//

// anything typed here

Statement - Anything after // is ignored, allowing comments in programs. In direct mode, // allows you to overtype on a full screen line. After your command, just type // and hit return; the rest of the line is ignored, and your command executed.

```
FUNC test'comment CLOSED
  // this is a comment
  RETURN TRUE // comments allowed
ENDFUNC test'comment
```

1. Tests single line comment and comment at the end of a statement.

ABS

ABS(«numeric expression»)
PRINT ABS(standard'number)

Function - Gives the absolute value of the number. Positive numbers and zero are unaffected, while negative numbers become positive.

```
FUNC test'abs CLOSED
  IF ABS(1)<>1 THEN RETURN FALSE
  IF ABS(-1)<>1 THEN RETURN FALSE
  IF ABS(0)<>0 THEN RETURN FALSE
  IF ABS(-3.9)<>3.9 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'abs
```

2. Tests positive, negative, zero and non-integer values.

AND

«expression» AND «expression»
IF number>0 AND number<100 THEN

Operator - Gives the result of a logical AND of two expressions, as shown by the following table:

AND	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

This is different than most BASICs in which AND is a bitwise operator. For bitwise AND in COMAL see BITAND.

```
FUNC test'and CLOSED
  IF (TRUE AND TRUE)<>TRUE THEN RETURN FALSE
  IF (TRUE AND FALSE)<>FALSE THEN RETURN FALSE
  IF (FALSE AND TRUE)<>FALSE THEN RETURN FALSE
  IF (FALSE AND FALSE)<>FALSE THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'and
```

3. Tests all four AND operation possibilities.

APPEND

OPEN [FILE] «file#»,«filename»,APPEND
OPEN FILE 2,"test",APPEND

File Type - Part of the OPEN statement. The sequential file must already exist on disk, and is opened in APPEND mode. New data is written to the file immediately after the existing data.

AT

PRINT AT «row»,«col»: [«print list»[«mark»]]
INPUT AT «row»,«col»[,«len»]:[«prompt»:] [«vars»[«mark»]]
PRINT AT 1,1: "Section number:"; num;
INPUT AT 10,1,1:"Yes or No? ":reply$

Special - Part of INPUT or PRINT statements, specifying a specific location to start at, similar to having a CURSOR statement immediately before a PRINT or INPUT statement. PRINT AT may also be combined with a USING format. Remember, the cursor location is specified row then column, similar to finding your seat at a theater. Including a comma or semicolon at the end of the statement causes the cursor to remain on the current line, and not go down to the next line. A comma means stay where it is, while a semicolon means space to the next zone then stay there (default is one space).

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not accept a comma at the end of an INPUT AT statement. They do accept a semi-colon.

```
FUNC test'at CLOSED
  row:=CURROW; col:=CURCOL
  PRINT AT 10,30: "",
  IF CURROW<>10 THEN RETURN FALSE
  IF CURCOL<>30 THEN RETURN FALSE
  PRINT AT 0,20: "",
  IF CURROW<>10 THEN RETURN FALSE
  PRINT AT 4,0: "",
  IF CURCOL<>20 THEN RETURN FALSE
  CURSOR row,col
  RETURN TRUE
ENDFUNC test'at
```

4. Tests row and column placement, then column only, then row only.

ATN

ATN(«numeric expression»)
PRINT ATN(num1+num2)

Function - Returns the arctangent in radians of the number. *CP/M COMAL allows you to choose degrees rather than radians, but has radians as the default.*

```
FUNC test'atn CLOSED
  IF ATN(0)<>0 THEN RETURN FALSE
  IF ABS(ATN(1)-PI/4)>0.000001 THEN RETURN FALSE
  IF ABS(ATN(TAN(0.5))-0.5)>0.000001 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'atn
```

5. Tests the arctangent trig function.

AUTO

AUTO [«start line»][,«increment»]
AUTO 9000
AUTO 100,100
AUTO ,5

Command - Makes the COMAL system generate line numbers automatically as a program is typed in. Valid line numbers are 1 - 9999. Each line number is always four characters. Some systems pad with leading 0's (0030), others with leading spaces (30). AUTO begins with the next available line number if you do not specify a starting line. If you don't specify an increment, 10 is used. If you don't specify a starting line number, 10 is used, unless program lines already exist. Then the line number to start at is the last line plus the increment.

Alder COMAL will not allow you to cursor up to correct a previous line while in AUTO mode, other COMALs do.

Just hitting «return» after a line number inserts a blank line in the program, except in C64 Power Driver, C64 COMAL 1.0 and Apple COMAL, which use a blank line to terminate AUTO mode. C64 Power Driver and C64 COMAL 1.0 always start at line 10 unless otherwise specified.

BASIC : exit COMAL to BASIC, see BYE

BITAND

«argument» BITAND «argument»
show(bnum BITAND %00001000)

Operator - Returns the bitwise AND of the two numbers, similar to most BASICs AND. Binary constants are prefixed by a %. The following table shows how BITAND works:

BITAND	00	01	10	11
00	00	00	00	00
01	00	01	00	01
10	00	00	10	10
11	00	01	10	11

```
FUNC test'bitand CLOSED
  IF (3 BITAND 3)<>3 THEN RETURN FALSE
  // (%11 BITAND %11)<>%11  <==binary
  IF (3 BITAND 0)<>0 THEN RETURN FALSE
  // (%11 BITAND %0)<>%0  <==binary
  IF (5 BITAND 6)<>4 THEN RETURN FALSE
  // (%101 BITAND %110)<>%100  <==binary
  RETURN TRUE
ENDFUNC test'bitand
```

6. Tests three types of BITAND operations: both bits on, both bits off, and one bit on - one bit off.

BITOR

«argument» BITOR «argument»
PRINT (bnum BITOR flag)

Operator - Returns the bitwise OR of the two numbers, similar to most BASICs OR. Binary constants are prefixed by a %. The following table shows how BITOR works:

BITOR	00	01	10	11
00	00	01	10	11
01	01	01	11	11
10	10	11	10	11
11	11	11	11	11

```
FUNC test'bitor CLOSED
  IF (3 BITOR 3)<>3 THEN RETURN FALSE
  // (%11 BITOR %11)<>%11  <==binary
  IF (3 BITOR 0)<>3 THEN RETURN FALSE
  // (%11 BITOR %0)<>%11  <==binary
  IF (5 BITOR 6)<>7 THEN RETURN FALSE
  // %101 BITOR %110)<>%111  <==binary
  IF (0 BITOR 0)<>0 THEN RETURN FALSE
  // (%0 BITOR %0)<>%0  <==binary
  RETURN TRUE
ENDFUNC test'bitor
```

7. Tests three types of BITOR operations: both bits on, both bits off, and one bit on - one bit off.

BITXOR

«argument» BITXOR «argument»
bnum=(num1+num2) BITXOR %10000000

Operator - Returns the bitwise exclusive OR of the two numbers. BITXOR performs the bitwise XOR operation bit by bit on the two numbers. Binary constants are prefixed by a %. The following table shows how BITXOR works:

BITXOR	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

```
FUNC test'bitxor CLOSED
  IF (3 BITXOR 3)<>0 THEN RETURN FALSE
  // (%11 BITXOR %11)<>%0  <==binary
  IF (3 BITXOR 0)<>3 THEN RETURN FALSE
  // (%11 BITXOR %0)<>%11  <==binary
  IF (5 BITXOR 6)<>3 THEN RETURN FALSE
  // (%101 BITXOR %110)<>%11  <==binary
  IF (0 BITXOR 0)<>0 THEN RETURN FALSE
  // (%0 BITXOR %0)<>%0  <==binary
  RETURN TRUE
ENDFUNC test'bitxor
```

8. Tests the three types of BITXOR operations: both bits on, both bits off, and one bit on - one bit off.

BYE

BYE

Command - Returns you to the computer's operating system.

Both Amiga COMALs ask you to confirm the exit from COMAL as a safeguard. Amiga COMAL also lets you click on the close window gadget in the top left corner of the COMMAND window to exit COMAL. C64 and C128 carts use the keyword BASIC instead of BYE, since BASIC is the main operating system.

CASE

CASE «control expression» [OF]
CASE reply$ OF
CASE choice OF

Statement - Begins a CASE structure, allowing a multiple choice decision with as many specific WHEN sections as needed. A default OTHERWISE section may be included that is executed if none of the WHEN sections match the condition (which can be either string or numeric). Statement blocks following each WHEN are indented when listed, but the CASE, WHEN and OTHERWISE statements are not). The system will insert the word OF for you if you don't type it.

```
FUNC test'case CLOSED
  DIM reply$ OF 4, choice$ OF 1
  reply$="abcd"
  FOR x:=1 TO 4 DO
    CASE x OF
    WHEN 1
      IF x<>1 THEN RETURN FALSE
    WHEN 2
      IF x<>2 THEN RETURN FALSE
    OTHERWISE
      IF x<3 THEN RETURN FALSE
    ENDCASE
    choice$=reply$(x:x)
    CASE choice$ OF
    WHEN "a"
      IF x<>1 THEN RETURN FALSE
    WHEN "b"
      IF x<>2 THEN RETURN FALSE
    OTHERWISE
      IF x<3 THEN RETURN FALSE
    ENDCASE
  ENDFOR x
  RETURN TRUE
ENDFUNC test'case
```

9. Tests numeric and string CASE matching, with multiple WHEN statements and an OTHERWISE.

CAT

CAT [«filename»]
CAT

Command - Gives a catalog (directory) of the files on a disk. It uses the default drive if none is specified. Pattern and wild card matching is allowed and should match the way the operating system works. For example, with Commodore and IBM computers, the following is true:

? matches any one character
* matches any string of characters

With the Amiga, the * means the current window, and other characters (such as #) are used for pattern matching. On IBM and Amiga, a period (.) is a significant part of a filename, while on Commodore it is just another character. These and other differences are to be expected, since the COMAL system must adapt to the computers operating system methods.

Most COMALs include both CAT and DIR even though they may be identical in purpose, to allow users to use the one they are used to. C64 Power Driver and C64 COMAL 1.0 allow DIR in programs, but not CAT.

COMMON COMAL - Definition and Test Functions

CHAIN

CHAIN «filename»
CHAIN "menu"

Command/Statement - Loads and runs a program from a disk file. The program must have previously been SAVEd to disk.

CHANGE

CHANGE "«old text»","«new text»"
CHANGE "zz","print'report"

Command - Changes parts of a program line into another string of characters, just like in word processors.

Both Amiga COMALs do not have this command, but have their own methods of accomplishing it. C64 Power Driver and Apple COMAL do not have the CHANGE command.

CHR$

CHR$(«numeric expression»)
PRINT CHR$(num)

Function - Returns the character with the specified numeric (ASCII) code. ORD is the complimentary function to CHR$.

Note that ASCII codes may vary from system to system (Commodore in particular).

```
FUNC test'chr CLOSED
  IF CHR$(53)<>"5" THEN RETURN FALSE
  IF ORD(CHR$(65))<>65 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'chr
```

10. Tests for ASCII character "5" and verifies that ORD is the complement of CHR$.

CLOSE

CLOSE [[FILE] «filenum»]
CLOSE FILE 2

Command/Statement - Closes the file specified. If no specific file is specified, all files are closed (but does not affect files opened via the SELECT OUTPUT statement). No error should occur if you attempt to close a file that is not open.

CP/M COMAL gives an error if you try to close a file that is not open.

CLOSED

PROC «procname»[(params)] [CLOSED]
FUNC «funcname»[(params)] [CLOSED]
PROC newpage(header$) CLOSED
FUNC gcd(n1,n2) CLOSED

Procedure/Function Type - Declares that all variables and arrays inside the procedure or function are to be local - hidden from the main program. Likewise, all variables and arrays in the main program are not known inside a CLOSED procedure or function. However, specific variables and arrays may become known inside a CLOSED procedure or function by use of parameters or the IMPORT statement. Data statements inside a CLOSED procedure or function are considered local *(except for C64 Power Driver, C64 COMAL 1.0 and Apple COMAL). Many COMALs allow a CLOSED procedure or function to be made EXTERNAL. See EXTERNAL.*

```
FUNC test'closed CLOSED
  temp:=TRUE
  closed'proc
  RETURN temp
  //
  PROC closed'proc CLOSED
    temp:=FALSE
  ENDPROC closed'proc
  //
ENDFUNC test'closed
```

11. Tests that a variable changed inside a closed procedure does not affect a variable with the same name outside that procedure.

CON

CON

Command - Restarts a program that was previously stopped by a STOP statement or the stop/break key.

Due to the internal linking system used by most COMALs, if lines are added, deleted, or modified, or if new variables are introduced, the program may not be able to be continued.

COS

COS(«numeric expression»)
PRINT COS(number)

Function - Returns the cosine of the number in radians. *CP/M COMAL allows you to choose degrees rather than radians, but has radians as the default.*

```
FUNC test'cos CLOSED
  IF ABS(COS(0)-1)>0.000001 THEN RETURN FALSE
  IF ABS(COS(PI/3)-0.5)>0.000001 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'cos
```

12. Tests the cosine trig function.

CREATE

CREATE «filename»,«# records»,«record size»
CREATE "names",128,200

Command/Statement - Creates a random access file of the specified size. Most COMALs count records in a random file beginning with record number 1. (*CP/M COMAL begins with record number 0.*)

CURCOL

CURCOL
column:=CURCOL

Function - Returns the current column position of the cursor on the text screen. Columns are counted from left to right. The leftmost column is 1.

CURCOL is not implemented by Alder. CURCOL is part of the SYSTEM package in C64 & C128 2.0 cartridge implementations, and requires a USE SYSTEM command prior to use, and is not available within CLOSED procedures or functions without being imported (or having a separate USE SYSTEM within the procedure or function).

```
FUNC test'curcol CLOSED
  col:=CURCOL
  CURSOR 0,30
  IF CURCOL<>30 THEN RETURN FALSE
  CURSOR 0,col
  RETURN TRUE
ENDFUNC test'curcol
```

13. Tests that the correct column is returned by setting only the column position, checking and then replacing the cursor where it originally was.

CURROW

CURROW
row:=CURROW

Function - Returns the current row of the text screen that the cursor is on. Rows are counted top to bottom. The top row is 1.

CURROW is not implemented by Alder. CURROW is part of the SYSTEM package in C64 & C128 2.0 cartridge implementations, and requires a USE SYSTEM command prior to use, and is not available within CLOSED procedures or functions without being imported (or having a separate USE SYSTEM within the procedure or function).

```
FUNC test'currow CLOSED
  row:=CURROW
  CURSOR 10,0
  IF CURROW<>10 THEN RETURN FALSE
  CURSOR row,0
  RETURN TRUE
ENDFUNC test'currow
```

14. Tests that the correct value is returned by setting the cursor location by row, checking the value, then returning the cursor to its original row location.

CURSOR

CURSOR «line»,«position»
CURSOR 1,1

Command/Statement - Positions the cursor to the specified row and column. Rows are counted from top to bottom; columns from left to right. The top row is line 1. The leftmost column is column 1. Cursor positioning is similar to finding your seat at a theater. First find the row, then the position in that row.

Specifying 0 as a row or column means not to change it, thus CURSOR 0,9 would move to position 9 on the current row.

If used as a direct command, CURSOR correctly positions the cursor, but then moves to the first position on the next line for the next command (which surprises many beginners).

```
FUNC test'cursor CLOSED
  row:=CURROW; col:=CURCOL
  CURSOR 10,30
  IF CURROW<>10 THEN RETURN FALSE
  IF CURCOL<>30 THEN RETURN FALSE
  CURSOR 0,20
  IF CURROW<>10 THEN RETURN FALSE
  CURSOR 15,0
  IF CURCOL<>20 THEN RETURN FALSE
  CURSOR row,col
  RETURN TRUE
ENDFUNC test'cursor
```

15. Saves the current cursor position, then moves it to a specific location and checks the values. Next it moves just the column and checks. Then it moves just the row and checks. Finally the cursor is returned to its original location.

DATA

DATA «value»{,«value»}
DATA "Sam",134,"Fred",22,"Gloria",46

Statement - Declares data constants that may be assigned to variables via a READ statement. Data may be text strings within quotes, or numbers.

Multiple items may follow a DATA keyword, separated by commas. When the last DATA item is read, EOD is set to TRUE. Data can be reused following a RESTORE command.

To include a quote mark as part of the string data, use two consecutive quote marks ("abc""defg" is read as abc"defg).

Data inside a CLOSED procedure or function is regarded as local data. Likewise, a READ statement inside a CLOSED procedure or function may only read data inside that procedure or function.

In C64 Power Driver, C64 COMAL 1.0 and Apple COMAL data is always global.

```
FUNC test'data CLOSED
  DIM s$ OF 6
  READ s$
  IF s$<>"passed" THEN RETURN FALSE
  READ x
  IF x<>13 THEN RETURN FALSE
  RETURN TRUE
  DATA "passed",13
ENDFUNC test'data
```

16. Tests reading string and numeric data from DATA statements.

DEL

DEL «range»
DEL 460
DEL pause

Command - Removes (deletes) lines from the program currently in the computer's memory. Lines may be deleted one at a time or in consecutive blocks, all at once.

DEL 10 deletes line 10. DEL pause deletes the procedure or function named pause. DEL 10-30 deletes all the lines in the range of 10 through 30. DEL -90 deletes all program lines up to and including line 90. DEL 9000- deletes all program lines after and including line 9000.

CP/M COMAL requires that a specified line must exist.

DELETE

DELETE «filename»
DELETE "test5"

Command/Statement - Removes files from disk. Several files may be deleted at once by using pattern matching and wildcard characters. These will match the methods used by the computers operating system and will vary between systems. For example, the following are used by Commodore 64 and 128:

? matches any one character
* matches any string of characters

IBM PC COMAL can only delete one file at a time. C64 Power Driver and C64 COMAL 1.0 require that a drive be specified as part of the filename.

DIM

DIM «string var» OF «max char»
DIM «str array»(«index») OF «max char»
DIM «array name»(«index»)
DIM name$ of 30
DIM players$(1:4) OF 10
DIM scores(min:max)

Command/Statement - Allocates (dimensions) space for strings and arrays. Arrays begin with element 1 unless otherwise specified. Multiple DIMs may be in one statement, separated by commas. Redimensioning is not allowed.

Each element of a numeric array is initially set to 0 when dimensioned (*except for Alder, but they may have this corrected now)*). Arrays may have multiple dimensions, with whatever top and bottom limits you wish, within memory limitations.

```
FUNC test'dim CLOSED
  DIM s$ OF 4, x(1), z(-1:2)
  s$:="passed"
  IF s$<>"pass" THEN RETURN FALSE
  IF x(1)<>0 THEN RETURN FALSE
  x(1):=TRUE; z(-1):=5
  IF x(1)<>TRUE THEN RETURN FALSE
  IF z(-1)<>5 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'dim
```

17. Tests dimensioning string and numeric arrays, including one with a negative element index. It also checks that strings are truncated if they are longer than the limit and that an array is initialized to 0 when it is dimensioned.

DIR

DIR [«filename»]
DIR

Command - Gives a directory of the files on a disk. It uses the default drive if no other is specified. Pattern matching and wild card characters are allowed and should match the way the operating system works. For example, with IBM and Commodore 64 computers, the following is true:

? matches any one character
* matches any string of characters

With the Amiga, the * means the current window, and other characters (such as #) are used for

pattern matching. On IBM and Amiga, a period (.) is a significant part of a filename, while on Commodore it is just another character. These and other differences are to be expected, since the COMAL system must adapt to the computers operating system methods.

Most COMALs include both CAT and DIR even though they may be identical in purpose, to allow users to use the one they are used to. C64 Power Driver and C64 COMAL 1.0 allow DIR in programs, but not CAT.

DISCARD

DISCARD

Command - Discards all previously linked packages and libraries.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support packages and thus do not have a DISCARD command.

DISPLAY

DISPLAY [«range»] [TO] [«filename»]
DISPLAY "names.lst"
DISPLAY init

Command - Lists a program without line numbers. Ranges of lines may be specified, as with LIST. Program lines may be displayed to disk, allowing them to be inserted into word processing documents and such. Some systems allow program lines displayed to disk to be re-entered with the ENTER or MERGE commands.

In CP/M COMAL, if there are more lines to be displayed than fit on the screen, it automatically pauses after each screenful. Hit <space> for the next screen. In Alder COMAL, <CTRL>+S is used to pause a display. Other COMALs use <space> to pause and restart a listing.

If you cursor up a program listing on the screen, when you hit the top line, Amiga, IBM and C128 COMALs will re-list the previous line, and scroll the listing down. This allows you to backtrack up a listing, and is very handy.

DIV

«dividend» DIV «divisor»
result=guess DIV count

Operator - Provides division with an integer answer. It can be used in conjunction with the MOD operator. DIV defines x DIV z as INT(x/z).

C64 cart and CP/M COMAL do not follow the definition when one of the numbers is negative.

```
FUNC test'div CLOSED
  IF 500 DIV 256<>1 THEN RETURN FALSE
  IF 1500 DIV 5<>300 THEN RETURN FALSE
  IF (-7) DIV 3<>(-3) THEN RETURN FALSE
  IF 7 DIV (-3)<>(-3) THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'div
```

18. Tests two types of division, with and without remainder, followed by two types of division with negative numbers.

DO : see FOR and WHILE

EDIT

EDIT [«range»]
EDIT pause
EDIT

Command - Lists program lines to the screen, similar to LIST, but without indentations, one line at a time, for you to edit. Line ranges may be specified as with LIST. *C64 Power Driver, C64 COMAL 1.0 and Apple COMAL list lines continuously, not one at a time.*

ELIF

ELIF «expression» [THEN]
ELIF reply$ IN "YyNn" THEN

Statement - Allows conditional statement execution. ELIF is short for ELSE IF and is part of the IF structure. The statement block following the ELIF is executed only if the condition is TRUE, otherwise it is skipped (the statement block is automatically indented in listings). If you omit the word THEN, the system will insert it for you.

```
FUNC test'elif CLOSED
  IF FALSE THEN
    RETURN FALSE
  ELIF TRUE THEN
    NULL
  ELSE
    RETURN FALSE
  ENDIF
  RETURN TRUE
ENDFUNC test'elif
```

19. Tests that a TRUE ELIF condition has its statements executed, and that execution then continues after the ENDIF, skipping the ELSE section statements.

ELSE

ELSE

Statement - Provides alternative statements to execute when all IF and ELIF conditions in the IF structure evaluate to FALSE (the statement block is automatically indented in listings).

```
FUNC test'else CLOSED
  IF FALSE THEN
    RETURN FALSE
  ELSE
    RETURN TRUE
  ENDIF
ENDFUNC test'else
```

20. Tests that the ELSE section is executed by default.

END

END [«message»]
END "All Done."

Statement - Terminates program execution. END is optional. Without an END statement, a program ends automatically after its last line is executed. There may be more than one END statement in a program. Programs ending at an END statement may not be restarted via CON (use STOP for this capability). A message may be included to replace the system default end message (usually End At Line 0100 or something similar).

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not allow the optional message after END. CP/M COMAL seems more flexible with the optional message, and thus ending messages can be slightly different. Amiga COMAL's end message is printed in the COMMAND Window.

ENDCASE

ENDCASE

Statement - Marks the end of a CASE structure.

```
FUNC test'endcase CLOSED
  CASE 1 OF
  WHEN 1
    NULL
  OTHERWISE
    RETURN FALSE
  ENDCASE
  RETURN TRUE
ENDFUNC test'endcase
```

21. Tests that program execution skips to after the ENDCASE following a successful WHEN match.

ENDFOR

ENDFOR [«control variable»]
ENDFOR sides#
ENDFOR increment

Statement - Marks the end of a FOR loop. The system will insert the variable name after ENDFOR for you if you omit it (after a SCAN or RUN). Single line FOR statements do not use ENDFOR.

Some systems convert NEXT into ENDFOR for you (making the transition from BASIC easier).

The control variable is considered local to the FOR structure by Amiga, IBM, CP/M, C128 and C64 cartridge COMALs. Thus a for loop variable will not conflict with a variable of the same name in the main program.

```
FUNC test'endfor CLOSED
  FOR x#:=1 TO 2 DO
    y#:=x#
  ENDFOR x#
  IF y#<>2 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'endfor
```

22. Tests that ENDFOR marks the end of a FOR loop.

ENDFUNC

ENDFUNC [«function name»]
ENDFUNC even

Statement - Marks the end of a user defined function. The system will insert the function name after the ENDFUNC for you if you do not type it (after a SCAN or RUN). ENDFUNC is not used with EXTERNAL function header lines. See EXTERNAL for more information.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support nested functions or EXTERNAL functions.

```
FUNC test'endfunc CLOSED
  IF int'func#<>5 THEN RETURN FALSE
  RETURN TRUE
  //
  FUNC int'func# CLOSED
    RETURN 5
  ENDFUNC int'func#
  //
ENDFUNC test'endfunc
```

23. Tests that ENDFUNC marks the end of a function.

ENDIF

ENDIF

Statement - Marks the end of a multi-line IF structure. One line IF statements do not use an ENDIF.

```
FUNC test'endif CLOSED
  IF FALSE THEN
    RETURN FALSE
  ENDIF
  RETURN TRUE
ENDFUNC test'endif
```

24. Tests that program execution skips to after the ENDIF for a failed IF structure.

ENDLOOP

ENDLOOP

Statement - Marks the end of a multi-line LOOP structure.

```
FUNC test'endloop CLOSED
  LOOP
    EXIT
    RETURN FALSE
  ENDLOOP
  RETURN TRUE
ENDFUNC test'endloop
```

25. Tests that program execution continues after the ENDLOOP after an EXIT statement is executed.

ENDPROC

ENDPROC [«procedure name»]
ENDPROC show'item

Statement - Marks the end of a procedure. The system will insert the procedure name after the ENDPROC for you if you do not type it (after a SCAN or RUN). ENDPROC is not used for EXTERNAL procedure header statements. See EXTERNAL for more information.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support nested procedures or EXTERNAL procedures.

```
FUNC test'endproc CLOSED
  setvar(x)
  RETURN x
  //
  PROC setvar(REF var) CLOSED
    var:=TRUE
  ENDPROC setvar
  //
ENDFUNC test'endproc
```

26. Tests that ENDPROC marks the end of a procedure definition.

ENDTRAP

ENDTRAP

Statement - Marks the end of the error handler TRAP structure.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support the error handler structure and thus do not have ENDTRAP.

```
FUNC test'endtrap CLOSED
  TRAP
    div0=5/0
    RETURN FALSE
  HANDLER
    NULL
  ENDTRAP
  RETURN TRUE
ENDFUNC test'endtrap
```

27. Tests that execution of the error handler continues after the ENDTRAP following completion of the handler section.

ENDWHILE

ENDWHILE

Statement - Marks the end of a multi-line WHILE structure. ENDWHILE is not used with single line WHILE statements.

```
FUNC test'endwhile CLOSED
  WHILE FALSE DO
    RETURN FALSE
  ENDWHILE
  RETURN TRUE
ENDFUNC test'endwhile
```

28. Tests that statements are not executed if the WHILE condition fails, and that program execution skips to immediately after the ENDWHILE statement.

ENTER

ENTER «filename»
ENTER "testing.lst"

Command - Enters program lines from an ASCII format file (such as a file of a program previously LISTed to disk). Any current program is cleared from memory prior to entering the new lines (use MERGE to preserve the current program).

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not clear a program first, allowing ENTER to also merge program segments into the current program. CP/M and IBM PC COMALs do not require line numbers in the file being entered.

When transferring a COMAL program from one system to another, LIST the program to disk, then

ENTER or MERGE it into the other system. This may also be done via modem or networks.

EOD

EOD
WHILE NOT EOD DO

Function - Boolean function that returns TRUE if End Of Data has been reached. If there are no DATA statements in the program, EOD is always TRUE.

```
FUNC test'eod CLOSED
  IF EOD THEN RETURN FALSE
  WHILE NOT EOD DO READ x
  IF NOT EOD THEN RETURN FALSE
  RETURN TRUE
  DATA 13,14
ENDFUNC test'eod
```

29. Tests that EOD returns FALSE if there is data to be read, that EOD can be used to read all data, and that after all the data is read, EOD is TRUE.

EOF

EOF(«filenum»)
WHILE NOT EOF(infile) DO

Function - Boolean function that returns TRUE when End Of File has been reached. Since several files may be open at one time, you must specify the file number.

ERR

ERR
CASE err OF

Function - Returns the error number when an error occurs within an error handler structure. Error numbers are implementation specific.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support an error handler and thus do not have ERR.

```
FUNC test'err CLOSED
  TRAP
    REPORT 13
  HANDLER
    IF ERR<>13 THEN RETURN FALSE
  ENDTRAP
  RETURN TRUE
ENDFUNC test'err
```

30. Tests that the error number reported by ERR is the same one that has been trapped.

ERRTEXT$

ERRTEXT$
PRINT ERRTEXT$

Function - Returns the error message for the current error inside an error handler. Error messages are implementation specific, and methods of dealing with them varies between COMALs.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support an error handler and thus do not have ERRTEXT$.

```
FUNC test'errtext CLOSED
  TRAP
    IF LEN(ERRTEXT$)<>0 THEN RETURN FALSE
    div0:=5/0  //division by 0 error
  HANDLER
    IF LEN(ERRTEXT$)>0 THEN RETURN TRUE
  ENDTRAP
  RETURN FALSE
ENDFUNC test'errtext
```

31. Tests that there is no text in ERRTEXT$ prior to an error, and that after an error occurs, ERRTEXT$ contains the message.

ESC

ESC
EXIT WHEN ESC

Function - Returns TRUE if the stop/break key has been pressed. This is only useful if the stop/break key is disabled (see TRAP).

ESC seems unreliable in Alder COMAL.

```
FUNC test'esc CLOSED
  TRAP ESC-
  TRAP ESC+
  IF ESC THEN NULL
  IF ESC THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'esc
```

32. Tests that the stop/break key can be disabled and enabled. It then tests that once set, the ESC flag can be cleared back to its FALSE setting.

EXEC

[EXEC] «procname»[(«parameter list»)]
show'item(number)

Command/statement - Executes a procedure. May be used from direct mode. The word EXEC is optional and rarely typed. Multiple EXEC statements may be on one line, separated by semicolons. Only the procedure name needs to be typed to execute a procedure.

```
FUNC test'exec CLOSED
  x:=FALSE
  setx(x)
  RETURN x
  //
  PROC setx(REF y)
    y:=TRUE
  ENDPROC setx
  //
ENDFUNC test'exec
```

33. Tests that the original variable passed as a REF parameter is properly updated after the procedure call.

EXIT

EXIT [WHEN «condition»]
EXIT WHEN errors>3

Statement - Provides the method for leaving a LOOP structure. It can be conditional with the optional WHEN extension.

CP/M COMAL and Amiga COMAL use EXIT to exit from FOR, REPEAT and WHILE structures as well as LOOP. This can cause potential problems and is viewed as a flaw rather than an enhancement.

```
FUNC test'exit CLOSED
  LOOP
    EXIT
    RETURN FALSE
  ENDLOOP
  RETURN TRUE
ENDFUNC test'exit
```

34. Tests that EXIT leaves the loop structure and begins execution following the ENDLOOP statement.

EXP

EXP(«numeric expression»)
PRINT EXP(number)

Function - Returns the natural logarithm's base value e raised to the power specified. A good representation of e is 2.718281828.

```
FUNC test'exp CLOSED
  IF EXP(0)<>1 THEN RETURN FALSE
  IF ABS(EXP(1)-2.71828)>0.00001 THEN RETURN FALSE
  IF ABS(EXP(LOG(10))-10)>0.000001 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'exp
```

35. Tests a few conditions of the natural logarithms base value e raised to the specified power.

EXTERNAL

PROC «name»[(«parms»)][EXTERNAL «file»]
FUNC «name»[(«parms»)][EXTERNAL «file»]
FUNC rec'size(name$) EXTERNAL "rec.ext"
PROC set'up EXTERNAL "setup.ext"

Special - Identifies a procedure or function as an external one. This means that the body of the procedure or function is stored on disk, and is not part of the program itself. Thus ENDFUNC or ENDPROC are not used. An external procedure or function is considered CLOSED. To be used as external, a procedure or function must be CLOSED and previously SAVEd to disk.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support EXTERNAL procedure or functions. Amiga, IBM, C64 and C128 cartridge COMALs allow the filename to be a variable.

```
FUNC test'external CLOSED
  // this test'external is only valid if
  // called as an external func
  RETURN TRUE
ENDFUNC test'external
```

36. Test that a function can be external. Note that this test is only valid if the function is saved to be an external function.

FALSE

FALSE
ok:=FALSE

System Constant - Always equals 0. It can be used in comparisons or as a numeric expression. Example, test:=FALSE is the same as test:=0.

```
FUNC test'false CLOSED
  IF FALSE<>0 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'false
```

37. Tests that FALSE is equal to 0.

FILE : see CLOSE, INPUT, OPEN, PRINT, READ, WRITE

FIND

FIND "«text string»"
FIND " PROC "

Command - Searches the program for specified text. It is case sensitive in most COMALs, so that repeat would not match REPEAT.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not have the FIND command.

FOR

FOR «var»:=«#» TO «#» [STEP «#»] DO [«statement»]
FOR x:=10 TO 1 STEP -1 DO PRINT x
FOR delay:=1 TO amount DO NULL
FOR num:=1 TO total DO

Statement - Marks the start of a FOR structure or one line FOR statement. The variable is initialized to the start value before loop execution begins. A check is made that the variable value does not exceed the end value before executing the loop statements (it is possible for the loop to be skipped entirely if the start value exceeds the end value to begin with). If the step value is negative, the variable is decremented with each loop, rather than incremented. The variable may be an integer variable which yields faster execution in many systems. The statement block within a multi-line FOR structure is automatically indented when listed. The system will insert the word DO for you if you omit it.

The FOR loop variable is considered LOCAL to the FOR structure in Amiga, IBM, CP/M, C64 and C128 cartridge COMALs. Thus a for loop variable will not conflict with a variable of the same name in the main program.

```
FUNC test'for CLOSED
  FOR w:=5 TO 5 DO
    IF w<>5 THEN RETURN FALSE
  ENDFOR w
  FOR z:=1 TO 0 DO
    RETURN FALSE
  ENDFOR z
  FOR x:=10 TO 7 STEP -2 DO
    t:=x
  ENDFOR x
  IF t<>8 THEN RETURN FALSE
  FOR y#:=1 TO 1 DO
    RETURN TRUE
  ENDFOR y#
  RETURN FALSE
ENDFUNC test'for
```

38. Tests a FOR loop that is executed only once, then one that is not executed at all. Next it test a loop that is decremented, and finally an integer loop.

FUNC

FUNC «name»[(«parm»)] [CLOSED]
FUNC «name»[(«parm»)] EXTERNAL «filename»
FUNC call'answered EXTERNAL "call"
FUNC but'first$(text$) CLOSED
FUNC occurances#(text$,c$)

Statement - Marks the start of a user defined function. Parameter passing is allowed; parameters used are considered local to the function unless preceded by the REF keyword. If the statement ends with CLOSED, the function is considered a closed function, and all variables and arrays in it are unknown to the main program. Likewise, all variables and arrays in the main program are then unknown to the closed function. Use IMPORT or parameters to bring main program variables or arrays into a closed function. The block of statements inside the function definition are automatically indented when listed. Functions may be recursive.

String, external and nested functions are not supported by Apple COMAL, C64 COMAL 1.0 or C64 Power Driver. A closed function is more tightly closed in CP/M COMAL than other COMALs.

```
FUNC test'func CLOSED
  IF hope$<>"hope" THEN RETURN FALSE
  RETURN TRUE
  //
  FUNC hope$ CLOSED
    RETURN "hope"
  ENDFUNC hope$
  //
ENDFUNC test'func
```

39. Tests that ENDFUNC ends the function definition.

GET$

GET$(«filenum»,«# of characters»)
text$=GET$(2,16)

Function - Returns the specified number of characters from the specified file. The file must previously have been opened as a read type file. If the end of file is reached before the specified number of characters are retrieved, only those retrieved prior to EOF will be returned (there is no padding of spaces and no error occurs unless you attempt to read from the file again).

CP/M COMAL has difficulties with EOF while executing GET$ due to the manner that CP/M is structured.

GOTO

GOTO «label name»
GOTO jail

Statement - Transfers program execution to the line with the specified label name. Since COMAL has many structures and loop methods, GOTO is not required to be used other than in advanced programs. It is being considered to remove GOTO from the COMAL standard.

```
FUNC test'goto CLOSED
  GOTO this'line'is'making'me'ill
  RETURN FALSE
this'line'is'making'me'ill:
  RETURN TRUE
ENDFUNC test'goto
```

40. Tests that GOTO can jump over lines.

HANDLER

HANDLER

Statement - Marks the beginning of the error handling section of the TRAP HANDLER structure. The block of statements in the trapped section and the error handling section are automatically indented when listed.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support an error handler and thus do not have HANDLER.

```
FUNC test'handler CLOSED
  TRAP
    div0=5/0
    RETURN FALSE
  HANDLER
    RETURN TRUE
  ENDTRAP
  RETURN FALSE
ENDFUNC test'handler
```

41. Tests that the handler section is executed after an error.

IF

IF «condition» THEN [«statement»]
IF reply$ IN "yYnN" THEN

Statement - The start of a multi-line IF structure. May also be a one line IF statement (no ENDIF is used). IF allows conditional statement execution. The block of statements following the IF are only executed if the condition is TRUE. The block of statements are automatically indented when listed. The system will insert the word THEN for you if it is omitted.

```
FUNC test'if CLOSED
  t:=FALSE
  IF TRUE THEN
    t:=TRUE
  ENDIF
  IF t THEN RETURN TRUE
  RETURN FALSE
ENDFUNC test'if
```

42. Tests that statements are executed following a TRUE condition with a single line IF or multi-line IF structure.

IMPORT

IMPORT «identifier» {,«identifier»}
IMPORT running'total

Statement - Allows a closed procedure or function to use variables, arrays, procedures and functions from the main program. There may be more than one IMPORT statement in a procedure or function, and all IMPORT statements should come prior to any executable statement in that procedure or function.

IMPORT is not supported by C64 Power Driver, C64 COMAL 1.0 or Apple COMAL. In those systems all procedures and functions in the main program are automatically available inside closed procedures and functions; variables and arrays can be shared via parameter passing.

```
FUNC test'import CLOSED
  t:=FALSE
  p
  RETURN t
  //
  PROC p CLOSED
    IMPORT t
    t:=TRUE
  ENDPROC p
  //
ENDFUNC test'import
```

43. Tests that a variable outside a CLOSED procedure is accessible after an IMPORT statement.

IN

«string1» IN «string2»
IF guess$ IN word$ THEN winner

Operator - Returns the position of «string1» within «string2» (or 0 if not found). If «string1» is the null string ("") 0 is returned.

If «string1» is the null string, Alder returns 1; C64 cart, CP/M COMAL and C64 Power Driver return the length of the string plus 1.

```
FUNC test'in CLOSED
  DIM s$ OF 3
  s$:="abc"
  IF ("b" IN s$)<>2 THEN RETURN FALSE
  IF ("d" IN s$) THEN RETURN FALSE
  IF ("" IN s$) THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'in
```

44. Tests that substrings are found when possible; are not found when not possible; and that the null string is not found in any string.

INPUT

INPUT FILE «file#»[,«rec#»]: «var list»
INPUT [AT «row»,«col»[,«len»]:][«prompt»:]«vars»[«mark»]
INPUT FILE 2: text$
INPUT AT 5,2,10:"ZIP CODE: ": zip'code,
INPUT "Age? ":age
INPUT reply$

Statement - INPUT allows the user to enter data into a running program from the keyboard (the AT section is optional). INPUT FILE gets the data from the file specified, which must have been previously opened for reading. INPUT FILE reads ASCII files, such as those created by PRINT FILE or a Word Processor with ASCII file output (does not read files created by WRITE FILE statements). The prompt is optional and may be a variable.

During the INPUT from keyboard request, the input area is a protected field extending to the end of the line (unless the length part of the AT section is specified). A 0 length means only a carriage return will be accepted (*except in CP/M COMAL*). A 0 for the row or column means not to change it (stay in the same row or column). If the «mark» is a comma, the cursor remains where it is after the reply. If it is a semicolon, spaces are printed to the next zone (one space by default if ZONE is not specified), then the cursor remains at that position.

CP/M COMAL requires that the prompt be a constant if used. Alder, C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not allow a comma to be used as the mark at the end of the statement (only allow a semicolon).

INT

INT(«numeric expression»)
tally:+INT(number)

Function - Returns the nearest integer less than or equal to the specified number. Both positive and negative numbers are rounded down (-8.3 becomes integer -9).

```
FUNC test'int CLOSED
  IF INT(3)<>3 THEN RETURN FALSE
  IF INT(3.8)<>3 THEN RETURN FALSE
  IF INT(-3)<>-3 THEN RETURN FALSE
  IF INT(-3.3)<>-4 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'int
```

45. Tests that the value returned is the same for integers; and that the value returned is the nearest integer that is less than a non-integer.

KEY$

KEY$
WHILE KEY$="" DO NULL

Function - Returns the first character in the keyboard buffer. If no key has been pressed, the null string ("") is returned.

C64 cart and C64 Power Driver return CHR$(0) if no key has been pressed.

```
FUNC test'key CLOSED
  WHILE KEY$>CHR$(0) DO NULL
  IF KEY$>"" THEN RETURN FALSE
  IF LEN(KEY$)>0 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'key
```

46. Tests clearing the keyboard buffer, then checks that no key is left in the buffer. Finally it checks that the null string (length 0) is returned when there is no key pressed.

LABEL

[LABEL] «label name»:
months:

Identifier - Assigns a label name to the line. This label is only referenced by RESTORE or GOTO. It is non-executable and may be placed anywhere within a program as a one line statement. The line is not indented (*except in Amiga and CP/M COMAL*). You do not have to type the word label, and most systems do not list it either (similar to how EXEC is treated).

```
FUNC test'label CLOSED
  RESTORE right'here
  READ x
  IF x<> 5 THEN RETURN FALSE
  RETURN TRUE
  DATA 4
right'here:
  DATA 5
ENDFUNC test'label
```

47. Tests that a label can be used with RESTORE to reset the next data item pointer.

LEN

LEN(«string expression»)
length=LEN(text$)

Function - Returns the length of the specified string. All characters, even non-printing characters, are counted. The length of the null string "" is 0.

```
FUNC test'len CLOSED
  DIM s$ OF 5
  s$:=""
  IF LEN(s$)<>0 THEN RETURN FALSE
  s$:="12345"
  IF LEN(s$)<>5 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'len
```

48. Tests that the length of the null string is 0 and that the length of a string is equal to the number of characters assigned to it.

LINK

LINK «filename»
LINK "mouse"

Command - Loads a disk based package and links it to the program. A USE command is required in the program before the package commands are available to the program.

Amiga COMAL links a package into the system with the USE command.

LIST

LIST [«range»] [TO] [«filename»]
LIST header
LIST "myprog.lst"
LIST pause "pause.lst"

Command - Lists the specified program lines. If no lines are specified, all lines are listed. If a filename is specified, the lines are listed to that file (in ASCII form), otherwise they are listed to the current output location (screen by default). A procedure or function name can be used to specify a line range.

```
LIST 30     lists only line 30
LIST -30    lists all lines up to and including line 30
LIST 9000-  lists all lines after and including line 9000
LIST 100-200  lists lines 100 through 200 inclusive
LIST pause  lists all lines in procedure name pause
```

Lines LISTed to disk may be merged into the programs with the MERGE command (*C64 Power Driver, C64 COMAL 1.0 and Apple COMAL use the ENTER command for merges*). Statement blocks within structures are automatically indented when listed.

LIST and ENTER commands are useful when transferring programs from one system to another. LIST the program to disk. Then ENTER it into the other system. Modems and networks may also be used to aid the transfer.

If more lines than fit on one screen are to be listed, CP/M COMAL pauses after each screen... press space for the next screen. With Alder COMAL, <CTRL>+S is used to pause a listing. The other COMALs have <space> pause and restart a listing.

CP/M COMAL requires the filename to come before the line range (reverse of the other COMALs). CP/M COMAL requires that a line number used to specify a range must exist. Alder requires the keyword TO before a filename; without the word TO, the LIST becomes a global search for the text in quotes.

Amiga, IBM and C128 COMALs will re-list preceding lines if you cursor up on the top line of the screen. This is handy to see lines that have just scrolled off the top.

LOAD

LOAD «filename»
LOAD "menu"

Command - Loads a program from disk into the computers memory. Program memory is cleared before loading the program. The program being loaded must have previously been SAVEd to disk. You cannot LOAD a program SAVEd by a different COMAL implementation. To transfer programs between implementations, use LIST to disk, and ENTER to retrieve them.

LOG

LOG(«numeric expression»)
PRINT LOG(number);

Function - Returns the natural logarithm of the number specified. This is log to the base e. A good representation of e is 2.718281828.

```
FUNC test'log CLOSED
  IF LOG(1)<>0 THEN RETURN FALSE
  IF ABS(LOG(2.71828)-1)>0.00001 THEN RETURN FALSE
  IF ABS(EXP(LOG(10))-10)>0.000001 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'log
```

49. Tests the natural logarithm function.

LOOP

LOOP

Statement - Starts a multi-line loop structure that uses the EXIT statement as the exit method. The statement block within a LOOP structure are automatically indented when listed.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not have the LOOP structure.

```
FUNC test'loop CLOSED
  count:=0
  LOOP
    EXIT WHEN count=2
    count:+1
  ENDLOOP
  IF count<>2 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'loop
```

50. Tests exiting a loop after a certain number of passes.

MAIN

MAIN

Command - Returns to the main program section when a program is stopped while an external procedure or function is being executed.
Alder, C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not have a MAIN command.

MERGE

MERGE [«line#»[,«increment»]] «filename»
MERGE "readrec.lst"

Command - Merges program lines from a disk file (ASCII format). The lines are renumbered as they are merged into the current program.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL use the ENTER command to merge program segments. Alder, Amiga COMAL, and C64/C128 2.0 cartridge COMALs require line numbers in the file being merged, even though those line numbers are ignored.

MOD

«dividend» MOD «divisor»
color=number mod 16

Operator - Returns the modulo of the numbers. It can be used in conjunction with DIV. It defines x MOD z as x-(x DIV z)*z which expands into x-INT(x/z)*z. If z is negative, the result may be irrelevant, but should follow the definition.

When one of the numbers is negative, CP/M COMAL and C64 cartridge COMAL do not follow the definition.

```
FUNC test'mod CLOSED
  IF 500 MOD 256<>244 THEN RETURN FALSE
  IF 1500 MOD 5<>0 THEN RETURN FALSE
  IF (-7) MOD 3<>2 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'mod
```

51. Tests that MOD in cases where the result should be 0 as well as when the result should be a positive value. Then it tests MOD with a negative number.

NEW

NEW

Command - Erases the program currently in memory and clears all variables. Linked packages are also cleared.

Amiga COMALs request confirmation if a NEW command is issued and the program in memory has been modified and not saved.

NEXT : converted to ENDFOR, see ENDFOR

NOT

NOT «condition»
IF NOT ok THEN

Operator - Returns to reverse of the TRUE / FALSE evaluation:
NOT TRUE = FALSE
NOT FALSE = TRUE

```
FUNC test'not CLOSED
  IF (NOT TRUE)=TRUE THEN RETURN FALSE
  IF (NOT FALSE)<>TRUE THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'not
```

52. Tests the two possibilities of NOT: NOT TRUE and NOT FALSE.

NULL

NULL
WHILE KEY$="" DO NULL

Statement - Does nothing. It can be used as an empty statement, such as in a pause loop.

```
FUNC test'null CLOSED
  NULL
  RETURN TRUE
ENDFUNC test'null
```

53. Tests that a NULL statement is accepted.

OF : see DIM and CASE

OPEN

OPEN [FILE] «file#»,«filename»,«type»
OPEN FILE 2,"scores",READ

Command/Statement - Opens a file and assigns it a file number (that is used later with file operation statements). A file may be opened to the screen, printer and serial port as well as disk.

OR

«condition» OR «condition»
IF reply$<"a" OR reply$>"z" THEN

Operator - Returns the result of the logical OR of the two expressions. This is different than most BASICs in which OR is a bitwise operator. For bitwise OR in COMAL see BITOR.

OR	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

```
FUNC test'or CLOSED
  IF (TRUE OR TRUE)<>TRUE THEN RETURN FALSE
  IF (TRUE OR FALSE)<>TRUE THEN RETURN FALSE
  IF (FALSE OR TRUE)<>TRUE THEN RETURN FALSE
  IF (FALSE OR FALSE)<>FALSE THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'or
```

54. Tests the four possibilities of OR: both TRUE, both FALSE, and one TRUE - one FALSE.

ORD

ORD(«string expression»)
a=ORD("a")

Function - Returns an integer representing the ASCII code (ordinal number) of the specified string. If the string is longer than one character, ORD only looks at the first character. An error results if the null string is used. ORD is system dependent and may vary between systems (especially Commodore).

```
FUNC test'ord CLOSED
  IF ORD("5")<>53 THEN RETURN FALSE
  IF ORD(CHR$(65))<>65 THEN RETURN FALSE
  IF ORD("c")-ORD("a")<>2 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'ord
```

55. Tests that the ASCII value for "5" is correct, then that CHR$ is a correct complement to ORD, and that the letter c is 2 more than the letter a in ORD values.

OTHERWISE

OTHERWISE

Statement - Marks the start of the default case in the CASE structure. The block of statements after the OTHERWISE are executed if no WHEN case condition is met. The block of statements are indented automatically when listed.

```
FUNC test'otherwise CLOSED
  x:=3
  CASE x OF
  WHEN 1,2,4
    RETURN FALSE
  OTHERWISE
    RETURN TRUE
  ENDCASE
  RETURN FALSE
ENDFUNC test'otherwise
```

56. Tests that if no WHEN case is matched, the OTHERWISE section is executed.

PAGE

PAGE

Statement/Command - Clears the screen and puts the cursor at the top left corner (1,1). If output is to another device, a CHR$(12) is sent (form feed).

```
FUNC test'page CLOSED
  PAGE
  IF CURCOL<>1 THEN RETURN FALSE
  IF CURROW<>1 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'page
```

57. Tests that PAGE clears the screen and leaves the cursor in position 1,1 (top left corner).

PEEK

PEEK(«memory address»)
device=PEEK(4839)

Function - Returns the decimal value of the contents in the specified memory location. PEEK is very machine dependent.

PEEK is a package command in Amiga, Alder and IBM PC COMAL and requires a USE command prior to using it.

PI

PI
PRINT "Value of PI is";PI

Function - Returns the value of pi. The number of digits varies between systems. Generally, pi is equal to 3.14159266.

```
FUNC test'pi CLOSED
  IF PI<3.141592 THEN RETURN FALSE
  IF PI>3.141593 THEN RETURN FALSE
  RETURN ABS(PI-4*ATN(1))<1e-06
ENDFUNC test'pi
```

58. Tests that PI returns a close representation of the value of PI.

POKE

POKE «memory address»,«contents»
POKE 4839,13

Function - Places the specified decimal value into the memory indicated memory location. POKEing the wrong value into some memory locations may "lock out" your machine.

POKE is a package command in Alder, Amiga and IBM PC COMAL, and requires a USE statement prior to using it.

PRINT

PRINT [AT «row»,«col»:] [USING «form»:] «list»[«mark»]
PRINT [FILE «#»[,«rec»]:] [USING «form»:]«list»[«mark»]
PRINT FILE 2: text$
PRINT AT 9,1: USING "$###.##": amount

Statement/Command - Prints items as specified. More than one item may be specified in one PRINT statement, separated by a , or ;. A comma is a null separator (no spaces between items). A semicolon ; prints spaces to the next zone (one space by default if ZONE has not been specified).

PRINT FILE statements write items in ASCII to the file (it may be preferable to use WRITE FILE instead for data files). The AT and USING sections are optional parts of a PRINT statement, and are part of COMMON COMAL. They provide added flexibility.

CP/M COMAL issues a CHR$(9) for each comma (as a tab mark). C64 Power Driver and C64 cartridge COMAL use the original meanings of the comma and semicolon separators. Unless you use a ZONE statement, the results are the same. See ZONE.

```
FUNC test'print CLOSED
  cc=CURCOL
  PRINT "",
  IF CURCOL<>cc THEN RETURN FALSE
  zz=ZONE
  ZONE 5
  PRINT "";
  cc=CURCOL
  PRINT "";
  IF cc+5<>CURCOL THEN RETURN FALSE
  ZONE zz //reset zone
  RETURN TRUE
ENDFUNC test'print'print
```

59. Tests that printing a null string with a comma endmark does not move the cursor. Then it checks that zone works with the semicolon separator. The zone is reset after the tests.

PROC

PROC «name»[(«parm»)] [CLOSED]
PROC «name»[(«parm»)] [EXTERNAL «file»]
PROC readrec(number)
PROC compare(t1$,t2$) EXTERNAL "comp.ext"

Statement - Marks the start of a multi-line procedure definition, including parameter passing (parameters are considered local unless preceded by the REF keyword). A procedure may recursive. The CLOSED keyword is included at the end of the statement to make a procedure closed. A closed procedure does not know about variables or arrays in the main program (unless they are IMPORTed). Likewise, variables and arrays inside a closed procedure are local, and remain unknown to the main program.

A closed procedure can be used as an EXTERNAL procedure by SAVEing it to disk. You can define a procedure within another procedure (nested).

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not allow nested procedures or EXTERNAL procedures.

```
FUNC test'proc CLOSED
  x:=FALSE
  setx
  RETURN x
  //
  PROC setx
    x:=TRUE
  ENDPROC setx
  //
ENDFUNC test'proc
```

60. Tests that variables changed in an open procedure are also changed in the main program.

RANDOM

OPEN FILE «file#»,«filename»,RANDOM «len»
OPEN FILE 2,"subs",RANDOM 88

File Type - Identifies a file as random access, for both reading and writing.

Commodore COMALs have a 254 byte limit on record length. Alder COMAL also allows READONLY to be specified after the record length, in which case the file may only be read and not written to.

RANDOMIZE

RANDOMIZE [«seed»]
RANDOMIZE
RANDOMIZE num

Statement/Command - Randomizes the random number generator. This generates a series of pseudo random numbers. You only need to use the RANDOMIZE command once in your program (such as right the the very beginning).

Specify a "seed" number after RANDOMIZE and you cause a specific series of random numbers to be generated. The series of numbers will be the same each time that specific seed is used. This is helpful while testing a program that uses random numbers.

```
FUNC test'randomize CLOSED
  RANDOMIZE
  FOR x:=1 TO 50 DO
    r:=RND(1,20)
    IF (r<1) OR (r>20) THEN RETURN FALSE
  ENDFOR x
  DIM rarray(1:9)
  RANDOMIZE 9 //specific set of numbers
  FOR x=1 TO 9 DO rarray(x)=RND(1,99)
  RANDOMIZE 9 //reset to specific set
  FOR x=1 TO 9 DO
    IF RND(1,99)<>rarray(x) THEN RETURN FALSE
  ENDFOR x
  RETURN TRUE
ENDFUNC test'randomize
```

61. Tests that no seed is needed with RANDOMIZE. Then checks that a random number is in its correct range. Next it checks that a specified seed gives the same set of random numbers each time.

READ

READ [FILE «file#»[,«rec#»]:] «var list»
OPEN [FILE] «filenum»,«filename»,READ
READ name$,age
READ FILE 2,record: name$,adr$,city$,st$
OPEN FILE 3,filename$,READ

File Type or Statement - In an OPEN statement, specifies a sequential file to be read. READ also can be used as a statement to read data from DATA statements. Finally, READ FILE statements read data from sequential or random files that were created with WRITE FILE statements (these are binary files, not ASCII).

```
FUNC test'read CLOSED
  DIM s$ OF 6
  READ s$
  IF s$<>"passed" THEN RETURN FALSE
  READ x
  IF x<>13 THEN RETURN FALSE
  RETURN TRUE
  DATA "passed",13
ENDFUNC test'read
```

62. Tests that string and numeric data are properly read.

REF

REF «var»
PROC alter(REF text$) CLOSED
FUNC slide(REF text$)

Parameter Type - Specifies that the parameter will be an alias for the matching variable or array in the calling statement (passed by reference rather than by value). The value of the calling statement changes as its REF parameter is changed.

```
FUNC test'ref CLOSED
  setvar(x)
  RETURN x
  //
  PROC setvar(REF var) CLOSED
    var:=TRUE
  ENDPROC setvar
  //
ENDFUNC test'ref
```

63. Tests that variables used as REF parameters have their value updated as the parameter is changed within the procedure.

RENAME

RENAME «old filename»,«new filename»
RENAME "temp","final"

Statement/Command - Renames a disk file. Takes an existing file and gives it a new name.

RENUM

RENUM [«target start»][,«increment»]
RENUM 100
RENUM ,5
RENUM 9000,1

Command - Renumbers the program in memory. Valid line numbers are 1-9999. By default, it renumbers a program to start at line 10 and increment by 10, unless you specify otherwise.

REPEAT

REPEAT

Statement - Marks the start of a multi-line REPEAT structure. The block of statements after the REPEAT are automatically indented when listed. They are continually executed until the condition after the UNTIL evaluates to FALSE. The statements will always be executed at least once.

```
FUNC test'repeat CLOSED
  REPEAT
    READ x
    IF x=4 THEN RETURN FALSE
  UNTIL x=5
  IF x=5 THEN RETURN TRUE
  RETURN FALSE
  DATA 10,3,5,4
ENDFUNC test'repeat
```

64. Tests that a REPEAT loop is repeated and exited properly.

REPORT

REPORT [«error code»]
REPORT
REPORT 256

Statement - Part of the error handler structure. REPORT causes an error (optionally you can specify what error number to generate). This is useful when using multiple nested handlers. REPORT puts you into the next outer handler. If REPORT is issued while not in a handler section, the error is reported to the system. REPORT is very system dependent. All COMALs (*except CP/M COMAL and Alder COMAL*) allow you to include an error message along with the error number.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not support the error handler structure, thus do not have the REPORT statement.

```
FUNC test'report CLOSED
  TRAP
    REPORT 13
    RETURN FALSE
  HANDLER
    IF ERR=13 THEN RETURN TRUE
  ENDTRAP
  RETURN FALSE
ENDFUNC test'report
```

65. Tests that REPORT can issue a specific error number.

RESTORE

RESTORE [«label»]
RESTORE month'names
RESTORE

Statement/Command - Allows data in DATA statements to be re-used. The pointer to the next data item is reset back to the first data item, unless a label is specified. Then the next data item pointer points at the first data item following the label.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not allow labels to be used with RESTORE, but always restore to the first data item.

```
FUNC test'restore CLOSED
  DIM l$ OF 5
  RESTORE good'language
  READ l$
  IF l$<>"COMAL" THEN RETURN FALSE
  RETURN TRUE
good'language:
  DATA "COMAL"
another'language:
  DATA "BASIC"
ENDFUNC test'restore
```

66. Tests that RESTORE will restore the pointer to the next data item following the specified label.

RETURN

RETURN [«value»]
RETURN TRUE
RETURN text$

Statement - Assigns the value specified after the RETURN to the function and returns control to the calling statement. RETURN may also be used to terminate a procedure early.

```
FUNC test'return CLOSED
  x=0
  rtest'return
  IF x=0 THEN RETURN TRUE
  RETURN FALSE
  //
  PROC rtest'return CLOSED
    RETURN
    x=5
  ENDPROC rtest'return
  //
ENDFUNC test'return
```

67. Tests that RETURN will cause an early exit to a procedure and that RETURN will return values from a function.

RND

RND [(«start num»,«end num»)]
dice=RND(1,6)+RND(1,6)

Function - Returns a random number greater than or equal to zero, and less than 1. If start and end limits are specified, RND returns an integer within the specified limits, inclusive.

Alder COMAL uses RND# when specifying a random integer in a range. C64 Power Driver uses RND(0) rather than RND.

```
FUNC test'rnd CLOSED
  RANDOMIZE
  FOR x:=1 TO 50 DO
    r:=RND(1,20)
    IF (r<1) OR (r>20) THEN RETURN FALSE
    r:=RND
    IF (r<0) OR (r>1) THEN RETURN FALSE
  ENDFOR x
  RETURN TRUE
ENDFUNC test'rnd
```

68. Tests that RND can return integers within a specified range and that RND can return a random number between 0 and 1.

RUN

RUN [«filename»]
RUN
RUN "menu"

Command - Begins execution of the program currently in memory. If a file is specified, the memory is cleared and the file is loaded and run.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not allow a filename to be specified.

SAVE

SAVE «filename»
SAVE "zombies"

Command - Stores the program in memory to the specified file in compressed form. Comments are not removed. Later the program can be retrieved with the LOAD, RUN, or CHAIN command. Procedures or functions stored with the SAVE command can be used as EXTERNAL procedures and functions.

A SAVEd program may not be transferred to another COMAL system. To transfer a program it must be in ASCII form as with the LIST to disk command. Use the ENTER command to retrieve it to the other system. Modems or networks may also be used to transfer the file.

Some COMALs save linked packages with the program. Commodore COMALs will not save a program if one with the same name already exists. IBM COMAL overwrites any file with the same name (without warning or notification). Amiga COMAL will rename a file found with the same name (adding the .BKUP extension) and then save the program (along with an ICON image for it).

SCAN

SCAN

Command - Scans the program in memory for structure errors. Once a program has been SCANned or RUN procedures and functions may be called from direct mode.

SELECT

SELECT [OUTPUT] «type»
SELECT loc$

Statement/Command - Selects the output location. You do not have to type the keyword OUTPUT. If it is omitted, COMAL will insert it for you. Default location is the screen. Possible locations are system dependent, but most COMALs include a standard set of destinations among their choices. Amiga COMAL, CP/M COMAL, Apple COMAL, C64 Power Driver, C64 COMAL 1.0, C64 2.0 cartridge, C128 2.0 cartridge, PET COMAL 0.14, PET COMAL 1.02, and PET COMAL 2.0 include:

SELECT "ds:" Data Screen
SELECT "lp:" Line Printer

Most also include:

SELECT "sp:" Serial Port
SELECT "filename"

IBM PC COMAL does not recognize the standard destinations in addition to its own. It only uses the identifications set up by MS-DOS:

SELECT "con:" Console
SELECT "prn:" Printer
SELECT "lpt1:" Printer (also lpt2: and lpt3:)
SELECT "com1:" Serial Port (also com2:)

```
FUNC test'select CLOSED
  TRAP
    SELECT OUTPUT "lp:" //printer
    SELECT OUTPUT "ds:" //screen
  HANDLER
    RETURN FALSE
  ENDTRAP
  RETURN TRUE
ENDFUNC test'select'select
```

69. Tests that SELECT will accept the two values common between COMALs: "ds:" and "lp:".

SGN

SGN(«numeric expression»)
flag=SGN(number)

Function - Returns -1 if the number is negative. Returns 1 if the number is positive. Returns 0 if the number is 0.

```
FUNC test'sgn CLOSED
  IF SGN(1)<>1 THEN RETURN FALSE
  IF SGN(-1)<>-1 THEN RETURN FALSE
  IF SGN(0)<>0 THEN RETURN FALSE
  IF SGN(1000)<>1 THEN RETURN FALSE
  IF SGN(-0.01)<>-1 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'sgn
```

70. Tests that SGN returns -1, 1 or 0 as appropriate.

SIN

SIN(«numeric expression»)
plot(SIN(num),y)

Function - Returns the sine of the number in radians. *CP/M COMAL allows you to choose radians or degrees, but has radians as the default.*

```
FUNC test'sin CLOSED
  IF SIN(0)<>0 THEN RETURN FALSE
  IF ABS(SIN(PI/3)-SQR(3)/2)>0.000001 THEN RETURN
  FALSE //wrap line
  RETURN TRUE
ENDFUNC test'sin
```

71. Tests the sine trig function.

SIZE

SIZE

Command - Displays the amount of available free memory. Some COMALs display more information, such as program size and data area size. This is an informational display only. See FREE for a function that may be used in a program.

SPC$

SPC$(«number of spaces»)
PRINT SPC$(39)

Function - Returns the number of spaces specified.

```
FUNC test'spc CLOSED
  IF SPC$(5)<>"     " THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'spc
```

72. Tests that SPC$ returns the number of spaces specified.

SQR

SQR(«numeric expression»)
root=SQR(number)

Function - Returns the square root of the number.

```
FUNC test'sqr CLOSED
  IF ABS(SQR(1000)-31.62277)>0.00001 THEN RETURN FALSE
  IF ABS(SQR(8)*SQR(8)-8)>0.00001 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'sqr
```

73. Tests the square roots returned in two ways: subtracting the correct value from the SQR call, and multiplying the SQR with itself and then subtracting the original number. Both should have 0 for an answer.

STEP

STEP «numeric expression»
FOR x=1 TO max STEP 2 DO

Part of FOR statement - Sets the amount the FOR variable is incremented after each loop. If it is negative, the loop variable is decremented rather than incremented, and terminates when the variable value is less than the end amount. The step amount can be an integer or real numeric expression.

Note: a non-integer step size can lead to some "round off" problems due to the way addition is performed on the numbers.

```
FUNC test'step CLOSED
  count:=0
  FOR x:=1 TO 11 STEP 2 DO
    IF x MOD 2<>1 THEN RETURN FALSE
    count:+1
  ENDFOR x
  IF count<>6 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'step
```

74. Tests that a step of 2, starting with an odd number, will only result in odd numbers, and that the correct number of loops are made.

STOP

STOP [«message»]
STOP "now inside PROC remove'blank"

Statement - terminates program execution. Execution may be continued with the CON command. Variables may be displayed or changed before continuing. Lines may also be listed. However, if any lines are added, deleted, or modified the program may not be able to be restarted (due to internal tables).

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not allow a message as part of a STOP statement.

STR$

STR$(«number»)
zip$=STR$(number)

Function - Returns a string that is the equivalent of the number. The number 567 becomes "567". The VAL function does the reverse, converting a string into a number.

```
FUNC test'str CLOSED
  IF STR$(56)<>"56" THEN RETURN FALSE
  IF STR$(1.61803)<>"1.61803" THEN RETURN FALSE
  IF STR$(-13)<>"-13" THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'str
```

75. Tests the STR$ function with an integer, real number, and negative number.

TAB

TAB(«column number»)
PRINT TAB(col), name$

Function - Prints spaces up to the column specified. If that position is already exceeded, it goes to the specified position on the next line. TAB is always part of a PRINT statement.

CP/M COMAL converts the TAB function into: TAB «column number»

```
FUNC test'tab CLOSED
  PRINT TAB(25),"",
  IF CURCOL<>25 THEN RETURN FALSE
  CURSOR 0,1
  RETURN TRUE
ENDFUNC test'tab
```

76. Tests that TAB locates the cursor properly.

TAN

TAN(«numeric expression»)
PRINT TAN(number)

Function - Returns the tangent of the number in radians. *CP/M COMAL allows you to choose radians or degrees, but radians is the default.*

```
FUNC test'tan CLOSED
  IF TAN(0)<>0 THEN RETURN FALSE
  IF ABS(TAN(PI/4)-1)>0.000001 THEN RETURN FALSE
  IF  ABS(TAN(2*PI/3)+SQR(3))>0.000001  THEN  RETURN
FALSE
  RETURN TRUE
ENDFUNC test'tan
```

77. Tests the tangent trig function.

THEN

THEN
IF NOT ok THEN RETURN FALSE
ELIF errors>3 THEN

Part of IF - Part of the IF and ELIF statements. The system will insert the word THEN for you if you omit it.

```
FUNC test'then CLOSED
  IF FALSE THEN RETURN FALSE
  IF TRUE THEN
    RETURN TRUE
  ENDIF
  RETURN FALSE
ENDFUNC test'then
```

78. Tests that THEN is accepted in a single line and multi-line IF statement.

TO

«start num» TO «end num»
FOR x:=1 TO 4 DO

Part of FOR - Part of the FOR statement, separating the start and end numbers.

```
FUNC test'to CLOSED
  FOR x:=1 TO 0 DO
    RETURN FALSE
  ENDFOR x
  RETURN TRUE
ENDFUNC test'to
```

79. Tests that TO is accepted in a FOR statement, and that if the start value excedes the end value the loop is skipped.

TRAP

TRAP // start of error handler
TRAP ESC- // disable stop/break key
TRAP ESC+ // enable stop/break key

Statement - Marks the start of the error trap structure. Also is used to disable/enable the stop/break key.

C64 Power Driver, C64 COMAL 1.0 and Apple COMAL do not have the error handler structure. Alder uses TRAPESC (one word) to set the stop/break key disable, and 0 or 1 instead of + or -.

```
FUNC test'trap CLOSED
  t:=FALSE
  TRAP
    PRINT 2/0 // division by 0
    t:=FALSE
  HANDLER
    TRAP
      TRAP ESC- //disabled
      TRAP ESC+ //enabled
      t:=TRUE
    HANDLER
      NULL
    ENDTRAP
  ENDTRAP
  RETURN t
ENDFUNC test'trap
```

80. Tests the error trap structure. Then tests that the stop/break key can be disabled and enabled.

TRUE

TRUE
RETURN TRUE

System constant - Always equal to 1 when used as assignment. Other times it means not FALSE (a value that is not equal to 0).

```
FUNC test'true CLOSED
  IF TRUE<>1 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'true
```

81. Tests that TRUE equals 1 when used as an assignment.

UNIT

UNIT [«string expression»]
UNIT data'drive$
drive$:=UNIT$

Statement/Function - Sets the default unit when used as a statement. As a function, it returns the current unit name.

```
FUNC test'unit CLOSED
  RETURN ":" IN UNIT$
ENDFUNC test'unit
```

82. Tests that a proper unit is returned: it must contain a colon (:).

UNTIL

UNTIL «condition»
UNTIL reply$="q"

Statement - Marks the end of the REPEAT structure. Statements inside the structure are executed until the condition is TRUE.

```
FUNC test'until CLOSED
  x:=256
  REPEAT
    x:=x/2
  UNTIL x<1
  RETURN x<1
ENDFUNC test'until
```

83. Test that UNTIL exits a REPEAT loop correctly.

USE

USE «package name»
USE system

Statement - Activates a package that is linked to the program, making all its procedures and function accessible. Package "commands" are not available to CLOSED procedures and functions unless they are imported or another USE command is included in the CLOSED procedure or function.

Amiga COMAL also automatically does a LINK if needed. CP/M COMAL requires quotes around the package name: USE "mouse"

USING

PRINT USING «format»: «var list»
PRINT USING "##> $###.##": x, cash(x)
PRINT AT 8,5: USING "##": item

Special - Part of a PRINT statement allowing formatted output. Within the format string a # reserves a position for each possible digit of the

number; a period "." marks the decimal point location; a minus sign "-" is optional reserving a position for the negative sign. On the right of the decimal point, zeroes are padded where necessary. On the left of the decimal point, spaces are padded. All other characters (other than # . -) are printed as supplied. If the number has more digits than reserved, a * is printed in each reserved position.

```
FUNC test'using CLOSED
  PRINT // move to next line
  x=42.8923
  PRINT USING "Test'using -##.#": x,
  IF CURCOL<>11 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'using'using
```

84. Tests that the cursor is at the correct location after a PRINT USING statement is executed.

VAL

VAL(«numeric string»)
age=VAL(reply$)

Function - Returns the numeric value of a numeric string. This allows you to input data as strings, check them for errors, then convert them into numbers. VAL accepts the digits, + and - signs, decimal point, and exponential notation. Leading spaces are ignored by VAL.

Requesting a VAL of a non-numeric string results in an error in most systems. UniComal ignores the rest of the string beginning at the first non-numeric character (but gives an error if it's the first character). C64 Power Driver, C64 COMAL 1.0 and Apple COMAL return 0 as the VAL of non-numeric strings (because they have no error handling system to trap a failed VAL).

```
FUNC test'val CLOSED
  IF VAL("56")<>56 THEN RETURN FALSE
  IF VAL("1.125")<>1.125 THEN RETURN FALSE
  IF VAL("-13")<>-13 THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'val
```

85. Tests that the proper value is returned for an integer, real number and negative number.

WHEN

WHEN «list of values»
WHEN "Jan","jan"
WHEN 1,2

Statement - Provides a specific case within a CASE structure. One or more values are listed after the WHEN. If any match the current case value, its following statements are executed. Statement blocks are indented automatically when listed.

Some COMALs also allow WHEN to be used as a conditional EXIT from the LOOP structure:
EXIT WHEN «condition»

In CP/M COMAL and Amiga COMAL, EXIT and EXIT WHEN statements will exit all loops, not just the LOOP structure. Since this may cause COMMON programs to run incorrectly, we consider it a flaw.

```
FUNC test'when CLOSED
  x:=5
  CASE x OF
  WHEN 5
    x:=6
  WHEN 6
    RETURN FALSE
  OTHERWISE
    RETURN FALSE
  ENDCASE
  RETURN TRUE
ENDFUNC test'when
```

86. Tests that only one WHEN case statements are executed by changing the case variable in one section to a valid value on a following WHEN condition. Only the first WHEN section should be executed.

WHILE

WHILE «expression» [DO] [«statement»]
WHILE NOT EOF(infile) DO process
WHILE errors<3 DO

Statement - Marks the start of a multi-line WHILE structure. As long as its condition is true the statements are executed. If the condition is FALSE right at the start, the statements are skipped over. Some COMALs also allow a one line WHILE statement that does not need an ENDWHILE. The system will insert the word DO if you omit it.

```
FUNC test'while CLOSED
  DIM a$ OF 20
  a$:="never stop"
  WHILE a$<>"stop" DO
    a$:=a$(2:)
  ENDWHILE
  IF a$<>"stop" THEN RETURN FALSE
  RETURN TRUE
ENDFUNC test'while
```

87. Tests that a WHILE loop continues executing until its condition is met.

WRITE

WRITE FILE «file#»[,«rec#»]:«var»
OPEN [FILE] «filenum»,«filename»,WRITE
WRITE FILE 2: name$
OPEN FILE 3,"scores",WRITE

Statement/Command/File Type - Writes data to a file in binary form (not ASCII). As a file type, it specifies that the file is to be written to. Data written by a WRITE FILE statement may be read with a READ FILE statement, and is not compatible between implementations.

ZONE

ZONE [«tab interval»]
ZONE 5
z:=ZONE

Statement/Function - Returns the current zone setting as a function. As a statement it sets the zone to the number specified. The zone determines the interval in tab positions on an output line. The default zone is 1 (a zone at each column). The semicolon is the zone separator (for PRINT statements and at the end of INPUT statements). By default, a semicolon outputs one space (if a ZONE statement is previously used, it outputs spaces to the next zone). When a comma is used as a separator in a PRINT statement (or at the end of an INPUT statement) no spaces are printed, and the cursor remains where it is (null separator).

C64 Power Driver, C64 2.0 cartridge, Alder and CP/M COMAL follow the original standard: the comma as the zone separator (with default zone 0) and the semicolon always outputing one space. If no ZONE statement is used in a program, it will work with the either standard. Alder uses GETZONE as the function that returns the current zone.

```
FUNC test'zone CLOSED
  PRINT // force new line
  z:=ZONE
  ZONE 10
  PRINT "";"";
  IF CURCOL<>21 THEN RETURN FALSE
  PRINT "",
  IF CURCOL<>21 THEN RETURN FALSE
  CURSOR 0,1
  ZONE z
  RETURN TRUE
ENDFUNC test'zone
```

88. Tests that the zone settings work with the semicolon zone separator.

Common COMAL is a trademark of COMAL Users Group, U.S.A., Limited.

Special Notes

C64
Power Driver is the current disk loaded COMAL for the C64 or C128. It has more commands, more user memory, and more capabilities than COMAL 0.14. It has been upgraded to C64 COMAL 1.0, but that upgrade has a fatal flaw with the NEW command. After months of efforts to find and fix this bug, the project was postponed and it will be rewritten from scratch.

The COMAL 2.0 cartridge is more powerful, faster, and easier to use, but its programs require the cartridge to be run, while Power Driver programs can be compiled with the compiler in Power Box.

IBM PC, compatibles, and PS/2
UniComal IBM PC COMAL 2.2 is the current version of COMAL for the IBM. The Mytech COMAL project was cancelled.

CP/M
CP/M COMAL 2.10 is the original and current release of COMAL for CP/M. It will run on the C128 in CP/M mode. A demo disk is available if you wish to test it (and its installation).

Amiga
AmigaCOMAL from ComWare is being distributed in Scandanavia and Germany. It will not be released in the USA / Canada until fall according to ComWare. We previously incorrectly referred to it as German Amiga COMAL. Borge Christensen, founder of COMAL, is involved in this project. The system is written in machine code for speed.

Another company, Alder (formerly Mytech), has a COMAL for the Amiga and said that it is shipping now, however, we have not yet seen it ourselves. Alder COMAL is written in C.

When we refer to Amiga COMAL in our articles, we are usually referring to ComWare AmigaCOMAL (Germany/Denmark). The other implementation from Alder is referred to as Alder COMAL (formerly Mytech COMAL).

Apple
Apple COMAL 1.0 should be released by the time you read this. There is no graphics in this first release. A future release of version 2.0 is being planned now, and will include more features.

PET
One of the original implementations of COMAL was on the Commodore PET computer. It is still available and being used. Disk loaded PET COMAL 0.14 is current. A plug in board for version 2.0 was also available.