COMPUTE! ISSUE 54 / NOVEMBER 1984 / PAGE 102

Update On COMAL: A SuperBASIC

Jim Butterfield, Associate Editor

COMAL is a computer language which was developed in Denmark as an offshoot of BASIC. Some programmers prefer it and think of it as a streamlined, extended, and systematized BASIC.

COMAL has been available for some time, mostly for Commodore 32K PET/CBM machines. Previous versions have had some success despite a few drawbacks and implementation problems. Now, new versions of COMAL are being released, and the new packages look promising.

The New Generation

COMAL has traditionally been available in the public domain (that means free). It's a sound language: The loose ends of traditional BASIC have been tightened into a much smoother system. The ease of use for beginners has been maintained, and added features significantly enhance the programming power. Yet this free, powerful language has enjoyed only modest success.

The problem has been limited resources. Traditional COMAL would fit only into a 32K PET/CBM; thus, only users with the biggest systems (of that era) could use COMAL. Second, the COMAL interpreter took up a good deal of memory, leaving room for only a small user program. To offset this difficulty, a "split" COMAL was developed which used a separate editor and interpreter; but this proved awkward to use.

Until recently, the best COMAL arrangement was obtained by using a CBM 8096 computer; with the extra 64K memory, there was plenty of room for both the interpreter and the user's program. Alternatively, a ROM board could be purchased to implement COMAL on a CBM 8032; this allowed large programs to be written, but the board was costly—about $400.

It looks like things are changing. New versions of COMAL are being released that will make it an attractive language. A disk version for the Commodore 64 is now in the public domain; a cartridge version is soon to be made available for sale; and even the ROM board for the PET/CBM 8032 is being redesigned to incorporate interesting new features.

What Is COMAL?

COMAL may be described in a number of ways. It's as easy as BASIC for beginners, but has the power of Pascal when advanced features are needed. It's a tightly defined superversion of BASIC, with much more precisely defined keywords and with statements that interrelate more closely. It's a BASIC extension, with extra graphics, sound, and sprite commands. COMAL is a structured version of BASIC, complete with IF-THEN-ELSE, WHILE-ENDWHILE, and similar features. It's an extensible language—you may even write your own keywords.

COMAL programs run as fast as—or faster than—the equivalent BASIC programs. COMAL will never have certain BASIC problems such as garbage collection.

For beginners, COMAL can be crudely described as follows: You type in a BASIC program; when you list it back, it looks as if it has been changed to Pascal. Indentation has been added and formats trimmed; it looks much neater. Some errors are checked at time of entry; some are checked before the run gets under way. Meaningful error messages are given.

Advanced programmers will be interested in other features of COMAL. The structured forms are all there, but there's more: procedures (subroutines) and functions with parameter passing; "closed" procedures that are isolated from the main coding; end-of-file and end-of-data flags; recursion; and in some versions, error trapping.

Users will find new commands and features that make graphics and sound easier. To change the background color to black, type BACK-GROUND 0. On the high-power end, COMAL 64 comes complete with turtle graphics commands: FORWARD 40 and RIGHT 45 have the same meaning as they would have in Logo.

Versions Of COMAL

Disk-based COMAL for the Commodore 64 is named COMAL 0.14. It's free in North America, but it's not public domain. The package prints a copyright statement, but permits unrestricted distribution and copying. COMAL 0.14 is available from clubs, user groups, and the COMAL user group in Madison, Wisconsin. It's a good language implementation, complete with graphics and a complete set of error messages drawn from disk as needed.

COMAL 2.0 is a plug-in cartridge for the 64 which is expected to be available from Commodore in 1985 for less than $100. It's significantly more powerful than COMAL 0.14—it offers much more program memory and includes extra features such as error trapping and program chaining or overlaying.

The COMAL board has been redesigned for the 8032 computer. The price of the revised board is expected to be similar to the previous version—that is, around $400—but there are some new features. First, if you don't want to use COMAL immediately, you can use BASIC, and the computer becomes an 8096 with 64K extra memory supplied by the COMAL board. Second, a video board can be added to the assembly to perform high-resolution graphics, including turtle geometry. Third, the board contains a time-of-day and date clock which is battery-powered and keeps good time even when the unit is off. I installed a board and checked the clock; before I reset it, it gave the correct time for Copenhagen, where it was manufactured.

Documentation

The *COMAL Handbook*, by Len Lindsay (published by Reston Publishing), is a reference manual for the language. It's just that: a reference manual, and not a tutorial. You can look things up, but it's not for learning the language.

Fortunately, most COMAL versions come with a disk of sample programs that illustrate the features of the language well. And to be fair to Lindsay's book, it contains a considerable quantity of sample programming.

If you're interested in the new 64 versions of COMAL, look specifically for the second edition of the *COMAL Handbook.* The publication date for this new edition hasn't yet been set. The original handbook gives a great deal of information on COMAL, but doesn't show the new color and graphics commands or other new features such as error trapping.

Most users will benefit from the fact that COMAL is derived from BASIC. Their BASIC experience will generally carry over to the new language.

Beginning Characteristics

Users may start COMAL programming as they did BASIC. Direct commands are allowed so that statements such as PRINT 5 + 7 can be executed.

Spaces must be used after keywords. FORK = 1TO5 must be written as FOR K = l TO 5. There are two reasons for this: COMAL encourages legibility, and COMAL allows long variable names. FORK could be a variable; it won't be confused with FOR K because of the space. By the way, the whole variable name is used, so that FORK is distinct from FORM, and both are different from the variable called FOR'LOVE'OF'IVY.

Even though COMAL has a full set of structured statements available, it allows the use of a GOTO statement. However, you can't GOTO a line number; you must GOTO a labeled statement in your program. COMAL is quite insistent that line numbers should be used only for editing purposes. The use of GOTO is strongly discouraged; there are usually plenty of alternative ways of programming what you need.

Subroutines are called, not with GOSUB, but with EXEC for execute. COMAL uses the term *procedure* rather than subroutine; a procedure is given a name. So instead of GOSUB 500 we might code EXEC SUMMARY. Since procedures have names, we don't need the keyword EXEC. Instead of EXEC SUMMARY we may just write SUMMARY, and the interpreter will call the procedure when it reaches that point. In this way, the language is extensible. We might code a program starting as follows:

```
100 START
110 CONTINUE
120 FINISH
130....
```

What this sequence means is: Call procedure START, then call procedure CONTINUE, then call procedure FINISH. Each of these will need to be defined as a procedure (a type of subroutine) somewhere in the program, but here's the point: The above coding is quite readable; and we have essentially defined new program commands.

It's impossible in this short article to deal with the whole spectrum of commands. Perhaps the above will give a flavor of how COMAL extends the capabilities of BASIC.

A Mix Of Features

COMAL isn't new, but there's a new COMAL. If languages interest you, this one is well worth a look. It's easy on the beginner, yet it's not limiting to the expert programmer. It's more than BASIC, not exactly a Pascal, and different from Logo—but it has some of the characteristics of all three languages.

It's inexpensive to try. You may find that it's just what you have been looking for.

Further information on COMAL may be obtained from:

COMAL User's Group
5501 Groveland Terrace
Madison, WI 53716

Reston Publishing
11480 Sunset Hills Road
Reston, VA 22090

COMAL Examples

The user may input what appears to be a BASIC program; when it is listed, it appears in a significantly different format, but does the same thing.

User Input	**COMAL Listing**
10 FOR J = 1 TO 20	**0010 FOR J:=1 TO 20 DO**
20 T=0	**0020 T:=0**
30 FOR K=1 TO J	**0030 FOR K:=1 TO J DO**
40 T=T+K	**0040 T:=T+K**
50 NEXT K	**0050 NEXT K**
PRINT T	**0060 PRINT T**
70 NEXT J	**0070 NEXT J**

Note that the COMAL editor has provided indentation to more easily identify the loops, and has changed the syntax of FOR and assignment statements slightly.

An experienced COMAL programmer might streamline the coding along the following lines:

```
0010 FOR J : = 1 TO 20 DO SUM
0020 PROC SUM
0030 T : = 0
0040 FOR K : = 1 TO J DO T : + K
0050 PRINT T
0060 END PROC
```

Here there are no NEXT statements, and both FOR statements are one-line loops. The coding within the FOR J-NEXT J loop has now been defined as a procedure called SUM and is called as

needed. The statement T = T + K has been replaced by the more efficient T : + K. The program will run a little faster in its new coding, but its major advantage is that it's neater.

The user might take things a step further by using longer variable names and passing a value to the procedure:

```
0010 FOR J : = 1 TO 20 DO SUM (J)
0020 PROC SUM(VALUE)
0030 TOTAL : = 0
0040 FOR INDEX : = 1 TO VALUE DO TOTAL : + INDEX
0050 PRINT TOTAL
0060 ENDPROC
```

Labels such as TOTAL and VALUE would be forbidden in BASIC (the keywords TO and VAL are there), but not in COMAL. The new program takes slightly more space than before and runs at the same speed. Procedure SUM could now be completely disconnected from the main routine (via PROC SUM(VALUE) CLOSED) since it is passed all the required data; it doesn't need to "raid" the main program variables.

We can go a step further by using a defined function in this case:

```
0010 FOR J : = l TO 20 PRINT SUM(J)
0020 FUNC SUM(VALUE)
0030 TOTAL : = 0
0040 FOR INDEX : = l TO VALUE DO TOTAL : + INDEX
0050 RETURN TOTAL
0060 ENDFUNC
```

The value of SUM is calculated each time it is used. Using a defined function can generate a very readable program. The statement RETURN is different from a BASIC subroutine RETURN. It means, "give back a value of TOTAL to the calling program."

Common Sense And COMAL

This program generates the sums of various series of numbers. It's meant to show COMAL options; it's certainly not the best way to do the job. As any math teacher will tell you, the sum of the numbers from 1 to J can be calculated by the formula (J + l) * J/2. We could therefore reduce the function to a single line and write the program as follows:

```
0010 FOR J : = l TO 20 PRINT SUM(J)
0020 FUNC SUM(VALUE)
0030 RETURN (VALUE + l) * VALUE/2
0040 ENDFUNC
```

That's not an advantage caused by the language—that's just a little math and common sense.