resource'.
Memory-types	Will usually return 'expansion memory chip memory'.
Waiting	A list of all the many tasks waiting for something to happen on the system.
Task-ready	A list of tasks ready to be called to task by the scheduler.
Semaphores	Used by some software to prevent conflicting access to facilities it controls. (Since AmigaGuide uses semaphores, there may be an item on the list for this application.)
Interrupts	A list of node names on the list of interrupts in the Exec Library structure.

<option> can, reportedly, be given as the (4-byte) absolute address of a list header; the function performs several tests to make sure that it really is a header.

Next, Prev. & Contents: SHOWLIST()

1.36 ARexxGuide | Functions reference | Informative (5 of 5) | TIME

```
                rv = TIME(<option>)
     rv is a formatted  string
        or a  number

Without arguments, the result is the current system time in Normal 24-hour
clock format -- hh:mm:ss

The
                 <option>
                 argument (C|E|H|M|N|R|S) determines the format
of the result and controls the
                 elapsed time counter
                .

   Examples:
         say time()              >>> 20:08:52
         say time(c)             >>> 8:08PM
         say time(h)             >>> 20
         say time(m)             >>> 1208
         say time(s)             >>> 72532
         call time(r);call delay 500;say time(e)  >>> 10.06

   Also see
                 DATE
                             DELAY

               More information:
                 Persistence of TIME() value
                Next: Information func. | Prev: SHOWLIST() | Contents: Information ↩
                    func.
```

1.37 ARexxGuide | Tutorials | Techniques (of) The elapsed time counter

```
               The 'E' and 'R' options to
                TIME()
                control a clock that allows an ARexx
script to measure time intervals. The clock is started with the first call
to either TIME(E) or TIME(R). The result of the first call will always be
'0.00'. The next call to TIME(E) will report the interval in the form
<ss.tt> where <s> is seconds and <t> is ticks of the internal clock (1/50
second on NTSC systems).

TIME(R) will reset the interval counter to 0.00.

Changes to the interval counter made within a  subroutine  are local to
that subroutine and do not affect the settings of the clock in the calling
environment.

Next, Prev, & Contents: TIME()
```

1.38 ARexxGuide | Functions reference | Informative | TIME (1 of 1) | OPTIONS

The following <option> keywords are available for the TIME():

Keyword	Description
Civil	Current time in civil format: hh:mm[AM|PM]
Elapsed	Elapsed time in seconds
Hours	Current time in hours since midnight
Minutes	Current time in minutes since midnight
Normal	Default 24-hour format: hh:mm:ss
Reset	Reset the elapsed-time clock
Seconds	Current time in seconds since midnight

Only the first letter of the option need be used.

Next, Prev. & Contents: TIME()

1.39 ARexxGuide | Functions reference (8 of 12) | FILE MANAGEMENT

```
 DELETE
(<filespec>)

 EXISTS
(<filespec>)

 MAKEDIR
(<dirname>)

 RENAME
(<oldfile>, <newfile>)

 STATEF
(<filespec>)
```

Also see File input/output functions

Although each of these functions could be replaced by calls to AmigaDOS commands such as 'address command "delete" <file>', the functions here are significantly quicker more informative since they return a value within variable space of the calling script.

Next: ARexx control func. | Prev: File I/O func. | Contents: Function ref.

1.40 ARexxGuide | Functions reference | File Mgt. (1 of 5) | DELETE

a rexxsupport.library function

```
rv = DELETE(<filespec>)
     rv is a  Boolean value
```

```
Deletes the file specified by <filespec>. Returns 1 if the file was found
and successfully deleted.

   Example:
         say delete('t:tempfile');    >>> 1 /* if the file was found */

                  Technique note:  Getting output from a command
                                   Get/set environmental variables

 Compatibility issues:
   All support functions are system specific.

Next: EXISTS() | Prev: File mgt. func. | Contents: File mgt. func.
```

1.41 ARexxGuide | Functions reference | File Mgt. (2 of 5) | EXISTS

```
                                                      a rexxsupport.library  ↩
                                                         function
rv = EXISTS(<filespec>)
     rv is a  Boolean value

Checks the Amiga file system for the presence of a file named <filespec>,
which may include full path specifications. If only a partial path
specification is included, the search is made relative to the current
directory.

   Example:
         say exists('sys:system/rexxmast');      >>> 1

   Also see
                  SHOWLIST
                              PRAGMA

                  MAKEDIR
                                   Technique note:  Get/set environmental variables

Note:
                  SHOWLIST('A')
                  returns a list (in upper case and without the ':') of
all currently assigned directories. SHOWLIST('V') returns a similar list
of currently mounted volumes. The lists can be used to check for the
presence of a file device specification.

When EXISTS() is used to check for the existence of a file on a device
that might not be available, the system requester that asks "Please insert
volume..." can be suppressed through use of  PRAGMA('W', 'N') .

PRAGMA('D', <dir>) will change the default directory examined by EXISTS()
to that specified by <dir>.

 Compatibility issues:
   All support functions are system specific.

Next: MAKEDIR() | Prev: DELETE() | Contents: File mgt. func.
```

1.42 ARexxGuide | Functions reference | File Mgt. (3 of 5) | MAKEDIR

```
                                        a rexxsupport.library function
rv = MAKEDIR(<dirname>)
     rv is a  Boolean value

Creates a new directory, like the AmigaDOS command of the same name.

This is one of the rare cases where an ARexx function works differently
with different versions of the Amiga operating system. Under AmigaDOS 1.3,
the function returns 1 (TRUE) even if the directory already exists, so the
call can be made to ensure that a directory exists. Under Release 2.04 and
higher, however, the return value is 0 (FALSE) if the directory already
exists.

A return of FALSE might also occur under any version of the OS if the
specified volume is not available or is full.

   Example:
         say makedir('env:ARexxGuide')    >>> 1

                 Technique note:  Get/set environmental variables

 Compatibility issues:
   All support functions are system specific.

Next: RENAME() | Prev: EXISTS() | Contents: File mgt. func.
```

1.43 ARexxGuide | Functions reference | File Mgt. (4 of 5) | RENAME

```
                                        a rexxsupport.library function
rv = RENAME(<oldfile>, <newfile>)
     rv is a  Boolean value

Renames <oldfile> to <newfile>.

 Compatibility issues:
   All support functions are system specific.

Next: STATEF() | Prev: MAKEDIR() | Contents: File mgt. func.
```

1.44 ARexxGuide | Functions reference | File Mgt. (5 of 5) | STATEF

```
                                                    a rexxsupport.library  ↩
                                                       function
rv = STATEF(<filespec>)
     rv is a  string
```

```
Returns information about the file named <filespec>. The status string for
a file is formatted as

   FILE|DIR <bytes> <blocks> <protect-flags> <days> <min> <ticks> <comment>

   <protect-flags> are reported in the order HSPARWED with a dash "-" if
      the attribute isn't present.
   <days> is the number of days since January 1, 1978
   <min> is the number of minutes since midnight
   <ticks> is the number of tick intervals (1/50 second) in the minute.

   Examples:
         say statef('sys:rexxc');     >>> DIR 0 0 ----RWED 5362 727 2702
         say statef('sys:rexxc/tco'); >>> FILE 364 1 --P-RWED 5362 727 2688

   Also see
                 SHOWDIR
                              PRAGMA

 Compatibility issues:
   All support functions are system specific.

Next: File mgt. func. | Prev: RENAME() | Contents: File mgt. func.
```