CENTURY COMMUNICATIONS

COMMODORE 64

MICRO GUIDE

A Quick Reference Guide to the BASIC AND SYSTEM OPERATIONS of the

COMMODORE 64

Copyright © Peter Morse and Brian Hancock 1984

All rights reserved

The authors gratefully acknowledge the permission of Commodore Business Machines U.K. Ltd. to include the copyright material from their *User Guide* pages 132, 135 and 152

First published in Great Britain in 1984
by Century Communications Ltd
Portland House, 12-13 Greek Street,
London W1V 5LE

ISBN 0 7126 0354 9

Printed in Great Britain

MICROGUIDE
FOR THE
COMMODORE 64

Professor Peter Morse
Brian Hancock

CENTURY
COMMUNICATIONS
London

© *1984 P. Morse and B. Hancock*

CONTENTS

	Commodore keywords	3
1	Conventions	4
2	Arithmetic and Logic Operations	6
3	Control Keys	7
4	Operating Commands	9
5	BASIC Statements	10
6	BASIC Functions	11
7	Error Messages	16
8	Screen Display and Graphics	18
9	Sound	24
10	Peripheral Control Commands	28

COMMODORE KEYWORDS

Keyword	Brief meaning	Page
ABS	absolute value	14
AND	logical AND	6
ASC	American standard code	15
ATN	arc-tangent	14
CHR$	character string	15
CLOSE	close file	28
CLR	clear variables	10
CMD	redirect output to other devices	28
CONT	continue	9
COPY	copy program	28
COS	cosine	14
DATA	data	10
DEF	define function	10
DIM	dimension	10
END	end of program	10
EXP	exponentiation	14
FOR	begins loop	10
FRE	free	14
GET	get from keyboard	11
GET#	get from file	11, 28
GOSUB	goto subroutine	11
GOTO	goto address	11
IF	if	11
INITIALIZE	initialize	28
INPUT	input data from keyboard	11
INPUT#	input data from file	28
INT	integer	14
LEFT$	left string	16
LEN	length of string	16
LET	let	11
LIST	list program on screen	9
LOAD	load program	9
LOG	natural logarithm	14
MID$	middle of string	16
NEW	erase old program/format disc	9, 28
NEXT	ends loop	12
NOT	logical inverse	6
ONGOSUB	on condition goto subroutine	12
ONGOTO	on condition goto address	12
OR	logical or	6
OPEN	open file/data channel	28
PEEK	peek memory	14
POKE	poke into memory	13
POS	print position	14
PRINT	print to screen	13
PRINT#	print to file	28
PRINT SPC	print with spacing	13
PRINT TAB	print with tabulation	13
READ	read data	12
REM	remark	12

Keyword	Brief meaning	Page
RENAME	rename program	28
RESTORE	restore data pointer	12
RETURN	return from subroutine	13
RIGHT$	right string	16
RND	random number	14
RUN	execute program	9
SAVE	save on cassette	9
SCRATCH	remove files from disc	28
SGN	sign	15
SIN	sine	15
SPC	space	15
SQR	square root	15
STEP	step	10
STOP	stop execution	13
STR$	string representation	16
SYS	system (as in USR)	13
TAB	tabulation	15
TAN	tangent	15
USR	user defined	15
VAL	value	16
VALIDATE	validate disc	28
VERIFY	verify file existence	9, 28
WAIT	wait	13

CONVENTIONS

Expression
Any legal combination of constants, variables, functions, and arithmetic operators.

Line number
Any number between 1 and 63999 at the beginning of the line which serves to identify the information on the line as a statement.

List
A one dimensional array.
example:
A(15), A$(20)

Number
A positive or negative decimal quantity which is significant to about eight digits, and whose magnitude is between a minimum of
$\pm 2.93873588\,E-39 \approx \pm 3 \times 10^{-39}$
and maximum of
$\pm 1.70141183E+38 \approx \pm 1.7 \times 10^{+38}$
Numbers larger than $1.7 \times 10^{+38}$ produce an OVERFLOW ERROR, and numbers smaller than 2.938×10^{-39} are taken as zero.

Numeric variable
Numeric variables are used to name a numeric value or a collection of numeric values. Two types:–

1 Real variable – may be represented by any set of alpha numeric characters (only the first two are recognised by computer) provided
(i) they start with a letter
(ii) they do not start with a BASIC keyword.

2 Integer variables – are distinguished by having a % sign after the name. Numbers between +32767 and −32767 are stored with complete accuracy.
Examples:
SUN, SUM, AGE, A, A1 are real variables. Note: SUM and SUN will be the same variable name as far as the computer is concerned. A%, AREA%, ABC% are integer variable names. Note that names like TOTAL, LENGTH and other basic keywords cannot be used as a variable name.

String
A sequence of characters each of which is a letter, digit, space or some character other than a carriage return. ASCII code used to represent these characters in computer.

String constant
A string enclosed in double quotes.
Example: "COMMODORE 64"

String variable
Used to store strings. The rules for naming string variables are similar to those for numeric variables, except that the variable name must be followed by a dollar sign "$".
Examples:
NAME$, A$, A1$, CITY$

Substring
Any set of consecutive characters taken in sequence from the parent string.
Example:
A$="COMPUTER", "PUT" is a substring.

Table
A two- or multi-dimensional array.

add	address (0-65535)
n, m	numbers
dn	device number (0-8)
fn	file number (0-127)
ln	line number (0-63999)
S	string
V	numeric string
V$	string variable
[ ]	optional item

ARITHMETIC AND LOGIC OPERATIONS

Arithmetic operators

symbol	operation	priority
+	addition	4
−	subtraction	4
*	multiplication	3
/	division	3
−	unary minus	2
↑	exponentiation	1

Logical operators

AND

Combines relations so that (condition 1) AND (condition 2) is only TRUE when both conditions are true.
Example:
IF X>1 AND X<10 THEN PRINT "BETWEEN 1 AND 10".
Also used as a bitwise operator for binary numbers.
Example:
PRINT 85 AND 28 gives 20

	128	64	32	16	8	4	2	1	
85 in binary is	0	1	0	1	0	1	0	1	
28 in binary is	0	0	0	1	1	1	0	0	
85 AND 28 is	0	0	0	1	0	1	0	0	=20

NOT

Logically gives reverse of expression.
Example:
IF NOT (A=B) THEN PRINT "NOT EQUAL"
Also used as bitwise operator.
Example:
PRINT NOT 56 gives −57
PRINT NOT 6 gives −7

OR

Combines relations so that (condition 1) OR (condition 2) is TRUE when either or both conditions are true.
Example:
IF X>0 or Y>0 THEN PRINT "ONE OR BOTH +VE".
Also used as bitwise operator.
Example:
PRINT 85 OR 28 gives 93

	128 64 32 16 8 4 2 1
85	0 1 0 1 0 1 0 1
28	0 0 0 1 1 1 0 0
	0 1 0 1 1 1 0 1 = 93

Relational operators

Symbol	Operation
=	equal to
<	less than
<=	less than or equal to
<>	not equal to
>	greater than
>=	greater than or equal to

CONTROL KEYS

Display modes

Mode 1
Upper case/Graphics – obtained on switching on

Mode 2
Upper case/lower case – obtained by depressing [THE COMMODORE KEY] [SHIFT] keys once; depressing keys again will set the display to mode 1 (ie toggles between mode 1 and mode 2).

[CRSR]
Moves cursor down.

[CRSR]
Moves cursor right.

[SHIFT] [CRSR]
Moves cursor up.

[SHIFT] [CRSR]
Moves cursor left.

[CTRL] [number key]
Sets text colour, and performs certain operations.
(see table 1).

[INST/DEL]
Erases previous character

[SHIFT] [INST/DEL]
Allows insertion.

[CLR/HOME]
Moves cursor to top left hand corner of screen.

[SHIFT] [CLR/HOME]
Clears screen and moves cursor to top left hand corner of screen.

[RETURN]
Causes command to be acted upon.

[RUN/STOP]
Stops program execution or stops loading program.

[SHIFT] [RUN/STOP]
Automatically loads program on tape into memory and executes it.

[RUN/STOP] [RESTORE]
Clears screen and restores the computer to normal state. Memory remains unchanged. The Commodore key loads program into memory after FOUND program is displayed on the screen in response to LOAD command.

[C=][key]
Gives graphics character on left of key when in upper/lower case mode.

[C=] [number key]
Sets the additional text colours (see table 1).

[SHIFT]
Normal standard typewriter.
upper case/lower case mode – upper case character
upper case/graphics mode – graphics character on right side of key.

Table 1
[CTRL][9]
Inverse video.

[CTRL][0]
Normal video.

Text colours

Keys	Colour	Code
[CTRL] [1]	BLACK	144
[CTRL] [2]	WHITE	5
[CTRL] [3]	RED	28
[CTRL] [4]	CYAN	159
[CTRL] [5]	PURPLE	156
[CTRL] [6]	GREEN	30
[CTRL] [7]	BLUE	31
[CTRL] [8]	YELLOW	158
[C=] [1]	ORANGE	129
[C=] [2]	BROWN	149
[C=] [3]	LIGHT RED	150
[C=] [4]	GREY 1	151
[C=] [5]	GREY 2	152
[C=] [6]	LIGHT GREEN	153
[C=] [7]	LIGHT BLUE	154
[C=] [8]	GREY 3	155

OPERATING COMMANDS

CONT
Continues program execution that is interrupted by pressing [RUN/STOP], STOP statement, or END statement. The execution continues from the line of interrupt.

LIST [In - Im]
Produces a listing of the program between lines In and Im inclusive. Default on In is first line and on Im is last line.
Examples:

LIST	lists whole program
LIST 200–	lists line 200 onwards
LIST 200–300	lists lines 200 to 300
LIST –100	lists upto line 100
LIST 100	lists line 100

LOAD p, dn
Loads program p from device dn. The default for dn is 1, ie tape. Printer is device 4 and disc is 8.
Examples:
Load loads the next program on tape
LOAD "PROG" searches for PROG on tape and loads it when found.
LOAD "PROG",8 loads the specified program from disc.
LOAD "*",8 loads the first program on disc.

NEW
Clears computer memory, ie deletes program and variables.

RUN In
Executes program in memory from the specified line number. Default for In is lowest line number. Direct command GOTO In has the same effect.

SAVE p, dn, m
Saves program p on device dn with end-of-tape marker if m=1 (ie if m=0 no end-of-tape marker). Default on dn is 1, ie tape. Disc is 8.
Examples:
SAVE "PROG" saves PROG on tape
SAVE "PROG",8 saves PROG on disc
SAVE A$ saves program whose name is A$ on tape
SAVE "PROG",1,1 saves PROG on tape with an end-of-tape marker at the end of program

VERIFY p, dn
Checks program on disc or tape against the one in memory, ie checks if program is SAVEd.
Examples:
VERIFY "PROG",8 checks for file PROG on disc
VERIFY "PROG" checks for file PROG on tape
VERIFY checks next program on tape

BASIC STATEMENTS

CLR
Clears all variables from memory, leaving the program in memory unchanged.

DATA d1, d2, d3 . . .
Provides constant data (numbers or strings) for READ statement
Example:

```
10  FOR I = 1 TO 5
20  READ A(I):NEXT I
30  DATA 35.0,-20.0, 7.25, -2.5, 18.0
    assigns 35.0 to A(1), -20.0 to A(2), and so on.
```

DEF FNV(dv)=expression
Defines a user-specified function, where V is any legal variable name (upto two characters long). The dummy variable dv is a letter which will be replaced by the function argument when the function is called.
Example:

```
10  DEF FNA(X) = X*X+1
20  Y=3 : B = FNA(Y) : PRINT B
```

Prints 10 on the screen. This is because the dummy variable X is replaced by the value of Y, and the expression is evaluated when the function FNA is called at line 20.

DIM V[$](k[,m])
Specifies storage space to be allocated for list or table V[$]. If a list or a table is not specified by DIM statement, the default for K (number of rows) is 10, and the default for m (number of columns) is 10. That means the list has 10 elements in it and the table has 10 rows and 10 columns (100 elements).
Examples:

```
10  DIM A(5)
```

reserves storage for six elements in numeric list A. Note zeroth element.

```
10 DIM X(9, 19), XY$(9, 9)
```

reserves storage for 10 rows and 20 columns for numeric table X, and, 10 rows and 20 columns for string table XY.

END
Indicates the end of program and is usually the statement with the highest line number.

FOR V = k TO m [STEP s]
Specifies a loop and must be used with a NEXT statement. The loop is executed for a range of specified by k and m.
s indicates increment in step (ie next n=n+s), and if omitted default is 1. k, m and s may be numbers or variables (numeric).
Examples:

```
10  FOR I=1 to 20
20  PRINT I:NEXT I
30  FOR J= 0 TO 2 STEP 0.2
40  PRINT J:NEXT J
```

line 20 prints 1, 2, 3, ... 20
line 40 prints 0, 0.2, 0.4, ... 1.8

GET V$
Allows inputting data from the keyboard one letter or number at a time. The character entered is assigned to the string variable V$.
Example:

```
10  GET A$
20  IF A$=" " GOTO 10
30  PRINT A$
```

line 20 checks if a key is pressed: if it is , prints the key, otherwise waits until a key is pressed.

GET #d, V$
Inputs one character at a time from a previously opened data file on device d, and assigns the character entered to the variable V$.
Example:

```
10  GET#1,A$
```

inputs one character from the opened date file and assigns it to A$.

GOSUB ln
Enters a subroutine at the specified line. The subroutine is exited by executing a RETURN statement which returns control to the line following the GOSUB call.
Example:

```
100    GOSUB 1000
110
.
.
.
200    STOP
1000   REM SUBROUTINE
.
.
.
1100   RETURN
```

GOTO ln
Transfers control to the statement at the specified line.

```
10  GOTO 50
```

IF condition THEN statement
The condition is evaluated and control is transferred to the statement following THEN if condition is true. If condition is not true, then control is transferred to the next line.
Example:

```
100  IF A$<>"E" THEN[GOTO]10
110  END
```

INPUT ["PROMPT";]V[$]
Allows data to be entered from the keyboard. This statement, unlike the GET statement, causes a "?" to be output on the screen so that you can respond by typing in values for the requested variables.

LET V[$]=expression
Assigns value of the expression to the variable V. The word LET may be omitted.

Examples:

```
10   LET A=3.5
20   B=A*3+A↑2
```

NEXT V[,V]

Terminates FOR . . . NEXT loop (see FOR) if the limit of loop is reached. If loop is not finished, the loop variable V is incremented by the specified step value. If more than one variable is given, they are completed in order from left to right.

Examples:

```
10     FOR I=1 to 10
20     FOR J=10 to 20
.
.
.
100    NEXT J       or 100 NEXT J, I
110    NEXT I
```

ON NV GOSUB ln1, ln2, ln3, . . .
GOTO

Allows several possible transfers of control to a subroutine or a line depending on the value of NV.(Numeric variable)

Examples:

```
30      ON X GOTO 100, 200, 300
        will cause control to move to line 100
        if X =1, line 200 if X=2, and line 300
        if X=3
40      ON X GOSUB 1000, 3000
        transfers control to subroutine at line
1000    if X=1, and subroutine at line
3000    if X=2
```

Example:

```
10   PRINT A;SPC(10);B
```

Prints value of A, inserts 10 spaces and prints value of B.

READ V1[$],V2[$], . . .

Assigns the data (numeric or string) in DATA statement to the specified variables. This statement must be used with at least one DATA statement.

Example:

```
 10   READ A$, B, C, D%
.
.
.
100   DATA SMITH, 3.5, 2.6, 25
```

REM

Inserts comment lines in the program. Everything after REM is ignored by the computer.

Example:

```
1000   REM **SUB1**
```

RESTORE

Allows data in DATA statements to be read more than once. That

is, it sets the data block pointer back to the beginning of the collection of data values.
Example:
100 RESTORE
sets data pointer to first data item in the program.

RETURN
Exits subroutine and transfers control to the statement following the last GOSUB from which it transferred. A subroutine must have at least one RETURN statement (see GOSUB).

POKE add, n
Is a command that alters the contents of the specified memory location (ie add) to the value specified by n (0-255). The addresses (add) that can be POKEd are between 0 and 65535.
Example:
POKE 1024,4 2 puts * (ASCII 42) in memory location 1024.

PRINT["prompt";V[,V]]
Types out results of computation, messages and/or types out blank line. The results tobe printed are separated by format control characters. These characters are:
Comma "," – causes the output to be printed in the next print zone (four print zones on a line).
Semicolon ";" – causes the output to be printed in a close packed form (ie print position remains at the end of last print).
Examples:
10 PRINT Prints blank line
20 PRINT "AVERAGE=";AVR prints the message inside quotes followed by value of AVR
30 PRINT A+B, A−B prints the sum of A and B and the result of subtracting B from A in the next print zone.

PRINT TAB(n);V
Types out results of computation, messages, etc, at column n.
Example:
10 PRINT TAB(5);A
prints value of A in column 5.

PRINT SPC(n);V
Types out results, etc after n spaces from the current print position.

STOP
Breaks program execution at line containing the STOP statement. Program execution can be resumed by typing CONT [ENTER], which will continue execution from the line following the STOP statement.

SYS(add)
Executes machine language program starting at address specified by add (between 0 and 65535).

WAIT add,V1[,V2]
Takes the contents of the memory location given by add, and XORs it with the second variable V2 and ANDs the result with the

first variable V1. If result is zero then the process is repeated until result is non zero, in which case next statement is executed.
Example:

```
10  POKE 198,0
20  WAIT 198,1
30  PRINT "OK"
```

waits until a key is pressed, and then prints OK.

BASIC FUNCTIONS

Section A: Numeric and Trigonometric Functions

ABS(n)
Returns the absolute value of n (ignores the sign).
Example:
PRINT ABS(−3.5) Prints 3.5
PRINT ABS(A*B) Prints absolute value of A×B

ATN(n)
Returns the arctangent (ie $\tan^{-1}$) of n in radians.

COS(n)
Returns the cosine of angle n given in radians.

EXP(n)
Returns the result of e(2.718) to the power of n.

FRE(n)
Returns the number of free bytes available, regardless of what value n has.

INT(n)
Returns the integer part of n (ie removes decimal fraction)
Example:
PRINT INT(2.99) Prints 2
PRINT INT(−2.01) Prints −3

LOG(n)
Returns the logarithm to base e (ie natural log) of n.

PEEK(add)
Returns the contents of the specified memory location (0-65535).

POS(n)
Returns the column number of the print position on the screen, regardless of what value n has.

RND(n)
Returns random whole number between 0 and 1. If n is zero, the random number generated will be the same as the last one. If n is not zero the generated number will be different.
Example:
PRINT INT(6*RND(1)) prints a random number between 1 and 5.

SGN(n)
Determines whether n is positive, zero or negative. It returns
−1 if n is negative
0 if n is zero
1 if n is positive

SIN(n)
Returns the sine of angle n given in radians.

SPC(n)
Used with PRINT statement to move the print position n space forward.
Example:
PRINT SPC(5);"COMMODORE"
Prints COMMODORE after moving five spaces forward.

SQR(n)
Returns square root of n (will produce error if n is negative).

TAB(n)
Used with PRINT statement to move the print position to the specified column of the current line.
Example:
PRINT TAB(10);"COMPUTER"
prints COMPUTER at column 10 of the current line.

TAN(n)
Returns the tangent of n given in radians.

USR(n)
Transfers control from BASIC to machine language routine whose address is given by the contents of memory locations 784 and 785 (user function jump). The parameter n is passed to the machine language programme, and returns a value back to BASIC program.

SECTION B: STRING FUNCTIONS

ASC(s)
Gives the ASCII code number for the first character of the string s.
Example:
PRINT ASC("COMP")
gives ASCII value of C which is 67
PRINT ASC(A$)
gives ASCII value of first character in A$.

CHR$(n)
Returns the character whose ASCII code is given by n (0-255).
Example:
CHR$(70) gives character F.

LEFT$(S, n)
Returns a substring of string S. The substring begins at the leftmost character of the string S and contains the number of characters specified by n.
Example:
```
10  LET A$="ABCDEFGH"
20  PRINT LEFT$(A$,5)
```
Prints ABCDE

LEN(S)
Returns the number of characters in the string.
Example:
PRINT LEN("COMPUTER")
Prints 8

MID$(S, n, m)
Returns the substring of string S, starting at character position specified by n, and containing the number of characters specified by m. If m is omitted the substring will continue to the end of string S.
Example:
MID$("STRING",2,3)
will give "TRI".

RIGHT$(S, n)
Returns a substring of the string S, which ends at the rightmost character of the string s and contains n characters.

STR$(n)
Returns the string representation of n (ie converts numbers to strings).
Example:
```
10  LET A$="DECEMBER▽" + STR$(26)
```
A$ will contain "DECEMBER▽26". Note: ▽ stands for space.

VAL(S)
Returns the numeric characters of strings as a number (ie inverse of STR$)
Example:
NUM=VAL("1234")
assigns value 1234 to NUM
PRINT VAL("IB1234")
prints 0
PRINT VAL("23.4GRAMS")
prints 23.4

ERROR MESSAGES

BAD SUBSCRIPT
Value of the subscript is greater than the declared dimension.

CAN'T CONTINUE
Program execution will not continue by using CONT command eg error occurred during last execution, or a line has been edited since.

DEVICE NOT PRESENT
The specified I/O device is not connected to the computer.

DIVISION BY ZERO
Attempting to divide a number by zero.

EXTRA IGNORED
A comma is included in data item which is keyed in in response to an INPUT statement. Every item after the comma is ignored.

FILE DATA
String data was received from a data file when numeric data was expected.

FILE NOT FOUND
No such file on tape or disc.

FILE NOT OPEN
Using CLOSE, CMD, PRINT#, etc, statements without using an OPEN statement first.

FILE OPEN
Attempting to open a file that is already open.

FORMULA TOO COMPLEX
String expression needs to be broken into smaller parts for evaluation.

ILLEGAL DIRECT
Using a statement as a direct command, where it can only be used as a statement in program, eg INPUT A.

ILLEGAL QUANTITY
The specified parameters are out of the allowable range.

LOAD
Problems with the program on tape.

NEXT WITHOUT FOR
Incorrect FOR . . . NEXT loop, eg wrong index variable or incorrect nesting.

NOT INPUT FILE
Attempting to input or get data from a file that is specified as an output file.

NOT OUTPUT FILE
Attempting to output data into file that is specified as an input file.

OUT OF DATA
Attempting to read more data when the data block pointer has reached the end of data.

OUT OF MEMORY
All available RAM is being used by the program and variables. Can also be caused by having too many GOSUBS or nesting too many FOR loops.

OVERFLOW
Number too large to be handled by computer (maximum value $\approx \pm 1.7 \times 10^{38}$):

REDIM'D ARRAY
Attempting to redimension an array. Arrays can only be dimensioned once.

REDO FROM START
Attempting to input character data during an input, when numeric data is expected. Re-enter correct data.

RETURN WITHOUT GOSUB
Caused by entering into a subroutine without using a GOSUB statement, eg using GOTO.

STRING TOO LONG
More than 255 characters in string.

SYNTAX ERROR
Unrecognisable command, eg misspelling, incorrect punctuation.

TYPE MISMATCH
Assigning string data to numeric variable or vice-versa.

UNDEF'D FUNCTION
Failure to define referenced user defined function (use DEF FN to define function).

UNDEF'D STATEMENT
Attempting to branch to a line which does not exist.

VERIFY
Program on tape or disc does not match program currently in memory.

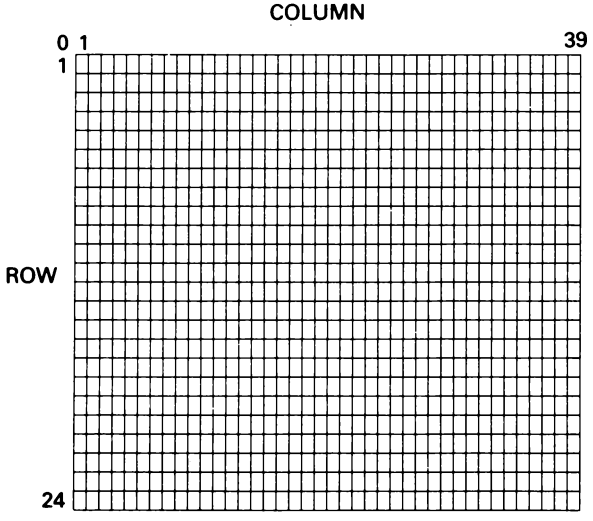
SCREEN DISPLAY

Two modes of Display:
1 Upper case/Graphics POKE 53272,29 Switches to mode i.
2 Upper case/Lower case POKE 53272,31 Switches to mode ii.

Two POKE statements are required to put a character on the screen.
1 POKE screen character code into screen memory (address 1024-2023).
2 POKE colour code into colour memory (address 55296- 56295), corresponding to the same row and column on screen memory.

Screen & colour memory maps

	Screen memory location	Colour memory location
Top left of screen 0th row and 0th column	1024	55296
Top right of screen 0th row and 39th column	1063	55355
. . . and so on until . . .		
Bottom left of screen 24th row and 0th column	1984	56256
Bottom right of screen 24th row and 39th column	2023	56295



Example:
10 POKE 1024+12+40*20 , 83 (83 code for heart)
20 POKE 55296+12+40*20 , 2 (2 code for red)
Puts a red heart at column 12 row 20 on the screen.

Colour codes
Sixteen colours are available.

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

SCREEN DISPLAY CHARACTERS AND CODES

POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2
0	@	@	21	U	u	42	*	
1	A	a	22	V	v	43	+	+
2	B	b	23	W	w	44	,	,
3	C	c	24	X	x	45	–	–
4	D	d	25	Y	y	46		
5	E	e	26	Z	z	47	/	/
6	F	f	27	[	[	48	0	0
7	G	g	28	£	£	49	1	1
8	H	h	29	]	]	50	2	2
9	I	i	30	↑	↑	51	3	3
10	J	j	31	←	←	52	4	4
11	K	k	32	SPACE	SPACE	53	5	5
12	L	l	33	!	!	54	6	6
13	M	m	34	"	"	55	7	7
14	N	n	35	#	#	56	8	8
15	O	o	36	$	$	57	9	9
16	P	p	37	%	%	58		
17	Q	q	38	&	&	59	;	;
18	R	r	39	'	'	60	<	<
19	S	s	40	(	(	61	=	=
20	T	t	41	)	)	62	>	>

POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2
63	?	?	85		U	107		
64			86		V	108		
65	♠	A	87		W	109		
66		B	88	♣	X	110		
67		C	89		Y	111		
68		D	90	♦	Z	112		
69		E	91			113		
70		F	92			114		
71		G	93			115		
72		H	94	π		116		
73		I	95			117		
74		J	96	SPACE	SPACE	118		
75		K	97			119		
76		L	98			120		
77		M	99			121		
78		N	100			122		✓
79		O	101			123		
80		P	102			124		
81	●	Q	103			125		
82		R	104			126		
83	♥	S	105			127		
84		T	106					

Codes from 128-255 are reversed images of codes 0-127.

GRAPHICS

Modes are:

1 Character mode
Taken on switch on, and allows to display all the available symbols on the keyboard. Symbol size is 8 pixels by 8 pixels.

2 High-resolution (bit-map) mode
64000 individually controllable pixels. That is 320 pixels across by 200 pixels down.
Memory requirement 8K (64000/8), where 8 is the number of bits per byte

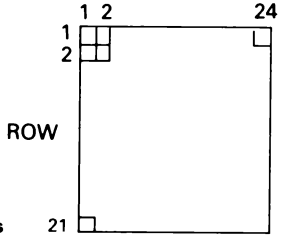
To put 64 into high resolution mode use:
POKE 53265, PEEK (53265) OR 32
To get out of high-resolution mode use:
POKE 53265, PEEK (53265) AND 223

To assign 8K of memory for high-resolution screen use:
POKE 53272, PEEK (53272) OR 8
This moves the high-resolution screen map to locations 8192-16191 and colour screen map to location 1024-2023.
Example:

```
10  POKE 53265, PEEK (53265) OR 32
20  POKE 53272, PEEK (53272) OR 8
30  FOR I=8192 TO 16191:POKEI, 0:NEXT I
40  FOR I=1024 TO 2023:POKE I, 14;NEXT I
50  FOR I=12038 TO 12354:POKE I, 255:NEXT I
60  GOTO 40
```

line 30 clears the contents of memory locations (screen map) 8192-16191
line 40 sets the foreground and backgrounds colours. ie
16* foreground + background
16* 0 + 14 black, light blue
line 50 draws a black horizontal line in the middle of the screen

3 Sprites
A sprite is a graphics object, and is formed from 3 bytes (24 pixels) across by 21 bits (21 pixels) down. This requires 3×21=63 bytes of memory which forms a block, in which sprite data is stored.



Sprite registers

Memory location	**Description**
53248	Sprite 0 x position
53249	Sprite 0 y position
53250-53263	Sprites 1-7 x, y positions
53264	Most significant Bit of x position
53265	Bit-mapped mode and vertical pixel scrolling
53266	Raster register
53267	Light pen x position
53268	Light pen y position
53269	Turn sprite ON/OFF
53270	Multicolour character mode and horizontal pixel scrolling
53271	Sprite expands in y direction
53272	Character memory pointer
53273	Interrupt request
53274	Disable interrupt
53275	Sprite priority
53276	Multicolour sprite mode
53277	Sprite expands x direction
53278	Sprite/sprite collision

53279	Sprite/background collision
53280	Border colour
53281	Background colour 0
53282	Background colour 1
53283	Background colour 2
53284	Background colour 3
53285	Sprite multicolour 0
53286	Sprite multicolour 1
53287-53294	Colour for sprites 0-7

To form a sprite:–

i Work out the data for the required sprite and place it in a 64 byte block in memory starting at block 13 (ie address 13×64=832). Use DATA and READ statements for this purpose.
For example:

row 1 data

bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0
value	128	64	32	16	8	4	2	1																

byte 1 — byte 2

32+16+8=56 — 128+64+32+16+8+4+2+1=255 — 16+8+4=28

row 1
DATA 56, 255, 28, ...

ii Turn on the required sprite use:–
POKE53269, value where value is obtained from:

sprite number	7	6	5	4	3	2	1	0
value	128	64	32	16	8	4	2	1

Example:
POKE 53269,4 turns on sprite 2
POKE 53269,46 turns on sprites 1, 2, 3 and 5.

iii Set sprite point to position where data to be read (ie which block?)

memory location	2040	2041	2042	2043	2044	2045	2046	2047
sprite	0	1	2	3	4	5	6	7

Example:
POKE 2044, 13 indicates data for sprite.
iv Move the sprite on to screen.
POKE 0-255 into the required sprites x position address.
and POKE 0-255 into the required sprites y position address.
Example:
POKE 53256, 85
POKE 53257, 100
moves sprite 4 to point 85,100 on the screen.
v Add colour
Format is:–
POKE 53287-53294, Colour code (0-15)
Example:
POKE 53291,13 changes the sprite 4 colour to light green.

SOUND

Creating sound on Commodore 64 is done with the aid of the 6581 SID (Sound Interface Device) chip. The 64 can generate three individual voices and one, two or all three can be played simultaneously (each having eight octaves, but there is one volume control for all three voices).
Sound on 64 is controlled by the POKE command, and the areas of the memory concerned with sound are 54272 to 54300.

Voice 1

Memory location		**Action**
Decimal	**HEX**	
54272	D400	Low frequency value of note
54273	D401	High frequency value of note
54274	D402	Low pulse
54275	D403	High pulse
54276	D404	Waveform
54277	D405	Attack/Decay
54278	D406	Sustain/Release
Voice 2		
54279	D407	Low frequency value of note
54280	D408	High frequency value of note
54281	D409	Low pulse
54282	D40A	High pulse
54283	D40B	Waveform
54284	D40C	Attack/Decay
54285	D40D	Sustain/Release
Voice 3		
54286	D40E	Low frequency value of note
54287	D40F	High frequency value of note
54288	D410	Low pulse
54289	D411	High pulse
54290	D412	Waveform
54291	D413	Attack/Decay
54292	D414	Sustain/Release
54293	D415	Low frequency cut-off (0-7)
54294	D416	High frequency cut-off (0-255)
54295	D417	Resonance (bits 4-7) Filter voice, 1, bit 0 (turn off voice 1) Filter voice 2, bit 1 (turn off voice 2) Filter voice 3, bit 2 (turn off voice 3)
54296	D418	Volume control for all three voices (bits 0-3). Volume value (0-15) bit 4 low pass filter bit 5 band pass filter bit 6 high pass filter
54297	D419	Access to output of envelope generator of voice 3
54299	D41B	Digitized output from voice 3
54300	D41C	Digitized output from envelope generator

In order to create any sound, the following parameters must be set in the order specified.

1 Volume:– You should POKE a value between 0 (lowest) and 15 (highest).
Example:
POKE 54296,15 sets volume to full.

2 Attack/Decay:– Controls the rate at which the note rises to its peak volume. Each voice (1, 2 and 3) is set separately, but attack and decay are controlled by one register.

AT4	AT3	AT2	AT1	DEC4	DEC3	DEC2	DEC1
128	64	32	16	8	4	2	1

VOICE 1 POKE 54277,
VOICE 2 POKE 54284,
VOICE 3 POKE 54291,
Example:
POKE 54284,240 (128+64+32+16) produces a long attack but no decay for voice 2.

3 Sustain/Release:– Prolongs note at a certain volume and releases it. As in 2, this is set separately for each voice, and both values are controlled by one register.

SUS4	SUS3	SUS2	SUS1	REL4	REL3	REL2	REL1
128	64	32	16	8	4	2	1

VOICE 1 POKE 54278,
VOICE 2 POKE 54285,
VOICE 3 POKE 54292,
Example:
POKE 54278,40 (32+8) gives a low sustain and high release for voice 1.

4 Waveform:– One of four types:–

		Code
(i)	Triangle waveform	17
(ii)	Sawtooth waveform	33
(iii)	Pulse	65
(iv)	Noise	129

Set separately for each voice
POKE 54276, code
POKE 54283, code
POKE 54290, code
Example:
POKE 54290,33 produces a sawtooth waveform for voice 3.

5 High frequency/Low frequency:– To play a note on 64, you should POKE two values for each voice. The values to be POKEd are given in the following table.

Voice 1 POKE 54272, low frequency note value: POKE 54273, high frequency note value
Voice 2 POKE 54279, low frequency note value: POKE 54280, high frequency note value
Voice 3 POKE 54286 low frequency note value: POKE 54287, high frequency note value

Example:
POKE 54272,37: POKE 54273,17 plays middle C

Music note values

The 64 note values to be POKEd into low and high frequency registers of SID chip.

Note	Note-Octave	Hi Freq	Low Freq
0	C–0	1	18
1	C#–0	1	35
2	D–0	1	52
3	D#–0	1	70
4	E–0	1	90
5	F–0	1	110
6	F#–0	1	132
7	G–0	1	155
8	G#–0	1	179
9	A–0	1	205
10	A#–0	1	233
11	B–0	2	6
12	C–1	2	37
13	C#–1	2	69
14	D–1	2	104
15	D#–1	2	140
16	E–1	2	179
17	F–1	2	220
18	F#–1	3	8
19	G–1	3	54
20	G#–1	3	103
21	A–1	3	155
22	A#–1	3	210
23	B–1	4	12
24	C–2	4	73
25	C#–2	4	139
26	D–2	4	208
27	D#–2	5	25
28	E–2	5	103
29	F–2	5	185
30	F#–2	6	16
31	G–2	6	108
32	G#–2	6	206
33	A–2	7	53
34	A#–2	7	163
35	B–2	8	23
36	C–3	8	147
37	C#–3	9	21
38	D–3	9	159
39	D#–3	10	60
40	E–3	10	205

Note	Note-Octave	Hi Freq	Low Freq
41	F–3	11	114
42	F#–3	12	32
43	G–3	12	216
44	G#–3	13	156
45	A–3	14	107
46	A#–3	15	70
47	B–3	16	47
48	C–4	17	37
49	C#–4	18	42
50	D–4	19	63
51	D#–4	20	100
52	E–4	21	154
53	F–4	22	227
54	F#–4	24	63
55	G–4	25	177
56	G#–4	27	56
57	A–4	28	214
58	A#–4	30	141
59	B–4	32	94
60	C–5	34	75
61	C#–5	36	85
62	D–5	38	126
63	D#–5	40	200
64	E–5	43	52
65	F–5	45	198
66	F#–5	48	127
67	G–5	51	97
68	G#–5	54	111
69	A–5	57	172
70	A#–5	61	126
71	B–5	64	188
72	C–6	68	149
73	C#–6	72	169
74	D–6	76	252
75	D#–6	81	161
76	E–6	86	105
77	F6	91	140
78	F#–6	96	254
79	G–6	102	194
80	G#–6	108	223
81	A–6	115	88
82	A#–6	122	52
83	B–6	129	120
84	C–7	137	43
85	C#–7	145	83
86	D–7	153	247
87	D#–7	163	31
88	E–7	172	210
89	F–7	183	25
90	F#–7	193	252
91	G–7	205	133
92	G#–7	217	189
93	A–7	230	176
94	A#–7	244	103

PERIPHERAL CONTROL COMMANDS

Device	Device Number (dn)
Keyboard	0
Cassette recorder	1
Screen	3
Printer	4
Disc drive	8

CLOSE fn
Closes the specified file that has been OPENed for either input or output. If file not closed at the end of writing, data may get lost.
Examples:
CLOSE #1 closes cassette file
CLOSE #4 closes printer channel
CLOSE #2-14 closes the specified disc channel
CLOSE #15 closes the error channel
Note: The error channel should be OPENed first and CLOSEd last.

CMD dn
Sends the output to the specified device instead of the screen.
Example:
OPEN 1,4:CMD4:LIST lists the entire program on the printer.

COPY
Makes a duplicate of a file or program on the same disc drive.
Format is:
PRINT #fn,"COPY0:NEWNAME=0:OLDNAME"
Example:
PRINT #5,"COPY0:TEST2=0:TEST1"

GET#fn,v[$]
Gets one character from the specified file at a time and assigns it to the specified variable. Characters read may include commas, colons, etc, whereas they cannot be read using INPUT # command.
Example:
GET#6, A$ gets a character from file 6 and assigns it to A$

INITIALIZE
Initializes the state of disc as if it has just been switched on.
Format is:
PRINT #fn, "INITIALIZE"
Example: PRINT #6, "INITIALIZE"

INPUT #fn,V[$][,V[$]]
Reads data from specified file and assigns it to the specified varaibles.
Example:
INPUT #1,A,B,C

NEW
Reformats an old disc, erasing all the files. Also used to format a new disc.
Format is:–
PRINT #fn ,"NEW0:name, Id"
Where Id is any two characters.

Example:
PRINT #15,"NEW0:FIRST,00"

OPEN fn, dn, sn, "name"
This command is used before any other peripheral control commands are used. This OPENs file fn to be used by INPUT#, GET#, CMD, CLOSE and PRINT# statements.
Sn = secondary number

Sn	device	description
0	tape	read from file
1	tape	write to file
2	tape	write to file with an End-of-tape marker
4	printer	printer channel
2-14	disc drive	disc channels
15	disc	disc error channel

PRINT # fn, V[$][,V[$]]
Writes the specified variable values to the specified file.
Examples:–
PRINT#5 , A ; CHR$(13);B;CHR$(13);C
writes each value to the file and inserts a RETURN in between to separate data. This is done on cassette file
PRINT#5, A, B$, C$, D
writes the specified values to the specified disc file.

RENAME
Renames a program or a file that is stored on disc.
Format is:
PRINT# fn, "RENAME0:NEWNAME=OLDNAME"
Example:
PRINT#6, "RENAME0:SORT=TOM"

SCRATCH
Erases the unwanted programs or files from disc.
Format is:
PRINT# fn, "SCRATCH0:name"
Example:
PRINT#6, "SCRATCH0:SORT"
deletes SORT from directory.

VALIDATE
Reorganises the files on disc so that the small blocks in between the files are joined to make a bigger block of storage.
Format is:
PRINT# fn, "VALIDATE"
Example:
PRINT #6, "VALIDATE"

VERIFY "name", dn
Checks the program in memory against the one on the specified device for match. If no match VERIFY ERROR is given.
Examples:
VERIFY "PROG1" checks with program stored on tape
VERIFY "PROG1",8 checks with program store on disc

The Century Microguide to the Commodore 64 is a conveniently sized, clearly laid out, quick reference guide for the busy Commodore 64 owner. It comprehensively summarizes all the essential information needed by the Commodore 64 enthusiast and includes:

- Special Keyboard Features
- Alphabetical Quick Reference
- BASIC commands
- Sound, Graphics and Colour
- Input/Output Instructions
- Numeric, Trigonometric and String Functions
- Arithmetic and Logic Operations
- Print and Plot Screens
- Error Handling
- Memory Maps
- System Variables
- Character Sets and Codes
- Disc Operating System Commands

Each command is illustrated with simple examples to show how it is used in context and there are practical hints throughout the book.

CENTURY COMMUNICATIONS LTD.

This was brought to you

from the archives of

http://retro-commodore.eu