

ARBGI

COLLABORATORS

	<i>TITLE :</i> ARBGI		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 31, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ARBGI	1
1.1	ARexx For Beginners - Glossary	1
1.2	Glossary - AmigaDOS	2
1.3	Glossary - Arguments	3
1.4	Glossary - ASCII	3
1.5	Glossary - Assignment	4
1.6	Glossary - Boolean Values	5
1.7	ARexx For Beginners - Glossary - C.H.U.G.	6
1.8	Glossary - Clauses	6
1.9	Glossary - CLI and Shell	7
1.10	Glossary - Commands	8
1.11	Glossary - Comments	8
1.12	Glossary - Logical Devices	9
1.13	Glossary - Directory Utilities	10
1.14	Glossary - Directory Utilities	11
1.15	Glossary - Expressions	12
1.16	Glossary - Files	12
1.17	Glossary - Format of Instructions	13
1.18	Glossary - Functions	15
1.19	instructions	16
1.20	Glossary - Integer Numbers	16
1.21	Glossary - Inter Process Communication	17
1.22	Glossary - Keywords	17
1.23	Glossary - Labels	18
1.24	Glossary - Scientific & Engineering Notations	19
1.25	Glossary - Number Systems	20
1.26	Glossary - Number Systems - Decimal	21
1.27	Glossary - Number Systems - Binary	21
1.28	Glossary - Number Systems - Hexadecimal	22
1.29	Glossary - Operators	23

1.30 Glossary - Path	24
1.31 Glossary - Public Domain	25
1.32 Glossary - Program Elements	26
1.33 Glossary - Special Characters	26
1.34 Glossary - Statements	28
1.35 Glossary - Streams	28
1.36 Glossary - Strings	29
1.37 Glossary - Symbols	30
1.38 Glossary - Tokens	30
1.39 Glossary - Using The CLI or Shell	31
1.40 Glossary - Viruses	32

Chapter 1

ARBGI

1.1 ARexx For Beginners - Glossary

AREXX FOR BEGINNERS

GLOSSARY

BY FRANK BUNTON

COPYRIGHT © FRANK P. BUNTON 1995-1998

AmigaDOS

Arguments

ASCII

Assignments

Boolean Values

Commodore Hornsby User Group

Clauses

CLI and Shell

Commands

Comments

Console Window '*'

Directory Utilities

Expressions

Files

Format of Instructions

Functions

Instructions

Integer Numbers

Inter Process Communication

Keywords

Labels

Logical Devices

Number Notations

Number Systems

Operators

Path

Public Domain

Program Elements

Special Characters

Statements

Streams

Strings

Symbols

Tokens

Using The Shell & CLI

Viruses

=== End of Text ===

1.2 Glossary - AmigaDOS

AREXX FOR BEGINNERS - GLOSSARY

AMIGADOS

AmigaDOS is the term applied to Amiga's equivalent of MS-DOS that is used on IBM compatible computers.

DOS was originally an acronym for "Disk Operating System". Some say it should be "Disk Based Operating System" as it does a lot more than operate a disk and that it was really an operating system based (stored) on a disks. Some say it should be "Device Operating System". These days a lot of the operating system is hard coded into the Amiga's ROM chips so that it is no longer disk based. It is best to forget that DOS ever stood for anything.

AmigaDOS is not just a set of commands nor is it just a Shell or CLI window. It is the software side of the Amiga's complete operating system. It controls all the functions of the computer such as multitasking, peripheral control and disk filing systems.

To learn all about AmigaDOS you should get hold of a copy of my disk called AmigaDOS For Beginners.

=== End of Text ===

1.3 Glossary - Arguments

AREXX FOR BEGINNERS - GLOSSARY

ARGUMENTS

An ARGUMENT is a piece of information that is transferred:-

1. - FROM the CLI/Shell which starts the ARexx program TO the program
2. - FROM the ARexx program TO a function
3. - FROM the ARexx program TO another program with which it is communicating.

Examples of these three categories are shown below.

1. Passing information to the ARexx program when it is started up from Shell/CLI can be illustrated with:-

```
> RX MyProgram.rexx 10,'Fred Bloggs'
```

The items 10 and 'Fred Bloggs' are two arguments whose values can be picked by the program "MyProgram.rexx" and used during its operation.

2. Passing information from the ARexx program to a function can be illustrated with:-

```
SAY CENTRE('This is a Test',55)
```

The two items 'This is a Test' and 55 are the arguments that are passed from the program to the function CENTRE.

3. Passing information to another program can be illustrated with:-

```
ADDRESS 'COMMAND'  
'DIR' 'Disk:Directory/fileName'
```

After using the ADDRESS instruction to tell the program "AMIGADOS" that we are going to talk to it, we send the command "DIR" together with the argument 'Disk:Directory/fileName'.

The contents of the argument can be varied at the discretion of the programmer, within the guidelines of the syntax of the function or external program, to suit the way the program is meant to work. Compare this to a Sub Keyword which cannot be varied by the programmer.

Arguments could be considered as a special type of Expression.

=== End of Text ===

1.4 Glossary - ASCII

AREXX FOR BEGINNERS - GLOSSARY

ASCII

ASCII, pronounced "Askey", stands for American Standard Code for Information Interchange. Basically, it is a system of allocating a unique CODE number for each character such as 65 for "A", 66 for "B", etc. Most computers store files containing only text in standard ASCII codes thus making them easily transferrable.

Text files containing only standard ASCII codes can be read by programs called "Text Readers" which are capable of reading ASCII only files. A (poor) example of such a text reader is the "More" program provided on all workbench disks. There are much better ones in the Public Domain.

Programs such as Word Processors, Desk Top Publishers and Data Base Programs, although they contain a lot of straight text, also contain a lot of non ASCII codes that control the formatting of the document. For instance, the bold, underlining, tabulation, page layout, etc. in text, and the column and row information, formulae, etc. in data bases.

Some of these programs (word processors, data bases, etc.) will store the document as ASCII codes for text and other codes for formatting. In this case, a text reader program will show you readable text interspersed with strange characters in place of the formatting codes.

Other such programs will use some other method than ASCII codes to store all the data, i.e. formatting and text. In this case, a text reader program will not show you any text.

An ASCII file (in word processing or data base terminology) is one saved without all the special formatting codes used by the word processor to indicate bold, underlining and other special features or such as those used by the data base program to indicate column and row information, formulae, etc. Thus the ASCII file contains only the ASCII codes for the text itself.

=== End of Text ===

1.5 Glossary - Assignment

AREXX FOR BEGINNERS - GLOSSARY

ASSIGNMENT

Assignment is the process of giving a value to a Variable Symbol.

For example:-

```
Name = 'Joe Bloggs'
Age = 53
Score = Score + 10
```


The variable symbol "Name" has been assigned a value of "Joe Bloggs"

The variable symbol "Age" has been assigned a value of 53

The symbol "Score" has been assigned a new value equal to what it was before plus 10.

Assignments can also be made via the keyboard by using the PULL instruction.

=== End of Text ===

1.6 Glossary - Boolean Values

AREXX FOR BEGINNERS - GLOSSARY

BOOLEAN VALUES

The term BOOLEAN comes from the English mathematician George Boole (1815-1864) who developed a system for formulating logical statements symbolically.

Briefly, it works on the premise that all expressions can be evaluated as TRUE or FALSE and the values of 1 and 0 are assigned as the Boolean Value of the expression.

```
TRUE  = 1
FALSE = 0
```

Thus the expression:-

```
12 = 12
```

is TRUE so that if you SAY the expression you will get 1:-

```
SAY 12 = 12
--> 1
```

but the expression:-

```
'Me' = 'You'
```

is FALSE so that if you SAY the expression you will get 0:-

```
SAY 'Me' = 'You'
--> 0
```

Some functions can take on a Boolean value. For example, the function EXISTS() tests to see if a disk file exists so that:-

```
EXISTS('s:startup-sequence')
```

has a value of 1 if the file "s:startup-sequence" exists and 0 if it does not exists.

=== End of Text ===

1.7 AREXX For Beginners - Glossary - C.H.U.G.

AREXX FOR BEGINNERS - GLOSSARY

COMMODORE HORNSBY USER GROUP INC. (C.H.U.G.)

P.O. BOX 1578 HORNSBY NORTHGATE N.S.W. 2077 AUSTRALIA

C.H.U.G. is a club where owners of Commodore and Amiga computers can get together to help each other to better understanding their computers and to get the most possible out of them.

There are two subgroups - The Amiga Group (all models)
- The Commodore Group (64 and 128 machines)

Commodore PC (IBM compatible) computers are not supported

FACILITIES

- Club meetings twice a month
- SIG (Special Interest Group) meetings on demand
- Public Domain Software Library (free catalogue disk for members)
- Club Magazine 6 times per year
- Club Newsletter every month
- Magazine Library of commercial publications
- Help for Beginners and others with problems
- Club "shop" (at meetings only) for purchase of disks and some other goods

For more information about C.H.U.G., please write to:-

Commodore Hornsby User Group Inc.
P.O. Box 1578
Hornsby Northgate
N.S.W. 2077
AUSTRALIA

=== End of Text ===

1.8 Glossary - Clauses

AREXX FOR BEGINNERS - GLOSSARY

CLAUSES

A clause is the smallest piece of program coding that can be executed by ARExx without an error resulting. A clause is made up of a number of TOKENS. A clause could be compared, in plain English, to a sentence made up of a number of words.

There are a number of different types of clauses.

INSTRUCTION CLAUSE is one that contains one or more of the many ARExx INSTRUCTIONS and the expressions that the programmer uses with the instruction. For example, this line is an Instruction clause:-

```
IF Count = 10 THEN SAY 'The End'
```

COMMAND CLAUSE is one that sends a COMMAND to an external program. For example, this line is a command clause:-

```
'Dir DH0:System'
```

This clause cannot be recognised by ARExx as any of the other types of clauses so ARExx assumes it must be a command clause and therefore sends it off to the external program with which it is currently communicating. If ARExx were currently communicating with AmigaDOS then AmigaDOS would receive the above command line and show you the contents of the "System" directory on drive "DH0:".

LABEL CLAUSE is one that contains only a LABEL to mark the start of a piece of coding. For example, this is a label clause:-

```
Multiply:
```

ASSIGNMENT CLAUSE is one that is used to ASSIGN values to SYMBOLS. For example, this is an assignment clause:-

```
Score = Score + 10
```

NULL CLAUSE is one that does nothing. It may be one of:-

- A blank line inserted to make the program easier to read
- A COMMENT

For example, these three lines, including the blank line between the two comments, are all null clauses:-

```
/* A comment line is a null clause */
```

```
/* This is another one */
```

```
=== End of Text ===
```

1.9 Glossary - CLI and Shell

AREXX FOR BEGINNERS - GLOSSARY

CLI AND SHELL

CLI is an acronym for "Command Line Interface". It is a program that displays a window into which the user can "Interface" with the Amiga by typing "Lines" of "Commands".

Shell is the more recent version of CLI (Command Line Interface). Workbench 1.2 and earlier had only CLI. Workbench 1.3 had the old CLI plus the new CLI enhancements which they called "Shell". In wb1.3 CLI and Shell could both be used individually. From Workbench 2 onwards, the Shell enhancements were incorporated into the standard CLI so that they became one and the same thing.

=== End of Text ===

1.10 Glossary - Commands

AREXX FOR BEGINNERS - GLOSSARY

COMMANDS

In ARexx, a COMMAND is an order sent from ARexx to an external program with which ARexx is communicating. The command is part of the external program's own system. Apart from being able to be sent from ARexx to the program, a command is not really a part of the ARexx system at all.

Commands should not be confused with Instructions.

For example, in this line:-

```
ADDRESS 'AMIGAGUIDE.1' 'WINDOWTOFRONT'
```

the word "ADDRESS" is an ARexx instruction which tells the program "AmigaGuide" to use its own command "WINDOWTOFRONT" to bring its own window to the front of all the others.

=== End of Text ===

1.11 Glossary - Comments

AREXX FOR BEGINNERS - GLOSSARY

COMMENTS

A COMMENT is a part of a program that explains something about the program. It has no effect on the operation of the program as ARexx completely ignores comments.

A comment is started with:-

```
/*
```

and ended with:-

```
*/
```

For example:-

```
/* This is a comment */
```

See Article 2 Program Elements for more details.

```
=== End of Text ===
```

1.12 Glossary - Logical Devices

AREXX FOR BEGINNERS - GLOSSARY

LOGICAL DEVICES

Most people think of DEVICES as physical things such as disk drives and printers. These physical devices all have names ending with a colon (:) such as:-

```
df0: df1: dh0: Prt:
```

The colon is an indication to the Amiga that the text to the left of it is a device name or, if referring to the name of a disk, the colon is an indication of a volume (disk) name that is in, or can be put in, a drive. For example:-

```
Workbench2.1:
```

The Amiga can be set up to recognise logical devices . Such logical device names also have a colon at the end of them. For example, the following logical device names are some of those that are automatically set up when your Amiga is booted:-

Logical Device Name	Directory that the logical device is referencing
SYS:	root directory of the boot disk
Fonts:	fonts directory of the boot disk
Libs:	libs directory of the boot disk
C:	c directory of the boot disk
S:	s directory of the boot disk
L:	l directory of the boot disk
Devs:	devs directory of the boot disk

Assignments to logical device names can be made with the AmigaDOS command ASSIGN as follows:-

```
ASSIGN Name: Directory
```

where:-

- Name: is the name you are going to give the logical device
- Directory is the name of the directory including its path to which the "Name:" is to refer.

For example, to change the assigned "Fonts:" directory, you could use:-

```
ASSIGN Fonts: df1:Fonts
```

Now, when the system wants some fonts, it will no longer look in the fonts directory of the boot disk but, instead, it will look in the fonts directory of the disk in df1:

When using ARexx, the RX command needs an assigned logical device named "Rexx:" in which to look for its programs to run. This assignment is usually made with the line:-

```
ASSIGN Rexx: Sys:S
```

However, if you wanted your ARexx programs in a separate directory to all the other "S" directory files, then you could make a new directory called, say, ArexxS, and use:-

```
ASSIGN Rexx: Sys:ARexxS
```

and then, when RX looks in the logical device "Rexx:" it will look in your directory "Sys:ARexxS".

=== End of Text ===

1.13 Glossary - Directory Utilities

AREXX FOR BEGINNERS - GLOSSARY

CONSOLE WINDOW "*"

A Console is a window into which input (e.g. from the keyboard or disk drive) can be displayed and into which a user can enter data via the keyboard.

The input and output are referred to as Streams.

A Console Handler is the program that controls this window and its streams.

The current console can be referred to as "*". For example, if you enter into a Shell/CLI window:-

```
> COPY * Ram:MyFile
```

or

```
> COPY * Prt:
```

Then anything typed into the the Shell/CLI after that is sent to the file in Ram: called "MyFile" or to the printer respectively.

For a Shell/CLI window, the output is returned to that window by entering:-

```
CTRL\
```

This means:-

Hold down the CTRL key and press \

=== End of Text ===

1.14 Glossary - Directory Utilities

AREXX FOR BEGINNERS - GLOSSARY

DIRECTORY UTILITY

A Directory Utility is a program that lets you examine the contents of disks and directories by directory and file name rather than by workbench icons. A good Directory Utility will have two windows side by side in which are shown the contents of two different directories, either on the same or different disks.

The program allows you to see files not normally visible by icons.

It allows you to do a lot of AmigaDOS functions such as copying, deleting, renaming, viewing, etc. without having to learn all the CLI commands. You can, for instance, click on a file name in one window and then click on a button called COPY and the file will be copied from the directory represented in that window to the one represented in the other window.

Modern directory utilities such as "DirWork 2" and "Directory Opus" (both commercial products) have an extremely large range of capabilities and can be configured to suit you own purposes.

There are also quite a few in the Public Domain.

=== End of Text ===

1.15 Glossary - Expressions

AREXX FOR BEGINNERS - GLOSSARY

EXPRESSIONS

An expression is one or more TOKENS that are put together by the programmer at his/her discretion. I guess you could say that it is a statement (expression) made by the programmer as opposed to the rigidity of the KEYWORDS that must conform to the syntax of ARexx.

An expression can consist of these tokens:-

SYMBOLS
STRINGS
OPERATORS
PARENTHESES

In these examples, the KEYWORDS are in upper case and the expressions are in lower case and in bold:-

```
SAY 'This string is an expression'  
Count = Count + 1  
DO WHILE Number < 100  
IF Name = 'Q' THEN SAY 'Quitting Program'  
SAY (Count + Score) * 100
```

1st Line - SAY is the Keyword
- 'This string is an expression' is the expression

2nd Line - there is no Keyword
- Count = Count + 1 is the expression

3rd Line - DO and WHILE are Keywords
- Number < 100 is the expression

4th Line - IF, THEN and SAY are Keywords
- Name = 'Q' is the expression linked to IF ... THEN
- 'Quitting program' is the expression linked to THEN SAY

5th Line - SAY is the Keyword
- (Count + Score) * 100 is the expression

=== End of Text ===

1.16 Glossary - Files

AREXX FOR BEGINNERS - GLOSSARY

FILES

In general terms, a file is:-

- A collection of data that is:-
- organised into the one unit and,
- can be referenced by name.

The most common form of file is a disk file which can be a program, a letter, a picture, an icon, or anything else that can be saved to disk.

Other types of files are not quite so easy to understand but can still be referred to as files by an ARexx program.

A printer file is a collection of data sent to your printer. When you select Print from your word processor menu, it collects all the data in your document and sends it to the printer as one unit. At the end of the document, the word processor sends an End Of File signal to the printer.

In the same way, files can be sent over modems or can be sent to a console window.

=== End of Text ===

1.17 Glossary - Format of Instructions

AREXX FOR BEGINNERS - GLOSSARY

FORMAT OF INSTRUCTIONS

ARexx manuals all tend to use the FORMAT at the start of the instruction description.

This format has a number of items in it (see examples below):-

- KEYWORD - This is the instruction or function itself
 or one of its sub keywords.
 - SUB-KEYWORD - This is another keyword used in conjunction with the
 main keyword. It may be essential or optional.
 - KEYWORDS & - These are shown in UPPER CASE to distinguish them from
SUB-KEYWORDS arguments. However, their case can be upper or lower when
 used by the programmer. The spelling of keywords and
 sub-keywords must be exactly as shown.
 - ARGUMENTS - These are items that are inserted by the programmer to
 give ARexx some information about the way the instruction
 should work or the string or number on which the instruction
-

should work.

Arguments are usually shown in lower case to distinguish them from keywords.

The text of arguments is determined by the programmer and thus is not normally the word shown in the manuals. The word used is merely an indication to the programmer of the type of insert that should be made at that spot. For example "filename" might indicate that the name of a disk file should be inserted in lieu of "filename".

The case may or may not be important. This will have to be determined by the context of the instructions.

Whereas the NAME of the instruction or function (i.e. the MAIN keyword) is always necessary, other parts of the format (including SUBKEYWORDS) may be optional or essential. This is indicated by the use of these symbols:-

| the vertical bar is used to separate alternate selections of which only one can be used.

{ } braces (curly brackets) are used to enclose arguments or sub-keywords where it is essential that one of the alternatives separated by the | be included.

[] brackets are used to enclose arguments or sub-keywords that are optional. These may be single items or alternative items separated by the | symbol.

Items not enclosed by { } or [] are essential.

() parentheses are used to enclose the arguments used for functions

Examples

SELECT

Nothing follows the keyword so it is used on its own.

SAY [expression]

The "expression" is enclosed in [] so it is optional. It is in lower case so it is not a keyword but an argument the text of which is set by the programmer.

NUMERIC FORM {SCIENTIFIC|ENGINEERING}

The sub-keyword "FORM" is not enclosed in any form of brackets so it is essential.

The SCIENTIFIC|ENGINEERING is enclosed in braces and the two sub-keywords are separated by the vertical bar | so one and only one of them is essential.

OPEN(file,filename [, 'APPEND' | 'WRITE' | 'READ']

The parentheses () indicate this is a function, not an instruction. The "file" and "filename" are in lowercase so they are not keywords but arguments. Their actual text is therefore set by the programmer. The part in brackets [] is optional but if used, must be only one of the sub-keywords separated by the vertical bars.

=== End of Text ===

1.18 Glossary - Functions

AREXX FOR BEGINNERS - GLOSSARY

FUNCTIONS

A FUNCTION is a piece of programming code written for a special purpose, i.e. to carry out specific function. These pieces of coding can be invoked, or called into use, at any point of a program whenever that function is to be performed. After the function has completed its job the program returns to the point in the program from which the program was called.

There are three types of functions recognised by ARexx. They are:-

- Internal Functions
- Built In Functions
- Function Libraries

Internal Functions are ones that are written into the program's coding by you, the programmer. They could be considered as a "sub-program" within the main program and need to be called or invoked from the main program. Some programming languages call them "Sub Routines". Read Article 16 for more details.

Built In Functions are ones that are a part of the ARexx interpreter system and so are not part of the program. However, they can still be likened to "sub-programs" and can be called or invoked from within your own program. They are often distinguished from Instructions by the parentheses () that follow the function name. For example CENTRE(). Read Articles 17 for more details.

Function Libraries are files external to both the ARexx program and the ARexx interpreter system. A "Function Library" is a single file that is a collection of functions organised in the manner of "Amiga Shared Libraries"*. However, it must have been written in such a way as to make it suitable for use by ARexx. An example of a "Function Library" is the "rexsupport.library" file that contains 11 functions. Read Articles 39 for more details.

* An "Amiga Shared Library" is one of the many files stored in the "Libs" directory (or, in later Amigas, may be stored in ROM chips) that can be used by any program that calls upon it. They can be "shared" by all programs. Examples are:-

```
diskfont.library  
icon.library  
mathtrans.library
```

=== End of Text ===

1.19 instructions

AREXX FOR BEGINNERS - GLOSSARY

INSTRUCTIONS

An ARExx INSTRUCTION is one of the 29 Keywords that the ARExx interpreter recognises as an order to do something. In this statement for example:-

```
SAY "Peter Piper picked a peck of pickled peppers"
```

The word SAY is an instruction to ARExx to put the string that follows it into the display window so that the user can see it.

Instructions should not be confused with Commands.

For example, in this line:-

```
ADDRESS 'AMIGAGUIDE.1' 'WINDOWTOFRONT'
```

the word "ADDRESS" is an ARExx instruction which tells the program "AmigaGuide" to use its own command "WINDOWTOFRONT" to bring its own window to the front of all the others.

=== End of Text ===

1.20 Glossary - Integer Numbers

AREXX FOR BEGINNERS - GLOSSARY

INTEGER NUMBERS

An INTEGER number is, simply, a number without a fractional part.

These numbers are integers:-

```
1      -345      59000
```

These numbers are NOT integers:-

```
1.234      -0.456      1-1/2
```

An integer can be positive or negative.

Zero is also considered to be an integer number.

In some circumstances, the syntax of ARexx instructions or functions will require you to use only integer numbers and an error stoppage will result if a non integer number is used.

=== End of Text ===

1.21 Glossary - Inter Process Communication

AREXX FOR BEGINNERS - GLOSSARY

INTER PROCESS COMMUNICATION (IPC)

"Inter Process Communication", or "IPC" as it is frequently called, is the process by which a program (process) can send messages to, and receive messages from, another program.

To participate in IPC a program must have been written with a "Message Port" which is really just a small segment of the program's coding which allows it to "Listen" for messages sent to it and to send messages out to other programs.

The multitasking ability of the Amiga makes it ideally suited for IPC.

=== End of Text ===

1.22 Glossary - Keywords

AREXX FOR BEGINNERS - GLOSSARY

KEYWORDS

A KEYWORD is a special word that is recognised by the ARexx interpreter as being the name of an Instruction or function. In some contexts, the term KEYWORD can also include SUBKEYWORD.

A SUB KEYWORD is a special word that comes after a keyword and modifies the way in which the instruction will act.

Both keywords and sub keywords are contained within the coding of the ARExx interpreter and therefore their usage (spelling, etc.) cannot be varied by the programmer. Compare this with arguments the contents of which can be varied to suit the purposes of the program.

To make the distinction in these articles I will put:-

KEYWORDS in upper case

Arguments in lower case with, possibly, an upper case initial

For example, the keyword, or instruction DO can have these sub keywords used with it:-

TO BY FOR FOREVER WHILE UNTIL

Keywords and sub keywords can, in some cases, be used in conjunction with user defined arguments.

In the following examples the keywords are in bold upper case and the arguments are in non bold lower case:-

DO Count = 1 TO 20 BY 2

DO FOR 20

DO FOREVER

DO WHILE Count < 20

DO UNTIL Count = 20

=== End of Text ===

1.23 Glossary - Labels

AREXX FOR BEGINNERS - GLOSSARY

LABELS

A Label is a special type of Fixed Symbol that is followed by a colon (:).

For example:-

Multiply:

is the label "Multiply" which could be used to mark the section of the program that carries out multiplications.

A label can be used to mark the start of a section of the program coding for one of these purposes:-

- making the program easier to read,

- indicating the start of an Internal function.
- indicating the point in a program to which the SIGNAL instruction is to divert program flow.

=== End of Text ===

1.24 Glossary - Scientific & Engineering Notations

AREXX FOR BEGINNERS - GLOSSARY

SCIENTIFIC AND ENGINEERING NOTATIONS

These two methods of number notation have the format:-

mantissa E exponent

For example:-

4.567E+4

where:-

"mantissa" is a number between:-

1 and 9.99999999 for SCIENTIFIC Notation
1 and 999.99999999 for ENGINEERING Notation

"E" is the "Exponential" indicator

"exponent" is the power to which 10 is raised and is a positive or negative integer number

For SCIENTIFIC it can be any integer number
For ENGINEERING it is any integer number that is a multiple of three.

The "mantissa" is multiplied by 10 raised to the power of the "exponent".
For example:-

$4.567E+4 = 4.567 \text{ times } 10 \text{ to power } 4$
 $= 4.567 \text{ times } 10000$
 $= 45670$

or, if the exponent is negative:-

$4.567E-4 = 4.567 \text{ times } 10 \text{ to power } -4$
 $= 4.567 \text{ times } 0.0001$
 $= 0.0004567$

Note that the + or - sign used in the exponent is not a sign of addition or subtraction. It is a sign of positivity or negativity.

When does ARExx Use These Notations?

One of these two notations is used for very large or very small numbers as it makes it easier to display them. Any number resulting from a calculation that is:-

10000000000 or more i.e. 1E+9 or more

or

less than 0.00000000001 i.e. less than 1E-10

will appear in scientific or engineering notation.

AREXX uses SCIENTIFIC as the default format for very large or very small numbers.

If ENGINEERING is needed for these large or small numbers then the user must tell ARExx to use it with this instruction:-

NUMERIC FORM Engineering

then, if SCIENTIFIC is later required in the same program, revert to it with the instruction:-

NUMERIC FORM Scientific

=== End of Text ===

1.25 Glossary - Number Systems

AREXX FOR BEGINNERS - GLOSSARY

NUMBER SYSTEMS

There are at least three number systems used in the computer world. These are:-

Decimal System	- Base 10
Binary System	- Base 2
Hexadecimal System	- Base 16

Prefixes are sometimes used to distinguish between the systems:-

No prefix	- Decimal
Prefix %	- Binary
Prefix \$	- Hexadecimal

For example, the decimal number 13 can be represented by the other two systems as:-

%00001101

\$0D

=== End of Text ===

1.26 Glossary - Number Systems - Decimal

AREXX FOR BEGINNERS - GLOSSARY

NUMBER SYSTEMS - DECIMAL

To say a number system has a base ten means that there are 10 digits (0-9) that can be used before we start to use double digit numbers. This system has developed because humans have 10 fingers and primitive humans counted on their fingers. Maybe some of us still do! If the powers that be had created us with 6 fingers on each hand we would have a number system based on 12 digits!

=== End of Text ===

1.27 Glossary - Number Systems - Binary

AREXX FOR BEGINNERS - GLOSSARY

NUMBER SYSTEMS - BINARY

Computers, to create an analogy with humans, have only one finger on each hand, or a total of two digits. So computers use a Binary System with only the two digits of 0 and 1. This is really not a good analogy. A better one is to consider that a computer can be compared to millions of little switches which can only have two states - on and off:-

- When a switch is off it holds a value of 0
- When a switch is on it holds a value of 1

The switches are the bits of the computer memory. 8 bits make up one byte.

Thus a byte can have each of its switches on or off, i.e. holding a value of 0 or 1. The byte can therefore hold numbers of decimal value from 0 to 255. In the following example the bits are numbered from 0 (rightmost) to 7 (leftmost) and the binary numbers are represented by the 0's and 1's below the bit numbers. For example, decimal 5 is binary 00000101.

Bit number values	Decimal Number
7 6 5 4 3 2 1 0	
0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 1	1
0 0 0 0 0 0 1 0	2
0 0 0 0 0 0 1 1	3

0 0 0 0 0 1 0 0	4
0 0 0 0 0 1 0 1	5
0 0 0 0 0 1 1 0	6
0 0 0 0 0 1 1 1	7
0 0 0 0 1 0 0 0	8
.....	...
0 0 0 0 1 1 1 1	15
0 0 0 1 0 0 0 0	16
.....	...
.....	...
1 1 1 1 1 1 1 1	255

Thus the binary number, which is represented by the 0's and 1's in the above table, is directly related to the on and off condition of each bit of a byte

We can convert from decimal to binary and vice versa with the use of these functions:-

```
BinaryNumber = C2B(D2C(DecimalNumber))
```

```
DecimalNumber = C2D(B2C(BinaryNumber))
```

For example:-

```
SAY C2B(D2C(200))      --> 11001000
SAY C2D(B2C(11001000)) --> 200
```

=== End of Text ===

1.28 Glossary - Number Systems - Hexadecimal

AREXX FOR BEGINNERS - GLOSSARY

NUMBER SYSTEMS - HEXADECIMAL

The hexadecimal system is used because it is too hard for humans to work in the binary system and hexadecimal relates more closely to binary than does decimal.

Hexadecimal uses 16 digits for the decimal numbers 0 to 15. It does not start to get into double digit numbers until the decimal number 16. Because we only have 10 digits at our disposal, the letter A - F are used for the extra 6 digits and represent the decimal number 10 to 15 respectively.

It is common practice to always represent hexadecimal numbers with two digits, using a leading zero if necessary (as shown below).

A short table of hexadecimal numbers and their decimal and binary equivalents is as follows:-

Hexadecimal	Binary	Decimal
-------------	--------	---------

01	00000001	1
02	00000010	2
03	00000011	3
04	00000100	4
05	00000101	5
06	00000110	6
07	00000111	7
08	00001000	8
09	00001001	9
0A	00001010	10
0B	00001011	11
0C	00001100	12
0D	00001101	13
0E	00001110	14
0F	00001111	15
10	00010000	16
11	00010001	17
..
1F	00011111	31
20	00100000	32
..
3F	00111111	63
40	01000000	64
..
..
FF	11111111	255
100	100000000	256

We can convert from decimal to hexadecimal and vice versa with the use of these functions:-

Hexadecimal = D2X(Decimal)

Decimal = X2D(Hexadecimal)

For example:-

SAY D2X(7998) --> 1F3E
 SAY X2D(1F3E) --> 7998

=== End of Text ===

1.29 Glossary - Operators

AREXX FOR BEGINNERS - GLOSSARY

OPERATORS

An Operator is a token that can perform operations such as arithmetic and comparisons on symbols, numbers and strings.

They can be any one, or a combination of two, of these characters:-

~ % ^ & * - + = / | < >

There are four types of operators which are:-

ARITHMETIC OPERATORS	+	-	*	/	**	%	//			
COMPARISON OPERATORS	=	~=	==	~==	>	<	>=	<=	~>	~<
LOGICAL OPERATORS	~	&		^	&&					
CONCATENATION OPERATORS		Space	None							

=== End of Text ===

1.30 Glossary - Path

AREXX FOR BEGINNERS - GLOSSARY

PATH

The term Path refers to the device or disk name and directory (drawer) names for the specified file.

The name of the path consists of:-

* The name of a device or disk followed by a colon. For example:-

```
df1:
dh0:
Work:
My_Disk:
Libs:
```

* The name of the directory or directories (if any) in which the file is held. These names come after the colon. For example:-

```
df1:Programs
= The directory "Programs" on disk in df1:

Workbench2.1:Devs
= The directory "Devs" on the Workbench disk.
```

If there is a sub directory in a directory then the two names are separated by a slash (/). For example:-

```
df1:Programs/ARexxPrgs
= The sub directory "ARexxPrgs" in the "Programs" directory.

Workbench2.1:Devs/Printers
= The sub directory "Printers" in the "Devs" directory.
```

* The name of the file itself.

If it is in a directory then the file name and the directory name are separated by a slash. For example:-

```
df1:Programs/Phonebook.rexx
```

= The file "Phonebook.rexx" in the "Programs" directory.

Workbench2.1:Devs/Printers/EpsonQ

= The file "EpsonQ" in the "Printers" sub directory
of the "Devs" directory.

If the file is in the root directory of the disk then the file
name comes immediately after the colon. For example:-

dfl:database.rexx

= The file "database.rexx" in the root directory of the disk in
dfl:

Workbench2.1:System.info

= The file "System.info" in the root directory of the workbench
disk.

=== End of Text ===

1.31 Glossary - Public Domain

AREXX FOR BEGINNERS - GLOSSARY

PUBLIC DOMAIN

Public Domain

Public Domain (P.D.) is software that has no copyright on it. It can be legally copied and distributed freely by anybody to anybody. The author has donated the program to the public. Anyone is free to do what they like with it such as use it in their own programs or disks, modify it to enhance its abilities, etc.

Also loosely described as Public domain, but not really so, are the following categories of software. These categories are often included in what is commonly called Public Domain libraries.

Freeware

Freeware is similar to public domain in that it can be legally copied and distributed freely by private individuals to private individuals but it usually has a copyright on it and restrictions as to its usage.

It would be necessary to read the copyright notices that come with each individual program but some common restrictions on its usage are:-

- Copying fees must be nominal only, i.e. only enough to cover disks and copying costs. No one is allowed to profit from it.
 - Distribution can only be made on the proviso that all nominated files are distributed along with the program.
 - No modifications can be made to the program.
-

- No commercial usage or copying is allowed without the authors written permission.

Shareware

Shareware is copyright software with similar distribution restrictions as those mentioned above for Freeware.

It works under an honour system. Normally, it can be freely copied and used for a short trial period. If you intend to continue using it then, legally, you must pay the author the requested shareware fee. If you do not pay the shareware fee then you are supposed to discontinue using it.

Sometimes some key part of the program is missing which is provided when the fee is paid. Sometimes fee payment requests annoyingly pop up every few minutes. Other means of encouraging payment are also employed.

On payment of the fee you will often receive the latest version of the program which has the missing parts included and/or the pestering requesters deleted. You may also go on a list and be advised of future updates.

=== End of Text ===

1.32 Glossary - Program Elements

AREXX FOR BEGINNERS - GLOSSARY

PROGRAM ELEMENTS

An ARexx Program is made up of a number of elements which are:-

TOKENS
CLAUSES
EXPRESSIONS

In learning to program in ARexx, you will learn how to put these various elements together.

=== End of Text ===

1.33 Glossary - Special Characters

AREXX FOR BEGINNERS - GLOSSARY

SPECIAL CHARACTERS

One group in the definition of TOKEN are called, for want of something better, Special Characters.

They are:-

Colon :

Semicolon ;

Comma ,

Parentheses ()

A Colon is used to distinguish a Label (which is a special form of fixed symbol) from a variable Symbol. For example, this is a label:-

Multiply:

A Semicolon is used to separate clauses or statements that are on the same line. For example, if you had:-

```
x = 3
y = 4
SAY x + y
```

you could replace it with:-

```
x = 3 ; y = 4 ; SAY x + y
```

A Comma is used in situations where a single clause or statement cannot all fit on the one line in your text editor. By putting a comma at the end of one line we are telling ARExx that the following line is part of the same clause or statement. For example:-

```
Quotation = 'Now is the time for all good men to come to',
'the aid of their country'
```

SAY Quotation

As the first line ends in a comma, all of the quotation "Now is the time for all good men to come to the aid of their country" is considered as the one item and is assigned to the symbol "Quotation".

A Comma is also used as a separator for the Arguments that are used for Functions. For example:-

```
SAY COPIES('*',5)
```

In this statement the comma separates the two arguments enclosed in the parentheses.

The Parentheses are the () characters. They are used to indicate to ARExx

that a group of symbols or tokens etc. are to be taken as the one unit. They can be used after a function name (as in the SAY COPIES example above) or to indicate the order in which calculations will occur. For example, in this:-

```
5 * (3 + 4)
```

The parentheses tell ARexx to add 3 and 4 before multiplying by 5.

=== End of Text ===

1.34 Glossary - Statements

AREXX FOR BEGINNERS - GLOSSARY

STATEMENTS

A Statement can be any of:-

- Assignment
- Command
- Instruction

For example:-

Count = 10	is an assignment statement
SAY 'Hi There Jack'	is an instruction statement
ADDRESS 'COMMAND' ; 'Dir'	is an instruction statement followed by a command statement

=== End of Text ===

1.35 Glossary - Streams

AREXX FOR BEGINNERS - GLOSSARY

STREAMS

To put it simply for the beginner, a stream could be said to be a pathway through which data can flow.

ARexx uses three streams which are called:-

- STDIN
- STDOUT
- STDERR

The default streams for ARexx are:-

STDIN - the normal input source, i.e. the console window into which data is entered via the keyboard.

STDOUT - the normal output source, i.e. the console window to which data is sent for display.

STDERR - a special "Global Tracing Console" which can be opened by the command utility called TCO.

If an ARExx program is started from a Shell/CLI window, and if it does not open any other window for input or output, then it uses the same console window for both STDIN and STDOUT, i.e. the window used to start it from.

=== End of Text ===

1.36 Glossary - Strings

AREXX FOR BEGINNERS - GLOSSARY

STRINGS

To put it rather simply, a "string" is a single unit of data that contains one or more characters all of which have been enclosed within the one set of quotation marks, or string delimiters. For example:-

```
'This is a string'
```

A string is distinguished from a SYMBOL by the fact that the string is enclosed in delimiters (e.g. quotation marks) whereas a symbol is not enclosed in delimiters.

A String Delimiter is a character that sets the limits of a string, i.e. it marks the start and end of the string.

The delimiter characters in ARExx are:-

```
Single opening quote -  '  
Double quote -        '"
```

No other characters, including the closing single quote `'`, can be used.

Each string must use only one type of delimiter so that :-

```
'This is a string'  is legitimate  
"This is a string"  is legitimate  
'This is a string"  is illegitimate  
"This is a string'  is illegitimate
```

Read Article 5 for more information about string delimiters.

A string can contain any of the characters that can be entered via the

keyboard (as well as some that can't!!). For example, all these are strings:-

```
'A String is not a Symbol'  
'!@#$%^&*()'  
' A1 b2 C3 '  
' 12 + 3 is Fifteen'  
' : ; / ? . < '
```

Note that strings can contain the space character so that a single strings can be made up of a number of words separated by spaces.

=== End of Text ===

1.37 Glossary - Symbols

AREXX FOR BEGINNERS - GLOSSARY

SYMBOLS

A SYMBOL is a token that can hold a value.

A Fixed symbol is one whose value does not change during the course of the program.

A Variable symbol is one whose value can change during the course of a program by having assignments of values made to it.

A symbol is usually distinguished from a String by the fact that it is not enclosed within quotation marks.

See Article 7 Symbol Introduction for more details.

=== End of Text ===

1.38 Glossary - Tokens

AREXX FOR BEGINNERS - GLOSSARY

TOKENS

A TOKEN is the smallest individual part of a CLAUSE. It could be compared, in English, to a word within a sentence.

There are a number of different types of tokens. These are:-

```
COMMENTS  
SYMBOLS
```

STRINGS
OPERATORS
SPECIAL CHARACTERS

=== End of Text ===

1.39 Glossary - Using The CLI or Shell

AREXX FOR BEGINNERS - GLOSSARY

USING THE CLI OR SHELL

I would expect that anyone learning ARexx should have at least an elementary knowledge of how to use a Shell or CLI window.

In these articles, when I want to indicate that a line is to be typed in at a CLI or Shell prompt, I will use the following sort of thing:-

```
> RX MyProgram
```

the ">" represents the CLI or Shell window's prompt. It may appear as one of the following or something similar, depending on your version of AmigaDOS:-

```
>
```

```
1.>
```

```
1.Workbench2.1 >
```

Whatever you have in your CLI or Shell window, you can take it that the ">" in these articles means the prompt that appears in your CLI or Shell window.

When you enter command lines at the prompt as a result of seeing things like:-

```
> RX MyProgram
```

then do not enter the ">!!!"

Start your entry at the text that appears after the prompt indicator. so that for the above example you would type in:-

```
RX MyProgram
```

I only show the ">" symbol to indicate to you that it is an entry to be made in a Shell or CLI window.

You must press return at the end of each command line.

=== End of Text ===

1.40 Glossary - Viruses

AREXX FOR BEGINNERS - GLOSSARY

VIRUSES

A virus is a small program that has been written to cause problems to computer users. Some viruses are in a disk's boot block and some are attached to other programs such as AmigaDOS commands. Viruses can be "benign" and only cause minor problems or silly messages, or they can be "malignant" and cause considerable damage to the data on your disks. You could find that ALL data has been completely wiped from your hard drive. To the best of my knowledge, they cannot do physical damage to your computer.

A decent Virus Checking and Killing program (such as "Virus_Checker") to detect and remove viruses is essential!.

=== End of Text ===
