COMAL Commands

COMAL Flags & Reserved Variables

EOD	EOD	End Of Data flag
EOF	EOF(<filenum>)	End Of File flag
ESC	ESC TRAP ESC<type>	stop key pressed flag
FALSE	FALSE	predefined value = 0
STATUS$	STATUS$	status of disk channel
TRUE	TRUE	predefined value of 1

Note 1: Commodore BASIC, with the exception of a few commands, is a subset of COMAL. COMAL has all but ASC, CLR, DEF FN, GOSUB & RETURN, POS, REM, USR, VERIFY, WAIT, and BASIC 4.0 Disk Commands are sent via the COMAL PASS Command; other I/O commands (DLOAD, DCLOSE, RECORD#, etc) are much like BASIC 2.0 format.

Note 2: GOSUB (and ON...GOSUB) & RETURN are replaced by PROC Commands

Format: ( ) Numeric Brackets - numeric input required
< > Angle Brackets - denotes user supplied input
[ ] Square Brackets - indicates optional input

Thus: [(< >)] would specify the user supplied input must be of numeric nature, if the option is exercised.

Commands Common to COMAL and CBM BASIC With NO Differences

ABS	gives the absolute value
AND	logical AND
ATN	arctangent in radians
CHR$	gives that numbers character
COS	cosine in radians
DATA	provides data for a READ
END	halt program execution
EXP	natural log e to n
INT	gives nearest integer less than or equal
LEN	gives the length of string
LET	assign value to variable
LOG	natural logarithm of n
NEW	clears program from memory
NOT	logical NOT
OR	logical OR
PEEK	look at memory
POKE	change memory location
RESTORE	reuse DATA with READ
RUN	run program now in memory
SGN	-1 if neg, 0 if 0, 1 if pos
SIN	gives sine in radians
SQR	gives square root
STOP	halt program execution
SYS	transfer control to assembly language
TAB	print spaces up to specified column
TAN	gives tangent in radians
THEN	part of IF structure
TO	increment FOR variable start TO end

SPECIAL INFO

Line numbers allowed: 1-9999.
Identifiers up to 16 chars (unshifted alpha, digits, [,], ', <-, ↑)
Null input is accepted.

First time into graphics:	SETGRAPHIC 0
After that simply:	SETGRAPHIC

RUN/STOP RESTORE keys restore default colors.

To clean up the identifier name table:	LIST "PROGRAM.L"
(frees up memory, removes unused identifiers)	NEW ENTER "PROGRAM.L"
Save a program to disk:	SAVE "PROGRAM"
Load a program from disk:	LOAD "PROGRAM"
List a program to printer:	SELECT "LP:" LIST

COMAL 64 Colours List (COMAL 0.14/2.0)

Number	Colour	CHR$	Number	Colour	CHR$
0	BLACK	144	8	ORANGE	129
1	WHITE	5	9	BROWN	149
2	RED	28	10	LIGHT RED	150
3	CYAN	159	11	DARK GREY	151
4	PURPLE	156	12	MEDIUM GREY	152
5	GREEN	30	13	LIGHT GREEN	153
6	BLUE	31	14	LIGHT BLUE	154
7	YELLOW	158	15	LIGHT GREY	155

COMAL Commands NOT Found in CBM BASIC (* except BASIC 3.5)

***AUTO**	AUTO [<start line>][,<increment>]	automatic line numbering
BASIC	BASIC	back into BASIC mode
CASE	CASE <control expression> [OF]	multiple choice decisions
CHAIN	CHAIN <filename>	load & run program on disk
CLOSED	PROC <procname>[(params)] [CLOSED] FUNC <funcname>[(params)] [CLOSED]	all proc or func variables local
***DEL**	DEL <range>	deletes lines
DIV	<dividend> DIV <divisor>	division with integer answer
***DO**	DO <statements>	do the following statements
EDIT	EDIT [<range>]	lists lines without indentations
ELIF	ELIF <expression> [THEN]	short for ELSE IF condition
***ELSE**	ELSE	alternative statements in IF structure
ENDCASE	ENDCASE	end of CASE structure
ENDFOR	ENDFOR [<control variable>]	end of FOR structure
ENDFUNC	ENDFUNC [<function name>]	end of function
ENDIF	ENDIF	end of IF structure
ENDPROC	ENDPROC [<procedure name>]	end of procedure
ENDWHILE	ENDWHILE	end of WHILE structure
ENTER	ENTER <filename>	merge a program segment from disk
EXEC	[EXEC] <procname>[(<actual parameter list>)]	execute a procedure
FUNC	FUNC <name>[(<params>)] [EXTERNAL <filename>] FUNC <name>[(<params>)] [CLOSED]	start of a multiline function
IN	<string1> IN <string2>	locate position of string1 within string2
KEY$	KEY$	scans keyboard (not in PET COMAL 0.14)
LABEL	<label name>:	assigns a label name to the line
MOD	<dividend> MOD <divisor>	gives remainder of division (modulo)
NULL	NULL	does nothing (no op)
OF	CASE <expression> [OF] DIM <stringvar> OF <max char> DIM <stringarray>(array index) OF <max char>	part of DIM or CASE structure
OTHERWISE	OTHERWISE	default for CASE
PROC	PROC <name>[(<params>)] [EXTERNAL <filename>] PROC <name>[(<params>)] [CLOSED]	start of multiline procedure
RANDOM	OPEN FILE <filenum>,<filename>,RANDOM <recln>	random access disk file
RANDOMIZE	RANDOMIZE	generate new random numbers
REF	REF <var>	param var used in reference in proc
***RENUM**	RENUM [<targetstart>][,<increment>]	renumber program
REPEAT	REPEAT	start of REPEAT structure
***TRAP**	TRAP ESC<type>	disable stop key
***UNTIL**	UNTIL <expression>	end of REPEAT loop
***USING**	PRINT USING <format>: <var list> PRINT [FILE <filenum>:] USING <format>:<vars>	allows formatted output (not PET 0.14) including FILE output.
WHEN	WHEN <list of values>	choice in CASE structure
***WHILE**	WHILE <expression> [DO] [<statement>]	start of WHILE structure
WRITE	WRITE FILE <filenum>[,<recnum>]: <var list> OPEN [FILE] <filenum>,<filename>,WRITE	write to a file
ZONE	ZONE <tab interval> ZONE	tab increment

Commands Common to COMAL and CBM BASIC With SLIGHT Differences

//	//[<anything>]	allows comments in a program
APPEND	OPEN [FILE] <filenum>,<filename>,APPEND	start at end of seq file
CAT	CAT [<drive number>]	gives disk directory
CLOSE	CLOSE [FILE] [<filenum>]	closes files
CON	CON	continue program execution
DELETE	DELETE <filename>	deletes a file from disk
DIM	DIM <string var> OF <max char> DIM <str array>(<array index>) OF <max char> DIM <array name>(<array index>)	reserves/allocates string & array space
FILE	INPUT FILE <filenum>[,<recnum>]: <var list> PRINT FILE <filenum>[,<recnum>]: <val list> READ FILE <filenum>[,<recnum>]: <var list> WRITE FILE <filenum>[,<recnum>]: <var list> OPEN [FILE] <filenum>,<filename>[,<type>] CLOSE [FILE] [<filenum>]	specifies that a file is to be used
FOR	FOR <var> = <start> TO <end> [STEP <step>] [DO]	start of FOR loop structure
GOTO	GOTO <label name>	go to line with this name
IF	IF <condition> [THEN] IF <condition> THEN <statement>	start of conditional IF structure
INPUT	INPUT [<prompt>:] <var list> INPUT FILE <filenum>[,<recnum>]:<var list>	input from keyboard or file
LIST	LIST [<range>] [<filename>]	list program
LOAD	LOAD <filename>	load a program from disk
OPEN	OPEN [FILE] <filenum>,<filename>[,<type>]	open a file
ORD	ORD(<string expression>) (same as ASC(in BASIC)	returns integer representing the char
OUTPUT	SELECT [OUTPUT] <type>	select output location **Like CMD**
PASS	PASS <disk command>	passes a string to disk command channel
PRINT	PRINT [FILE <filenum>:] [<items>] PRINT [FILE <filenum>:] USING <format>:<vars> (RANDOM file use: [FILE <filnum>,<recnum>:])	prints items to screen/printer/file
READ	READ <var list> READ FILE <filenum>[,<rec num>]: <var list> OPEN [FILE] <filenum>,<filename>,READ	read data from DATA line or file
RND	RND(<num>) RND(<start num>:<end num>)	random number
SAVE	SAVE <filename>	record program on disk
SELECT	SELECT [OUTPUT] <type>	choose output location
SIZE	SIZE	reports on memory usage (free memory)
STEP	STEP <numeric expression>	increment FOR loop var by this amount
UNIT	OPEN FILE <#>,<nm>,UNIT <dev>[,<sec>][,<typ>]	specify unit (device)

SPRITES (COMAL 0.14/2.0)

DATACOLLISION	DATACOLLISION <sprite#>,<reset collisn flg?>	test for collision with data
DEFINE	DEFINE <sprite definition num>,<64 byte def$>	set up a sprite image for later use
HIDESPRITE	HIDESPRITE <sprite number>	turn off specified sprite
IDENTIFY	IDENTIFY <sprite number>,<definition number>	assign a sprite an image
PRIORITY	PRIORITY <sprite number>,<data priority?>	does data has priority over sprite
SPRITEBACK	SPRITEBACK <color1>,<color2>	set two multicolor sprite colors
SHOWSPRITE	SHOWSPRITE <sprite number>	turn on specified sprite
SPRITECOLLISION	SPRITECOLLISION <sprite#>,<reset collsn flg?>	test for sprite collision
SPRITECOLOR	SPRITECOLOR <sprite number>,<color number>	set color of sprite
SPRITEPOS	SPRITEPOS <sprite#>,<x coord>,<y coord>	position sprite at x,y location
SPRITESIZE	SPRITESIZE <sprite#>,<y expand?>,<x expand?>	set sprite size (expand or not)

HIGH RES and TURTLE Graphics (COMAL 0.14/2.0)

BACK	BACK <length>	move turtle backwards
BACKGROUND	BACKGROUND <color number>	set the screen background color
BORDER	BORDER <color number>	set the screen border color
CLEAR	CLEARSCREEN	clear the graphics screen (in frame)
DRAWTO	DRAWTO <x coordinate>,<y coordinate>	draws a line from current point
FILL	FILL <x coordinate>,<y coordinate>	fills in area with current color
FORWARD	FORWARD <length>	move turtle forward
FRAME	FRAME <x0>,<x1>,<y0>,<y1>	set up a screen window
FULLSCREEN	FULLSCREEN	fullscreen graphics (f5)
HIDETURTLE	HIDETURTLE	make the turtle invisible
HOME	HOME	put the turtle in its home position
LEFT	LEFT <degrees>	turn turtle left
MOVETO	MOVETO <x coordinate>,<y coordinate>	move to specified point without line
PENCOLOR	PENCOLOR <color number>	sets the current turtle pen color
PENDOWN	PENDOWN	put pen down, turtle draws line
PENUP	PENUP	pick up pen, turtle doesn't draw line
PLOT	PLOT <x coordinate>,<y coordinate>	plot a point in current color
PLOTTEXT	PLOTTEXT <x coord>,<y coord>,<text$>	print text on graphics screen
RIGHT	RIGHT <degrees>	turn turtle right
SETGRAPHIC	SETGRAPHIC [<type>]	turn on graphics screen
SETHEADING	SETHEADING <degree>	set turtle heading
SETTEXT	SETTEXT	turn on text screen (f1)
SETXY	SETXY <x coordinate>,<y coordinate>	set turtle x and y coordinates
SHOWTURTLE	SHOWTURTLE (note: sprite 7 is used for the turtle)	make turtle visible
SPLITSCREEN	SPLITSCREEN	2 text lines above graphics (f3)
TURTLESIZE	TURTLESIZE <size>	set turtle size (0 to 10)

TURTLE GRAPHICS CHART

Turtle Control:	**CBM LOGO**	**CBM COMAL**
Move forward length	FORWARD	FORWARD
Move backward length	BACK	BACK
Home turtle	HOME	HOME
Turn turtle left	LEFT	LEFT
Turn turtle right	RIGHT	RIGHT
Move to a point	SETXY	SETXY
Turn to specific heading	SETHEADING	SETHEADING
Make turtle visible	SHOWTURTLE	SHOWTURTLE
Make turtle invisible	HIDETURTLE	HIDETURTLE
Pen up off paper	PENUP	PENUP
Pen down on paper	PENDOWN	PENDOWN
Set pen color	PENCOLOR	PENCOLOR
Number of colors	16	16
Set size of turtle	–	TURTLESIZE
Plot a point	–	PLOT
Print text in graphics	?	PLOTTEXT
Screen And Colour Control:		
Set screen window	?	FRAME
Clear graphics screen	CLEARSCREEN	CLEAR
Set to graphics mode	DRAW	SETGRAPHIC
Set to text screen	NODRAW	SETTEXT
Set background color	BACKGROUND	BACKGROUND
Set border color	–	BORDER
Fill in an area	–	FILL
Full screen mode	FULLSCREEN	FULLSCREEN
Split screen mode	SPLITSCREEN	SPLITSCREEN
Function Key Actions:		
F1	TEXT SCREEN	TEXT SCREEN
F3	SPLITSCREEN	SPLITSCREEN
F5	FULLSCREEN	FULLSCREEN

COMAL 2.0 Library Descriptions

Library (page $80, $A59A-$BFF1):

A5C1 Sense routine

PACKAGE **english:**

A686 Init routine

PACKAGE **dansk:**

A68C Init routine

PACKAGE **system:**

CA2F Init routine
A80B PROC setprinter(str)
A96A PROC hardcopy(str)
A976 PROC setrecorddelay(int)
A97D PROC setpage(int)
A984 FUNC inkey
A9B6 FUNC free
A9C3 PROC keywords'in'upper'case(int)
A9C6 PROC names'in'upper'case(int)
A9C9 PROC quote'mode(int)
A9E1 FUNC currow
A9E9 FUNC curcol
A9F6 PROC textcolors(int,int,int)
AA34 PROC defkey(int,str)
AA7F PROC showkeys
AB21 PROC bell(int)
AB2D PROC serial(int)
A7FF PROC settime(str)
A805 FUNC gettime
A878 PROC getscreen(REF str)
A87B PROC setscreen(REF str)

Library (page $83, $800F-$C000):

8081 Sense routine

PACKAGE **graphics:**

8CDC Init routine
95CB PROC window(real,real,real,real)
8F15 PROC viewport(int,int,int,int)
8CA3 PROC drawto(real,real)
8ADA PROC draw(real,real)
8B06 PROC plot(real,real)
8C7C PROC moveto(real,real)
8AE8 PROC move(real,real)
A62A PROC circle(real,real,real)
A64F PROC arc(real,real,real,real,real)
A564 PROC arcl(real,real)
A55B PROC arcr(real,real)
9426 PROC textstyle(int,int,int,int)
9157 PROC plottext(real,real,str)
8D9B PROC pencolor(int)
8DBE PROC textcolor(int)
8FC3 FUNC getcolor(real,real)
A37B PROC fill(real,real)
A380 PROC paint(real,real)
9496 PROC background(int)
9483 PROC textbackground(int)
950B PROC border(int)
951E PROC textborder(int)
8E2A PROC graphicscreen(int)
90FC PROC textscreen
A25D PROC splitscreen
A258 PROC fullscreen
88FA PROC clearscreen
895E PROC clear
A23B PROC showturtle
A248 PROC hideturtle
A20F PROC turtlesize(real)
90A9 FUNC xcor
90D6 FUNC ycor
8CA3 PROC setxy(real,real)
904D PROC setheading(real)
9094 FUNC heading
903F PROC left(real)
903C PROC right(real)
901A PROC forward(real)
9017 PROC back(real)
9536 PROC penup
9542 PROC pendown
954E PROC home
9576 PROC wrap
9584 PROC nowrap
A8D7 FUNC inq(int)
AFD7 PROC savescreen(str)
B027 PROC loadscreen(str)
ADF4 PROC printscreen(str,int)

PACKAGE **turtle:**

8CE2 Init routine
9017 PROC bk(real)
9496 PROC bg(int)
88FA PROC cs
901A PROC fd(real)
A248 PROC ht
903F PROC lt(real)
8D9B PROC pc(int)
9542 PROC pd
9536 PROC pu
903C PROC rt(real)
904D PROC seth(real)
A23B PROC st
9483 PROC textbg(int)
95CB PROC window(real,real,real,real)
8F15 PROC viewport(int,int,int,int)
8CA3 PROC drawto(real,real)
8ADA PROC draw(real,real)
8B06 PROC plot(real,real)
8C7C PROC moveto(real,real)
8AE8 PROC move(real,real)
A62A PROC circle(real,real,real)
A64F PROC arc(real,real,real,real,real)
A564 PROC arcl(real,real)
A55B PROC arcr(real,real)
9426 PROC textstyle(int,int,int,int)
9157 PROC plottext(real,real,str)
8D9B PROC pencolor(int)
8DBE PROC textcolor(int)
8FC3 FUNC getcolor(real,real)
A37B PROC fill(real,real)
A380 PROC paint(real,real)
9496 PROC background(int)
9483 PROC textbackground(int)
950B PROC border(int)
951E PROC textborder(int)
8E2A PROC graphicscreen(int)
90FC PROC textscreen
A25D PROC splitscreen
A258 PROC fullscreen
88FA PROC clearscreen
895E PROC clear
A23B PROC showturtle
A248 PROC hideturtle
A20F PROC turtlesize(real)
90A9 FUNC xcor
90D6 FUNC ycor
8CA3 PROC setxy(real,real)
904D PROC setheading(real)
9094 FUNC heading
903F PROC left(real)
903C PROC right(real)
901A PROC forward(real)
9017 PROC back(real)
9536 PROC penup
9542 PROC pendown
954E PROC home
9576 PROC wrap
9584 PROC nowrap
A8D7 FUNC inq(int)
AFD7 PROC savescreen(str)
B027 PROC loadscreen(str)
ADF4 PROC printscreen(str,int)

PACKAGE **sprites:**

98B9 Init routine
9979 PROC define(int,str)
9B0D PROC identify(int,int)
99AC PROC spritecolor(int,int)
99BB PROC spritepos(int,int,int)
9A4A PROC spritesize(int,int,int)
9B46 PROC showsprite(int)
9B52 PROC hidesprite(int)
9A83 PROC spriteback(int,int)
9A93 FUNC spritecollision(int,int)
9A96 FUNC datacollision(int,int)
9ABF PROC priority(int,int)
AB54 PROC linkshape(int)
AB5A PROC loadshape(int,str)
AB6E PROC saveshape(int,str)
9B6F PROC movesprite(int,int,int,int,int)
9A11 PROC stopsprite(int)
9DFC PROC animate(int,str)
9D13 FUNC moving(int)
9D1F PROC startsprites
9CEB FUNC spritex(int)
9CFF FUNC spritey(int)
9D3F FUNC spriteinq(int,int)
9ECD PROC stampsprite(int)

PACKAGE **font:**

CA2F Init routine
ABD0 PROC linkfont
ABDF PROC loadfont(str)
AC49 PROC keepfont
ABF1 PROC savefont(str)
AC57 PROC getcharacter(int,int,REF str)
AC87 PROC putcharacter(int,int,str)

PACKAGE **sound:**

B287 Init routine
B2FE PROC note(int,str)
B3DE PROC pulse(int,int)
B3FA PROC gate(int,int)
B412 PROC soundtype(int,int)
B436 PROC ringmod(int,int)
B455 PROC sync(int,int)
B474 PROC adsr(int,int,int,int,int)
B4AD PROC filterfreq(int)
B4CD PROC resonance(int)
B4E6 PROC filter(int,int,int,int)
B508 PROC filtertype(int,int,int,int)
B52C PROC volume(int)
B543 FUNC env3
B549 FUNC osc3
B54F FUNC frequency(str)
B55B PROC setscore(int,REF int(),REF int(),REF int())
B59F PROC playscore(int,int,int)
B5CD PROC stopplay(int,int,int)
B5FC FUNC waitscore(int,int,int)
B2E3 PROC setfrequency(int,real)

PACKAGE **paddles:**

CA2F Init routine
B62C PROC paddle(int,REF real,REF real,REF real,REF real)

PACKAGE **joysticks:**

CA2F Init routine
B6B9 PROC joystick(int,REF real,REF real)

PACKAGE **lightpen:**

B77D Init routine
B7FA PROC offset(int,int)
B7D1 FUNC penon
B79B PROC readpen(REF real,REF real,REF real)
B820 PROC timeon(int)
B82A PROC delay(int)
B80D PROC accuracy(int,int)