STARTING
LOAD "BOOT*",8
RUN

COMAL (.14)

(Product of UNI COMAL APS)
COMAL USERS GROUP USA LTD.

LEROY'S CHEATSHEET® Keyboard Overlay for the COMMODORE 64

STEPS TO TYPE IN COMAL PGM

1. NEW
2. AUTO
3. TYPE PROGRAM
4. HIT RETURN TWICE When done
5. SAVE "NAME"
6. LIST (To Check it)
7. RUN

UTILITY COMMANDS

LOAD *"filename"*	LOAD PROGRAM from disk
CHAIN *"filename"*	LOAD PROGRAM & RUN
SAVE *"filename"*	SAVE PROGRAM TO DISK
NEW	CLEAR PROGRAM from memory
RUN	RUN CURRENT PROGRAM in memory
CAT [*disk#*]	VIEW DIRECTORY of *disk*
BASIC	RETURN to BASIC mode
SIZE	DISPLAY available memory size

EDIT COMMANDS

AUTO [*startline*][,*increment*]	AUTOMATIC LINE NUMBERING
RENUM [*startline*][,*increment*]	RENUMBER program
DELETE *"driven:filename"*	DELETE FILE from Disk
DEL [*startline—endline*]	DELETE range of LINES
EDIT [*startline—endline*]	LIST FILE without indentations
LIST [*startline—endline*]	LIST FILE
LIST *"filename"*	CREATE SOURCE program on disk
ENTER *"filename"*	MERGE SOURCE program from disk

DISK COMMANDS

PASS "NB:*diskname,id"*	FORMAT	DISK
PASS "CB:*newfile=***B:***oldfile"*	COPIES	FILE
PASS "RB:*newname=oldname"*	RENAME	FILE
PASS "SB:*filename"*	SCRATCH	FILE
PASS "VB"	VALIDATE	DISK
PASS "IB"	INITIALIZE	DISK
CAT	DISK DIRECTORY	
DELETE "B:*filename"*	SCRATCH	FILE

SUBROUTINE STATEMENTS

FUNC *name* [(*parameters*)] [**CLOSED**] *statements...*	START of FUNCTION (CLOSED=Local variables)
ENDFUNC [*functionname*]	END of FUNCTION
PROC *name* [(*parameters*)] [**CLOSED**] *statements...*	START of PROCEDURE (CLOSED=local variables)
ENDPROC [*procedurename*]	END of PROCEDURE
REF *variable*	Next variable is passed by reference
RETURN [*variable*]	Function: Return value Procedure: Go back to calling statement

STRING FUNCTIONS

CHR$(*exp*)	CONVERT *exp* to ASCII character
LEN(*string*)	RETURN LENGTH of *string*
ORD(*string*)	RETURN ASCII VALUE of FIRST CHARACTER in *string*
s1 **IN** *s2*	LOCATE *string (s1)* POSITION in string (*s2*)

COMAL (.14)

STATEMENTS

// text; COMMENT (up to return)
CON CONTINUE PROGRAM EXECUTION
DATA *c1* [,*c2*,...,*cn*] DEFINE READ DATA
DIM [*name*] **OF** [*maxchar*] Allocate STRING VARIABLE
DIM [*name*] (*index*) **OF** [*maxchar*] Allocate STRING ARRAY
DIM [*name*] (*index*) Allocate ARRAY
DIV INTEGER DIVISION
END [*labelname*] STOP PROGRAM execution
:= Assign *labelname* to line
= ASSIGN VALUE to variable
NULL NO-OP (no operation)
PEEK(*adress*) LOOK AT MEMORY *address*
POKE *address,value* PUT VALUE INTO MEMORY *address*
RESTORE RESET DATA FOR READ
SYS *address* JUMP TO ASSEMBLY ROUTINE *addr*
STOP STOP PROGRAM (CON continues)
TAB (*column*) PRINT SPACES TO *column*
ZONE *width* DEFINE PRINT ZONE WIDTH

SPRITE COMMANDS

DEFINE *image #,defstring* Setup sprint image (*defstring*=64 bytes)
IDENTIFY *sprite#* to *image#* Assign *sprite#* to *image#*
HIDESPRITE *n* Turn off sprite *n*
SPRITECOLOR *sprite#,n* Set sprite to color *n*
SPRITEPOS *sprite#,x,y* Position sprite at loc. *x,y*
SPRITESIZE *sprite#,x,y* IF *x*=1 Double width
IF *y*=1 Double Height
DATACOLLISION *sprite#,reset* True - COLLISION with data
reset 1 = Reset flag
0 = Flag stored
SPRITE COLLISION *sprite#,reset* True - COLLIDED with other sprite
reset 1 = Reset flag
0 = Flag stored
PRIORITY *sprite#,p* *p*=True -Sprite underneath graphics
False-Sprite above graphics
SPRITEBACK *color1,color2* Set multicolor sprite colors

CONDITIONALS

IF...THEN...ELSE If condition is TRUE, THEN is executed otherwise ELSE is executed.
IF...THEN...ELIF...THEN...ELSE ELIF is short for ELSE IF. It is used for multiple IF's.
ENDIF ENDIF terminates IF structure
IF *condition* **THEN** *statement* If condition TRUE then statement is executed. ENDIF is not needed.

CASE *exp* **OF**
WHEN *choicelist*
statements...
OTHERWISE
statements...
ENDCASE IF *exp* is equal to one of the items in a choice list the statements following are executed. If *exp* does not match choice list the statements following OTHERWISE are executed ENDCASE terminates case structure.

GOTO *name* GOTO line with this name.

I/O STATEMENTS

CLOSE FILE [*file#*] CLOSE FILE
INPUT# *file#,variablelist* READ DATA from *file#* to *variablelist*
KEY$ GET CHARACTER from KEYBOARD
OPEN FILE [*file#*],"*filename*",[*r*] OPEN SEQUENTIAL FILE *r* = R-READ. W-WRITE, A-APPEND
OPEN FILE [*file#*],"*filename*",**RANDOM**,*n* OPEN RANDOM FILE (rec. up to *n* bytes)
PRINT FILE [*file#*][:*items*] PRINT item to SCREEN, PRINTER, or FILE
PRINT FILE [*file#*]**:USING** (*tmt*) : *(vars)* PRINT the variables formatted
READ (*varlist*) READ DATA from PROGRAM LINE
READ FILE [*file#*],[*rec#*]: (*varlist*) READ DATA from FILE into variable list
SELECT "**LP:**" or "**DB:**" DIRECT PRINTOUT to SCREEN(DB) or printer(LP)
PASS *"diskcommand"* PASSES COMMAND to DISK
STATUS$ RETURN STATUS of DISK channel
WRITE FILE [*file#*][:*varlist*] WRITE to SEQUENTIAL file
WRITE FILE [*file#*],[*rec#*][:*varlist*] WRITE to RANDOM file

NOTE: DATA STORED by WRITE FILE can only be retrieved by READ FILE not input file, and cannot be used in tape

LOOP STRUCTURES

FOR *var*=*start* **TO** *end* **STEP** [*step*] **DO** Loop from start to end
statements...
ENDFOR [*var*] End of FOR loop

REPEAT REPEAT FOLLOWING STATEMENTS
statements...
UNTIL (*exp*) Stops repeat when exp is TRUE

WHILE (*exp*) **DO** Repeat WHILE until exp is false
statements...
ENDWHILE Terminates a WHILE structure

UnSHIFTed

TEXT SCREEN

SPLIT SCREEN

FULL SCREEN

COLORS

0 BLACK
1 WHITE
2 RED
3 CYAN
4 PURPLE
5 GREEN
6 BLUE
7 YELLOW
8 ORANGE
9 BROWN
10 Lt. RED
11 GRAY 1
12 GRAY 2
13 Lt. GREEN
14 Lt. BLUE
15 GRAY 3

Cut on other side of line

ARITHMETIC OPERATORS

↑	POWER
*	MULTIPLICATION
/	DIVISION (floating point)
DIV	DIVISION (integer)
+	ADDITION
–	SUBTRACTION
MOD	Returns REMAINDER

RELATIONAL OPERATORS

<	LESS THAN
<=	LESS THAN OR EQUAL
=	EQUAL
>=	GREATER THAN OR EQUAL
>	GREATER THAN
<>	NOT EQUAL

BOOLEAN OPERATORS

(*exp*) **AND** (*exp*)	AND Return: 1 (true) or **0** (false)
NOT (*exp*)	NOT Return: 1 (true) or **0** (false)
(*exp*) **OR** (*exp*)	OR Return: 1 (true) or **0** (false)
EOD	TRUE if END OF DATA
EOF(*x*)	TRUE if END OF FILE

NUMERIC FUNCTIONS

ABS(*exp*)	Absolute value to *exp*				
ATN(*exp*)	Arctangent of *exp*	**LOG**(*exp*)	Natural log of *exp*	**SGN**(*exp*)	Sign of *exp* (–1, **0**, or 1)
COS(*exp*)	COSINE of *exp*	*n1* **MOD** *n2*	DIVISION REMAINDER (*n1*/*n2*)	**SIN**(*exp*)	Sine of *exp* (radians)
EXP(*exp*)	Log Base e of *exp*	**RND**(*x,y*)	RANDOM NUMBER (*x* TO *y*)	**SQR**(*exp*)	Square root of *exp*
INT(*exp*)	RETURNS INTEGER Part	**RND**(*n*)	RANDOM NUMBER (**0.0** TO 1.**0**)	**TAN**(*exp*)	Tangent of *exp*

TURTLE MOVEMENT

HIDETURTLE	Make turtle invisible
SHOWTURTLE	Make turtle visible
FORWARD *n*	Forward *n* steps
BACK *n*	Backward *n* steps
RIGHT *n*	Right *n* degrees
LEFT *n*	Left *n* degrees
HOME	Home turtle
SETXY *x,y*	Set turtle *x* and *y* coordinates
SETHEADING *n*	Set heading *n* degrees
TURTLE SIZE *n*	Set turtle size **0** to 1**0**

GRAPHIC COMMANDS

BACKGROUND *n*	BACKGROUND color n
BORDER *n*	BORDER color n
CLEAR	CLEAR graphic SCREEN
DRAWTO *x,y*	Draw line from current point to *x,y*
MOVETO *x,y*	Move to *x,y* without line
FILL *x,y*	Fills closed area containing *x, y* to current color
GETCOLOR *x,y*	Return color of point *x,y*
SETGRAPHIC 0	Set high resolution graphics on
SETGRAPHIC 1	Set multicolor graphics on
SETGRAPHIC	Return to previous screen

PEN COMMANDS

PENCOLOR *n*	Turtle pen color n
PENDOWN	Turtle draws lines
PENUP	No turtle lines
PLOT *x,y,text*	Text displayed starting at *x,y*
FRAME *xmin,xmax,ymin,ymax*	Set up a screen window

P.O. Box 8299, Pgh., PA 15218

LC1SF-CM1

COMAL (.14)