Commander X16

Emulator and

System Manual

Table of Contents

Keyboard and Special Keys....1
Commander X64 Emulator Keyboard Notes....1
Variables....1
BASIC Command Vocabulary 2
ABS....2
ASC....2
ATN....2
BIN$....2
Boot....2
CHAR....3
CHR$....3
CLOSE....3
CLR....3
COLOR....4
Colors:....4
CONT....4
COS....5
DATA....5
DEF FN....5
DIM....5
DOS....5
EXP....6
FNxx....6
Frame....6
FRE....6
FOR....6
GEOS....6
GET....7
GET#....7
GOSUB....7
GOTO....7
HEX$....7
IF....8
INPUT....8
INPUT#....8
INT....8
JOY....9
Keyboard Joystick keys.9
KEYMAP....9
LEFT$....10
LEN....10
LET....10
LINE....10
LOAD....10
LOCATE....11
LOG....11
MID$....11
MON....11
MOUSE....12
Mouse Modes....12
MX/MY/MB....13
Mouse Button return values....13
NEXT....13
NEW....14
OLD....14
ON....14
OPEN....14
PEEK....14
POKE....14
POS....15
PRINT....15
PRINT#....15
PSET....15
READ....15
RECT....16
REM....16
RESET....16
RESTORE....16
RIGHT$....16
RND....17
RUN....17
SAVE....17
SCREEN....18
Screen Modes....18
SGN....18
SIN....18
SPC....18
SQR....19
STOP....19
STR....19
TAB....19
TAN....19
USR....19
VAL....20
VERIFY....20
VPEEK....20
VPOKE....20
VLOAD....20
WAIT....21

Keyboard and Special Keys

Commander X64 Emulator Keyboard Notes

Keystroke	Effect
^F / ^ Return	Fullscreen Toggle
^R	Reset (Runs AUTOBOOT.X16)
Shift Backspace	Insert a blank to type over
Shift Home	Clear Screen
Shift Alt	Toggle between ASCII and PETSCII

Variables

A letter, or a letter followed by a digit, or two letters.

Real variables get no special mark character. For integers follow the variable name with a ‘%’. For Strings follow the variable name with a ‘$’.

BASIC Command Vocabulary

ABS

C64 Function

n = abs(x)

ABS returns the positive value of x, which may be negative or positive.

ASC

C64 Function

n = asc(x$)

ASC returns the ascii code of the first character in X$.

ATN

C64 Function

n = atn(x)

Returns the angle, measured in radians, whose tangent is x.

BIN$

CX64 function

A$ = BIN$(n)

Returns a string representing the binary value of n

Boot

CX64 Statement, Command

BOOT

Load and run AUTOBOOT.X16

CHAR

CX64 Statement

CHAR X,Y,COLOR,STRING

Print a string on the graphical plane using the color and at the location specified

CHR$

C64 Function

a$ = chr$(48)

CHR$ returns the character for the code of the argument. 48 happens to be “0”.

CLOSE

C64 Statement

CLOSE <FILENUM>

Close a file that was opened with the OPEN command.

CLR

C64 Statement

CLR

CLR clears all variables in memory but leaves the program intact. RUN causes a CLR before starting execution.

COLOR

CX64 Statement, Command

COLOR <fgcol>[,<bgcol>]

This command sets the text mode foreground color, and optionally the background color

Colors:

0	Black
1	White
2	Red
3	Cyan
4	Purple
5	Green
6	Blue
7	Yellow
8	Orange
9	Brown
10	Light Red
11	Gray 1
12	Gray 2
13	Light Green
14	Light Blue
15	Gray 3

CONT

C64 Command

CONT

Continue a program that has been stopped with ^C or the stop key. If the program is edited while stopped it will no longer be possible to continue execution.

COS

C64 Function

c = cos(x)

Returns the cosine of the argument x, measured in radians

DATA

C64 Statement

DATA “VALUE”,42,3.1415,”String”,SingleWord

The DATA statement provides information that can be read by the READ statement. Strings with spaces, colons or commas must be enclosed in quotes. Two commas will be read as a numeric zero.

DEF FN

C64 Statement

10 DEF FN<variablename>(parameter) = <computable expression>

The Define Function statement allows you to create a function that can be used later in your program to compute the function.

```
10 DEF FNA( R ) = R * R * 3.1415
20 INPUT “RADIUS”; R: PRINT “AREA = “; FNA( R )
```

DIM

C64 Statement

10 DIM A$(50), B(20,20,20)

The DIM statement allocates memory for an array.

DOS

CX64 Command,Statement

DOS <string>

This command sends the string to the status channel of a device. The special “$” will print a directory of the media in the device. If no string is provided the status of the current device is reported

EXP

C64 Function

n = exp(x)

Returns e to the x

FNxx

C64 Function

n = FNA1(x)

Returns the value of the user-defined function.

Frame

CX64 Statement

FRAME <x1>,<y1>,<x2>,<y2>,<color>

This command draws a rectangle on the graphics screen in the given color.

FRE

C64 Function

n = FRE(x)

Returns the number of unused bytes available in memory. If that number of bytes is greater than 32K it may return as a negative number.

FOR

C64 statement

FOR <VARIABLE> = <INITIAL VALUE> TO <TERMINAL VALUE> [STEP <STEP VALUE>]

The FOR statement marks the top of a counted loop and can count up or down by any value with the optional STEP parameter. Any of the values may be constants or variables.

GEOS

CX64 command

GEOS

Enter the GEOS User Interface

GET

C64 Statement

10 GET A$

The GET statement places the next available character in the input buffer in the string variable specified. If the input buffer is empty the specified variable will be an empty string.

To wait for the user to type a character:

```
10 GET A$: IF A$="" THEN 10
```

GET#

C64 Statement

As the GET statement, except the next character is read from an opened file.

GOSUB

C64 Statement

Cause program execution to jump to a new location in the program, with the current location being saved for a return.

The number of nested subroutines is limited by memory. In a simple test 23 levels was obtained but the program was trivially small.

```
10 I = 0
20 GOSUB 30
30 PRINT I
40 I = I + 1
50 GOSUB 30
```

GOTO

C64 Statement

The GOTO statement causes the program flow to jump to another line without saving a return location.

HEX$

CX64 Function

S$ = HEX$(n)

This function returns a string containing the hexadecimal value of n. If n < 255 hex$(n) returns 2 characters, If n > 255 then hex$ returns 4 characters. Hex$ handles numbers up to 65535

IF

C64 Statement

10 IF <CONDITONAL EXPRESSION> THEN <STATEMENT>| <LINE NUMBER>

IF allows the program to make choices depending on the conditional expression. If the conditional expression is true the the statement in the THEN clause is executed.

INPUT

C64 Statement

10 INPUT [“PROMPT”;] <VARIABLE>{,VARIABLE}

The INPUT statement receives some input from the user and stores it into a variable. Multiple values may be received and placed into multiple variables. A prompt may be included that is printed to remind the user what value is needed.

INPUT#

C64 statement

The INPUT# statement works the same as the INPUT statement, without the prompting option, but instead reads values from an open file.

INT

C64 function

i = int(f)

Returns the integer part of a real number

JOY

CX64 function

JOY(n)

Returns the state of a joystick.

joy(n) where n is in the range of 1 to 4. Joy(0) returns the “keyboard joystick”

Keyboard Joystick keys

Value	Button
$800	A
$400	X
$200	L
$100	R
$080	B
$040	Y
$020	SELECT
$010	START
$008	UP
$004	DOWN
$002	LEFT
$001	RIGHT

KEYMAP

CX64 Statement

KEYMAP <string>

This command sets the current keyboard layout. It can be put into an AUTOBOOT file to always set the keyboard layout on boot.

```
10 KEYMAP"SV-SE"    :REM SMALL BASIC PROGRAM TO SET LAYOUT TO SWEDISH/SWEDEN
SAVE"AUTOBOOT.X16"  :REM SAVE AS AUTOBOOT FILE
```

LEFT$

C64 Function

a$ = left$(s$,c)

Left returns the leftmost c characters from the string s$

LEN

C64 Function

n = len(a$)

LEN returns the length of a string.

LET

C64 Statement,command

[LET] <VARIABLE> = <EXPRESSION>

The word let is optional, this statement assigns a value or the result of a computed expression to a variable.

LINE

CX64 statement

LINE <x1>,<y1>,<x2>,<y2>,<color>

This command draws a line on the graphics screen in a given color.

```
10 SCREEN128
20 FORA=0TO2*\XFFSTEP2*\XFF/200
30 :  LINE100,100,100+SIN(A)*100,100+COS(A)*100
40 NEXT
```

LOAD

C64 Command

LOAD “filename”<,device>

Load a file from the device

LOCATE

CX64 Statement

LOCATE <line>[,<column>]

This command positions the text mode cursor at the given location. The values are 1-based. If no column is given, only the line is changed.

```
100 REM DRAW CIRCLE ON TEXT SCREEN
110 SCREEN0
120 R=25
130 X0=40
140 Y0=30
150 FORT=0TO360STEP1
160 :  X=X0+R*COS(T)
170 :  Y=Y0+R*SIN(T)
180 :  LOCATEY,X:PRINTCHR$($12);" ";
190 NEXT
```

LOG

C64 Function

n = log(x)

Returns the natural log of x

MID$

C64 Function

r$ = mid$(x$,s,l)

MID$ returns a string that is l characters long starting from character s in string x$.

MON

CX64 Command

MON (Alternative: MONITOR)

This command enters the machine language monitor. See the dedicated chapter for a description.

MOUSE

CX64 Statement

MOUSE <mode>

This command configures the mouse pointer.

Mouse Modes

Mode	Description
0	Hide mouse
1	Show mouse, set default mouse pointer
-1	Show mouse, don't configure mouse cursor

`MOUSE 1` turns on the mouse pointer, `MOUSE 0` turns it off.

If the BASIC program has its own mouse pointer sprite configured, it can use `MOUSE -1`, which will turn the mouse pointer on, but not set the default pointer sprite.

The size of the mouse pointer's area will be configured according to the current screen mode. If the screen mode is changed, the MOUSE statement has to be repeated.

```
MOUSE 1 : REM ENABLE MOUSE
MOUSE 0 : REM DISABLE MOUSE
```

MX/MY/MB

CX64 Function

n=MX
n=MY
n=MB

Return the horizontal (`MX`) or vertical (`MY`) position of the mouse pointer, or the mouse button state (`MB`).

`MB` returns the sum of the following values depending on the state of the buttons:

Mouse Button return values

Value	Button
0	none
1	left
2	right
4	third

```
10 SCREEN$80
20 MOUSE1
25 OB=0
30 TX=MX:TY=MY:TB=MB
35 IFTB=0GOTO25
40 IFOBTHENLINEOX,OY,TX,TY,16
50 IFOB=0THENPSETTX,TY,16
60 OX=TX:OY=TY:OB=TB
70 GOTO 30
```

NEXT

C64 statement

10 NEXT [<variable>{,variable}]

The NEXT statement marks the end of a FOR loop. It causes the loop variable to be incremented and tested. If no variable is listed the inner most for loop is computed. If multiple variables are listed they must be listed in the order of closing loops: for a for b for c … next c,b,a

NEW

C64

NEW

Remove the current basic program and clear variables.

OLD

CX64

OLD

This command recovers the BASIC program in RAM that has been previously deleted using the `NEW` command or through a RESET.

ON

C64 Statement

ON <expression> GOTO | GOSUB <list of line numbers>

ON, used with GOTO or GOSUB forms a computed goto or gosub, selecting from the list of line numbers according to the expresson.

OPEN

C64 Statement

OPEN <file ID>,<device>[,<operation or buffer or channel>],”<filename>”

PEEK

C64 Function

x = peek(ad)

Returns the content of memory at address specified by ad.

POKE

C64 Statement,Command

Poke writes the value in a variable or a constant to a specific location in memory. Be careful with POKE as placing the wrong value into a memory based control register can hang the machine.

```
10 POKE <address>,<value>
```

POS

C64 Function

Returns the column where the next value printed will be placed.

PRINT

C64 Statement, Command

PRINT [<variable>|<constant>|<expression> {,|; <variable>|<constant>|<expression>}]

PRINT will print zero or more variables, constants, or the computed value of expressions to the screen. A PRINT by itself will print a carriage return / linefeed to move the cursor to the next line.

```
10 PRINT RIGHT$(“000000”+STR$(N),6);
```

The example prints the value of n, right justified in a field 6 characters wide.

PRINT#

C64 Statement

Works as print does, but prints to an open file.

```
100 PRINT#1, A(1);”,”;A(2);”,”;A(3)
```

PSET

CX64 Statement

PSET <x>,<y>,<color>

This command sets a pixel on the graphics screen to a given color.

```
10 SCREEN $80
20 FOR I=1 TO 20:PSET RND(1)*320, RND(1)*200, RND(1)*256:NEXT
30 GOTO20
```

READ

C64 Statement

READ <variable list>

READ will take information from DATA statements and put in variables in the variable list. READ must provide the correct variable for each item as it reads the DATA statements. If the next item of DATA is a string and READ is going to fill a numeric variable the result will be a “Type Mismatch” error.

RECT

CX64

RECT <x1>,<y1>,<x2>,<y2>,<color>

This command draws a solid rectangle on the graphics screen in a given color.

```
10 SCREEN$80
20 FORI=1TO20:RECTRND(1)*320,RND(1)*200,RND(1)*320,RND(1)*200,RND(1)*256:NEXT
30 GOTO20
```

REM

C64 Statement

REM is a remark statement. This provides a way for the programmer to leave notes to future programmers. REM does not affect the execution of the program in any way.

RESET

CX64

RESET

Performs a software reset of the system.

RESTORE

C64 Statement

Restore the DATA pointer to the first data item. RESTORE must be on its own line.

RIGHT$

C64 Function

r$=right$(s$,x)

RIGHT$ returns the x rightmost characters of string s$.

```
10 REM FORMATTING EXAMPLE: PRINT 8 IN AN 8 CHARACTER ZERO FILLED FIELD
20 PRINT RIGHT$(“000000000”+STR$(X),8)
```

RND

C64 Function

n = rnd(x)

If x is positive, rnd returns the next pseudo random number in the range 0-1. If x is negative then x is used to seed the pseudo random number generator (prng). If x = 0 then the same number as the last call is returned.

RUN

C64 Command

RUN <line number>

Begin executing the program in memory, optionally at a line number. Note that a variable may not be used instead of a line number.

SAVE

C64

SAVE “[@0:]filename”[,device]

Save the program to the device, optionally overwriting an existing file with the same name.

SCREEN

SCX64 Statement

SCREEN <mode>

This command switches screen modes. The value of -1 toggles between modes $00 and $03.

Screen Modes

Mode	Description
$00	80x60 text
$01	80x30 text
$02	40x60 text
$03	40x30 text
$04	40x15 text
$05	20x30 text
$06	20x15 text
$80	320x200 256 color grapics and 40x25 text. Mode $80 displays two layers: a text layer on top of a graphics screen. In this mode, text color 0 is translucent instead of black.

```
SCREEN 3 : REM SWITCH TO 40 CHARACTER MODE
SCREEN 0 : REM SWITCH TO 80 CHARACTER MODE
SCREEN -1 : REM SWITCH BETWEEN 40 and 80 CHARACTER MODE
```

SGN

C64 Function

x = sgn(n)

This function returns a 1 for a positive number, a -1 for any negative number and zero for zero.

SIN

C64 Function

n = sin(x)

SIN(x) returns the sin of x, measured in radians.

SPC

C64 Function

10 print a;spc(5);b

This function may be used in PRINT statements to skip a number of spaces

SQR

C64 Function

n = sqr(x)

The SQR function returns the square root of its argument.

STOP

C64 Statement

STOP causes program execution to stop. The program, if not edited, may be restarted with a CONT command.

STR

C64 Function

r$ = str$(x)

Returns a string that is identical to the printed format of x.

TAB

C64 Function

10 print tab(15);”At column 15”

This function is used in print statements to set the column where the next item will print

TAN

C64 Function

The TAN function returns the tangent of its argument, measured in radians.

USR

C64 Function

n = usr(x)

This function calls an assembly language routine and supplies x, The assembly language will return a value which is then returned by the USR function. For details on the assembly language routine see the section on assembly language.

VAL

C64 Function

n = VAL(x$)

The VAL function reads a number from the string x$. This is the inverse of the STR$() function

VERIFY

C64 Command

VERIFY “filename”[,device]

Read a file on the device by name and compare it to the program in memory. This command verifies that the program was properly written to the device.

VPEEK

VPEEK (<bank>, <address>)

Return a byte from the video address space. The video address space has 20 bit addresses, which is exposed as 16 banks of 65536 addresses each.

```
  PRINT VPEEK(1,$B000) : REM SCREEN CODE OF CHARACTER AT 0/0 ON SCREEN
```

VPOKE

VPOKE <bank>, <address>, <value>

Set a byte in the video address space. The video address space has 20 bit addresses, which is exposed as 16 banks of 65536 addresses each.

```
  VPOKE 1,$B000+1,1 * 16 + 2 : REM SETS THE COLORS OF THE CHARACTER
                               REM AT 0/0 TO RED ON WHITE
```

VLOAD

VLOAD <filename>, <device>, <VERA_high_address>, <VERA_low_address>

Loads a file directly into VERA RAM.

```
VLOAD "MYFILE.BIN", 8, 0, $4000  :REM LOADS MYFILE.BIN FROM DEVICE 8 TO VRAM
$4000.
VLOAD "MYFONT.BIN", 8, 1, $F000  :REM LOAD A FONT INTO THE DEFAULT FONT LOCATION
($1F000).
```

WAIT

C64 Statement

Wait <address>,<and value>[,<xor value>]

The content of the memory location are xored with the xor value, if present, then anded with the and value. If the result is non-zero the program continues, if the result is zero the program continues to check at this line.