THE AREXX INTERVIEW

An AW correspondent pumps his subject for answers to ten common questions about ARexx.

#1 ***AmigaWorld:*** *What the heck is ARexx, and why would I need it, anyway?*

ARexx: ARexx is a language—like BASIC or C or Pascal—that is used to write programs that control your computer. The programs are simple text files, editable by any text editor or word processor, which are then "interpreted" in real time as they are run. The Amiga version of the REXX language is a commercial product sold (and programmed) by William S. Hawes. Commodore is including ARexx free with the 2.04 and later releases of the operating system. If you buy it separately, it sells in the mail-order houses for less than $30, and (in my humble opinion) is well worth the money.

Let me try to give you the short answer on why you need it.

- First of all, it replaces the AmigaDOS script language. ARexx can do everything AmigaDOS can do (in fact, it can even use AmigaDOS to do it), and it is at least as easy to work with, but much more flexible and powerful.

- It replaces BASIC for those little "I've got a job to do, and I want it programmed quick" kind of problems, and through the use of external libraries it can take over for most of the "bigger" tasks you might previously have done using BASIC.

- It acts as a go-between to let other programs exchange information—program A sending data through ARexx to program B, and maybe back again. This interprocess communication makes possible the automation of all sorts of things, and it will become more and more useful and important as more and more programs become "ARexx-compatible."

#2 ***AW:*** *"Function Not Found!" errors! Explain this library business, okay?*

AR: This can be a confusing area. Some libraries come with ►

BY RICHARD STOCKTON

ILLUSTRATED BY GWYN STRAMLER

The best advice I can give you for adding new libraries is "read the docs" for each one.

ARexx, some don't. Some should be loaded by you, and some definitely should not. ARexx comes with two libraries standard: the rexxsyslib.library and the rexxsupport.library. The first is loaded automatically by the "rexxmast" program that runs in your startup-sequence, but the support lib needs to be loaded by you before the functions in it can be called by ARexx. Functions in the rexxsupport.library include SHOWDIR(), SHOWLIST(), STATEF(), OPENPORT(), CLOSEPORT(), and the message-handling routines like WAITPKT() and GETPKT().

As you can see, there is a lot of good stuff in there, so this library is often needed. The mere fact that it exists in libs: is not enough; you have to tell ARexx that you want to make it available for use. You can do this in your startup-sequence by adding a line such as

```
rx "CALL ADDLIB('rexxsupport.library',0,-30,0)"

*** DO NOT load the 'rexxsyslib.library'! ***
```

Among the freely distributable ARexx-compatible libraries, the rexxarplib.library, written by Willy Langeveld, is probably the most widely used. It adds screens, windows, gadgets, and graphics to the ARexx arsenal of commands. This is how you can quickly get a nice Graphic User Inferface up and running with a minimum of fuss.

If you use the rexxarplib (and lots of demo programs do), the rexxarplib.library (and possibly the rexxmathlib.library) needs to be loaded, but the arp.library and the screenshare.library, although required for the rexxarplib to work, should not be loaded. (They are loaded automatically when needed.) To accomplish this, add the following:

```
rx "CALL ADDLIB('rexxmathlib.library',0,-30,0)"
rx "CALL ADDLIB('rexxarplib.library',0,-30,0)"

*** DO NOT load the 'arp.library' or the 'screenshare.library'! ***
```

There are other ARexx-compatible libraries available, which brings up one of the most powerful features of ARexx: its ability to access commands from libraries or even other running programs. The best advice I can give you for adding new libraries is "read the docs" for each one. The author should tell you if it needs to be loaded and, if you're lucky, include an example of the preferred method.

#3 ***AW:*** *How can I find out if my script is being used with the new 2.0 OS?*

AR: Try out the following routine:

```
a = SHOWLIST('L','exec.library',,'A')       /* get library address */

b = OFFSET(a,20)                                /* add 20 to address */

c = IMPORT(b,2)                    /* get 2 chars from that address */

d = C2D(c)                  /* and turn them into a decimal number */

IF d >= 37 THEN SAY 'WorkBench 2.x version V'd
ELSE SAY 'WorkBench 1.3 (or less) version V'd
```

#4 ***AW:*** *I've heard you say that routines could be written that would somehow "TRAP" errors, and show me the offending code. Show me!*

AR: The followimg will automatically display the line in your code that ARexx is having a problem with, as well as the lines on either side of the probable error line. The internal ARexx "Special" variable RC will hold the error number and SIGL will hold the line number where the signal occurred. First, turn on the required feature by putting this line near the top of your ARexx program:

```
SIGNAL ON SYNTAX
```

Then, at the bottom of your program add:

```
EXIT        /* we don't want to accidentally execute this routine */
SYNTAX:
          /* ":" denotes a "label", can be CALLed or SIGNALed */
SAY 'SYNTAX ERROR:' RC ERRORTEXT(RC)
                                  /* built-in English messages */
SAY SOURCELINE(SIGL-1)
SAY SOURCELINE(SIGL)
                         /* <<== this line SHOULD be the error */
SAY SOURCELINE(SIGL+1)
EXIT
```

If you replace the EXIT with RETURN, it will force the program to continue from the point of the error. This is dangerous in most circumstances, because you don't know what the syntax error might have been. There could be times when all you want to do is trap the signal (if the user presses BREAK_C, for example), so that the program can exit gracefully, saving data, and so forth. In that case, you might have the BREAK_C: routine RETURN rather than EXIT.

SIGNAL ON SYNTAX is usually the most valuable debugging tool, but you can "SIGNAL ON" a whole variety of conditions including BREAK_C thru BREAK_F, ERROR, FAILURE, HALT, IOERR, and NOVALUE.

#5 ***AW:*** *How do you change the current directory "inside" an ARexx program?*

AR: It's easy—simply use the following:

```
old_dir = PRAGMA('D',new_dir)
```

All internal ARexx commands, like OPEN(), RENAME(), DELETE(), and so on, will now be pointed at new_dir as their current directory.

If you are using ADDRESS COMMAND to launch a CLI-based program, however, it is a bit trickier, since the Shell or CLI that ADDRESS COMMAND uses may or may not know the current directory, depending on your environment. The safest thing to do, the thing that works every time in any ARexx environment, is to set the directory as the first command and connect it to the commands that follow, using the linefeed character, ASCII character 10, hex 0A. In this way, all of these commands will be executed as if they had been typed, one line per linefeed character, at a single CLI window. Note that ADDRESS COMMAND sends the final linefeed automatically.

You can put an implicit "CD" into the ADDRESS COMMAND by doing this:

```
LF = '0A'x
ADDRESS COMMAND 'CD DF0:' LF 'DIR' LF 'CD RAM:' LF 'DIR'
```

This will display a directory listing of DF0: and then one of RAM:

#6 ***AW:*** *How can one send a text string directly to the printer? In BASIC, you'd simply LPRINT "Hello World." How would you do it in ARexx?*

AR: Well, here are three ways. First, you could simply enter

```
ADDRESS COMMAND 'Type >PRT: Hello World'
```

but that method is somewhat limiting, and is slow if you need to send more than a single line of text. Here's what I normally do:

```
x = OPEN(p,'PRT:','W')      /* using the printer as if it were a file */
IF x == 0 THEN EXIT 20
CALL WRITELN(p,'Hello World')                     /* write one line */
CALL CLOSE(p)  /* close the printer so the next task can use it */
```

Or, if you put your text into a stem (let's call it "text.") and keep the total number of lines in text.0, you can use this more general routine:

```
text. = ''                              /* blank our sample text file */
text.0 = 3                                               /* just 3 lines */
text.1 = 'This is line 1'
text.2 = 'line 2'
text.3 = 'now is the time for all good men to...'

x = OPEN(p,'PRT:','W')
IF x == 0 THEN EXIT 20
DO i = 1 TO text.0
  CALL WRITELN(p,text.i)
END
CALL CLOSE(p)
```

Remember that WRITELN() appends a linefeed character (ASCII 10) to the text sent automatically, but WRITECH() (not used here) does not.

#7 ***AW:*** *I need to format commas into numbers like 980432009 to make them more readable. Is there a command to do this?*

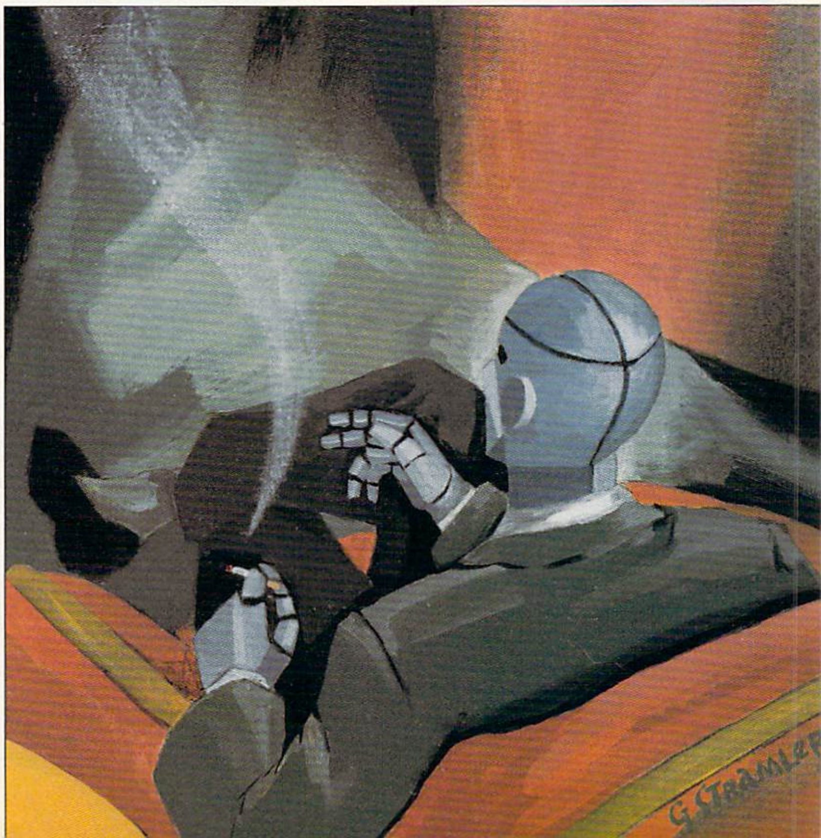
AR: ARexx does not have a built-in command to do this, but here is a subroutine you can use that will handle up to 15 digits—large enough to show the national debt! It also remembers any digits to the right of the decimal point, passing them back unchanged.

```
comma: NUMERIC DIGITS 14
                  /* largest number of digits ARexx can handle */
ARG num .
mant = num - num%1
                  /* anything to the right of the decimal point?*/
IF mant = 0 THEN mant = ''
ELSE
  DO
    mant = SUBSTR(mant,2)
                              /* if yes, then remove leading "0" */
    num = num % 1                   /* make num into an integer */
  END
dgt = LENGTH(num)                   /* number of digits in num */
numtext = ''
IF dgt>3 THEN numtext = ','RIGHT(num,3)
IF dgt>6 THEN numtext = ','LEFT(RIGHT(num,6),3)||numtext
IF dgt>9 THEN numtext = ','LEFT(RIGHT(num,9),3)||numtext
IF dgt>12 THEN
  DO
    numtext= ','LEFT(RIGHT(num,12),3)||numtext
    numtext = LEFT(num,dgt-12)||numtext
  END
ELSE IF dgt>9 THEN numtext = LEFT(num,dgt-9)||numtext
ELSE IF dgt>6 THEN numtext = LEFT(num,dgt-6)||numtext
ELSE IF dgt>3 THEN numtext = LEFT(num,dgt-3)||numtext
ELSE numtext = num numtext = numtext||mant
                              /* replace any decimal point data */

RETURN(numtext)
```

#8 ***AW:*** *Could you give me an example of a way to wait for a disk to be inserted into a certain drive and then report the volume name of the disk that was inserted—with no action* ►

...but a subroutine ... will handle up to 15 digits—large enough to show the national debt!



performed on the part of the user except to insert the disk?

AR: Let's say the "certain" drive is DF0:. Now try this:

```
CALL PRAGMA('W','N')              /* turn off disk requesters */
DO loop = 1
  IF EXISTS('DF0:') THEN
    DO
      olddir = PRAGMA('D','DF0:')
                                /* make df0: the current directory */
      SAY 'The disk in DF0: is called' PRAGMA('D')
      LEAVE loop
    END
  CALL DELAY(100)                          /* wait 2 seconds */
END
```

This will tell you the name of the disk in DF0: If there is no disk in the drive, it will keep checking every two seconds until a disk is inserted.

#9 *AW: Are there any good REXX books specific to the Amiga?*

AR: Using ARexx on the Amiga, by Chris Zamara and Nick Sullivan (Abacus, 425pp., $34.95), includes a disk with examples. It covers all areas of ARexx well and assumes nothing about your programming knowledge. Included are numerous examples of using ARexx with programs. This is ARexx from the end user's viewpoint.

Amiga Programmer's Guide to ARexx, by Eric Giguere (Commodore, 218 pp., $20), also includes a disk with examples. This one, however, covers ARexx from the programmer's viewpoint: How to put an ARexx port in your program, standard commands, and so forth. At the moment, this book is available only from CATS (Commodore Applications & Technical Support). To order, write to: Commodore Business Machines, Department C, 1600 Wilson Drive, West Chester, PA 19380. Specify CATS Part Number: AREXX01 and add $3 for shipping and handling. CATS insists on check or money order—no plastic. Add sales tax if you live in Pennsylvania, Virginia, California, Tennessee, or Illinois.

#10 *AW: Where can I get answers to my ARexx questions?*

AR: I run an ARexx question-and-answer session on the first Saturday of every month at 6 PM PST on Portal (408/973-9111 to sign up). You can also leave electronic mail for me on Portal, BIX, or GEnie (I am rlstockton on all of these). I also run my own BBS at 206/744-1254. If you are among the modem-impaired, you can mail me your questions care of *AmigaWorld,* and I'll try to help.

Bill Hawes, the author of ARexx, is also available on BIX (whawes), where he moderates a public message conference (amiga.vendors/wishful) for support of his products. If I can't answer a question, this is where I usually turn for the "ultimate" answer. ■

Richard Stockton is one of the leading ARexx programmers in the country and a cofounder of Gramma Software, a publisher of ARexx-compatible software.