

**the
black
book
of**



Robert H. & Dell Taylor



**the
black
book
of
C-128**

Robert H. & Dell Taylor

Pet, Vic-20, C-64, 1541, 1571 and Commodore 128 are trademarks of Commodore Business Machines.
Radio Shack is a trademark of Tandy Corporation.
CP/M Plus Version 3.0, copyright by Digital Research Inc. All rights reserved
CP/M is a registered trademark of Digital Research Inc.

Copyright by Value-Soft Inc., 9513 S.W. Barbur Blvd.
Suite 56, Portland Oregon 97219. All rights reserved.

This manual contains copyrighted material. No part of this manual, may be reproduced, stored in a retrieval system, transmitted in any form, by any means, electronic, mechanical, photocopying, recording or otherwise, without written permission of Value-Soft Inc. No liability is assumed with the respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the authors assume no responsibility for errors or omissions. Neither is any liability assumed for its use or damages resulting from the use of the information contained herein.

TABLE of CONTENTS

WHY BLACK BOOK ?	1
INTRODUCTION	3
HOW TO	8
C-64 MODE	RED 9
BASIC 2.0 Dictionary	10
BASIC 2.0 Abbreviations	16
BASIC 2.0 Error Codes	18
Control Codes	20
Color	22
Graphics	24
Sound	31
Disk Command Summary	36
POKES and PEEKS	38
Memory Map	40
Kernal Routines	63
C-128 MODE	BLUE 65
BASIC 7.0 Dictionary	66
BASIC 7.0 Abbreviations	79
BASIC 7.0 Error Messages	84
Control Codes	86
Escape Codes	89
Function Keys	90
Color	91
Graphics	93
Sound	98
Disk Command Summary	102
POKES and PEEKS	103
M/L Monitor	104
Memory Map	107
Kernal Routines	116

CP/M PLUS 3.0

DARK GREEN 119

Introduction	120
Transient Utility Commands	121
Resident Utility Commands	137
Assignment/Logical Devices	138
Reserved Characters	138
Reserved File Types	139
Control Codes	139

1541 DISK DRIVE

LIGHT GREEN 141

1541 Specifications	142
Diskette Format	142
Sector Distribution	143
1541 BAM	144
1541 Directory Header	144
Directory Format	144
Error Checking	145
Change Device Number	145
DOS Wedge Commands	146
1541 File Types	146
1541 File Formats	147
File Commands	150
Direct Access Commands	152
Internal Disk Commands	152
GCR Conversion	154
Bit Values	154
BASIC 2.0 Commands	155
1541 DOS Error Messages	157
1541 RAM	160

1571 DISK DRIVE

YELLOW 173

1571 Specifications	174
MFM Format	175
GCR Format	175
1571 Diskette Format	176
1571 Sector Distribution	177
1571 BAM	178
1571 Directory Header	178
DOS Shell	179
1571 File Types	183

1571 File Formats	184
File Commands	186
Direct Access Commands	189
Internal Disk Commands	188
Kernal Routines	190
Error Checking	190
Change Device Number	191
Basic 7.0 Disk Commands	192
1571 DOS Error Messages	193

GENERAL INFORMATION ORANGE 197

Numeric Conversion Chart	198
ASCII Conversion Chart	205
Multiplication Table	209
Hex Addition Table	210
Hex Multiplication Table	210
Super Help Chart	211
MCS6510 Instruction Set	218
Symbols & Operators	220
Screen Format Tables	221
Standard Abbreviations	224
Cassette Commands	229

PERSONAL DATA GOLD 230

Equipment Purchases	231
Equipment Repair	237
Supplies	243
Disk IDs	247
Bulletin Boards Local	252
Bulletin Boards Non-Local	256

REFERENCE 260



WHY BLACK BOOK ?

The initial idea, for a manual of this type began in the period of the Vic-20. I had accumulated my notes and tid-bits of information, between the covers of a small black note-book. Soon it bulged with homespun charts and long lists of facts. By the time the 1541 disk drive, and C-64 arrived on the scene, the black note-book had become entirely out of hand. Odd notes were written on scraps of paper with no order whatsoever. My son became curious, and read what I had. I was soon informed, that the information I had been saving would be of use to him also, PLEASE SHARE.

The note-book continued to grow, but with a new symbolism of order. The memory of the C-64 and the 1541 became of special interest. I dumped everything to a printer and triggered the laborious investigation of what was happening. This went on for months. Just as I was about to finish along came the C-128.

The idea of three computers in one case was a challenge beyond my dreams. I purchased the first C-128 I could find. The box sat un-opened, for better than a week. Finally my youngest son ask the question, 'mom why did you buy that thing if you are going to leave it in the box?'. When I told him I had what I wanted , the owner's manual, I received very strange looks. As it turned out, he got the computer, and I got the book. The questions in my mind, soon rose beyond the book, I needed the computer too. My son was so deeply involved with the C-128, that I couldn't get near it. Can you believe, I was forced to buy another entire computer system.

The original black note-book was ragged from use. It had become the most read book in the house. At times we almost fought over that poor tattered thing. Then came the final word, both my sons wanted it up-dated for the C-128 as well as the 1571. Well, if I was to devote that much time to the additions, why not have it printed and share the end result with everyone.

It was a blessing that my youngest son became interested in the project, without his help Black Book of C-128 would not be in print. In the months that followed, he continually reminded me of an old computer terminology, 'KISS' (Keep It Simple Stupid). Kiss became a household word. Every time I sat down to my desk to write, a member of the family, would remind me to KISS. A hug goes to Robert, without his help, I could never have finished.

This is a book about a computer. I felt it would be appropriate, that the text be printed, directly from my print-outs, and not type-set by a print shop. The following equipment/software was used:

- Commodore 128
- Commodore 1571 Disk Drive
- Commodore 1541 Disk Drive
- Commodore 1701 Video Monitor
- Panasonic Printer K-X 1092
- Easyscript (word processor)

Welcome to the pages of my little black note-book. I sincerely hope you find it as useful, as the members of my family do. If you have suggestions or subjects you would like to see added to a future update, please drop me a note via the publisher.

Dell Taylor

'granny'
Feb. 1986

INTRODUCTION

The Black Book of C-128 has been developed as a vast reference source to assist programmers of all levels from the beginning BASIC student, to the experienced machine language writer. Think of it as a dictionary or extensive reference source. The aim of this book, is to support the user, not teach BASIC programming.

The Commodore 128 is one of a long line of computers from Commodore Electronics Ltd. Originally the company repaired typewriters. The first computer designed by Commodore, was for Radio Shack Inc.. After review of the new machine, they decided to design and manufacture the TRS-80 in it's place. Commodore went on to market the successful Pet (Personal Electronic Transactor). Many claim the Pet was really named for the then popular 'Pet Rock'. The series was a leader in the industry and many are still in use today.

Commodore had the foresight to see the coming market for personal computers. The vision of a computer in every home was the dream of all computer developers. There were three questions yet to be answered, why, how, and is the public really ready for the move.

The next computer designed by Commodore, was the TOI (The Other Intellect) or as some people joked, Tools Of Idiots. The TOI was put aside to develop an inexpensive color computer, the Vic-20. The TOI was not to be, but many of it's exceptional features found a place in the future C-64.

The Vic-20 was the breakthrough, to the home market, for Commodore. With sound, color, and low price tag, the stage was set for the entire industry. Other new computers appeared, and dis-appeared, most could not or would not compete with the volume marketing techniques of Commodore.

The arrival of the Commodore 64, set industry standards for the home computer. A personal computer must have color, sound, graphics, and most of all be easy

to use. The new version of the 6502 chip, opened new worlds for the purchaser. The flexibility of RAM and ROM, allowed use of languages other than BASIC, plus better control for the user.

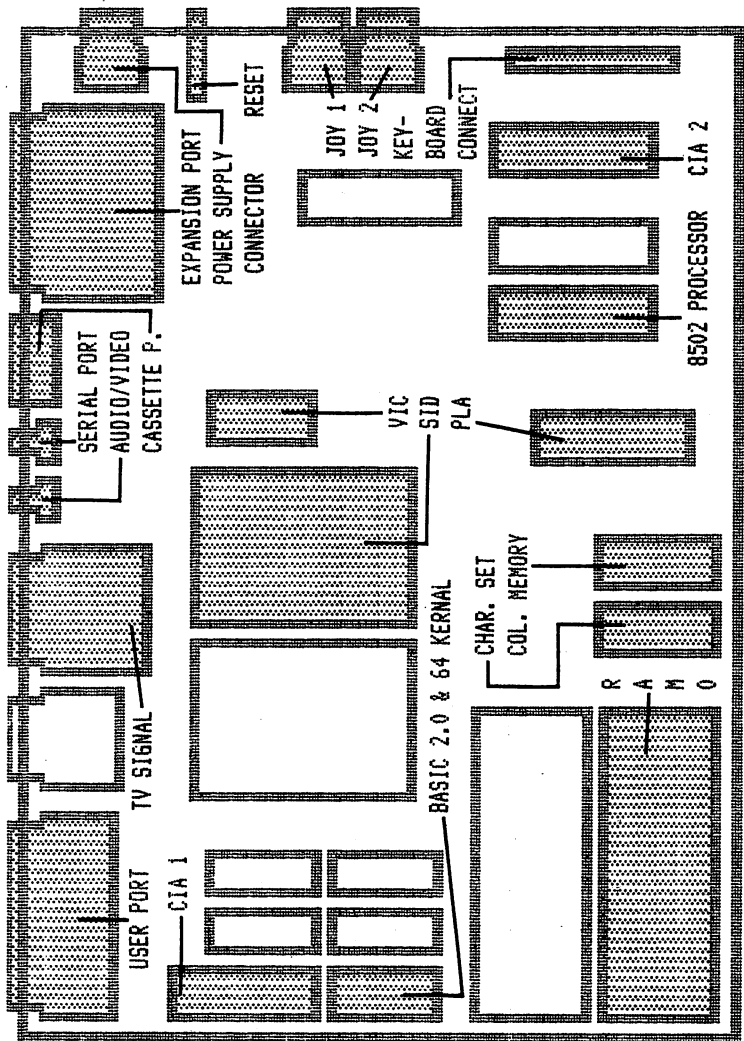
Today the Commodore 128 is taking the personal computer market by storm. Three computers, in one case, that is a bargain.

* C-64 MODE..All the features of the reliable C-64, compatibility with hundreds of the existing software packages and the C-64 peripherals. The color, graphics, and sound abilities, are identical to the original. BASIC 2.0, 40 column output and access to 64K RAM (Random Access Memory) have been retained.

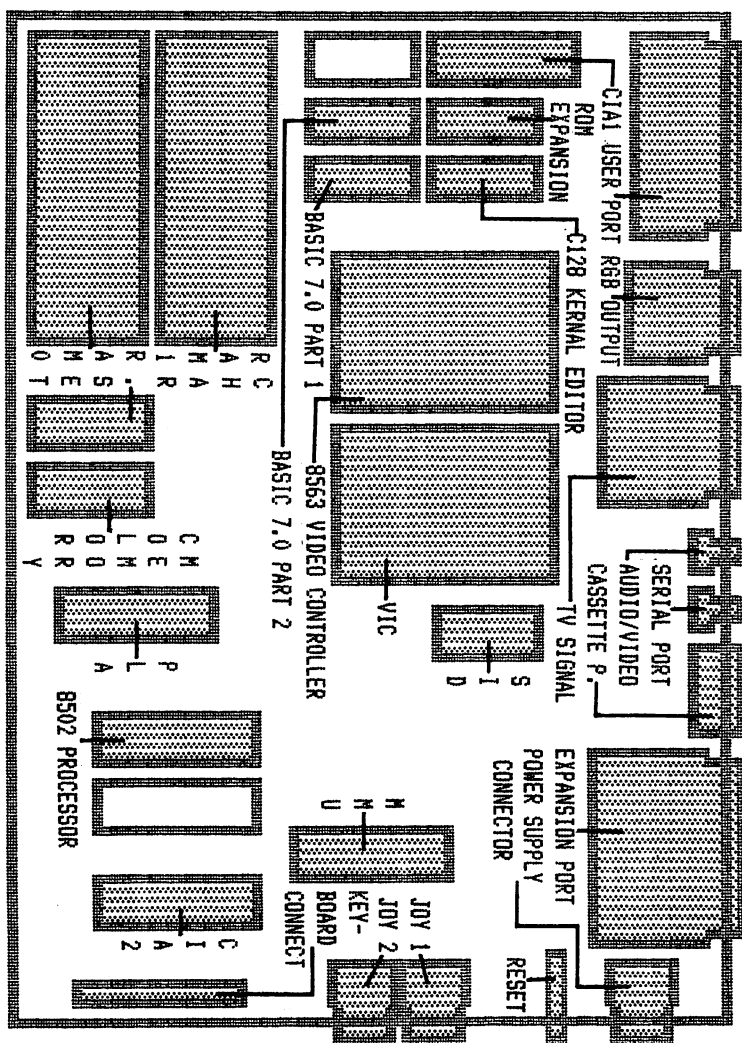
* C-128 MODE...128K of available RAM (Random Access Memory), expandable to 512K. BASIC 7.0 has over 150 easy to use commands. The 128 mode, offers 40 or 80 column output, plus a 92 key keyboard including numeric keypad. A machine language monitor is available via pre-programmed function keys.

*CP/M MODE.....CP/M is the abbreviation for Control Program for Microprocessors. This mode enables a 280 microprocessor chip, 59K of RAM (Random Access Memory), 40 or 80 column video output, full 92 key keyboard, and access to the new fast serial disk drive.

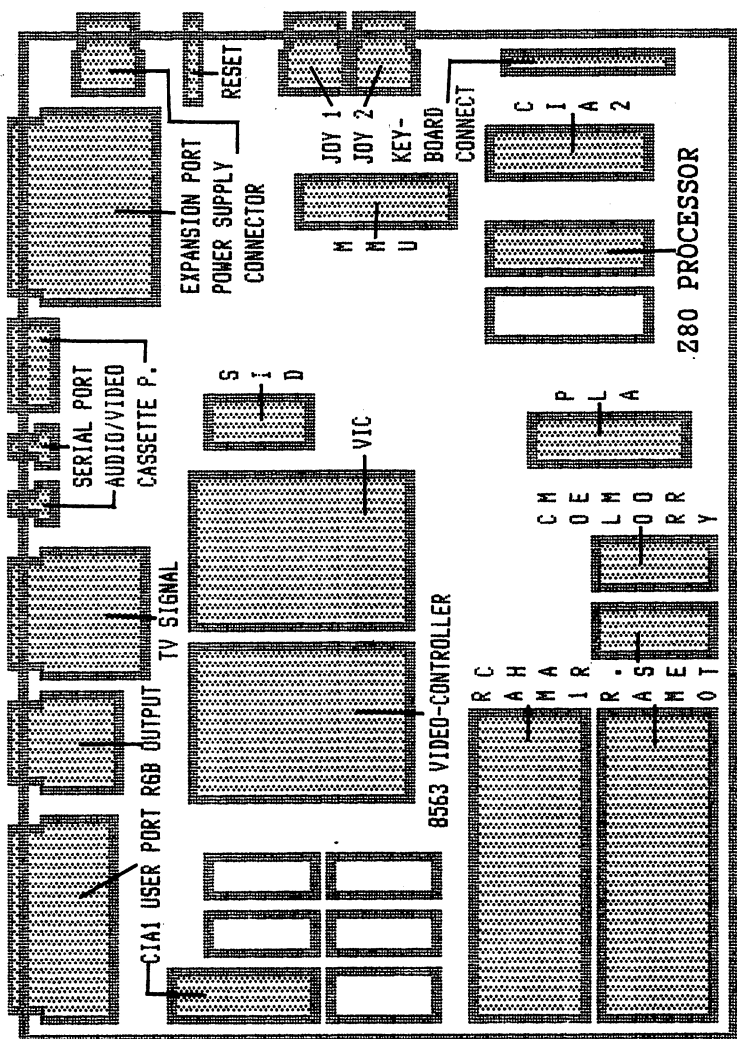
As you proceed from one mode to another, the configuration of components within C-128 change. The next three pages, are diagrams of which components are in use for each of the three modes. The shaded areas of the drawings are in use, the blank ones are not.



C-64 MODE



C-128 MODEL



CP/M MODE

HOW TO USE THE BLACK BOOK of C-128

This book was written in a manner, to quickly answer questions on a given subject. The standard format of text often becomes to lengthy, for the quick identification of material. Information has been converted to chart or table form when ever possible. Text has purposely been kept to a minimum.

Each of the following subjects, are identified by a different color.

C-64 MODE	RED
C-128 MODE	BLUE
CP/M MODE	DARK GREEN
1541 DISK DRIVE	LIGHT GREEN
1571 DISK DRIVE	YELLOW
GENERAL INFORMATION	ORANGE
PERSONAL	GOLD

A color divider, has been placed at the beginning of each of the above subjects. The color is also at the top of each page in the given section.

The page following each colored divider, contains a detailed table of contents for the chapter.

When ever possible, an example of use, or the proper syntax (command sentence structure) is included. The following format has been used for all syntax of BASIC 2.0 and 7.0:

1. BASIC keywords are shown in capital letters.
2. Items shown within quote marks (" ") are user definable.
3. Items shown within square brackets ([]) are an optional range.

BASIC 2.0 Dictionary	10
BASIC 2.0 Abbreviations	16
BASIC 2.0 Error Messages	18
Control Codes	20
Color	22
Graphic Summary	24
Sprite Functions	25
Sprite Pointers	25
Sprite Off/On	25
Sprite Color	26
Multi-color Sprite	26
Vertical (Y) Position	27
Horizontal (X) Position	27
Horizontal (X) Position (256-511)	28
Expanded Sprite	29
Sprite Priorities	29
Sprite Collisions	30
Sound	31
Sound Registers	31
Musical Notes	32
Attack, Decay, Release Rates	35
Disk Command Summary	36
POKES and PEEKS	38
Memory Map	40
RS-232 Pseudo 6551 Registers	45
MOS 6566 Video Interface Controller (VIC II)	53
MOS 6581 Sound Interface Device (SID)	54
MOS 6526 Complex Interface Adapter (CIA) #1	55
MOS 6526 Complex Interface Adapter (CIA) #2	56
Operating System	56
Kernal Routines	63
Hardware Vectors	64

BASIC 2.0 DICTIONARY

The computer language most commonly used by Commodore computers, is referred to as BASIC, (Beginner's All Purpose Symbolic Instruction Code). It was developed by John Kemeny and Thomas Kurtz in the mid 60's as an alternate to machine language. People could relate to common words much easier than a lot of numbers. As in any language, BASIC has it's own vocabulary. The format, in which BASIC code must be entered, is referred to as syntax. Computers are very fussy about how you talk to them, infact at times, down right un-friendly if code is not written just the way it likes. When an error has transpired, the message 'SYNTAX ERROR' will appear on your monitor. This is telling you to check spelling, punctuation, etc.

The BASIC 2.0 reserved words, are the language of the C-64. These words are also included, in BASIC 7.0 of the C-128. You may use both 2.0 and 7.0 in the C-128 mode.

The following pages are a mini-dictionary of 7.0 keywords. On the first line of each entry is the keyword and an example of the proper syntax. The second line contains a brief definition, of the function the word performs.

ABS 20 x=ABS(8.25)
Returns the absolute value of a number.

AND if k=3 AND y=0 then500
Checks the truth of multipule operands.

ASC 100 print ASC(a\$)
Returns the ASCII value of a character.

ATN 230 print ATN(0)
Returns the artangent of a number.

CHR\$ 200 print CHR\$(48)
Converts Commodore ASCII to it's char. equivalent.

CLOSE 100 CLOSE15
To CLOSE an open logical file or device.

CLR 10 CLR
Clears a program file of all variables.

CMD 200 CMD1
To send a file to the desired channel.

CONT CONT
Restarts a program after a STOP or END command

COS 200 COS(0)
Calculates the cosine of a number.

DATA 40 DATA 48,0,255
Used to store information within a program.

DEF FN 20 DEF FN z(3)=x + y
A user defined function for later use in a program.

DIM 10 DIM a\$(100)
Defines an array or matrix of variables.

END 900 END
Ends a program run and displays the READY message.

EXP 100 print EXP(3)
Returns the value 2.71 raised to the x power.

FN 100 print FN b(x)
The value of a function created by DEF FN statement.

FOR 100 FOR x=1 to 200
Used with NEXT to repeat a section of code for a set number of times.

FRE 100 print FRE(0)
Returns the number of unused bytes. The value will be negative if over 32K.

GET 100 GET a\$:if a\$="" then100
To GET one character from the keyboard.

GET# 100 GET#5,a\$
Get one byte from an external device.

GOSUB 100 GOSUB500
Tells the computer to branch to a subroutine.

GOTO 100 GOTO450
Tells the computer to GOTO a specified line number.

IF THEN 100 IF a(>)0 THEN600
A statement to evaluate and command to do.

INPUT 100 INPUT n\$
Prints question mark and waits for user input.

INPUT# 100 INPUT#5,a\$
To retrieve data stored in a file.

INT 200 print INT(a)
Returns the integer value of a number.

LEFT\$ 200 a\$=LEFT\$(x\$,1,3)
The left most character of a string.

LEN 140 if LEN(a\$)=4 then450
Returns number of characters in string expression.

LET 100 LET a=20
To assign a value to an array.

LIST LIST
To look at a program in memory, screen or print same

LOAD LOAD "menu",8
Reads contents of file from disk or tape into memory

memory

LOG 100 x=LOG(45/7)
Returns the natural logarithm of argument

MID\$ 100 a\$=MID\$(x\$,1,5)
A sub-string of a larger string.

NEW NEW
Clears BASIC memory and all variables.

NEXT 300 NEXT X1
Used with FOR to create an end of a FOR NEXT loop.

NOT 500 if NOT dd = xx then 850
Reverses the meaning of a statement.

ON GOTO 600 ON a GOT500,600,700
GOTO the line number shown after the GOTO command.

ON GOSUB 600 ON a GOSUB500,600,700
Goto the line number shown after the GOSUB command.

OPEN 700 OPEN15,8,15
OPENS channel for input/output to peripheral device.

OR 700 if a=3 OR z=1 then 450
Links two expressions into a single compound.

PEEK 300 if PEEK(49152)=162 then 800
Returns the value of a memory location.

POKE 100 POKE53280,1
To change the contents of a memory location.

POS 200 if POS(0)>40 then print chr\$(13)
Identifies the current cursor position.

PRINT 100 PRINT "hello"
To output data to the screen.

PRINT# 150 PRINT#4,"hello"
To output data to a file or device.

READ 160 READ d\$
To fill variables from a DATA statement.

REM 10 REM program name: mail
To be used for user REMarks or re-minders.

RESTORE 20 RESTORE
Resets pointer to the first statement.

RETURN 100 RETURN
To exit a sub-routine called by a GOSUB.

RIGHT\$ 120 if RIGHT\$(x\$,1)="y" then 450
The right most character of a string.

RND 100 a=(RND(0)*100)
Generates a random number sequence.

RUN RUN
To start a program currently in memory.

SAVE SAVE "menu",8
To store a file to disk or tape.

SGN 400 if SGN(d)<0 then print "error":end
Returns the sign (pos., neg., or zero) of X.

SIN 300 x=SIN(a)
Gives the sine of the <numeric> argument, in radians

SPC(400 print SPC(20) "hello"
Skip X spaces forward on the same line.

SQR 350 a=SQR(d)
Returns the square root value of a positive number.

STATUS 400 if ST<>0 then print "error"
Shows STATUS of last input or output performed on an open file.

STEP 450 for x=1 to 250 STEP 2
Defines the value of increment for a loop

STOP 900 STOP
To stop the execution of a program.

STR\$ a\$=STR\$(a)
Converts a number to a string.

SYS 800 if st<0 then SYS64738
Execute a machine language program at xxxxx memory location.

TAB(100 print TAB(15) "welcome"
Move X spaces forward on the same line.

TAN 355 x=TAN(a)
Returns the tangent value of expression in radians.

TI 675 print TI
Reads the interval timer (jiffy clock).

TI\$ 850 TI\$=x\$
Used to set time in the internal timer.

TO 555 for x=1 TO 500:next x
Used as part of the FOR statement.

USR 855 d=USR(a)
Transfer to a machine language program.

VAL 655 v=VAL(x\$)
To convert a string to a number.

VERIFY VERIFY "menu",8
Compares a saved file against one in memory.

WAIT 335 WAIT1,32,32
To pause until data matches.

BASIC 2.0 ABBREVIATIONS

Most BASIC commands, statements, and functions can be abbreviated to a shorter form. Listed in column one, are the keywords. Columns two and three are the equivalent in hexadecimal and decimal. Column four is the abbreviated version.

To enter an abbreviation: type the letter or letters shown before the word shift. Next hold down the shift key and type the remaining letters.

KEYWORD	HEX	DECIMAL	SHORT-HAND
ABS	\$B6	00182	A shift B
AND	\$AF	00175	A shift N
ASC	\$C6	00198	A shift S
ATN	\$C1	00193	A shift T
CHR\$	\$07	00199	C shift H
CLOSE	\$A0	00160	CL shift O
CLR	\$9C	00156	C shift L
CMD	\$9D	00157	C shift M
CONT	\$9A	00154	C shift O
COS	\$BE	00190	N/A
DATA	\$83	00131	D shift A
DEF	\$96 \$A5	00150 00165	D shift E
DIM	\$86	00134	D shift I
END	\$80	00128	E shift N

KEYWORD	HEX	DECIMAL	SHORT-HAND
EXP	\$BD	00189	E shift X
FN	\$A5	00165	N/A
FOR	\$81	00129	F shift O
FRE	\$B8	00184	F shift R
GET	\$A1	00161	G shift E
GET #	\$A1 \$23	00161 00035	N/A
GOSUB	\$8D	00141	GO shift S
GOTO	\$89	00137	G shift O
IF	\$8B	00139	N/A
INPUT	\$85	00133	N/A
INPUT#	\$84	00132	I shift N
INT	\$B5	00181	N/A
LEFT\$	\$C8	00200	LE shift F
LEN	\$C3	00195	N/A
LET	\$88	00136	L shift E
LIST	\$9B	00155	L shift I
LOAD	\$93	00147	L shift O
LOG	\$8C	00188	N/A
MID\$	\$CA	00202	M shift I
NEW	\$A2	00162	N/A
NEXT	\$82	00130	N shift E
ON	\$91	00145	N/A
OPEN	\$9F	00159	O shift P
OR	\$B0	00176	N/A
PEEK	\$C2	00194	P shift E
POKE	\$97	00151	P shift O
POS	\$B9	00185	N/A
PRINT	\$99	00153	?
PRINT#	\$98	00152	P shift R
READ	\$87	00135	R shift E
REM	\$8F	00143	N/A
RESTORE	\$8C	00140	RE shift S
RETURN	\$8E	00142	RE shift T
RIGHT\$	\$C9	00201	R shift I
RND	\$BB	00187	R shift N
RUN	\$8A	00138	R shift U
SAVE	\$94	00148	S shift A
SGN	\$B4	00180	S shift G
SIN	\$BF	00191	S shift I

KEYWORD	HEX	DECIMAL	SHORT-HAND
SPC(	\$A6	00166	S shift P
SQR	\$BA	00186	S shift Q
STATUS	\$53 \$54	00083 00084	ST
STEP	\$A9	00168	ST shift E
STOP	\$90	00144	S shift T
STR\$	\$C4	00196	ST shift R
SYS	\$9E	00158	S shift Y
TAB(	\$A3	00163	T shift A
TAN	\$C0	00192	N/A
TIME	\$54 \$49	00084 00073	TI
TIME\$	\$54 \$49 \$24	00084 00073 00036	TI\$
TO	\$A4	00164	N/A
USR	\$B7	00183	U shift S
VAL	\$C5	00197	V shift A
VERIFY	\$95	00149	V shift E
WAIT	\$92	00146	W shift A

BASIC 2.0 ERROR MESSAGES

An error message is the computer's way of telling you something is wrong. When exploring for the cause, you must remember that a computer is very exacting in its demands. One forgotten comma, will generate a syntax error. I recently talked to a gentleman, that went so far as to name his computer shop 'Syntax Error'. The idea came from using his computer. Programmers have seen the message so often, that they never forget it, talk about free advertising.

The clues to an error, are often subtle. Stop, think, take time to review the problem completely. The answer is usually in plain sight. The computer can only follow your instructions.

BAD DATA	String data received from open file, numeric data expected
BAD SUBSCRIPT	Array number outside range in Dim statement

BREAK IN XX	STOP pressed during execution
CAN'T CONTINUE	CONT not possible at this time
DEVICE NOT PRESENT	Required I/O device not on line
DIVISION BY ZERO	By zero not allowed
EXTRA IGNORED	Extra input data ignored
FILE NOT FOUND	File not found on device
FILE NOT OPEN	Tried to access a closed file
FILE OPEN	Tried to open an open file
FORMULA TOO COMPLEX	String operation too complex for the computer
ILLEGAL DIRECT	INPUT may be used in program only, not in direct mode
ILLEGAL QUANTITY	Argument out of allowable range
LOAD ERROR	Tape PRG has loading error
NEXT WITHOUT FOR	Executed NEXT before FOR found
NOT INPUT FILE	Tried to read a file opened for write only
NOT OUTPUT FILE	Tried to write file opened for read only
OUT OF DATA	No data left to read
OUT OF MEMORY	All available memory in use
OVERFLOW	Numerical result greater than maximum
REDIM'D ARRAY	Tried to DIM an array more than once, this is not allowed
REDO FROM START	Wrong data type entered
RETURN WITHOUT GOSUB	RETURN found before a GOSUB
STRING TOO LONG	Limit is 255 characters
SYNTAX	Error in punctuation, spelling, etc.
TYPE MISMATCH	Wrong data type
UNDEF'D FUNCTION	Function un-defined by DEF FN
UNDEF'D STATEMENT	Line number does not exist
VERIFY ERROR	Program in memory does not match one in device

C-64 CONTROL CODES

The CONTROL may be found at the middle left-hand side of the keyboard. The key generates control functions. These are often special tasks, or short cuts, to much larger jobs. To perform a control function, hold down the CTRL key, and press the other key that is called, for at the same time. The CHR\$ functions are an another way of doing these same tasks within BASIC code. The CHR\$ refers to a table, of character string codes in the computers memory. The CHR\$ function converts a Commodore ASCII code to its character equivalent.

C= Commodore logo key, location, lower left corner of the keyboard.

char. = character

CHR\$	CTRL KEY	FUNCTION
CHR\$(5)	CTRL2/E	Set char./color to white
CHR\$(8)	CTRL H	Disable char. set change
CHR\$(9)	CTRL I	Enable char. set change
CHR\$(10)	CTRL J	Carriage RETURN/line feed
CHR\$(13)	CTRL M	Send carriage RETURN
CHR\$(14)	CTRL N	Char. set to up/low/case
CHR\$(17)	CRSR DN/CTRL Q	Cursor down one row
CHR\$(18)	CTRL 9	Reverse field characters
CHR\$(19)	HOME/CTRL S	Cursor to home position
CHR\$(20)	DEL/CTRL T	Delete char./fill hole
CHR\$(24)	CTRL X	Tab set/clear
CHR\$(28)	CTRL3/CTRL/	Character color to red
CHR\$(29)	CRSR/CTRL]	Cursor one column right
CHR\$(30)	CTRL6/CTRL	Character color to green
CHR\$(31)	CTRL7	Character color to blue
CHR\$(34)	"	Prints a quote mark
CHR\$(129)	C=1	Character color to orange
CHR\$(131)		Run PRG, keyboard only
CHR\$(133)	F1	Reserved CHR\$ for F1 key
CHR\$(134)	F3	Reserved CHR\$ for F3 key
CHR\$(135)	F5	Reserved CHR\$ for F5 key
CHR\$(136)	F7	Reserved CHR\$ for F7 key

CHR\$	CTRL KEY	FUNCTION
CHR\$(137)	F2	Reserved CHR\$ for F2 key
CHR\$(138)	F4	Reserved CHR\$ for F4 key
CHR\$(139)	F6	Reserved CHR\$ for F6 key
CHR\$(140)	F8	Reserved CHR\$ for F8 key
CHR\$(141)	SHIFT/RETURN	Carriage RETURN/line feed without BASIC code
CHR\$(142)		Char. set up/case/graphic
CHR\$(144)	CTRL 1	Character color to black
CHR\$(145)	CRSR UP	Move cursor up one row
CHR\$(146)	CTRL 0	Reverse field off
CHR\$(147)	HOME	Cursor to home position
CHR\$(148)	INST	Insert command
CHR\$(149)	C=2	Character color to brown
CHR\$(150)	C=3	Character color light red
CHR\$(151)	C=4	Character color dk.gray
CHR\$(152)	C=5	Character color med. gray
CHR\$(153)	C=6	Character color light grn.
CHR\$(154)	C=7	Character color lt. blue
CHR\$(155)	C=8	Character color lt. gray
CHR\$(156)	CTRL 5	Character color to purple
CHR\$(157)	CRSR LEFT	Cursor left one column
CHR\$(158)	CTRL8	Character color to yellow
CHR\$(159)	CTRL 4	Character color to cyan

C-64 COLOR

The Commodore 64 offers the user a rainbow of sixteen colors. The screen, border, or character color may be changed to any of the sixteen hues. There are several options accessible, to the user, to alter color. The chart, on the next page, lists the attainable colors, plus the three modes available to change color on the Commodore 64.

In the first column are the sixteen available colors. Many color combinations may be used, but some may become blurred when placed next to another. With trial and error, you will soon learn, which hues compliment each other.

The second column, is the value to be added to a POKE command to change border, screen, or character color.

EXAMPLE: POKE 53280,2 This command will change the border of the screen, to red. By changing the number after the comma, the border can be changed to any of the sixteen colors. POKE 53281,2 changes the entire screen to red, but not the border.

The commands in column three (labeled KEY) are used to modify character color. The CONTROL and COMMODORE keys are found on the left hand side of the keyboard. To change the character color, hold down the CONTROL or COMMODORE LOGO key plus the number of the color at the same time. While in the quote mode, colors may be altered as often as needed.

EXAMPLE: Press the CONTROL key and the number seven. From now, until another color command is entered, all characters will be in yellow.

The fourth column displays the CHR\$ value for each of the sixteen colors. The primary use for these, are to alter the character color within program code.

EXAMPLE: CHR\$(129) All characters will be printed to the screen in orange.

COLOR	VALUE	KEY	CHR\$
BLACK	0	CONTROL+1	CHR\$(144)
WHITE	1	CONTROL+2	CHR\$(5)
RED	2	CONTROL+3	CHR\$(28)
CYAN	3	CONTROL+4	CHR\$(159)
PURPLE	4	CONTROL+5	CHR\$(156)
GREEN	5	CONTROL+6	CHR\$(30)
BLUE	6	CONTROL+7	CHR\$(31)
YELLOW	7	CONTROL+8	CHR\$(158)
ORANGE	8	COMMODORE+1	CHR\$(129)
BROWN	9	COMMODORE+2	CHR\$(149)
LIGHT RED	10	COMMODORE+3	CHR\$(150)
DARK GRAY	11	COMMODORE+4	CHR\$(151)
MED. GRAY	12	COMMODORE+5	CHR\$(152)
LIGHT GREEN	13	COMMODORE+6	CHR\$(153)
LIGHT BLUE	14	COMMODORE+7	CHR\$(154)
LIGHT GRAY	15	COMMODORE+8	CHR\$(155)

SCREEN, BORDER, and CHARACTER COLOR

Border color	POKE 53280,value X
Screen color	POKE 53281,value X
Character color	POKE 55296,value X

GRAPHIC SUMMARY

The Commodore 64 has several types, of unique graphic features. Text may be displayed, in a rainbow of various colors. Elaborate graphic images can be created by connecting letters, numbers, special symbols, and graphic characters of the three character sets.

A second type of graphic display, is the bit-map mode for drawing intricate pictures and designs. Included in this category are both standard bit-map and multi-color bit-map mode. In the bit-map modes, you control each pixel, thus allowing great detail in your work.

The third type of screen display, is a combination of text and bit-map graphics. This allows a split screen as you might see in an adventure game, picture at the top and text at the bottom.

Another graphic feature is the sprite. A sprite is a movable graphic shape, designed by a user. The image may be animated, colored, or multi-colored. A sprite is 24 pixels wide X 21 pixels high in regular bit map mode, or 12 pixels wide X 21 high in multi-color bit map mode.

IMPORTANT MEMORY LOCATIONS FOR GRAPHICS

1024	\$0400	Start of screen memory
2023	\$07E7	End of screen memory
53265	\$D011	Bit map mode on or off
		Screen blanking on or off
53270	\$D016	Multi-color mode on or off
53272	\$D018	Location of screen memory
53272	\$D018	Controls choice of character sets
53280	\$D020	To change the color of the border
53281	\$D021	To change the color of the screen
55295	\$D7FF	Start of color memory
55296	\$D800	End of color memory
56334	\$DC0E	Turns interrupts on or off
56576	\$DD00	Bank selection

SPRITE FUNCTIONS

SPRITE POINTERS

Sprite number	0	1	2	3	4	5	6	7
Pointer	2040	2041	2042	2043	2044	2045	2046	2047
Value	192	193	194	195	196	197	198	199
Address	12288	12352	12416	12480	12544	12608	12672	12736
EXAMPLE: VALUE X 64 = ADDRESS								

SPRITE OFF/ON

Sprite number	0	1	2	3	4	5	6	7
Sprite ON Value	1	2	4	8	16	32	64	128
EXAMPLE: POKE 53269,value = sprite on 0 = sprite off								
If more than one sprite is to be on at the same time, add the two figures together and enter in the place of 'value'								
EXAMPLE: POKE 53269,69 = sprites 0, 2, 6, (1+4+64=69) are now on.								

SPRITE COLOR

Sprite number	0	1	2	3	4	5	6	7
Address	53287	53288	53289	53290	53291	53292	53293	53294
Colors	0 = black	4 = purple	8 = orange	12 = med. gray				
	1 = white	5 = green	9 = brown	13 = lt. green				
	2 = red	6 = blue	10 = lt. red	14 = lt. blue				
	3 = cyan	7 = yellow	11 = dark gray	15 = lt. gray				
EXAMPLE: POKE 53289,5 = sprite green Choose the sprite to color (0-15) the address to POKE is below that number. After the five digit number, add a comma and the number of the color value.								

MULTI-COLOR SPRITE

Sprite number	0	1	2	3	4	5	6	7
Value	1	2	4	8	16	32	64	128
EXAMPLE: POKE 53276,value = multi-color sprite If more than one sprite is to be multi-color at the same time, add the two figures together and enter in the place of 'value'. 0 = multi-color off. EXAMPLE: POKE 53276,168 = sprites 3, 5, 7 (8+32+128=168) in multi-color.								

VERTICAL (Y) POSITION

Sprite number	0	1	2	3	4	5	6	7
Poke	53249	53251	53253	53255	53257	53259	53261	53263
The physical screen begins in the upper left corner of the screen (out in the border). The vertical range for each sprite is from 0 to 255, 0 = upper left, 255 = lower left corner. Visible viewing begins at line 50, this must be taken into account in placing a sprite. EXAMPLE: POKE 53253,150 = the top of sprite 2 at the about center of the left edge of the screen.								

HORIZONTAL (X) POSITION

Sprite number	0	1	2	3	4	5	6	7
Address	53248	53250	53252	53254	53256	53258	53260	53262

The visible screen begins in upper left corner. The horizontal range for each sprite is from 0 to 511, 0 = left, 511 = right. Only values 24 to 343 are visible on the screen. You may only poke from 0 to 255, therefore, an additional register is needed for the positions beyond 255 (256-511).

HORIZONTAL (X) POSITION (256-511)

Sprite number	0	1	2	3	4	5	6	7
Value	1	2	4	8	16	32	64	128

This second register can only hold either a zero or a sprite value (0 = off, value = on). This tells the computer if the right side of the screen is to be used.

EXAMPLE: POKE 53264,0 = Right side not used

If the position of the sprite is less than 255, enter the value for the position.

EXAMPLE: POKE 53252,100 = Sprite 2 in the 100 position.

If the position is to be greater than 255, change the value after POKE 53264 to a sprite value. This equals the first 255 positions. The value of the second POKE equals plus 256.

EXAMPLE: POKE 53264,2:POKE 53252,25 = Sprite 1 in the 280th (256+25 = 280) position.

If more than one sprite is to be on the right side of the screen, add the values together and POKE the value into 53264. EXAMPLE: POKE 53264,84 = Sprites 2, 4, 6, (4+16+64=84) will use the right side of the screen.

EXPANDED SPRITE

Sprite number	0	1	2	3	4	5	6	7
Value	1	2	4	8	16	32	64	128
EXAMPLE: POKE 53277, value = wide sprite POKE 53271, value = tall sprite If more than one sprite is to be expanded, add the values together. EXAMPLE: POKE 53277, 20 = sprites 2 and 4 (4+16=20) expanded in width.								

SPRITE PRIORITIES

Sprite number	0	1	2	3	4	5	6	7
Value	1	2	4	8	16	32	64	128
EXAMPLE: POKE 53275, value = gives the sprite priority over other figures on the screen. The lower the value of a sprite, the higher the priority on the screen. if sprite 1 were to pass in front of sprite 6, 1 would block 6 from view. If more than one sprite should have priority at the same time, add the two figures together and enter in the place of 'value' EXAMPLE: POKE 53275,6 = Sprite 1 has priority over sprite 2 (2+4=6).								

SPRITE COLLISIONS

Sprite number	0	1	2	3	4	5	6	7
Value	1	2	4	8	16	32	64	128
If a sprite overlaps another sprite or character, it is a collision. You may PEEK the following location to see if one has occurred: EXAMPLE: PEEK 53278, value = sprite collision PEEK 53279, value = character collision To test more than one sprite add the values together. EXAMPLE: PEEK 53278,10 = sprites 1 and 3 (2+8=10) have encountered a collision.								

C-64 SOUND

The Commodore 64 is a musical wonder. The other day I listened to the thundering 1812 Overture (with cannon of course) played on a C-64. It was truly fantastic. A friend of mine says he can see the day when we attend a concert of computers. I can envision 76 C-64's marching on parade. This might sound like a joke, but stop and think of the possibilities it presents.

The chip that commands the sound for the C-64, is SID (Sound Interface Device). SID contains three separate registers (voices). There are twenty five separate memory locations, that control these registers. Each of the three voices, have seven memory locations that control the type of sound produced by that voice. The following table, is read from left to right, starting with the memory location.

64 SOUND REGISTERS**54272-54278 VOICE 1**

ADDRESS	REGISTER DESCRIPTION
54272	Lower half of tone frequency value
54273	Upper half of tone frequency value
54274	Lower half of pulse width value
54275	Upper half of pulse width value
54276	Waveform control register
54277	Attack and Decay control register
54278	Sustain/Release control register

54279-54285 VOICE 2

54279	Lower half of tone frequency value
54280	Upper half of tone frequency value
54281	Lower half of pulse width value
54282	Upper half of pulse width value
54283	Waveform control register
54284	Attack and Decay control register
54285	Sustain/Release control register

54286-54292 VOICE 3

54286	Lower half of tone frequency value
54287	Upper half of tone frequency value
54288	Lower half of pulse width value
54289	Upper half of pulse width value
54290	Waveform control register
54291	Attack and Decay control register
54292	Sustain/Release control register

54293-54296 FILTERING

54293	Lower half of cutoff filter value
54294	Upper half of cutoff filter value
54295	Resonance/sound filtering values
54296	Mode and volume controls

The following chart illustrates octave, musical note, frequency produced, frequency produced by SID, high and low byte POKE values. Sid can produce sound waves at any frequency between 0 and 3995 hertz. This compares to about the same pitch range as a piano.

64 MUSICAL NOTES

OCTAVE	MUSICAL NOTE	FREQUENCY PRODUCED	SID FREQUENCY	HIGH BYTE VALUE	LOW BYTE VALUE
0	C	16.4	269	1	13
0	C#	17.3	284	1	28
0	D	18.4	302	1	46
0	D#	19.4	318	1	62
0	E	20.6	338	1	82
0	F	21.8	358	1	102
0	F#	23.1	379	1	123
0	G	24.5	402	1	146
0	G#	26.0	427	1	171
0	A	27.5	451	1	195
0	A#	29.1	477	1	221
0	B	30.9	507	1	251

OCTAVE	MUSICAL NOTE	FREQUENCY PRODUCED	SID FREQUENCY	HIGH BYTE VALUE	LOW BYTE VALUE
1	C	32.7	536	2	24
1	C#	34.6	568	2	56
1	D	36.7	602	2	90
1	D#	38.9	638	2	126
1	E	41.2	676	2	164
1	F	43.7	717	2	205
1	F#	46.2	758	2	246
1	G	49.0	804	3	36
1	G#	51.9	851	3	83
1	A	55.0	902	3	134
1	A#	58.3	956	3	188
1	B	61.7	1012	3	244
2	C	65.4	1073	4	49
2	C#	69.3	1137	4	113
2	D	73.4	1204	4	180
2	D#	77.8	1276	4	252
2	E	82.4	1352	5	72
2	F	87.3	1432	5	152
2	F#	92.5	1517	5	237
2	G	98.0	1608	6	72
2	G#	103.8	1703	6	167
2	A	110.0	1804	7	12
2	A#	116.5	1911	7	119
2	B	123.5	2026	7	234
3	C	130.8	2146	8	98
3	C#	138.6	2274	8	226
3	D	146.8	2408	9	104
3	D#	155.6	2553	9	249
3	E	164.8	2703	10	143
3	F	174.6	2864	11	48
3	F#	185.0	3035	11	219
3	G	196.0	3215	12	143
3	G#	207.7	3407	13	79
3	A	220.0	3609	14	25
3	A#	233.1	3824	14	240
3	B	246.9	4050	15	210

OCTAVE	MUSICAL NOTE	FREQUENCY PRODUCED	SID FREQUENCY	HIGH BYTE VALUE	LOW BYTE VALUE
4	C	261.6	4291	16	195
4	C#	277.2	4547	17	195
4	D	293.7	4818	18	210
4	D#	311.1	5103	19	239
4	E	329.6	5407	21	31
4	F	349.2	5728	22	96
4	F#	370.0	6070	23	182
4	G	392.0	6431	25	31
4	G#	415.3	6813	26	157
4	A	440.0	7218	28	52
4	A#	466.2	7648	29	224
4	B	493.9	8102	31	166
5	C	523.3	8584	33	136
5	C#	554.4	9095	35	135
5	D	587.3	9634	37	162
5	D#	622.3	10208	39	224
5	E	659.3	10815	42	63
5	F	698.5	11458	44	194
5	F#	740.0	12139	47	107
5	G	784.0	12861	50	61
5	G#	830.6	13625	53	57
5	A	880.0	14436	56	100
5	A#	932.3	15294	59	190
5	B	987.8	16204	63	76
6	C	1046.5	17167	67	15
6	C#	1108.7	18188	71	12
6	D	1174.7	19270	75	70
6	D#	1244.5	20415	79	191
6	E	1318.5	21629	84	125
6	F	1396.9	22915	89	131
6	F#	1480.0	24278	94	214
6	G	1568.0	25722	100	122
6	G#	1661.2	27251	106	115
6	A	1760.0	28872	112	200
6	A#	1864.7	30589	119	125
6	B	1975.5	32407	126	151

OCTAVE	MUSICAL NOTE	FREQUENCY PRODUCED	SID FREQUENCY	HIGH BYTE VALUE	LOW BYTE VALUE
7	C	2093.0	34334	134	30
7	C#	2217.5	36377	142	25
7	D	2349.3	38539	150	139
7	D#	2489.0	40831	159	127
7	E	2637.0	43258	168	250
7	F	2793.8	45831	179	7
7	F#	2960.0	48557	189	173
7	G	3136.0	51444	200	244
7	G#	3322.4	54502	212	230
7	A	3520.0	57743	225	143
7	A#	3729.3	61177	238	249
7	B	3951.1	64815	253	47

ATTACK, DECAY, RELEASE RATES

VALUE	ATTACK RATE TIME CYCLE	DECAY/RELEASE RATE TIME CYCLE
0	2 ms	6 ms
1	8 ms	24 ms
2	16 ms	48 ms
3	24 ms	72 ms
4	38 ms	114 ms
5	56 ms	168 ms
6	68 ms	204 ms
7	80 ms	240 ms
8	100 ms	300 ms
9	250 ms	750 ms
10	500 ms	1.5 s
11	800 ms	2.4 s
12	1 s	3 s
13	3 s	9 s
14	5 s	15 s
15	8 s	24 s

C-64 DISK COMMAND SUMMARY

The commands listed below are used with the 1541 disk drive or the 1571 in C-64 mode.

CLOSE PRINT#15:CLOSE15
To close channel that has been opened to receive data

CMD CMD1
To redirect screen output to a disk file.

COPY PRINT#15,"C:new file=old file"
To make a copy (duplicate) of a file.

GET #
To read one byte from a record.

INITIALIZE PRINT#15,"I"
Returns the disk drive to power-up condition.

INPUT# GET#5,a\$
Reads a whole record from the disk.

LOAD

LOAD"file name",8	Load a BASIC program.
LOAD"*,8	Load first program of DIR.
LOAD"file name",8,1	Load machine language program.
LOAD"file name",8,3	Automatic load and run.
LOAD"\$",8	Loads directory of disk.
LOAD"A\$",8	Loads contents of variable (A\$)
LOAD"T*",8	Loads first file beginning with 'T'.
LOAD"JAN*",8	Loads first file beginning with 'JAN'.

NEW

PRINT#15,"N:name,ID" To NEW a disk (format).
PRINT#15,"N:name" To clear a directory.

OPEN OPEN 15,8,15
Opens a file or channel for input or output.

PRINT# PRINT#5,p\$
To output data to file or channel.

RENAME PRINT#15,"R:new name=old name"
To rename a file on a diskette.

SAVE SAVE"file name",8
To SAVE a file in memory to disk.

SAVE/REPLACE SAVE"@:file name",8
To SAVE a new program and replace old.

SCRATCH PRINT#15,"S:name"
To erase (SCRATCH) a file.

VALIDATE PRINT#15,"V"
To re-organize or collect scattered blocks and make space available on the diskette.

VERIFY VERIFY"file name",8
To compare a saved file with one in memory

ERROR CHANNEL To read the error channel,
 type the following:
 10 OPEN15,8,15
 20 INPUT#15,A\$,B\$,C\$,D\$
 30 PRINT A\$,B\$,C\$,D\$
 40 CLOSE 15

C-64 HANDY POKES and PEEKS

The C-64 mode has 65,536 individual memory locations, each may store a number ranging from 0 to 255. A PEEK allows the user, to read the number stored at a given address. A POKE lets you write data into a RAM (Random Access Memory) location. ROM (Read Only Memory) may be read, but cannot be written to. POKE and PEEK commands may be used to extend your BASIC programming skills. As you can see, by the following chart, many useful and unusual things can take place. The list is just a beginning, explore other locations. You won't damage your computer, and you may learn a lot.

POKE or PEEK	ACTION
POKE 1,0	Disable operating system. Default ,1
POKE 19,65	Disable ? in input statement. Default value: 0
POKE 22,35	LIST without line #'s. Default ,25
PEEK 57	Number of the current BASIC line
PEEK 59	Number of prior BASIC line
POKE 120,0	Computer will go crazy. Default ,123
POKE 145,127	Acts the same as the RUN/STOP key
PEEK 152	Number of devices currently open
PEEK 154	Output device number
PEEK 187	Current open device number in use
PEEK 197	Returns ASCII value of key pressed
POKE 198,0	Clears the keyboard buffer
POKE 199,1	Screen reverse off, 0 = on
POKE 204	Cursor flash toggle, 1 = off, 0 = on
PEEK 206	Character under cursor in ASCII
PEEK 209/210	Current screen line address
PEEK 211	Cursor location on current line
POKE 213	Screen line length, default: ,39
PEEK 214	Current cursor position line number
PEEK 243/244	Current screen color RAM location
PEEK 646	Current character color value
POKE 649	Size of keyboard buffer. Default:,10
POKE 649,0	Disables keyboard. Default:,10
POKE 650,0	Disables all keys, except cursor keys and space bar

POKE or PEEK	ACTION
POKE 650,64	Disables all repeat of keys
POKE 650,128	Makes all keys repeat
POKE 657,1	Disables SHIFT & COMMODORE keys
POKE 774,0	To vanish LISTing, leaves line #. Default value: ,26
POKE 774,141	Listing vanishes. Default value: ,26
POKE 775,168	To disable LIST. Prints ?UNDEF'D STATEMENT ERROR. Default: ,167
POKE 775,171	To cause the computer to crash if a LIST attempt made. Default: ,167
POKE 775,200	To disable LIST. Default: 775,167
POKE 788,52	Disable RUN/STOP/TI\$. Default: ,49
POKE 792,193	Disable STOP/RESTORE. Default: ,71
POKE 793,203	To disable RESTORE. Default: ,254
POKE 808,127	Disables RUN. Default ,237
POKE 808,225	Disable RUN/STOP/RESTORE,Default:,237
POKE 808,230	Disables proper listing. Default:,237
POKE 808,232	Disables LIST/SAVE, program may not be stopped during run. Default: ,237
POKE 808,234	Disables STOP/RESTORE/LIST. Default value: 237
POKE 808,239	Disables RUN/STOP. Default: ,237
POKE 818,32	Disables SAVE function. Default:,237
POKE 818,32	Press shifted RUN/STOP key, program in memory will auto run. Default 165
POKE 816,157	Disables LOAD function. Default ,165
POKE 819,246	Stops all attempts to SAVE. READY message is displayed. Default: ,245
POKE 53265,11	For screen blanking. Saves run time. Default value: 27
POKE 53269,0	Turns off sprites, saves 5% run time
POKE 53272	,21 = upper case ,23 = lower case
POKE 53280	Screen border color
POKE 53281	Screen color
POKE 53296,1	Change operating speed from 1 MHZ to 2 MHZ. Default value : ,0
POKE 56325,X	Changes cursor speed. 0 = fast, 58 = normal, 255 = slow
SYS 62913	Displays name of current file

64 MODE MEMORY MAP

The memory map of C-64 mode, is like a large filing cabinet. Imagine the memory locations, as single drawers, filled with information. The content of some of the drawers may be moved, modified, or read as is. A number of the drawers are locked, you can look but not make changes. By being aware of what is available, you can extend your programming skills. Remember, a computer can only do as you tell it.

The subsequent pages contain a broad memory map for C-64 mode. All of the addresses are identified in hexadecimal, decimal, and by label. This is followed by a description of how the data at this location is used by the computer.

C-64 MEMORY MAP

HEX	LABEL	DECIMAL	ACTION
0000	D6510	0	Chip directional register
0001	R6510	1	Chip input/output, memory and tape control
0002	ADRAY1	2	UNUSED
0003	ADRAY1	3	Vector: floating point to integer
0005	ADRAY2	5	Vector: integer to floating point
0007	CHARAC	7	Search character for BASIC text
0008	ENDCHR	8	Scan end of string for quote flag
0009	TRMPOS	9	Tab column save
000A	VERCK	10	LOAD/VERIFY flag:0=LOAD, 1=VERIFY
000B	COUNT	11	Input buffer pointer/# subscript
000C	DIMFLG	12	Default array DIM flag
000D	VALTYP	13	Flag: type of data
000E	INTFLG	14	Flag: type of numeric data
000F	GARBFL	15	Flag: data scan LIST,quote marks
0010	SUBFLG	16	Flag: subscript reference, FNX
0011	IMPFLG	17	Flag: 0=input, \$40=GET, \$98=READ
0012	TANSGN	18	Flag:TAN or SIN result comparison
0013	CHANNL	19	Flag: current input/output prompt
0014	LINNUM	20	Pointer: temporary string storage
0016	TEMPPT	22	Pointer: temporary string stack
0017	LASTPT	23	Pointer: last temp. string vector
0019	TEMPST	25	Stack for temporary strings
0022	INDEX	34	Pointer: for number transfer
0024	INDEX	36	Miscellaneous number pointer
0026	RESHO	38	Floating point area-mult./divide
0028	TXTTAB	43	Pointer: start of BASIC text
002B	VARTAB	45	Pointer: start of BASIC variables
002F	ARYTAB	47	Pointer: start of BASIC arrays
0031	STREND	49	Pointer: end of BASIC arrays
0033	FRETOP	51	Pointer: bottom of string storage
0035	FRESPC	53	Pointer: temporary for strings
0037	MEMSIZ	55	Pointer: limit of BASIC memory
0039	CURLIN	57	Pointer:current BASIC line number
003B	OLDLIN	59	Pointer: prior BASIC line number
003D	OLDTXT	61	Pointer: BASIC statement for cont
003F	DATLIN	63	Pointer: current data line number
0041	DATPTR	65	Pointer:address current data item

HEX	LABEL	DECIMAL	ACTION
0043	INPPTR	67	Temp.storage while INPUT/GET/READ
0045	VARNAM	69	Pointer: Current BASIC variable
0047	VARPNT	71	Pointer: current BASIC var. data
0049	FORPNT	73	Pointer: variable for FOR/NEXT
004B	OPPTR	75	Pointer: temporary data storage
004D	OPMASK	77	Comparison symbol accumulator
004E	DEFPNT	78	Misc. work area, pointers etc.
0050	DSCPNT	80	Pointer to string description
0052	DSCPNT	82	Length of above string
0053	FOURS	83	Constant used by garbage collect
0054	JMPER	84	Jump vector for functions
0057	TEMPF1	87	BASIC numeric work area
005C	TEMPF2	92	Temporary storage

Memory location 00097 through 00112 are arguments for the floating-point arithmetic routines stored in the KERNAL ROM. They are central to the execution of any BASIC math operation. They also provide the assembly language programmer, the ability to perform any math operation on two numbers, without writing the BASIC routines.

0061	FACEXP	97	Accumulator #1: exponent
0062	FACHO	98	Accumulator #1: mantissa
0066	FACSGN	102	Accumulator #1: sign
0067	SGNFLG	103	Pointer: evaluation constant
0068	BITS	104	Accumulator #1: overflow
0069	ARGEXP	105	Accumulator #2: exponent
006A	ARGHO	106	Accumulator #2: mantissa
006E	ARGSGN	110	Accumulator #2: sign
006F	ARISGN	111	Sign comparison result, math
0070	FACOV	112	Accumulator #1 lo-order(rounding)
0071	FBUFPT	113	Pointer: cassette buffer
0073	CHRGET	115	CHRGET subroutine, get BASIC char
007A	TXTPTR	122	Pointer: curr. byte of BASIC text
008B	RNDX	139	RND function seed value
0090	STATUS	144	Status word (ST)
0090		144	Kernal work storage area
0091	STKEY	145	Flag: has stop key been pressed?

HEX	LABEL	DECIMAL	ACTION
0092	SVXT	146	Timing for tape
0093	VERCK	147	Flag: load=0, verify=1
0094	C3PO	148	Flag:serial bus, output char buff
0095	BSOUR	149	Buffered character for serial bus
0096	SYNO	150	End of tape received
0097	XSAV	151	Temporary X register save area
0098	LDTND	152	Number of open files, maximum 10
0099	DFLTN	153	Input device, normally 0
009A	DFLTO	154	Default output (CMD) device
009B	PRTY	155	Tape character output parity
009C	DRSW	156	Flag: tape byte received
009D	MSGFLG	157	Flag: display of error message
009E	PTR1	158	Tape pass 1 error log/char buffer
009F	PTR2	159	Tape pass 2 error log corrected
00A0	TIME	160	Jiffy clock, hrs.,min.,sec.
00A3	TSFCNT	163	Temporary data storage
00A4	TBTCNT	164	Cycle count,serial input/output
00A5	CNTDN	165	Countdown, tape write/bit count
00A6	BUFPNT	166	Pointer:# of char. in tape buffer
00A7	INBIT	167	RS-232 input bit storage
00A8	BITC1	168	RS-232 input bit count
00A9	RINONE	169	RS-232 flag:start bit error check
00AA	RIDATA	170	RS-232 input byte buffer
00AB	RIPRTY	171	RS-232 input parity
00AC	SAL	172	Pointer: tape buffer
00AE	EAL	174	Pointer: tape, end address for LOAD, SAVE, VERIFY.
00B0	CMPO	176	Tape timing constants
00B2	TAPE1	178	Pointer:address start/tape buffer
00B4	BITTS	180	RS-232 tape bit counter
00B5	NXTBIT	181	RS-232 bit to send/read or end
00B6	RODATA	182	RS-232 char. error/output buffer
00B7	FNLEN	183	Number of char. in file name
00B8	LA	184	Current logical file number
00B9	SA	185	Current secondary address
00BA	FA	186	Current device number
00BB	FNADR	187	Pointer: current file name
00BD	ROPRTY	189	RS-232 output parity
00BE	FSBLK	190	Cassette read/write block count

HEX	LABEL	DECIMAL	ACTION
00BF	MYCH	191	Tape input word buffer
00C0	CAS1	192	Cassette motor interlock
00C1	STAL	193	Pointer: start address, cassette load and write
00C3	MEMUSS	195	Pointer: kernal setup
00C5	LSTX	197	Last key pressed
00C6	NDX	198	# characters in keyboard buffer
00C7	RVS	199	Flag: reverse char. MODE: normal=0, Value= 18
00C8	INDX	200	Pointer: end of line for input
00C9	LXSP	201	Cursor log (row, column)
00CB	SFDX	203	Keyscan interrupt routine
00CC	BLNSW	204	Flag: cursor blink, 0=on, 1=off
00CD	BLNCT	205	Timer: cursor blink
00CE	GOBLN	206	Character under cursor in ASCII
00CF	BLNON	207	Flag: last cursor blink on or off?
00D0	CRSW	208	Flag: input/get from scr./keyboard
00D1	PNT	209	Pointer: current screen line add.
00D3	PNTR	211	Position of cursor on curr. line
00D4	QTSW	212	Flag: editor in quote mode, 0=off
00D5	LNMX	213	Current screen line length
00D6	TBLX	214	Current line where cursor is
00D7	SCHAR	215	Temporary storage
00D8	INSRT	216	Flag: insert mode
00D9	LDTB1	217	Screen line link table
00F2		242	Screen row marker
00F3	USER	243	Pointer: to address of current screen color RAM
00F5	KEYTAB	245	Pointer: to keyboard decode table
00F7	RIBUF	247	Pointer: RS-232 input buffer
00F9	ROBUF	249	Pointer: RS-232 output buffer
00FB	FREKZP	251	Four free bytes for ML routines
00FF	BASZPT	255	BASIC temporary work area
0100		256	Floating point to ASCII work area
0100-103E		256-318	Tape input error log
0100-01FF		256-511	6510 processor hardware stack
013F-01FF		319-511	BASIC stack area
0200-0258		512-600	BASIC input buffer
0259	LAT	601	Logical file number table

HEX	LABEL	DECIMAL	ACTION
0263	FAT	611	Device numbers table
026D	SAT	621	Secondary active file add. table
0277	KEYD	631	Keyboard buffer
0281	MEMSTR	641	Pointer: start of BASIC memory
0283	MEMSIZ	643	Pointer: top of BASIC memory
0285	TIMOUT	645	Flag: serial bus IEEE
0286	COLOR	646	Current color code of character
0287	GDCOL	647	Color under cursor
0288	HIBASE	648	High byte screen memory address
0289	XMAX	649	Maximum size of keyboard buffer
028A	RPTFLG	650	Flag: repeat for keys
028B	KOUNT	651	Key repeat speed counter
028C	DELAY	652	Counter for timing delay
028C	SHFLAG	653	Flag: signals if SHIFT,CTRL, or COMMODORE LOGO key is pressed
028E	LSTSHF	654	Last keyboard shift pattern
028F	KEYLOG	655	Pointer: keyboard table setup
0291	MODE	657	Keyboard SHIFT mode
0292	AUTODN	658	Flag: screen scroll enabled

** RS-232 PSEUDO 6551 REGISTERS **

0293	M51CTR	659	RS-232 mock 6551 control register
0294	M51CDR	660	RS-232 mock 6551 comm. registers
0295	M51AJB	661	RS-232 bit timing
0297	RSSTAT	663	RS-232 status register
0298	BITNUM	664	RS-232 number of bits to send
0299	BAUDOF	665	Time needed to send a bit
029B	RIDBE	667	Beg. & end of input/output buffer
029B	RIDBE	667	Pointer: to end of RS-232 256 byte receive buffer
029C	RIDBS	668	Pointer: to start of RS-232 256 byte receive buffer
029D	RODBS	669	Pointer: to RS-232 output buffer
029E	RODBE	670	Pointer: to end of RS-232 256 byte transmit buffer
029F	IRQTMP	671	IRQ SAVE during tape input/output

HEX	LABEL	DECIMAL	ACTION
02A1	ENABL	673	RS-232 interrupt enabled
02A2	TODSNS	674	CIA #1 timer
02A3	TRDTMP	675	CIA #1 interrupt register B
02A4	TD1IRQ	676	CIA #1 control register A
02A5	TLNIDX	677	Screen row marker
02A6	TVSFLG	678	Flag: for PAL or NTSC
02A7-02FF		679-767	UNUSED
0300-030B		768-779	BASIC indirect vector table
0300	IERROR	768	Pointer: print BASIC error message
0302	IMAIN	770	Vector: BASIC warm start
0304	ICRNCH	772	Vector: tokenize BASIC text
0306	IQPLOP	774	Vector: prints tokens link list
0308	IGONE	776	Vector: execute BASIC prog. token
030A	IEVAL	778	Vector: evaluates a single-term arithmetic expression
030C		780	Register storage area
030C	SAREG	780	Temporary storage during SYS of A-register.
030D	SXREG	781	Temporary storage during SYS of X-register.
030E	SYREG	782	Temporary storage during SYS of Y-register.
030F	SPREG	783	Temporary storage during SYS of .P-register.
0310	USRPOK	784	USR function, jump instructions
0311	USRADD	785	High & low byte, USR start address
0313		787	UNUSED
0314	CINV	788	Vector: hardware interrupt address
0316	CBINV	790	Vector: break instruction interr.
0318	NMINV	792	Vector: Hardware Non-Maskable Interrupt (NMI)
031A-032D		794-813	Kernal indirect vectors
031A	IOPEN	794	Vector: to kernal OPEN routine
031C	ICLOSE	796	Vector: to Kernal CLOSE routine
031E	ICHKIN	798	Vector: to Kernal CHKIN routine
0320	ICKOUT	800	Vector: to Kernal CHKOUT routine
0322	ICLRCH	802	Vector: to Kernal CLRCHN routine
0324	IBASIN	804	Vector: to Kernal CHRIN routine
0326	IBSOUT	806	Vector: to Kernal CHROUT routine

HEX	LABEL	DECIMAL	ACTION
0328	ISTOP	808	Vector: to Kernal STOP routine.
032A	IGETIN	810	Get vector.
032C	ICLALL	812	Abort input/output vector.
032E	USRCMD	814	Warm start vector.
0330	ILOAD	816	LOAD link.
0332	ISAVE	818	Vector: to Kernal SAVE routine.
0334		820	UNUSED-8 bytes free
033C	TBUFFR	828-1019	Cassette input/output buffer
0340		832	Sprite 13
0380		896	Sprite 14
03C0		960	Sprite 15
03FC		1020	UNUSED
0400	VICSCN	1024	Vid. matrix 25 line X 40 column
07F8	SPNTRS	2040	Pointer: sprite shape data
1000-1FFF		4096-8191	Character ROM image
8000		32768	Autostart ROM cartridge
9000		36864	Char. ROM image, VIC II chip
A000		40960	Vector: warm start
A000		40960	ROM: BASIC interpreter or 8K/RAM
A002	BWARM	40962	Vector: warm start
A004		40964	ASCII Text characters CBM BASIC
A00C	STMDSP	40972	Dispatch vector table, statement
A052	FUNDSP	41042	Dispatch vector table, function
A080	OPTAB	41088	Dispatch vector table, operator
A09E	RESLST	41118	Keyword table
A19E		41374	ASCII text, BASIC error messages
A19E	ERRTAB	41374	BASIC error message #1
A1AE		41390	BASIC error message #2
A1B6		41398	BASIC error message #3
A1C6		41414	BASIC error message #4
A1CE		41422	BASIC error message #5
A1E6		41446	BASIC error message #6
A1F6		41462	BASIC error message #7
A206		41478	BASIC error message #8
A216		41494	BASIC error message #9
A226		41510	BASIC error message #10
A236		41526	BASIC error message #11
A23E		41534	BASIC error message #12
A24E		41550	BASIC error message #13

HEX	LABEL	DECIMAL	ACTION
A25E		41566	BASIC error message #14
A26E		41582	BASIC error message #15
A276		41590	BASIC error message #16
A27E		41598	BASIC error message #17
A296		41622	BASIC error message #18
A29E		41630	BASIC error message #19
A2AE		41646	BASIC error message #20
A2BE		41662	BASIC error message #21
A2CE		41678	BASIC error message #22
A2D6		41686	BASIC error message #23
A2EE		41710	BASIC error message #24
A2F6		41718	BASIC error message #25
A306		41734	BASIC error message #26
A30E		41742	BASIC error message #27
A31E		41758	BASIC error message #28
A326		41766	BASIC error message #29
A328	ERRPTR	41768	Error message vector table
A365		41829	BASIC interpreter messages
A38A	FNDFOR	41866	Stack routine, GOSUB/FOR-NEXT
A3B8	BLTU	41912	Checks for free memory
A3FB	GETSTK	41979	Test for stack depth
A408	REASON	41992	Check available space in memory
A435	OMERR	42037	Out of memory error handler
A437	ERROR	42039	General error handler
A469		42089	Prints error or break message
A474	READY	42100	Prints the message READY
A47E		42110	New BASIC line processing
A480	MAIN	42112	Main BASIC program loop
A49C	MAIN1	42140	Add or replace line of text
A4A9		42153	Erase program line
A4ED		42221	Insert program line
A533	LINKPRG	42291	BASIC line chaining
A560	INLIN	42336	Receive line from the keyboard
A579	CRUNCH	42361	Tokenize BASIC line
A613	FNDLIN	42515	Search for line number
A642	SCRATCH	42562	Perform BASIC command NEW
A65E	CLEAR	42590	Perform BASIC command CLR
A68E	RUNC	42638	Reset BASIC to start of program
A69C	LIST	42652	Perform BASIC command LIST

HEX	LABEL	DECIMAL	ACTION
A742	FOR	42818	Perform BASIC command FOR
A7AE	NEWSTT	42926	Set up next statement
A7ED	GONE	42980	Execute next BASIC statement
A81D	RESTOR	43037	Perform BASIC command RESTORE
A820		43052	Test STOP key
A82F	STOP	43055	Perform BASIC command STOP
A831	END	43057	Perform BASIC command END
A857	CONT	43095	Perform BASIC command CONT
A871	RUN	43121	Perform BASIC command RUN
A883	GOSUB	43139	Perform BASIC command GOSUB
A8A0	GOTO	43168	Perform BASIC command GOTO
A8D2	RETURN	43218	Perform BASIC command RETURN
A8F8	DATA	43256	Perform BASIC command DATA
A906	DATAN	43270	Scan for next statement
A909	REMN	43273	Scan for next line
A928	IF	43304	Perform BASIC command IF
A93B	REM	43323	Perform BASIC command REM
A94B	ONGOTO	43339	Perform ON GOTO or ON GOSUB
A96B	LINGET	43371	Get integer from text
A9A5	LET	43429	Perform BASIC command LET
AA80	PRINTN	43648	Perform BASIC command PRINT#
AA86	CMD	43654	Perform BASIC command CMD
AA9A	PRINT	43680	Perform BASIC command PRINT
AB1E	STROUT	43806	Print string from any memory
AB3B		43835	Print format character
AB4D	DOAGIN	43853	Process bad input
AB7B	GET	43899	Perform BASIC commands GET/GET#
ABA5	INPUTN	43941	Perform BASIC command INPUT#
ABBF	INPUT	43967	Perform BASIC command INPUT
ABF9	QINLIN	44025	Prompt and input
AC06	READ	44038	Perform BASIC command READ
ACFC	EXIGNT	44284	Input error messages
AD1E	NEXT	44318	Perform BASIC command NEXT
AD78		44408	Type match check
AD8A	FRMNUM	44426	Evaluate numeric expression
AD8F	CHKSTR	44431	Check input was a string
AD9E	FRMEVL	44446	Evaluate expression
AE83	EVAL	44675	Convert single arithmetic term
AEA8	PIVAL	44712	Constant - PI

HEX	LABEL	DECIMAL	ACTION
AEF1	PARCHK	44785	Evaluate within brackets
AEF7	CHKCLS	44791	Check for ")"
AEFA	CHKOPN	44794	Check for "("
AEFD		44797	Check for "--"
AEFF	CHKOPN	44799	Check for ", "
AF08	SNERR	44808	Prints SYNTAX error
AF14	RSVVAR	44820	Checks for variable name
AF2B	ISVAR	44843	Get the value of a variable
AFA7	ISFUN	44967	Set up function references
AFE6	OROP	45030	Perform BASIC command OR
AFE9	ANDOP	45033	Perform BASIC command AND
B016	DORE1	45078	Comparison routine for <, >, =
B081	DIM	45185	Perform BASIC command DIM
B08B	PTRGET	45195	Locate variable area
B0E7		45287	Find variable, return location
B113		45331	Check .A register for alpha ASCII character
B11D	NOTFNS	45341	Create new space
B185	FINPTR	45445	Return address of variable found
B194	ARYGET	45460	Pointer: array routine
B1A5	N32768	45477	32768 in Floating Point format
B1AA	FACINX	45482	Convert Floating Point number
B1B2	INTIDX	45490	Input/convert a Floating Point subscript to positive integer
B1BF	AYINT	45503	FAC #1 to integer
B1D1	ISARY	45521	Find or create array
B245	BSERR	45637	Print bad subscript error mess.
B248	FCERR	45640	Print illegal quantity err. mess.
B34C	UMULT	45900	Compute array size
B37D	FRE	45949	Perform BASIC function FRE
B391	GIVAYF	45969	Convert 16-bit signed integer to Floating Point...
B39E	POS	45982	Perform BASIC command POS
B3A6	ERRDIR	45990	Test for direct mode
B3B3	DEF	46003	Perform BASIC command DEF
B3E1	GETFNM	46049	Check for function syntax
B3F4	FNDOER	46068	Evaluate BASIC command function
B465	STRD	46181	Perform BASIC command STR\$
B475	STRINI	46197	String administration

HEX	LABEL	DECIMAL	ACTION
B487	STRLIT	46215	Set up string
B4F4	GETSPA	46324	Make room for strings
B526	GARBAG	46374	Collect garbage
B5BD	DVARS	46525	Check for eligible strings
B606		46598	Collect string
B63D	CAT	46653	Concatenate two strings
B67A	MOVINS	46714	Build string to memory
B6A3	FRESTR	46755	Discard unwanted string
B6BD	FRETMS	46811	Clean the discriptor stack
B6EC	CHRD	46828	Perform BASIC command CHR\$
B700	LEFTD	46848	Perform BASIC command LEFT\$
B72C	RIGHTD	46892	Perform BASIC command RIGHT\$
B737	MIDD	46903	Perform BASIC command MID\$
B761	PREAM	46945	Pull string parameters from the stack
B77C	LEN	46972	Perform BASIC command LEN
B782	LEN1	46978	Exit string mode
B78B	ASC	46987	Perform BASIC command ASC
B79B	GETBYTC	47003	Input byte parameters
B79E		47006	Place 1 byte from BASIC text in X-register
B7AD	VAL	47021	Perform BASIC command VAL
B7B5	STRVAL	47029	ASCII to FAC #1
B7B7		47031	Converts FAC #1 to a two byte unsigned integer at \$14
B7EB	GETNUM	47083	Get POKE/WAIT parameters
B7F7	GETADR	47095	FAC #1 to an interger
B80D	PEEK	47117	Perform BASIC command PEEK
B824	POKE	47140	Perform BASIC command POKE
B82D	FUWAIT	47149	Perform BASIC command WAIT
B849	FADDH	47177	Add 0.5 to FAC #1.
B850	FSUB	47184	Number in memory minus FAC #1
B853	FSUBT	47187	Perform subtraction
B867	FADD	47207	Number in memory plus FAC #1
B86A	FADDT	47210	Perform addition
B8A7	FADD4	47271	Make result negative
B8FE	NORMAL	47358	Normalize FAC #1
B947	NEGFAC	47431	Complement FAC #1
B97E	OVERR	47486	Print overflow error message

HEX	LABEL	DECIMAL	ACTION
B983	MULSHF	47491	Single byte multiply
B9BC	FONE	47548	Floating point constants
B9C1	LOGCN2	47553	FAC table for log functions
B9EA	LOG	47594	Perform BASIC command LOG
BA28	FMULT	47656	Multiply FAC #1 by memory.
BA30	FMULT	47664	Multiply FAC #2 by FAC #1.
BA33	FMULT	47667	Multiply FAC # 1 by FAC #2
BA59	MLTPLY	47705	Multiply a bit subroutine
BA8C	CONUPK	47756	Memory to FAC #2
BAB7	MULDIV	47799	Adjust FAC #1 & #2
BAD4	MLDVEX	47828	Handle underflow/overflow
BAE2	MUL10	47842	Multiply FAC #1 by 10.
BAF9	TENC	47865	Constant 10 in five-byte floating format
BAFE	DIV10	47870	Divide FAC #1 by 10.
BB07		47879	Divide FAC #2 by 10
BB0F	FDIV	47887	Divide a # in memory by FAC #1
BB12	FDIVT	47890	Divide FAC #2 by FAC #1.
BBA2	MOVFM	48034	Move a floating point number from memory to FAC #1
BBC7	MOV2F	48071	Move a floating point number from FAC #1 to memory.
BBD7		48087	FAC #1...to memory.
BBFC	MOVFA	48124	Move a floating point number from FAC #2 to FAC #1
BC0C	MOVFA	48140	Round and move a floating point number from FAC #1 to FAC #2
BC0F	MOVFA	48143	FAC #1 to FAC #2, no rounding
BC1B	ROUND	48155	Round off FAC #1.
BC2B	SIGN	48171	Find sign of FAC #1
BC39	SGN	48185	Perform BASIC command SGN
BC58	ABS	48216	Perform BASIC command ABS
BC5B	FCOMP	48219	Compare FAC #1, to memory.
BC9B	QINT	48283	Convert FAC #1...to integer
BCCC	INT	48332	Perform BASIC command INT
BCF3	FIN	48371	Convert ASCII string to floating point # in FAC #1.
BD7E	FINLOG	48510	Add an ASCII digit to FAC #1

HEX	LABEL	DECIMAL	ACTION
BDB3	N0999	48563	Constants table for ASCII string conversion.
BDC0	INPRT	48576	Print 'IN' followed by a line #
BDCD	LINPRT	48589	Output # in ASCII decimal digits
BDD7	FACOUT	48599	Print FAC #1, as an ASCII string
BDDD	FOUT	48605	Convert contents of FAC #1 to an ASCII string
BF11	FHALF	48913	More constants for rounding/SQR
BF1C	FOUTB	48924	Powers of - 10 constants table
BF3A	FDCEND	48954	Table of constants, TI\$ to ASCII
BF52	48978-49008		UNUSED AREA
BF71	SQR	49009	Perform BASIC command SQR
BF75	FPWRT	49013	Replace FAC #1 with FAC #2
IF7B	FPWRT	49019	Perform exponentiation
BFB4	NEGOP	49076	Perform negative statement NOT
BFBF	EXPCON	49087	Constant table - EXP/LOG routines
BFED	EXP	49133	Perform BASIC command EXP
C000-CFFF	49152-53247		4K FREE RAM, not for BASIC programs

D000-DFFF 53248-57343

4K MEMORY BLOCK...normal functions are: two Complex Interface Adapter (CIA) Input/Output chips, VIC-II video controller chip, Sound Interface Device (SID), and color RAM.

D000-D02E 53248-53294

MOS 6566 VIDEO INTERFACE CONTROLLER (VICII) generates video signals for the display of sprites, text, color, multicolor, bitmap graphics, etc.

D000	SPOX	53248	Sprite 0 horizontal (X) position
D001	SPOY	53249	Sprite 0 vertical (Y) position
D002	SP1X	53250	Sprite 1 horizontal (X) position
D003	SP1Y	53251	Sprite 1 vertical (Y) position
D004	SP2X	53252	Sprite 2 horizontal (X) position
D005	SP2Y	53253	Sprite 2 vertical (Y) position
D006	SP3X	53254	Sprite 3 horizontal (X) position
D007	SP3Y	53255	Sprite 3 vertical (Y) position

HEX	LABEL	DECIMAL	ACTION
D008	SP4X	53256	Sprite 4 horizontal (X) position
D009	SP4Y	53257	Sprite 4 vertical (Y) position
D00A	SP5X	53258	Sprite 5 horizontal (X) position
D00B	SP5Y	53259	Sprite 5 vertical (Y) position
D00C	SP6X	53260	Sprite 6 horizontal (X) position
D00D	SP6Y	53261	Sprite 6 vertical (Y) position
D00E	SP7X	53262	Sprite 7 horizontal (X) position
D00F	SP7Y	53263	Sprite 7 vertical (Y) position
D010	MSIGX	53264	Sprites 0-7 most significant bits horizontal positions
D011	SCROLY	53265	Cont. reg. vertical fine scroll
D012	RASTER	53266	Read/write current raster scan line, compare of raster IRQ
D020	EXTCOL	53280	Border color
D021	BGCOL0	53281	Background color 0: screen color
D022	BGCOL1	53282	Background col.1: char.multicol.1
D023	BGCOL2	53283	Background col.2: char.multicol.1
D024	BGCOL3	53284	Background color 3
D025	SPMCO	53285	Sprite multicolor register 0
D026	SPMC1	53286	Sprite multicolor register 1
D027	SP0COL	53287	Sprite 0 color register
D028	SP1COL	53288	Sprite 1 color register
D029	SP2COL	53289	Sprite 2 color register
D02A	SP3COL	53290	Sprite 3 color register
D02B	SP4COL	53291	Sprite 4 color register
D02C	SP5COL	53292	Sprite 5 color register
D02D	SP6COL	53293	Sprite 6 color register
D02E	SP7COL	53294	Sprite 7 color register
D02F-D03F		53295-53311	Not connected
D040-D3FF		53312-54271	Vic-II register images

**** MOS 6581 SOUND INTERFACE DEVICE (SID) ****

D400	FRELO1	54272	Voice 1:frequency cont.(low byte)
D401	FREHI1	54273	Voice 1:frequency cont.(hi. byte)
D402	PWLO1	54274	Voice 1: pulse waveform width
D404	VCREG1	54276	Voice 1: control register
D405	ATDCY1	54277	Envelope gen.1:attack/decay
D406	SUREL1	54278	Envelope gen.1:sustain/release

HEX	LABEL	DECIMAL	ACTION
D407	FRELO2	54279	Voice 2: frequency - low byte
D408	FREHI2	54280	Voice 2: frequency - high byte
D409	PWLO2	54281	Voice 2: pulse waveform-low byte
D40A	PWHI2	54282	Voice 2: pulse waveform-high byte
D40B	VCREG2	54283	Voice 2: control register
D40C	ATDCY2	54284	Envelope generator 2:attack/decay
D40D	SUREL2	54285	Envelope gen. 2:sustain/release
D40E	FRELO3	54286	Voice 3: frequency - low byte
D40F	FREHI3	54287	Voice 3: frequency - high byte
D410	PWLO3	54288	Voice 3: pulse waveform-low byte
D411	PWHI3	54289	Voice 3: pulse waveform-high byte
D412	VCREG3	54290	Voice 3: control register
D413	ATDCY3	54291	Envelope gen. 3: attack/decay cycle control
D414	SUREL3	54292	Envelope gen. 3: sustain/release cycle control
D415	CUTLO	54293	Filter cutoff freq.:low nibble
D416	CUTHI	54294	Filter cutoff frequency:high byte
D417	RESON	54295	Filter resonance/voice input con.
D418	SIGVOL	54296	Select filter mode and volume
D419	POTX	54297	Analog/digital converter:paddle 1
D41A	POTY	54298	Analog/digital converter:paddle 2
D41B	RANDOM	54299	Oscillator 3: random # generator
D41C	ENV3	54300	Envelope generator 3 output
D41D-D41F		54301-54303	Not connected
D420-D7FF		54304-55295	Sid images
D800-DBFF		55296-56319	Color RAM: fixer location

**** MOS 6526 COMPLEX INTERFACE ADAPTER (CIA) #1 ****

DC00	CIAPRA	56320	Data port A:joystick,keyboard,ect
DC01	CIAPRB	56321	Data port B:joystick,keyboard,ect
DC02	CIDDR A	56322	Data direction register:port A
DC03	CIDDR B	56323	Data direction register:port B
DC04	TIMALO	56324	Timer A: low byte
DC05	TIMAH I	56325	Timer A: high byte
DC06	TIMBLO	56326	Timer B: low byte
DC07	TIMBHI	56327	Timer B: high byte
DC08	TODTEN	56328	Time of day clock: 1/10 seconds

HEX	LABEL	DECIMAL	ACTION
DC09	TODSEC	56329	Time of day clock: seconds
DC0A	TODMIN	56330	Time of day clock: minutes
DC0B	TODHRS	56331	Time of day clock: hours & AM/PM
DC0C	CIASDR	56332	Serial data port
DC0D	CIAICR	56333	Interrupt control register
DC0E	CIACRA	56334	Control register A
DC0F	CIACRB	56335	Control register B
DC10-DCFF	56336-56575		CIA #1 register images

**** MOS 6526 COMPLEX INTERFACE ADAPTER (CIA) #2 ****

DD00	C12PRA	56576	Data port A (serial bus, RS-232)
DD01	C12PRB	56577	Data port B (user port, RS-232)
DD02	C2DDRA	56578	Data direction register: port A
DD03	C2DDRB	56579	Data direction register: port B
DD04	TI2ALO	56580	Timer A: low byte
DD05	TI2AHI	56581	Timer A: high byte
DD06	TI2BLO	56582	Timer B: low byte
DD07	TI2BHI	56583	Timer B: high byte
DD08	TO2TEN	56584	Time of day clock: 1/10 seconds
DD09	TO2SEC	56585	Time of day clock: seconds
DD0A	TO2MIN	56586	Time of day clock: minutes
DD0B	TO2HRS	56587	Time of day clock: hours & AM/PM
DD0C	CI2SDR	56588	Synchronous serial input output buffer
DD0D	CI2ICR	56589	CIA interrupt control register
DD0E	CI2CRA	56590	CIA control register A
DD0F	CI2CRB	56591	CIA control register B
DD10-DDFF	56592-56831		CIA #2 register images
DE00-DEFF	56832-57087		Reserved for future expansion
DF00-DFFF	57088-57343		Reserved for future expansion

Location 57344 to 65535 (E000 - FFFF) are used by the 8K operating system Kernal ROM. Kernal ROM is subject to revision (by the manufacturer), these routines may not stay at the same address. It is recommended that the user refer to the 'JUMP TABLE' at the end of ROM. By having the program 'jump' to a ROM addresse in the TABLE, it should run without change on future models.

HEX	LABEL	DECIMAL	ACTION
E000		57344	Continuation of EXP routine
E043	POLY1	57411	Evaluation of complex expressions
E059	POLY2	57433	Main series evaluation routine
E08D	RMULC	57485	Multiplicative constant for RND
E092	RADDC	57490	Additive constant for RND
E097	RND	57495	Perform the BASIC command RND
E0F9		57593	Call kernal input/output routines
E12A	SYS	57642	Perform the BASIC command SYS
E156	SAVE	57686	Perform the BASIC command SAVE
E165	VERIFY	57701	Perform the BASIC command VERIFY
E168	LOAD	57704	Perform the BASIC command LOAD
E1BE	OPEN	57790	Perform the BASIC command OPEN
E1C7	CLOSE	57799	Perform the BASIC command CLOSE
E1D4		57812	Set parameters, LOAD,VERIFY,SAVE
E200		57856	Skip comma and get integer in .X
E206		57862	Get current character and check for end of line
E20E		57870	Check for comma
E219		57881	Set parameters for OPEN and CLOSE
E264	COS	57956	Perform the BASIC command COS
E268	SIN	57960	Perform the BASIC command SIN
E26B		57963	FAC 1=SIN (FAC 1)
E2B4	TAN	58036	Perform the BASIC command TAN
E2B7		58039	FAC 1=TAN (FAC 1)
E2E0	PI2	58080	Constants for trig functions
E2E5	TWOPI	58085	5 byte floating point rep. of constant 2*PI
E2EA	FR4	58090	5 byte floating point rep. of constant 1/4
E2EF	SINCON	58095	Table of constants for eval. of SIN,COS,TAN
E30D		58125	FAC 1=ATN (FAC 1)
E30E	ATN	58126	Perform the BASIC command ATN
E33E	ATNCON	58174	Constants for ATN
E37B		58235	Warm start BASIC
E38B		58251	Error message handler
E394		58260	Cold start BASIC
E3A2	INITAT	58274	Character get for zero page
E3BA		58298	Initial RND seed value

HEX	LABEL	DECIMAL	ACTION
E3BF	INIT	58303	Initialize BASIC
E422		58402	Print BASIC start-up messages
E447		58439	Table of vectors to important BASIC routines
E453		58451	Copy BASIC vectors to RAM
E460	WORDS	58464	Power up messages
E4AD		58541	Program patch area
E4B7		58551	UNUSED 35 bytes free
E4DA		58586	Clear color RAM
E4E0		58592	Pause when file found on cassette
E4EC		58604	Baud rate table for european(PAL)
E000	IOBASE	58624	Get input output address
E505	SCREEN	58629	Set screen limits
E50A	PLOT	58634	Track cursor location
E518	CINT	58648	Initialize input output
E531		58673	Normalize screen
E544		58692	Clear screen
E566	NXTD	58726	Home the cursor
E56C	STUPT	58732	Set screen pointers to cur. line
E5A0	PANIC	58784	Set input output defaults
E5A9		58792	Set VIC chip defaults
E5B4	LP2	58804	Get input from keyboard
E5CA	KBINP	58826	Wait for carriage return
E632	KSINP	58930	Input from screen or keyboard
E684	QTSWC	59012	Quote mark test
E691	SCPUT	59025	Set up screen print
E6A8		59048	Return from outputting a char. to the screen
E6B6		59062	Advance the cursor
E6F7		59127	Retreat the cursor
E701		59137	Back into previous line
E716	SCNPNT	59158	Output to screen
E87C		59516	Cursor goto next line
E891		59537	Carriage return
E8A1	CKDECL	59553	Check line decrement
E8B3	CKINL	59571	Check line increment
E8CB	CKCOLR	59595	Set color code
E8A1	CKDECL	59601	Color code table
E8E2		59618	Code conversion

HEX	LABEL	DECIMAL	ACTION
E8EA	SCROL	59626	Scroll screen
E965		59749	Open space on screen
E9C8		59848	Move screen line
E9E0		59872	Synch color transfer
E9F0		59888	Set start of line
E9FF	CLRLN	59903	Clear screen line
EA13	DSPP	59923	Print to screen
EA1C		59932	Store on screen
EA24		59940	Synch color to character
EA31	KEY	59953	Interrupt (IRQ)
EA87	SCNKEY	60039	Check keyboard
EAE0		60128	Decode keystroke
EB48		60232	Set up keyboard decode table
EB59		60249	Set text mode
EB79	KEYCOD	60281	Keyboard vectors
EB81		60289	Keyboard 1 - unshifted
EB91		60305	Keyboard maps
EBC2		60354	Keyboard 2 - shifted
EC03		60419	Keyboard 3 - logo key shifted
EC44		60484	Graphics/text control
EC4F		60495	Set graphics mode
EC5E		60510	Check for special characters
EC78		60536	PET ASCII table for keyboard
ECB9		60601	Video chip default
ECE7		60647	Shift/run equivalent
ECF0		60656	Low byte table scr. line address
ED09	TALK	60681	Send 'TALK'
ED0C	LISTEN	60684	Send 'LISTEN'
ED11		60689	Send control character
ED40		60736	Send to serial bus
EDB0		60848	Time out on serial
EDB9	SECOND	60857	Send listen SA
EDBE		60862	Clear ATN
EDC7	TKSA	60871	Send SA to device after talk
EDDD		60893	Send byte to I/O device
EDEF	UNTLK	60911	Send 'UNTALK' to device
EDFE	UNLSN	60926	Send 'UNLISTEN' to device
EE13	ACPTR	60947	Receive a byte from device
EE85		61061	Set serial clock line low (on)

HEX	LABEL	DECIMAL	ACTION
EE8E		61070	Set serial clock line high (off)
EE97		61079	Set serial bus output line low
EEA0		61088	Set serial bus output high
EEA9		61097	Get serial bus input and clock
EEB3		61107	Delay of 1 MS.
EEBB		61115	Send next RS-232 bit NMI
EF06		61190	Send new RS-232 byte
EF2E		61230	Handle RS-232 errors
EF3B		61243	Disable timer
EF4A		61258	Compute word length/RS-232 char.
EF59		61273	Receive next RS-232 bit NMI
EF7E		61310	Set up to receive new byte/RS-232
EF90		61328	Test if above byte received
EF97		61335	Put byte in RS-232 receive buffer
EFC5		61381	Receive parity error
EFCA		61386	Receive overrun error
EFCD		61389	Receive break error
EFD0		61392	Receive frame error
EFE1		61409	CHKOUT for the RS-232 device
F00D		61453	No DSR error
F017		61463	Send to RS-232 buffer
F04D		61517	Input from RS-232 buffer
F086		61574	Get from RS-232 buffer
F0A4		61604	Check serial bus, see if idle
F0BD		61629	Kernal control messages
F12B		61739	Print kernal error messages
F13E	GETIN	61758	Get 1 byte from input device
F14E		61774	..from RS-232
F157	CHRIN	61783	Input byte from current device
F199		61849	Get byte from tape/serial/RS-232
F1CA	CHROUT	61898	Output a byte
F1CA		61909	Output
F1DD		61917	To tape.....
F20E	CHKIN	61966	Set input device
F250	CHKOUT	62032	Set output device
F291	CLOSE	62097	Close a logical input/output file
F30F		62223	Find file in logical file table
F31F		62239	Set file values
F32F	CLALL	62255	Abort all files

HEX	LABEL	DECIMAL	ACTION
F333	CLRCHN	62259	Restore default input/output
F34A	OPEN	62282	Do file opening
F3D5		62421	Send SA
F409		62473	Open RS-232
F49E	LOAD	62622	Load program from device
F5A5		62885	Print 'SEARCHING' message
F5B8		62904	Print file name
F5D2		62930	Print 'LOADING or VERIFY'
F5DD	SAVE	62941	Save PROGRAM to device
F68F		63119	Print 'SAVING' and filename
F69B	UDTIM	63131	Update software clock
F6BC		63164	Log PIA key reading
F6DD	RDTIM	63197	Read time from software clock
F6E4	SETTIM	63204	Set the software clock
F6ED	STOP	63213	Test STOP key
F6FB		63227	File error messages
F72C		63276	Find any tape header
F76A		63338	Write tape file header
F7D0		63440	Get tape buffer address
F7D7		63447	Set tape buffer start/end pointer
F7EA		63466	Find specific header
F80D		63501	Bump tape pointers
F817		63511	Print message 'PRESS PLAY'
F82E		63534	Check cassette status
F838		63544	Print message 'PRESS RECORD'
F841		63553	Initiate tape read
F864		63588	Initiate tape write
F875		63605	Common tape read/write
F8D0		63696	Check STOP during tape functions
F8E2		63714	Adjust tape bit timing
F92C		63788	Read tape data
FA60		64096	Store next character from tape
FB8E		64398	Reset pointer
FB97		64407	New tape character setup
FBA6		64422	Toggle tape data output line
FBC8		64456	Write data to cassette
FBCD		64461	Tape write (IRQ)
FC57		64599	Header write (IRQ)
FC6A		64618	Write data to cassette

HEX	LABEL	DECIMAL	ACTION
FC93		64659	Restore default vectors
FCB8		64696	Terminate cassette input/output
FCCA		64714	Stop tape motor
FCD1		64721	Check read/write pointers
FCDB		64731	Bump read/write pointers
FCE2		64738	Powerup reset routine
FD02		64770	Check for autostart cartridge
FD13		64787	Set kernal
FD15	RESTOR	64789	Restore RAM vectors
FD1A	VECTOR	64794	Set RAM vector table
FD30		64816	Table:RAM vector/Default I/O rout
FD30	RAMTAS	64848	Perform RAM test
FD52		64850	Initialize system constants
FD9B		64923	Table of IRQ vectors
FDA3	IOINIT	64931	Initialize input/output registers
FDDD		64989	Enable timer
FDF9	SETNAM	65017	Set data name parameters
FE00	SETLFS	65024	Save file details
FE07	READST	65031	Get status word
FE18	SETMSG	65048	Set message control flag
FE21	SETTMO	65057	Set time out flag for IEEE bus
FE25	MEMTOP	65061	Read/set top of memory
FE34	MEMBOT	65076	Read/set bottom of memory
FE43		65091	Non-maskable interrupt entry
FE66		65126	Reset/STOP warm start
FE72		65138	Handler for RS-232 NMI
FEBC		65214	Restore and exit
FEC2		65218	RS-232 timing table
FED6		65238	RS-232 receive next bit NMI
FF07		65287	RS-232 send
FF1A		65306	Load timer/ baud rate lookup table
FF43		65347	Fake IRQ
FF48		65352	Main IRQ entry point
FF5B	CINT	65371	Initialize screen editor/VIC chip
FF6E		65390	End of routine to set timer for sixtieth of a second IRQ
FF80		65408	Kernal version identifier byte

C-64 KERNAL ROUTINES

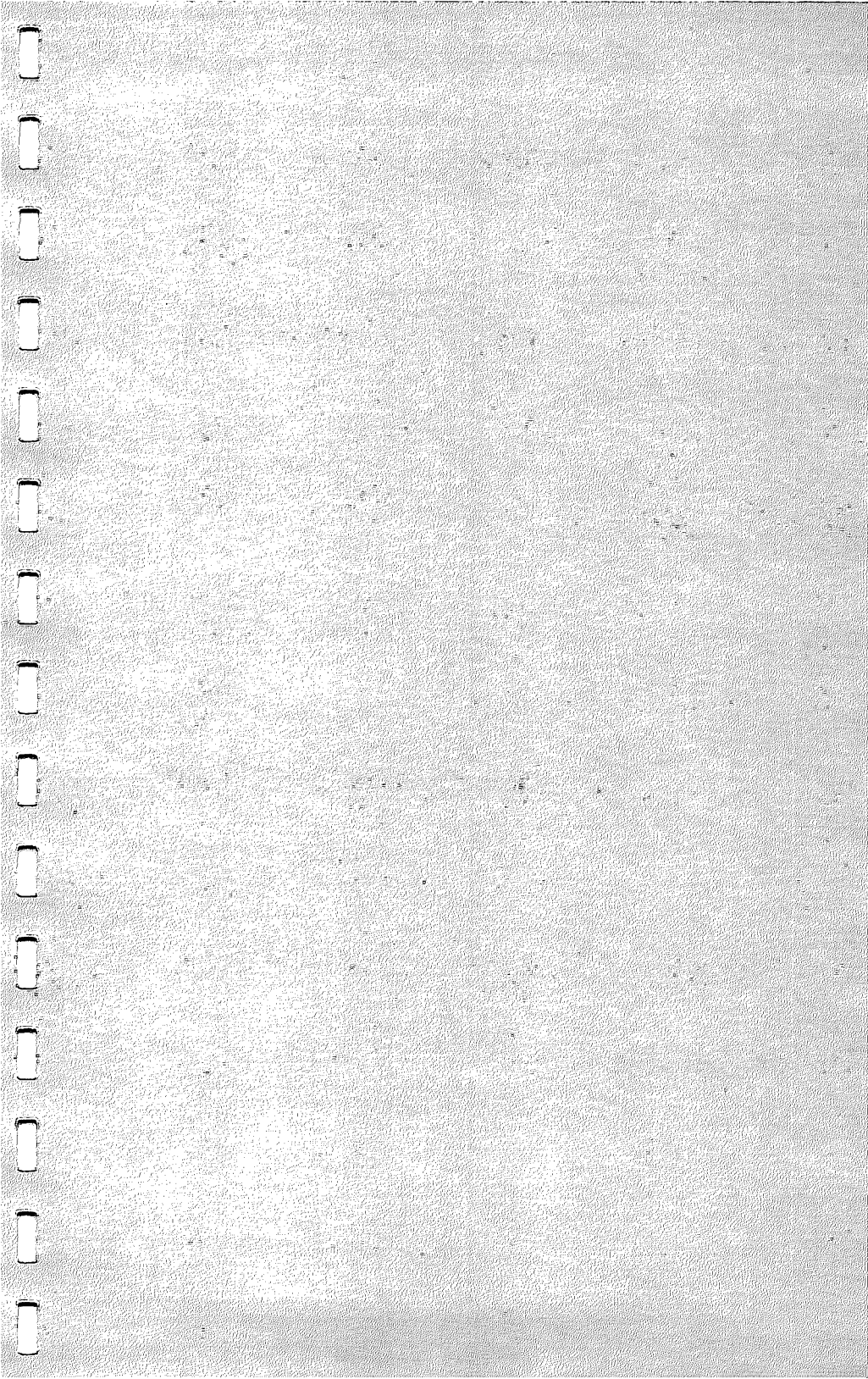
The Kernal ROM, is a table of pre-defined subroutines for C-64 mode. These addresses act as pointers to all the components needed to complete each routine. Commodore used this same set in the Vic-20. This enables programs using kernal ROM subroutines, to be converted from one machine to the other. Some of the memory chips in the C-64 have been up-graded or changed from time to time. The table has remained the same.

HEX	DECIMAL	LABEL	ACTION
\$FF81	65409	CINT	Initialize screen editor
\$FF84	65412	IOINIT	Initialize input/output
\$FF87	65415	RAMTAS	Initialize RAM/tape buffer, set screen
\$FF8A	65418	RESTOR	Restore default input/output vectors
\$FF8D	65421	VECTOR	Read/set vectored input/output
\$FF90	65424	SETMSG	Control KERNAL messages
\$FF93	65427	SECOND	Send a secondary address after a listen
\$FF96	65430	TKSA	Send a secondary address after a talk
\$FF99	65433	MEMTOP	Read and set top of memory
\$FF9C	65436	MEMBOT	Read and set bottom of memory
\$FF9F	65439	SCNKEY	Scan the keyboard
\$FFA2	65442	SETTMO	Set time out on the serial bus
\$FFA5	65445	ACPTR	Input a byte from the serial port
\$FFA8	65448	CIOUT	Output a byte to the serial port
\$FFAB	65451	UNTLK	Command the serial bus to untalk
\$FFAE	65454	UNLSN	Command the serial bus to unlisten
\$FFB1	65457	LISTEN	Command the serial bus to listen
\$FFB4	65460	TALK	Command the serial bus to talk
\$FFB7	65463	READST	Read the status of input/output

HEX	DECIMAL	LABEL	ACTION
\$FFBA	65466	SETLFS	Set logical, first and second address
\$FFBD	65469	SETNAM	Set file name
\$FFC0	65472	OPEN	Open a logical file
\$FFC3	65475	CLOSE	Close a logical file
\$FFC6	65478	CHKIN	Open a channel for input
\$FFC9	65481	CHKOUT	Open a channel for output
\$FFCC	65484	CLRCHN	Close input/output channels
\$FFCF	65487	CHRIN	Input a character from channel
\$FFD2	65490	CHROUT	Output a character to channel
\$FFD5	65493	LOAD	Load RAM from a device
\$FFD8	65496	SAVE	Save RAM to a device
\$FFDB	65499	SETTIM	Set the time clock
\$FFDE	65502	RDTIM	Read the time clock
\$FFE1	65505	STOP	Scan the STOP key
\$FFE4	65508	GETIN	Get a character from a device
\$FFE7	65511	CLALL	Close all channels and files
\$FFEA	65514	UDTIM	Increment the time clock
\$FFED	65517	SCREEN	Return X,Y organization of the screen
\$FFF0	65520	PLOT	Read and set the X,Y cursor position
\$FFF3	65523	IOBASE	Return the base address of input/output device

6510 HARDWARE VECTORS

\$FFFA	65530	TKSA	NMI hardware vectors
\$FFFC	65532		System reset
\$FFFE	65534		NMI interrupt request & break hardware vectors





BASIC 7.0 Dictionary	66
BASIC 7.0 Abbreviations	79
BASIC 7.0 Error Messages	84
Control Codes	86
Escape Codes	89
Function Keys	90
User Defined F-Keys	90
Color	91
Color 40 Column Mode	92
Color 80 Column Mode	92
Graphic Summary	93
SPRite DEFinition mode	96
Sound	98
Disk Command Summary	102
POKES and PEEKS	103
M/L Monitor	104
Memory Map	107
Bank Configuration	107
Bank 0	107
Bank 1	114
Bank 14	114
Bank 15	114
Kernal Routines	116

DICTIONARY of BASIC 7.0

The following pages are a mini-dictionary of 7.0 keywords. On the first line of each entry is the keyword and an example of the proper syntax. The second line contains a brief definition, of the function the word performs.

ABS 20 z=ABS(X)

Returns the absolute value of a number.

APPEND 800 APPEND#8,"first file"

Opens the file named/points to the end of the file.

AND if k=3 AND y=0 then 500

Checks the truth of multiple operands.

ASC 100 print ASC(a\$)

Returns the ASCII value of a character.

ATN 230 print ATN(N)

Returns the artangent of a number.

AUTO AUTO [line #]

Start up command for AUTO line numbering.

BACKUP BACKUP d0 to d1

Dual drive only, copy the files of disk to another.

BANK BANK X (0-15)

To choose one of 16 banks (0-15).

BEGIN 100 if a then BEGIN:print "ok"

Enables embedding lines of code in IF THEN statement.

BEND 300 BEND

Ends the BEGIN routine.

BLOAD BLOAD "menu",b0,94080

Loads a binary file to a specified memory location.

BOOT	BOOT
Loads and runs a machine language program from disk.	
BOX	235 BOX1,11,11,80,80
Used to draw a BOX at a specific place on the screen	
BSAVE	BSAVE "sprite 1",b0,p3584 to p4096
Saves a binary file to a specified memory locatation	
BUMP	230 BUMP[1-2]
Returns sprite collision information.	
CATALOG	CATALOG
Displays the directory of a disk.	
CHAR	50 CHAR0,20,20,"welcome",0
Display characters at a specific place on the screen	
CHR\$	200 print CHR\$(48)
Converts Commodore ASCII to character equivalent.	
CIRCLE	230 CIRCLE1,150,100,60,10
Draws CIRCLES, arcs, etc. at a specified position.	
CLOSE	100 CLOSE 15
To close a logical file or device.	
CLR	10 CLR
To clear all program variables.	
CMD	200 CMD1
Directs the output to another device.	
COLLECT	COLLECT
Gathers disk space allocated to files left open.	
COLLISION	550 COLLISION[1-3],900
Identifies sprite collisions, specifies action.	

COLOR 10 COLOR[0-6],[1-16]
COLOR source number, COLOR number
Assigns colors to one of the 7 screen color areas.

CONCAT CONCAT "menu2" to "menu1"
To merge two existing disk sequential files.

CONT CONT
Restarts a program after a STOP or END command.

COPY COPY "oldfile" to "newfile"
To COPY files from one disk to another.

COS 140 print COS(N)
Returns the cosine for the angle of X radins.

DATA 40 DATA 48,0,255
Used to store information within a program.

DEC DEC("DD00")
Returns the decimal value of a hex number string.

DCLEAR 800 DCLEAR D0
To clear all channels on the disk drive.

DCLOSE 800 DCLOSE
To close a disk file or all files of a disk.

DEF FN 20 DEF FN z(3) = x+y
A user defined function for later use in a program.

DELETE DELETE 200-500
To erases lines of BASIC code.

DIM 10 DIM a\$(100)
To declare the number of elements in an array.

DIRECTORY DIRECTORY
To view the directory of a disk.

DLOAD DLOAD "menu"
To LOAD a BASIC program from disk.

DO 300 DO until z=0
To define and control a program loop.

DOPEN 850 DOPEN#1,"data file"
Opens disk file(SEQ,REL,rand. access) for read/write

DRAW 250 DRAW[0-3],20,20 to 100,100
A graphics command, used to draw to the screen.

DSAVE DSAVE"menu"
To SAVE a BASIC program to a disk.

DVERIFY DVERIFY "menu"
Compares a program in memory to one on a disk.

ELSE 875 if x>0 then 450:ELSE print"error"
ELSE clause is only executed if the expression false

END 900 END
Ends a program run and displays the READY prompt.

ENVELOPE 235 ENVELOPE[0-9],[0-15],[0-15],
[0-15],[0-15],[0-4],[0-4095]
Defines a musical instrument, sets: Envelope #,
Attack rate, Decay rate, Sustain, Release, WaveForm,
Pulse Width.

ER 450 print ER
Holds the definition of the most recent error.

ERR\$ 460 print ERR\$(ER)
Reads an error message from error in BASIC program.

EXIT 500 if x=10 then EXIT
Used to exit a do loop.

EXP 100 print EXP(3)
Returns the value 2.71 raised to X power.

FAST 200 FAST
Puts the computer in 2 MHz mode of operation.

FETCH 300 FETCH30,3000,2,4000
Gets data from a RAM expansion module.

FILTER 300 FILTER200,1,0,0,14
A musical command to program SID filters.

FOR 100 FOR x=1 to 20
Used with NEXT command to repeat a section of code.

FN 100 print FN b(x)
The value of a function created by DEF FN statement.

FRE FRE [0-1]
Returns the number of unused bytes in memory.

GET 100 GET a\$: if a\$="" then 100
To GET one character from the keyboard.

GETKEY 200 GETKEY a\$
Receives input from keyboard, waits for user input.

GET # 100 GET#5,a\$
To GET one character from an external device.

GOSUB 100 GOSUB 500
Tells the computer to branch to a subroutine.

GO64 G064
Sends the computer from 128 mode to 64 mode.

GOTO 100 GOTO 450
Tells the computer to GOTO a specific line number.

GRAPHIC 10 GRAPHIC[0-1],1

Places the computer in one of the six graphic modes.

GSHAPE 455 GSHAPE a\$,[0-319],[0-199],[0-4]

Used to save or load rectangular areas of high resolution or multicolor screens.

HEADER HEADER "disk #01",I10

This is the command used to format a disk for use.

HELP HELP

Highlights a line where an error has occurred.

HEX\$ HEX\$(45000)

Converts a decimal number to hexadecimal.

IF THEN ELSE 35 if x=0 then500:ELSE end

Evaluates BASIC expression checks for true or false.

INPUT 340 INPUT n\$

Prints a question mark and waits for user input.

INPUT# 100 INPUT#5,a\$

To retrieve data stored in a file.

INSTR print INSTR("1234567","45"),[0-255]

Searchs for location of a string, displays the value of the starting position.

INT 200 print INT(a)

Returns the integer value of a number.

JOY 455 if JOY([1-2])=135 then 885

Displays position of joystick, status of firebutton.

KEY KEY[1-8],"print"

To define or list the function keys.

LEFT\$ 200 a\$=LEFT\$(x\$,1,3)
The left most character of a string.

LEN 140 if LEN(a\$)=4 then 450
Returns the number of characters in the string.

LET 100 LET a=20
To assign a value to an array.

LIST LIST
To look at a program in memory, screen or print same

LOAD LOAD "menu",8
Reads the contents of file (disk/tape) into memory.

LOCATE 230 LOCATE180,100
For placement of a pixel on a high resolution screen

LOG 100 x=LOG (45/7)
Returns the natural logarithm of an argument.

LOOP 200 LOOP
Tells the computer to loop back to the DO command.

MID\$ 100 a\$=MID\$(x\$,1,5)
A sub-string of a larger string.

MONITOR MONITOR
To enter the machine language monitor mode.

MOVSPR 255 MOVSPR[0-8],150,160
To move or position a sprite on the screen.

NEW NEW
Clears BASIC memory and all memory.

NEXT 300 NEXT x1
Used with FOR to create an end of a FOR NEXT LOOP.

ON . . GOTO 600 ON a GOTO 500,800,700
GOTO the line number shown after the GOTO command.

ON . . GOSUB 600 ON a GOSUB 500,700,800
Goto the line number shown after the GOSUB command.

OPEN 700 OPEN 15,8,15
OPENS a channel for input or output to a device.

PAINT 200 PAINT[0-3],20,20,[0-1]
A graphics command to fill an area with color.

PEEK 300 if PEEK(4000)=162 then 400
Returns the value of a memory location.

PEN 765 if PEN([0-4])=x then 900
To obtain the coordinates of a light pen.

PI 1100 print PI
Returns the value of PI (3.14159265).

PLAY 10 PLAYV[1-3],O[0-6],T[0-9],
U[0-15],x[0-1],[a,etc]
Musical command, allows selection of voice, octave,
envelope, volume, and notes.

POKE 100 POKE 53280,1
To change the contents of a memory location.

POS 230 print POS(0)
Returns the current cursor position.

POT 100 print POT([1-4])
Tells direction of game paddle/if firebutton pressed

PRINT 100 PRINT "hello"
To output data to the screen.

PRINT# 150 PRINT#4,"hello"
To output data to a file or device.

PRINT USING 150 PRINT USING "####";a\$
Defines format of material for screen/printer/device

PUDEF 160 PUDEF "*****"
To redefine blanks, ', 's, '. 's, '\$'s in PRINT USING.

RCLR 555 print RCLR([0-6])
Returns the color of the color source.

RDOT 775 print RDOT([0-2])
Locates a pixel on a high resolution screen/colors.

READ 100 READ a\$
To fill variables from a DATA statement.

RECORD 100 RECORD#3,100,1
To read or write data to a relative file.

REM 10 REM date: 02/04/86
To be used for user remarks or re-minders.

RENAME RENAME "old filename" to "newfilename"
Command used to change the name of a file on disk.

RENUMBER RENUMBER 10,10,1
A command to renumber the lines of a BASIC program.

RESTORE 300 RESTORE
To reset the pointers to next data.

RESUME 100 RESUME [line#/next]
Where computer should resume after an error trapped.

RETURN 800 RETURN
To exit a subroutine called by a GOSUB

RGR 235 print RGR([0-5])
Informs computer which graphic mode currently in use

RIGHT\$ 450 print RIGHT\$("hometown",5)
The right most character of a string.

RND 335 print RND(1)
Generates a random number sequence.

RSPCOLOR 445 print RSPCOLOR([1-2])
Identifies the color values of a multicolor sprite.

RSPPOS 555 print RSPPOS[0-15],[0-1],2
Identifies the speed and position of a sprite.

RSPRITE 775 print RSPRITE[0-15],[0-5]
Identifies the six characteristics of a sprite as defined in the SPRITE command.

RUN RUN
To start the program currently in memory.

RWINDOW 885 print RWINDOW([0-2])
Defines the size of a screen window.

SAVE SAVE "menu",[1-8]
To store a file to disk or tape.

SCALE 235 SCALE[0-1],[mult x=0-159],[y=0-199]
[bit x=0-319],[y=0-199]
To change the scale of bit-mapped images in multi-color/high resolution modes.

SCNCLR 350 SCNCLR[0-5]
Clears the screen when in graphics modes.

SCRATCH SCRATCH "menu",8
Erases a file from a disk and disk directory.

SGN 300 print SGN(x)
Returns the sign (pos., neg., or zero) of X.

SIN 400 print SIN(x)
Returns sine of (numeric) argument in radians.

SLEEP 250 SLEEP[0-65535]
Delays program execution for specified seconds.

SLOW 450 SLOW
Returns computer to 1 MHz speed from 2 MHz (FAST).

SOUND 670 SOUND[1-3],[0-65535],
[0-32767],[0-2],[0-65535],[0-32767],[0-3],[0-4095]
The command to create sound and music.

SPC (100 print SPC(0)"hello"
To skip X spaces forward on the same line.

SPRCOLOR 110 SPRCOLOR[1-16],[1-16]
Sets multicolor 1 and or 2 for sprites.

SPRDEF SPRDEF
To enter the sprite editor in C-128 mode.

SPRITE 100 SPRITE[1-8],[0-1],
[1-16],[0-1],[0-1],[0-1],[0-1]
Turns sprites controls off or on.

SPRSAV 235 SPRSAV1,a\$
Saves sprite data from a text string variable to a
storage area in memory.

SQR 350 SQR(d)
Returns the square root of a positive number.

SSHAPE 555 SSHAPEd\$,20,20
Used to save or load rectangular areas of high
resolution/ multicolor screens

STASH 650 STASH[1-65536],
[0-65535],[0-3],[0-65535]
Moves contents of memory into expansion RAM.

Status 900 print ST
Shows the STATUS of last input/output performed.

STEP 450 for x=1 to 250 STEP 2
Defines the value of increment for a loop.

STOP 545 STOP
To end the execution of a program.

STR\$ 555 x\$=STR\$(x)
Converts a number to a string.

SWAP 650 SWAP[1-65536],
[0-65535],[0-3],[0-65535]
Trades contents of internal RAM with contents of
external RAM expander.

SYS 675 SYS40965,{x},{y},{s}
To execute machine language program or subroutine.

TAB(20 print TAB(20) "welcome"
To move X number of spaces forward on the same line.

TAN 300 print TAN(x)
Returns the tangent value of (numeric) expression.

TEMPO 875 TEMPO[0-255]
The musical command to define speed of play.

TI 900 print TI
Current numeric value of the clock in 1/60 seconds.

TI\$ 990 print TI\$
Reads TI as a 24 hour clock.

TO 555 for x=1 TO 500:next
Used as part of the FOR statement.

TRAP 200 TRAP[line#]
For error detection within a running BASIC program.

TROFF 500 TROFF
Turns the trace mode off.

TRON 800 TRON
Turns on Trace mode for debugging BASIC programs.

UNTIL 500 do UNTIL x=30
Program continues until the condition is true.

USR 555 x=USR(z+
To transfer to a machine language program.

VAL 655 n=VAL(n\$)
To convert a string to a numeric value.

VERIFY VERIFY "menu",8
To compare a saved file against one in memory.

VOL 20 VOL[0-15]
A musical command to define level of sound.

WAIT 995 WAIT1,35,35
To pause until data matches.

WHILE 500 do WHILE x<200:loop
DO LOOP continues as long as the condition is TRUE.

WIDTH 650 WIDTH[1-2]
Determines width of pixel cursor for drawing lines.

WINDOW 20 WINDOW[0-39],[0-24],[0-39],[0-24]
Used to set up a screen window in 40/80 column mode.

XOR 650 print XOR(x1,x1)
Provides exclusive OR of argument values x1 and x2.

BASIC 7.0 ABBREVIATIONS

Most BASIC commands, statements, and functions can be abbreviated to a shorter form. Listed in column one, are the keywords. Columns two and three are the equivalent in hexadecimal and decimal. Column four is the abbreviated version.

To enter an abbreviation: type the letter or letters shown before the word shift. Next hold down the shift key and type the remaining letters.

KEYWORD	HEX	DECIMAL	SHORT-HAND
ABS	\$B6	00182	A shift B
APPEND	\$FE \$0E	00254 00014	A shift P
ASC	\$C6	00198	A shift S
ATN	\$C1	00193	A shift T
AUTO	\$DC	00220	A shift U
BACKUP	\$F6	00246	BA shift C
BANK	\$FE \$02	00254 00002	B shift A
BEGIN	\$FE \$18	00254 00024	B shift E
BEND	\$FE \$19	00254 00025	BE shift N
BLOAD	\$FE \$11	00254 00017	B shift L
BOOT	\$FE \$1B	00254 00027	B shift O
BOX	\$E1	00225	N/A
BSAVE	\$FE \$10	00254 00016	B shift S
BUMP	\$CE \$03	00206 00003	B shift U
CATALOG	\$FE \$0C	00254 00012	C shift A
CHAR	\$E0	00224	CH shift A
CHR\$	\$07	00199	C shift H
CIRCLE	\$E2	00226	C shift I
CLOSE	\$A0	00160	CL shift O
CLR	\$9C	00156	C shift L
CMD	\$9D	00157	C shift M
COLLECT	\$F3	00243	COLL shift E
COLLISION	\$FE \$17	00254 00023	COL shift L
COLOR	\$E7	00231	COL shift O
CONCAT	\$FE \$13	00254 00019	C shift O
CONT	\$9A	00154	N/A
COPY	\$F4	00244	CO shift P

KEYWORD	HEX	DECIMAL	SHORT-HAND
COS	\$BE	00190	N/A
DATA	\$83	00131	D shift A
DEC	\$D1	00209	N/A
DCLEAR	\$FE \$15	00254 00021	DCL shift E
DCLOSE	\$FE \$0F	00254 00015	D shift C
DEF FN	\$96 \$A5	00150 00165	N/A
DELETE	\$F7	00247	DE shift L
DIM	\$86	00134	D shift I
DIRECTORY	\$EE	00238	DI shift R
DLOAD	\$F0	00240	D shift L
DO	\$EB	00235	N/A
DOPEN	\$FE \$0D	00254 00013	D shift O
DRAW	\$E5	00229	D shift R
DSAVE	\$EF	00239	D shift S
DVERIFY	\$FE \$14	00254 00020	D shift V
ELSE	\$D5	00213	N/A
END	\$80	00128	N/A
ENVELOPE	\$FE \$0A	00254 00010	E shift N
ER	\$45 \$52	00069 00082	N/A
ERR\$	\$D3	00211	E shift R
EXIT	\$ED	00237	EX shift I
EXP	\$BD	00189	E shift X
FAST	\$FE \$25	00254 00037	N/A
FETCH	\$FE \$21	00254 00033	F shift E
FILTER	\$FE \$03	00254 00003	F shift I
FOR	\$81	00129	F shift O
FRE	\$B8	00184	F shift R
FNXX	\$A5	00165	N/A
GET	\$A1	00161	G shift E
GETKEY	\$A1 \$F9	00161 00241	GETK shift E
GET #	\$A1 \$23	00161 00035	N/A
GOSUB	\$8D	00141	GO shift S
GO64	\$CB \$64	00203	N/A
GOTO	\$89	00137	G shift O
GRAPHIC	\$DE	00222	G shift R
GSHAPE	\$E3	00227	G shift S
HEADER	\$F1	00241	HE shift A
HELP	\$EA	00234	N/A
HEX\$	\$D2	00210	H shift E

KEYWORD	HEX	DECIMAL	SHORT-HAND
IF THEN ELSE	\$8B \$A7 \$D5 00139 00167 00213		N/A
INPUT	\$85	00133	N/A
INPUT#	\$84	00132	I shift N
INSTR	\$D4	00212	IN shift S
INT	\$B5	00181	N/A
JOY	\$CF	00207	J shift O
KEY	\$F9	00248	K shift E
LEFT\$	\$C8	00200	LE shift F
LEN	\$C3	00195	N/A
LET	\$88	00136	L shift E
LIST	\$9B	00155	L shift I
LOAD	\$93	00147	L shift O
LOCATE	\$E6	00230	LO shift C
LOG	\$BC	00188	N/A
LOOP	\$EC	00236	LO shift O
MID\$	\$CA	00202	M shift I
MONITOR	\$FA	00250	MO shift N
MOVSPR	\$FE \$06	00254 00006	M shift O
NEW	\$A2	00162	N/A
NEXT	\$82	00130	N shift E
ON..GOSUB	\$91 \$8D	00145 00141	ON GO shift S
ON..GOTO	\$91 \$89	00145 00137	ON G shift O
OPEN	\$9F	00159	O shift P
PAINT	\$DF	00223	P shift A
PEEK	\$C2	00194	PE shift E
PEN	\$CE \$04	00206 00004	P shift E
PI	\$50 \$49	00080 00073	N/A
PLAY	\$FE \$04	00254 00004	P shift L
POKE	\$97	00151	PO shift K
POS	\$B9	00185	N/A
POT	\$CE \$02	00206 00002	P shift O
PRINT	\$99	00153	?
PRINT#	\$98	00152	P shift R
PRINT USING	\$99 \$FB	00152 00251	?US shift I
PUDEF	\$DD	00221	P shift U
RCLR	\$CD	00205	R shift C
RDOT	\$D0	00208	R shift D
READ	\$87	00135	RE shift A
RECORD	\$FE \$12	00254 00018	R shift E

KEYWORD	HEX	DECIMAL	SHORT-HAND
REM	\$8F	00143	N/A
RENAME	\$F5	00245	RE shift N
RENUMBER	\$F8	00248	REN shift U
RESTORE	\$8C	00140	RE shift S
RESUME	\$D6	00214	RES shift U
RETURN	\$8E	00142	RE shift U
RGR	\$CC	00204	R shift G
RIGHT\$	\$C9	00201	R shift I
RND	\$BB	0187	R shift N
RSPCOLOR	\$CE \$07	00206 00007	RSP shift C
RSPPOS	\$CE \$05	00206 00005	R shift S
RSPRITE	\$CE \$07	00206 00006	RSP shift R
RWINDOW	\$CE \$09	00206 00009	R shift W
SAVE	\$94	00148	S shift A
SCALE	\$E9	00233	SC shift A
SCNCLR	\$E8	00232	S shift C
SCRATCH	\$F2	00242	SC shift R
SGN	\$B4	00180	S shift G
SIN	\$BF	00191	S shift I
SLEEP	\$FE \$0B	00254 00011	S shift L
SLOW	\$FE \$26	00254 00038	N/A
SOUND	\$DA	00216	S shift O
SPC(	\$A6	00166	N/A
SPRCOLOR	\$FE \$08	00254 00008	SPR shift C
SPRDEF	\$FE \$1D	00254 00029	SPR shift D
SPRITE	\$FE \$07	00254 00007	S shift P
SPRSAV	\$FE \$16	00254 00022	SPR shift S
SQR	\$BA	00186	S shift Q
SSHAPE	\$E4	00228	S shift S
STASH	\$FE \$1F	00254 00031	S shift T
STatus	\$53 \$54	00083 00084	N/A
STEP	\$A9	00168	ST shift E
STOP	\$90	00144	ST shift O
STR\$	\$C4	00196	ST shift R
SWAP	\$FE \$23	00254 00035	S shift W
SYS	\$9E	00158	N/A
TAB(	\$A3	00163	T shift A
TAN	\$C0	00192	N/A
TEMPO	\$FE \$05	00254 00005	T shift E

KEYWORD	HEX	DECIMAL	SHORT-HAND
TI	\$54 \$49	00084 00073	N/A
TI\$	\$54 \$49 \$24	00084 00073 00036	N/A
TO	\$A4	00164	N/A
TRAP	\$D7	00215	T shift R
TROFF	\$D9	00217	TRO shift F
TRON	\$D8	00216	TR shift O
UNTIL	\$FC	00252	U shift N
USR	\$B7	00183	U shift S
VAL	\$C5	00197	N/A
VERIFY	\$95	00149	V shift E
VOL	\$DB	00219	V shift O
WAIT	\$92	00146	W shift A
WHILE	\$FD	00253	W shift H
WIDTH	\$FE \$1C	00254 00028	WI shift D
WINDOW	\$FE \$1A	00254	W shift I
XOR	\$CE \$08	00206 00008	X shift O

BASIC 7.0 ERROR MESSAGES

An error message is the computer's way of telling you something is wrong. When exploring for the cause, you must remember that a computer is very exacting in its demands. One forgotten comma, will generate a syntax error.

The C-128 has an added feature, to help the user find an error, the HELP key. The HELP key is located within the gray row of keys, at the top of the keyboard. When the HELP key is pressed, the line containing the error is shown on the screen. The code will be high-lighted, from the start of the problem, to the end of the line. Next review the high-lighted portion of the code very carefully. The clues to an error, are often subtle. Stop, think, take time to review the problem completely. The answer is usually in plain sight. The computer can only follow your instructions.

#	MESSAGE	DESCRIPTION
1.	TOO MANY FILES	Limit 10 files open at once
2.	FILE OPEN	Tried to open an open file
3.	FILE NOT OPEN	Must open file before use
4.	FILE NOT FOUND	File name not found on disk
5.	DEVICE NOT PRESENT	I/O device can not detected
6.	NOT INPUT FILE	Attempted to GET or input data from an output file
7.	NOT OUTPUT FILE	Attempted to send data to an input only file
8.	MISSING FILE NAME	file name not in the command
9.	ILLEGAL DEVICE #	Tried to use wrong device (Save to the screen, etc.)
10.	NEXT WITHOUT FOR	Executed NEXT without for
11.	SYNTAX	Wrong punctuation, spelling, line number missing, etc
12.	RETURN WITHOUT	A RETURN without a GOSUB has been detected
13.	OUT OF DATA	READ statement found, when there is no data left to read
14.	ILLEGAL QUANTITY	Number used outside of range

#	MESSAGE	DESCRIPTION
15.	OVERFLOW	Result larger than allowed
16.	OUT OF MEMORY	Out of K, all memory used
17.	UNDEF'D STATEMENT	Line number does not exist
18.	BAD SUBSCRIPT	Array number outside range defined in DIM statement
19.	REDIM'D ARRAY	Array may only be DIMed once
20.	DIVISION BY ZERO	Division by zero not allowed
21.	ILLEGAL DIRECT	INPUT,GET,INPUT#, get# are only allowed within program
22.	TYPE MISMATCH	Wrong data type
23.	STRING TO LONG	String limit 255 characters
24.	FILE DATA	Bad data from tape or disk received by the computer
25.	FORMULA TOO COMPLEX	Operation too complex for the computer to understand
26.	CAN'T CONTINUE	Program has an error or has been edited
27.	UNDEF'D FUNCTION	Function hasn't been defined defined by the user
28.	VERIFY	File in memory does not match one on the device
29.	LOAD	Loading problem
30.	BREAK	Stop key has been pressed
31.	CAN'T RESUME	RESUME without TRAP found
32.	LOOP NOT FOUND	PRG found DO without LOOP
33.	LOOP WITHOUT DO	LOOP found without DO
34.	DIRECT MODE ONLY	Use in direct mode only
35.	NO GRAPHICS AREA	Graphics area undefined
36.	BAD DISK	Failed to format, may be bad
37.	BEND NOT FOUND	"IF...THEN BEGIN" or "IF.. THEN...ELSE BEGIN" found without BEND to match BEGIN
38.	LINE # TOO LARGE	Line number is larger than the allowable 63999
39.	UNRESOLVED	Line number referred to is missing
40.	UNIMPLEMENTED	Command not in BASIC 7.0
41.	FILE READ	An error has been found while loading/reading from a device

128 CONTROL CODES

The CONTROL KEY is found at the middle left-hand side of the keyboard. The key generates control functions. These are often special tasks, or short cuts, to much larger jobs. To perform a control function, hold down the CTRL key, and press the other key that is called for at the same time. The CHR\$ functions are another way of doing these same tasks within BASIC code. The CHR\$ refers to a table, of character string codes in the computers memory. The CHR\$ function converts a Commodore ASCII code to its character equivalent.

C= Commodore logo key, location, lower left corner of the keyboard.

CHR\$	CTRL KEY	FUNCTION
CHR\$(2)	CTRL B	Underline (80 column)
CHR\$(5)	CTRL2/E	Character color white (40 or 80 column mode)
CHR\$(7)	CTRL G	Produce bell tone
CHR\$(9)	CTRL I	Send cursor to next set tab position
CHR\$(10)	CTRL J	Send a line feed
CHR\$(11)	CTRL K	Character set change on
CHR\$(12)	CTRL L	Character set change off
CHR\$(13)	CTRL M	Send carriage return/line feed to computer, enter a line of BASIC code
CHR\$(14)	CTRL N	Character set to upper & lowercase
CHR\$(15)	CTRL O	Turn flash on (80 column)
CHR\$(17)	CRSR DN/CTRL Q	Cursor down one row
CHR\$(18)	CTRL 9	Reverse field on
CHR\$(19)	HOME/CTRL S	Cursor to home position
CHR\$(20)	DEL/CTRL T	Delete character and fill space
CHR\$(24)	CTRL X	Tab set to clear
CHR\$(27)	ESC/CTRL I	Send ESC character

CHR\$	CTRL KEY	FUNCTION
CHR\$(28)	CTRL3/CTRL/	Character color to red (40 or 80 column)
CHR\$(29)	CRSR/CTRL]	Cursor one column right
CHR\$(30)	CTRL6/CTRL	Character color to green (40 or 80 column)
CHR\$(31)	CTRL 7	Character color to blue
CHR\$(34)	"	Prints a quote mark
CHR\$(129)	C=1	Character color to orange (40 column), dark purple (80 column mode)
CHR\$(130)		Underline off (80 column)
CHR\$(131)		Run PRG, keyboard only
CHR\$(133)	F1	Reserved CHR\$ for F1 key
CHR\$(134)	F3	Reserved CHR\$ for F3 key
CHR\$(135)	F5	Reserved CHR\$ for F5 key
CHR\$(136)	F7	Reserved CHR\$ for F7 key
CHR\$(137)	F2	Reserved CHR\$ for F2 key
CHR\$(138)	F4	Reserved CHR\$ for F4 key
CHR\$(139)	F6	Reserved CHR\$ for F6 key
CHR\$(140)	F8	Reserved CHR\$ for F8 key
CHR\$(141)	SHIFT/RETURN	Carriage return/line feed without BASIC code
CHR\$(142)		Character set upper case, graphics
CHR\$(143)		Flash off (80 column)
CHR\$(144)	CTRL 1	Character color black (40 or 80 column mode)
CHR\$(145)	CRSR UP	Move cursor up one row
CHR\$(146)	CTRL 0	Reverse field off
CHR\$(147)	HOME	Clear window screen and cursor home
CHR\$(148)	INST	Insert command
CHR\$(149)	C=2	Character color to brown (40 column mode). Dark yellow (80 column)
CHR\$(150)	C=3	Character color to light red (40 or 80 column)

CHR\$	CTRL KEY	FUNCTION
CHR\$(151)	C=4	Character color dark gray (40 column mode), dark cyan (80 column)
CHR\$(152)	C=5	Character color medium gray (40 or 80 column)
CHR\$(153)	C=6	Character color to light green (40 or 80 column)
CHR\$(154)	C=7	Character color light blue (40 or 80 column)
CHR\$(155)	C=8	Character color light gray (40 or 80 column)
CHR\$(156)	CTRL 5	Character color purple (40 or 80 column mode)
CHR\$(157)	CRSR LEFT	Cursor left one column
CHR\$(158)		Character color to yellow
CHR\$(159)	CTRL 4	Character color cyan (40 column mode) Light cyan (80 column)

ESCAPE CODES

The ESCAPE key is used to activate special functions, such as editing, scrolling, define windows, etc.. The ESCAPE key is gray and located in the upper left corner of the keyboard.

To activate one of the commands, first press and then RELEASE the ESCAPE key followed by the desired letter key. You MUST remember, to release the ESCAPE key before pressing the next character.

ESCAPE KEY	ESCAPE FUNCTION
ESC Q	Erase to the end of the current line
ESC P	Erase to the start of the current line
ESC D	Delete the current line
ESC I	Insert a line
ESC A	Auto-insert mode on
ESC O	Auto-insert mode off, 80 column only
ESC C	Quote and insert mode off
ESC @	Clear everything to the end of screen
ESC J	Move cursor to start of current line
ESC K	Move to the end of the current line
ESC Y	Set default tab stops (8 spaces)
ESC Z	Clear all tab stops
ESC L	Scrolling on
ESC M	Scrolling off
ESC V	Scroll up
ESC W	Scroll down
ESC G	Bell on (by control-G)
ESC H	BELL OFF
ESC E	Cursor to non-flashing mode
ESC F	Cursor to flashing mode
ESC B	Bottom of window at cursor position
ESC T	Top of window at cursor position
ESC X	Swap output device (40/80 col.)
ESC U	Underlined the cursor (80 column only)
ESC S	Change to block cursor (80 col. only)
ESC R	Reverse video on (80 column only)
ESC N	Reverse video off (80 column only)

C-128 FUNCTION KEYS

The four function keys are located in the upper right hand corner of the keyboard. To use the keys, press number one, three, five, or seven for the first four. Numbers two, four, six, or eight are activated by depressing SHIFT at the same time as the key. All eight of the keys have been pre-defined in the C-128 mode, by the manufacture as follows:

KEY	CHR\$	FUNCTION	ACTION
F1	(133)	GRAPHIC	To enter the GRAPHIC mode
F2	(137)	DLOAD	Prints DLOAD, you add the file name to load program
F3	(134)	DIRECTORY	Displays Directory of disk
F4	(138)	SCNCLR	Clears the screen
F5	(135)	DSAVE	Prints DSAVE, you add the file name to be saved
F6	(139)	RUN	RUNs the current program in memory
F7	(136)	LIST	LISTs the current program in memory to the screen
F8	(140)	MONITOR	To engage the machine language monitor

USER DEFINED F-KEYS

The Function keys may also be programmed by the user from a BASIC program or in direct mode.

EXAMPLE: KEY 8, "OPEN 15,8,15"

In the above example, key 8 has been user-defined to OPEN the disk drive channel. This may be a time saver if used often in your program.

C-128 COLOR

The Commodore 128 offers sixteen colors, in both the forty or eighty column modes. The screen, border, or character color, may be changed to any of the sixteen hues. The following charts list the colors available, plus the three modes available to change color in 128 mode.

COLOR COMMAND

To select the desired color, type the following:

COLOR (source),(color)

The choice of source (0-6) is as follows:

- 0 40 column background color
- 1 Foreground for the graphics screen
- 2 Foreground color 1 for multicolor screen
- 3 Foreground color 2 for multicolor screen
- 4 40 column border (text or graphics mode)
- 5 Character color (40 or 80 column text)
- 6 80 column background color

Color is a value, from column one, of the charts that follow (40 or 80 column).

CONTROL or COMMODORE key

The commands in column three (labeled KEY) are used to modify character color only. To use, hold down the CONTROL or COMMODORE LOGO key, plus the number of the color at the same time.

EXAMPLE: Press the CONTROL key and the number seven. From now, until another color command is entered, all characters will be in yellow.

CHR\$

The fourth column displays the CHR\$ value for each of the sixteen colors. The primary use for these, are to alter the character color within program code.

EXAMPLE: CHR\$(28) All characters will now be red.

COLOR 40 COLUMN MODE

COLOR	VALUE	KEY	CHR\$
BLACK	1	CTRL 1	CHR\$(144)
WHITE	2	CTRL 2	CHR\$(5)
RED	3	CTRL 3	CHR\$(28)
CYAN	4	CTRL 4	CHR\$(159)
PURPLE	5	CTRL 5	CHR\$(156)
GREEN	6	CTRL 6	CHR\$(30)
BLUE	7	CTRL 7	CHR\$(31)
YELLOW	8	CTRL 8	CHR\$(158)
ORANGE	9	C= 1	CHR\$(129)
BROWN	10	C= 2	CHR\$(149)
LIGHT RED	11	C= 3	CHR\$(150)
DARK GRAY	12	C= 4	CHR\$(151)
MEDIUM GRAY	13	C= 5	CHR\$(152)
LIGHT GREEN	14	C= 6	CHR\$(153)
LIGHT BLUE	15	C= 7	CHR\$(154)
LIGHT GRAY	16	C= 8	CHR\$(155)

COLOR 80 COLUMN MODE

COLOR	VALUE	KEY	CHR\$
BLACK	1	CTRL 1	CHR\$(144)
WHITE	2	CTRL 2	CHR\$(5)
DARK RED	3	CTRL 3	CHR\$(28)
LIGHT CYAN	4	CTRL 4	CHR\$(159)
LIGHT PURPLE	5	CTRL 5	CHR\$(156)
DARK GREEN	6	CTRL 6	CHR\$(30)
DARK BLUE	7	CTRL 7	CHR\$(31)
LIGHT YELLOW	8	CTRL 8	CHR\$(158)
DARK PURPLE	9	C= 1	CHR\$(129)
DARK YELLOW	10	C= 2	CHR\$(149)
LIGHT RED	11	C= 3	CHR\$(150)
DARK CYAN	12	C= 4	CHR\$(151)
MEDIUM GRAY	13	C= 5	CHR\$(152)
LIGHT GREEN	14	C= 6	CHR\$(153)
LIGHT BLUE	15	C= 7	CHR\$(154)
LIGHT GRAY	16	C= 8	CHR\$(155)

GRAPHIC COMMAND SUMMARY

The Commodore 128 has several types of unique graphic features. A creative programmer can become an accomplished artist in a short period of time

Text may be displayed in a rainbow of various colors. Elaborate graphic images can be created by connecting letters, numbers, special symbols, and graphic characters of the three available keyboards.

A second type of graphic display, is the bit-map mode for drawing intricate pictures and designs. Included in this category are both standard bit-map and multi-color bit-map mode. In the bit-map modes, you control each pixel, thus allowing great detail in your work.

The third type of screen display, is a combination of text and bit-map graphics. This allows a split screen as you might see in an adventure game, picture at the top of the screen and text at the bottom.

An additional graphic feature is available, known as a sprite. The sprite is a movable graphic shape, that may be in any user defined form. The sprite image may be animated, colored, or even multi-colored. The size of a sprite, is 24 pixels wide, by 21 pixels high, in regular bit map mode or 12 pixels in width by 21 high in multicolor bit map mode. There are three (3) methods for sprite creation in the C-128 mode:

1. Sprite statements within a program.
2. SPRite DEFinition mode (SPRDEF), this mode uses an editor within the C-128
3. Created the same as in the C-64 mode.

The first step in graphics programming, is selection of the graphic mode. The proper syntax is:

EXAMPLE: GRAPHIC1,1

In the example, the first number one (1), is GRAPHIC mode one (1), (standard bit map), the second one (1)

clears the cscreen (0= not clear, 1= clear.

MODE	DESCRIPTION
0	40 column standard text
1	Standard bit map
2	Standard bit map (split screen)
3	Multicolor bit map
4	Multicolor bit map (split screen)
5	80 column TEXT only

The following pages, are a list of graphic commands. On the first line of each entry is the command and an example of syntax. The second line is a brief definition of the function the command performs.

BOX 235 BOX1,11,11,80,80
Used to draw a BOX at a specific place on the screen

BUMP 230 BUMP[1-2]
Returns sprite collision information.

CHAR 50 CHAR0,20,20,"welcome",0
Display characters at a specific place on the screen

CIRCLE 230 CIRCLE1,150,100,60,10
Draws CIRCLES, arcs, etc. at a specified position.

COLLISION 550 COLLISION[1-3],900
Identifies sprite collisions, specifies action.

COLOR 10 COLOR[0-6],[1-16]
COLOR source number, COLOR number
Assigns colors to one of the 7 screen color areas.

DRAW 250 DRAW[0-3],20,20 to 100,100
A graphics command, used to draw to the screen.

GRAPHIC 10 GRAPHIC[0-1],1
Places the computer in one of the six graphic modes.

SPRCOLOR 110 SPRCOLOR[1-16],[1-16]
Sets multicolor 1 and or 2 for sprites.

SPRDEF SPRDEF
To enter the sprite editor in C-128 mode.

SPRITE 100 SPRITE[1-8],[0-1],
[1-16],[0-1],[0-1],[0-1],[0-1]
Turns sprites controls off or on.

SPRSAV 235 SPRSAV 1,a\$
Saves sprite data from a text string variable to a storage area in memory.

SSHAPE 555SSHAPE d\$,20,20
Used to save or load rectangular areas of high resolution/ multicolor screens

SPRite DEFinition mode

The following commands are used to create a sprite using the internal sprite editor in BASIC 7.0. To engage the editor type SPRDEF and press return.

Shift CLR key	Erases the entire work area
M key	Turns on/off multicolor sprite
CTRL 1-8	Selects sprite foregroundcolor 1-8
C= 1-8	Selects sprite foreground color 9-16
1 key	Turns off pixel color
2 key	Turns on pixels in foreground color
3 key	Turns on areas in multicolor 1
4 key	Turns on areas in multicolor 2
A key	Turns on/off automatic cursor movement
CRSR keys	Moves the cursor within the work area
RETURN	Moves cursor to start of next line
HOME key	Moves cursor/top left corner/work area
X key	Expands sprite horizontally

Y key	Expands sprite vertically
Shift RETURN	Saves sprite from work area/returns to SPRITE NUMBER prompt
C key	Copies one sprite to another
STOP key	Turns off displayed sprite/returns to SPRITE # prompt/no change to sprite
RETURN key	(at SPRITE # prompt) Exits SPRDEF mode

128 SOUND

The C-128 mode has the same musical capability as the C-64. The one exception is the usage of BASIC 7.0 for programming the SID (Sound Interface Device) chip. The new powerful commands, make creating music very easy. A note may be entered as a letter, as read from sheet music, rather than complicated poke values.

The following are the functions of SID:

- 3 independent programmable voices
- 4 mixable waveforms for each voice
- Mixable filters to choose or omit selected sounds
- Envelope generator (ADSR control) for each voice
- 10 predefined envelopes for musical instruments
- 2 cascadable ring modulators
- Alternation option for external signal sources
- 2 b-bit A/D converters

The BASIC 7.0 commands employed to program music, are listed below.

SOUND

EXAMPLE: SOUND[vc],[freq],[dur],[dir],[min],[sv],
[wf],[pw]

The commands to create sound and music.

- vc = Select VoiCe 1,2, or 3
- freq = Sets FREQuency level (0-65535)
- dur = Sets DURation of sound (0-32767)
- dir = Sets DIRection sound is incremented
0 = up, 1 = down, 2 = up and down
- min = Select MINimum frequency (0-65535)
only if DIR specified
- sv = Step Value for the sweep (0-32767)
- wf = Select WaveForm, 0 = triangle, 1 = sawtooth,
2 = variable pulse, 3 = white noise
- pw = Set Pulse Width (0-4095)

DIR,MIN,SV,Wf, and PW are optional parameters.

ENVELOPE

EXAMPLE: ENVELOPE [e],[a],[d],[s],[r],[wf],[pw]
Defines a musical instrument, sets the following:
Envelope#, Attack rate, Decay rate, Sustain, Release,
WaveForm, Pulse Width.

e = envelope number (0-9). There are 10 predefined envelopes for musical instruments. This area may also be defined by the user.

ENVEL- OPE #	INSTRU- MENT	DE- ATTACK	SUS- TAIN	RE- LEASE	WAVE- FORM	WIDTH
0	Piano	0	9	0	2	1536
1	Accordion	12	0	12	0	
2	Calliope	0	0	25	0	
3	Drum	0	5	5	0	
4	Flute	9	4	4	0	
5	Guitar	0	9	2	1	
6	Harpsichord	0	9	0	0	
7	Organ	0	9	9	0	
8	Trumpet	8	9	4	1	512
9	Xylophone	0	9	0	0	2048

a = attack rate (0-15). Attack is the rate at which a musical note reaches its peak.

d = decay rate (0-15). Rate at which a note decreases from peak volume.

s = sustain level (0-15). Level at which a note is played between attack and delay,

r = release rate (0-15). The rate at which a note decreases from its sustain level to zero.

wf = waveform (0-4). The type of sound wave. 0 = triangle, 1 = sawtooth, 2 = pulse (square), 3 = noise, 4 = ring modulation.

pw = pulse width (0-4095). Length of time between notes.

FILTER

EXAMPLE: FILTER[cf],[lp],[bp],[hp],[res]

The FILTER command is used to increase the sharpness and quality of sound.

cf = Cutoff Frequency (0-2047)
lp = LowPass filter 0 = off, 1 = on
bp = BandPass filter 0 = off, 1 = on
hp = High Pass filter 0 = off, 1 = on
res = RESonance (0-15)

PLAY

EXAMPLE: PLAY"CDEFGAB"

Allows selection of musical notes. Enclose the letter of the note or notes within quote marks. Many options may also be added between the SAME quote marks. To specify the duration of the note, place one of the following before the note:

W = whole note H = half note Q = quarter note
I = eighth note S = sixteenth note

R adds a rest, M says wait for end of measure

For further control, add one of the following control characters plus a value from the column labeled RANGE.

CONTROL CHARACTER	DESCRIPTION	RANGE	DEFAULT SETTING
----------------------	-------------	-------	--------------------

V	Voice	1-3	1
O	Octave	0-6	4
T	Envelope	0-9	0
U	Volumn	0-15	9
X	Filter	0=off,1=on	0

TEMPO

EXAMPLE: TEMPO [x]

The musical command to define speed of play. The x may be a value from the low of 0 to the high of 255

VOL

EXAMPLE: VOL [x]

Defines the level of sound. The x may be a value from the low of 0 to the high of 15

BASIC 7.0 DISK COMMANDS

The commands listed below are used with the 1571 disk drive in C-128 mode.

COMMAND	USE
APPEND	Append data to file
BLOAD	Load a binary file starting at specific memory location
BOOT	Load and execute program
BSAVE	Save a binary file to a specified memory location
CATALOG	Display contents of dir. on monitor
CMD	Redirect screen output to disk file
COLLECT	Free inaccessible disk space
CONCAT	Concatenate two data files
COPY	Copy files between devices
DCLEAR	Clear open channels on disk drives
DCLOSE	Close logical disk file
DIRECTORY	Display contents of dir. on monitor
DLOAD	Load a BASIC program from disk
DOPEN	Open disk file for read/write
DSAVE	Save a BASIC program to disk
DVERIFY	Compare file in mem to file on disk
GET#	Read one byte from a record
HEADER	To format a disk, make ready to use
INITIALIZE	Returns drive to power-up condition
INPUT#	Read whole record from disk
PRINT#	Output data to a file
RECORD	Position REL file pointers
RENAME	Change name of a file on disk
RUN	Execute BASIC program from disk
SCRATCH	Erase file(s) from a disk
VERIFY	Compare file in mem to one on disk

C-128 POKES and PEEKS

The C-128 (in 128 mode) has 131,072 individual memory locations. Each address stores a number ranging from 0 to 255. A PEEK allows the user, to read the number stored at a given address. A POKE lets you write data into a RAM (Random Access Memory) location. The ROM (Read Only Memory) may be read, but cannot be written to. POKES and PEEKS can extend your BASIC programming skills to a higher level. The following list is just a beginning, explore other locations. You won't hurt your computer, and you may learn a lot.

POKE or PEEK (bank 0)	ACTION
POKE 24,35	Replaces LINE # with # character default = 27
POKE 24,37	No line numbers, default = 27
POKE 24,1	Listing is garbage, default = 27
POKE 24,19	Double space listing, default = 27
POKE 24,53	Locks up the system, default = 27
PEEK (215)=0	40 column mode
PEEK (215)=128	80 column mode
POKE 775,171	Crash if list is tried, default=81
POKE 775,200	Line #s visible, code gone,default=81
POKE 808,222	Disables RUN/STOP/RESTORE,default=110
POKE 808,233	Disables RUN/STOP only, default=110
POKE 819,243	Disables SAVE, prints ? OUT OF MEMORY DEFAULT = 245
POKE 819,243	Disables LIST function
POKE 2594,96	Disables repeat of keys, default=128
POKE 53265,11	Screen blanking, default=27
POKE 53280,X	X = screen color value
POKE 53281,X	X = border color value
POKE 53296,1	Fast mode = 2 MHz
POKE 53296,252	Slow mode = 1 MHz
POKE 53220	Joystick port 1
POKE 56321	Joystick port 2
SYS 65357	Go to C-64 mode
SYS 65375	From 40 to 80 column mode or back

C128 M/L MONITOR

The C128 contains a built in machine language monitor that enables the user to read or write M/L. The monitor may be used in C128 mode, 40/80 column. To enter this mode from BASIC, type:

Example: MONITOR <RETURN> or press 'F8'.

NOTE: <> Required parameters [] Optional parameters

COMMANDS

ASSEMBLE

FUNCTION: To enter a line of assembly code.

FORMAT : A <start add><opcode>[operand]

COMPARE

FUNCTION: Compares 2 sections of memory and displays the differences.

FORMAT : C <start add><end add><new start add>

DISASSEMBLE

FUNCTION: Disassemble ML code into assem.

FORMAT : D [<start add><end add>]

FILL

FUNCTION: Fills a range of memory with specified byte

FORMAT : F <start add><end add><byte>

GO

FUNCTION: Begin program, at specific address.

FORMAT : G [add]

HUNT

FUNCTION: Look for specific byte occurrences of data.

FORMAT : H <star add><end add><byte 1>[<byte 2>...]
or
H <start add><end add><'ASCII string'>

JUMP

FUNCTION: Jumps to specified subroutine.

FORMAT : J [add]

LOAD

FUNCTION: Load a file from disk or tape.

FORMAT : L <"filename">[,<device#>[,<load add>]]

MEMORY

FUNCTION: Display of memory in HEX.

FORMAT : M [<start add>[<end add>]]

REGISTERS

FUNCTION: Displays the registers.

FORMAT : R

SAVE

FUNCTION: Saves a range of memory to disk or tape.

FORMAT : S <"filename">,<device#>,<start add>,
<end add+1>

TRANSFER

FUNCTION: Move segment of mem. to new area.

FORMAT : T <start add><end add><new start add>

VERIFY

FUNCTION: Check file on device with memory.

FORMAT : V <"filename">[,<device#>[,<load add>]]

EXIT

FUNCTION: Exits to BASIC.

FORMAT : X

FUNCTION: Indicates an assembler directive.

FORMAT : .

>

FUNCTION: Modifies memory.

FORMAT : >

;
FUNCTION: Modifies 8502 register displays.
FORMAT : ;

@
FUNCTION: Displays the following:
Disk status @[device#]
Disk command @[device#][,<command string>]
Catalog @[device#],\${[[<drive>]][:<file spec>]]

C-128 MEMORY

The following memory maps have been included, so that the user, may explore the architecture of C-128 mode. The BANKS allow memory, to be rearranged to different configurations. Bank 0 from \$0000 to \$03FF, is common to all of the BANK areas. Bank 0 contains work areas and the user's BASIC program. One might think of this system as a room of furniture, that may be rearranged to suit the user's programming need. The computer defaults to BANK 15 at power-up.

BANK	CONFIGURATION
0	RAM (0) only
1	RAM (1) only
2	RAM (2) only
3	RAM (3) only
4	Internal ROM, RAM (0), I/O
5	Internal ROM, RAM (1), I/O
6	Internal ROM, RAM (2), I/O
7	Internal ROM, RAM (3), I/O
8	External ROM, RAM (0), I/O
9	External ROM, RAM (1), I/O
10	External ROM, RAM (2), I/O
11	External ROM, RAM (3), I/O
12	Kernal/Internal ROM (low), RAM (0), I/O
13	Kernal/External ROM (low), RAM (0), I/O
14	Kernal/BASIC ROM, RAM (0), Character ROM
15	Kernal/BASIC ROM, RAM (0), I/O

BANK (0) 0000-03FF COMMON TO ALL BANKS

HEX	DECIMAL	DESCRIPTION
0000	0	6510 data direction, processor port
0001	1	6510 data register, processor port
0002	2	Storage for bank byte
0003	3	Storage/program counter, high
0004	4	Storage/program counter, low

HEX	DECIMAL	DESCRIPTION
0005	5	Storage for CPU status register
0006	6	Storage for accumulator
0007	7	Storage for X-register
0008	8	Storage for Y-register
0009	9	Storage for stack pointer
000A	10	Look for quote at end of string
000B	11	Screen color/last tab
000C	12	Disk flag: 0=load, 1=verify
000D	13	Pointer: #/subscripts/input buf.
000E	14	Flag: default dim
000F	15	Type: FF=string; 00=numeric
0010	16	Type: 80=integer; 00=floating point
0011	17	Flag: LIST,read DATA,garb. coll.
0012	18	Flag: subscript reference
0013	19	Flag: \$00=input, \$40=get, \$98=read
0014	20	Flag: tan sign/compare results
0015	21	Current I/O prompt flag
0016-0017	22-23	Integer value
0018	24	Pointer: temporary string stack
0019-0020	25-26	Last temporary string address
001B-001D	27-29	3 byte stack/temporary strings
001E-0020	30-32	3 byte stack/temporary strings
0021-0023	33-35	3 byte stack/temporary strings
0024-0025	36-37	Utility pointer area
0026-0027	38-39	2 byte help pointer index 2
0028	40	Floating point, product/multi.
002D-002E	45-46	Pointer: start of BASIC (bank 0)
002F-0030	47-48	Pointer: start of variables (bank 1)
0031-0032	49-50	Pointer: start of arrays
0033-0034	51-52	Pointer: end of arrays
0035-0036	53-54	Pointer: string storage (moving down)
0037	55	Lowest string vector
0039-003A	57-58	Pointer: limit of memory(bank 1)
003B-003C	59-60	Current BASIC line number
003D-003E	61-62	Text pointer: BASIC work point

HEX	DECIMAL	DESCRIPTION
003F-0040	63-64	Pointer: PRINT USING,char.search
0041-0042	65-66	Current DATA line number
0043-0044	67-68	Current DATA address
0045-0046	69-70	Pointer: INPUT routine hi/low
0047-0048	71-72	Current variable name
0049-004A	73-74	Current variable name
004B-004C	75-76	Pointer: mask/AND,LIST,FOR NEXT
004D-004E	77-78	Temp. storage for prog. pointer
004F	79	Mask for compare operation >:2, =:4,<:8
0050-0051	80-81	Pointer: FN and garbage collect.
0052-0054	82-84	Pointer: variable list-string compare
0055	85	Help flag: \$xx=HELP,\$xx=LIST
0056-0057	86-87	Jump vector/function evaluation
0059	89	Area for INSTRING operation or Temporary pointer 1
005A-005B	90-91	Pointer: block transfer,DIM init
005C-005D	92-93	Pointer: block transfer
005E	94	Temporary pointer 2
005F-0060	95-96	Number of places before/after Decimal for conversion
0062	98	Exponent sign of # read
0063	99	Accum.#1: exponent
0064-0067	100-103	Accum.#1: mantissa
0068	104	Accum.#1: sign
0069	105	Pointer: Polynomial evaluation
006A	106	Accum.#2: exponent
006B-006E	107-110	Accum.#2: mantissa
006F	111	Accum.#2: sign
0070	112	Sign comparison, accum.#1/#2
0071	113	Accum.#1 low order (rounding)
0072-0073	114-115	Pointer: cassette buffer
0074-0075	116-117	AUTO command offset value, 0=off
0076	118	Flag:1=BASIC-st. set 10K higher
0077	119	Sprite # counter/leading zeros
0078	120	Help counter
0079	121	Temp. storage/indirect loading
007A-007C	122-124	Description/error variable DS\$

HEX	DECIMAL	DESCRIPTION
007D-007E	125-126	End of stack during program run
007F	127	\$xx=RUN mode,\$xx=direct mode
0080	128	Pointer:statue of DOS parser
0083	131	Current color for graphic mode
0084	132	Multi-color mode: color 1
0085	133	Multi-color mode: color 2
0086	134	Foreground color
0087-0188	135-136	X-direction scale factor
0089-008A	137-138	Y-direction scale factor
008B	139	Stop draw,if not background col.
008C-008D	140-141	Pointer:address/graphic routines
008E	142	Temp storage 1/graphic routines
008F	143	Temp storage 2/graphic routines
0090	144	Status word ST
0091	145	Keyswitch IA:STOP and RVS flags
0092	146	Time constants/cassette
0093	147	Load flag: 0=LOAD,1=VERIFY
0094	148	Serial bus flag: char in buffer
0095	149	Character in buffer/serial bus
0096	150	End-of-tape received from tape
0097	151	Temporary data address
0098	152	Number of files open
0099	153	Input device, normally 0
009A	154	Output CMD device, normally 2
009B	155	Parity byte from cassette
009C	156	Tape flag: byte received
009D	157	I/O messages: 192=all, 64=errors 0=nil
00A0-00A2	160-162	Jiffy clock HMS
00A3-00A4	163-164	Temporary storage/serial bus
00A5	165	Pointer: SAVE to tape, help
00A6	166	Pointer: cassette buffer
00A7	167	RS-232 input bits,tape sh. count
00A8	168	RS-232 counter input bits
00A9	169	RS-232 start bit flag
00AA	170	RS-232 buffer input byte
00AB	171	RS-232 input parity
00AC-00AD	172-173	Pointer: screen scroll,cass buf.
00AE-00AF	174-175	Tape end adds/end of program

HEX	DECIMAL	DESCRIPTION
00B0-00B1	176-177	Cassette constant for time
00B2-00B3	178-179	Vector: tape buffer start
00B4	180	RS-232 next bit for scroll
00B5	181	RS-232 next byte for transfer
00B6	182	RS-232 byte buffer
00B7	183	Number of character in file name
00B8	184	Current logical file
00B9	185	Current secondary address
00BA	186	Current device
00BB-00BC	187-188	Pointer to file name
00BD	189	RS-232 parity buffer
00BE	190	# of remaining read/write blocks
00BF	191	Serial buffer
00C0	192	Tape motor interlock
00C1	193	Start address input/output,low
00C2	194	Start address input/output,high
00C3-00C4	195-196	Pointer: kernal vector address
00C5	197	Tape read/write data range
00C6	198	Bank # current LOAD,SAVE,VERIFY
00C7	199	Bank # current filename,\$BB,\$BC
00C8-00CB	200-203	RS-232 input/output buffer addr.
00CC-00CD	204-205	Keyboard decode pointer (bank 5)
00CE-00CF	206-207	Pointer: Kernal PRINT routine
00D0	208	Number of char. in keyboard buf.
00D1	209	# of programmed char. waiting
00D3	211	Key shift flag: 0=no shift
00D4	212	Flag for keypress
00D5	213	Last key code: 88 if no key
00D6	214	Input from screen/from keyboard
00D7	215	40/80 columns: 0=40 columns
00D8	216	Flag/text/graphic screen mode
00D9	217	Character base: 0=ROM, 4=RAM
00DA	218	Pointer: MOVLIN (low)
00DB	219	Pointer: MOVLIN (high)
00DC	220	Function key number
00DD	221	String length to current F-key
00DE	222	Bank /function key call
00DF	223	String length to current F-key
00E0-00E1	224-225	Pointer to screen line/cursor

HEX	DECIMAL	DESCRIPTION
00E2-00E3	226-227	Color line pointer
00E4-00E7	228-231	Screen margins: bottom, top, left, right
00E8-00E9	232-233	Input cursor log (row,column)
00EA	234	End/input cursor log(row,column)
00EB	235	Pos. of cursor on screen line
00EC	236	Row where cursor lives
00ED	237	Maximum number of screen lines
00EE	238	Maximum number of screen columns
00EF	239	Temporary storage/char. to go
00F0	240	Mem. last character (ESC test)
00F1	241	Color under cursor
00F3	243	Flag: RVS mode active
00F5	245	Flag: quote mode active
00F6	246	Flag: auto insert active
00F8	248	Screen scroll cutoff
00F9	249	Stop beep tones made by CTRL G
00FA-00FF	250-255	UNUSED
0100-01FF	256-511	Processor stack area
0100-01FF	256-318	Tape error log
0100-0124	256-292	DOS work area
0125-0138	293-312	PRINTUSING work area
0200-02A0	512-672	BASIC input buffer
02A2-02AE	674-686	BANK peek subroutine
02AF-02BD	687-701	BANK poke subroutine
02BE-02CC	702-716	BANK compare subroutine
02CD-02E2	717-738	JSR to another BANK
02E3-02FB	739-763	JMP to another BANK
02FC-02FD	764-765	Function execute hook
0300-0311	768-785	BASIC links
0302	770	BASIC warm start
0304	772	Tokenize BASIC text
0306	774	BASIC text list
0308	776	BASIC character dispatch
030A	778	BASIC token evaluation
0312-0313	786-787	UNUSED
0314-0315	788-789	IRQ vector
0316-0317	790-791	Break interrupt vector
0318-0319	792-793	NMI interrupt vector

HEX	DECIMAL	DESCRIPTION
031A-032D	794-813	Kernal vectors
032E-033D	814-829	Kernal links
033C-037F	828-895	Kernal tables
033E-0349	830-841	Keyboard matrix shift vectors
034A-0353	842-851	Keyboard buffer
0354-035D	852-861	Tab stop bits
035E-0361	862-865	Line wrap bits
0362-036B	866-875	Logical file table
036C-0375	876-885	Device number table
0376-037F	886-895	Secondary address table
0380-039E	896-926	CHRGOT subroutine
0386	902	CHRGOT entry
039F-03D1	927-938	Subroutine to fetch from RAM banks
03DF	991	Accum. #1: overflow
0400-04FF	1024-1279	VIC text screen (VM #1)
0400-07E7	1024-2023	40 column screen memory
0400-FEFF	1024-65279	BASIC variable storage
07F8-07FF	2040-2047	Sprite identity area (text)
0800-09FF	2048-2559	BASIC run time stack
0A00-0A01	2560-2561	Vector: BASIC restart
0A05-0A06	2565-2566	Pointer: Bottom of memory
0A07-0A08	2562-2563	Pointer: top of memory
0A18	2584	Pointer: RS-232 receive
0A19	2585	Pointer: RS-232 input
0A1A	2586	Pointer: RS-232 transmit
0A1B	2587	Pointer: RS-232 send
0A20	2592	Size of keyboard buffer
0A22	2594	Key repeat, 128 = all, 64 = none
0B00-0BBF	2816-3007	Cassette buffer
0BC0-0BFF	3008-3071	(disk boot page)
0C00-0CFF	3072-3327	RS-232 input buffer
0D00-0DFF	3328-3583	RS-232 output buffer
0E00-0FFF	3584-4095	Sprite definition area
1000-10FF	4096-4351	Function key buffer
1000-1009	4096-4105	number of bytes per key
100A		Function key strings
1100-1107	4352-4359	CP/M reset code
1108-11FF	4360-4607	BASIC DOS/VSP variables

HEX	DECIMAL	DESCRIPTION
11E9-11EA	4585-4586	Light pen values, X and Y
1200-1201	4608-4609	Previous BASIC line number
1202-1203	4610-4611	Pointer: BASIC statement CONT
1204-1207	4612-4615	Punctuation characters (.,\$)
1208	4616	Error type ER
1209-120A	4617-4618	Error line number EL
1210-1211	4624-4625	End of BASIC (bank 0)
1212-1213	4626-4627	BASIC program limit
1218-121A	4632-4634	USR program jump
121B-121F	4635-4639	RND seed value
1400-17FF	5120-6143	Reserved for foreign lang. sys.
1800-1BFF	6144-7162	Reserved for function key ML
1C00-FEFF	7168-65279	BASIC program area
1C00-1FFF	7168-8191	VIC bit-map color
2000-3FFF	8192-16383	Screen memory (hi-res)
4000-FBFF	16384-64511	BASIC RAM memory (hi-res)
4000-FEFF	16384-65279	BASIC program area continued
FF00	65280	MMU configuration register
FF01-FF04	65281-65284	MMU load config. registers
FF05-FF44	65285-65348	Kernal dispatch code
FFD0	65488	CP/M and kernal RAM code
FFFA-FFFF	65530-65535	NMI;RST; and IRQ vectors

BANK (1) 0000-03FF IS THE SAME AS RAM (0)

0400-FEFF	1024-64511	BASIC variables, strings, arrays
FF05-FF44	65285-65348	Kernal int. dispatch code
FFF5-FFF9	65525-65529	'CBM' and RST address \$E224
FFFA-FFFF	65530-65535	Vectors: NMI, RST, and IRQ

BANK 14 SAME AS BANK 15 EXCEPT:

D000-DFFF	53248-57343	ROM character generator
-----------	-------------	-------------------------

BANK 15

4000-CFFF	16348-53247	ROM: BASIC
D000-D030	53248-53296	40- column video chip 8564
D400-D41C	54272-54300	SID sound chip 6581

HEX	DECIMAL	DESCRIPTION
D500-D50A	54528-54538	MMU 8722 memory setup registers
D600-D601	54784-54785	80 column CRT control 8563
D800-D8E7	55296-56295	Color nybbles
DC00-DC0F	56320-56336	CIA 1 (IRQ) 6526
DD00-DD0F	56576-56591	CIA 2 (NM) 6526
DF00-DF0A	57088-57098	DMA controller
E000-FEFF	57344-65279	ROM: kernal
FF05-FFFF	65285-65535	ROM: Transfer, jump table

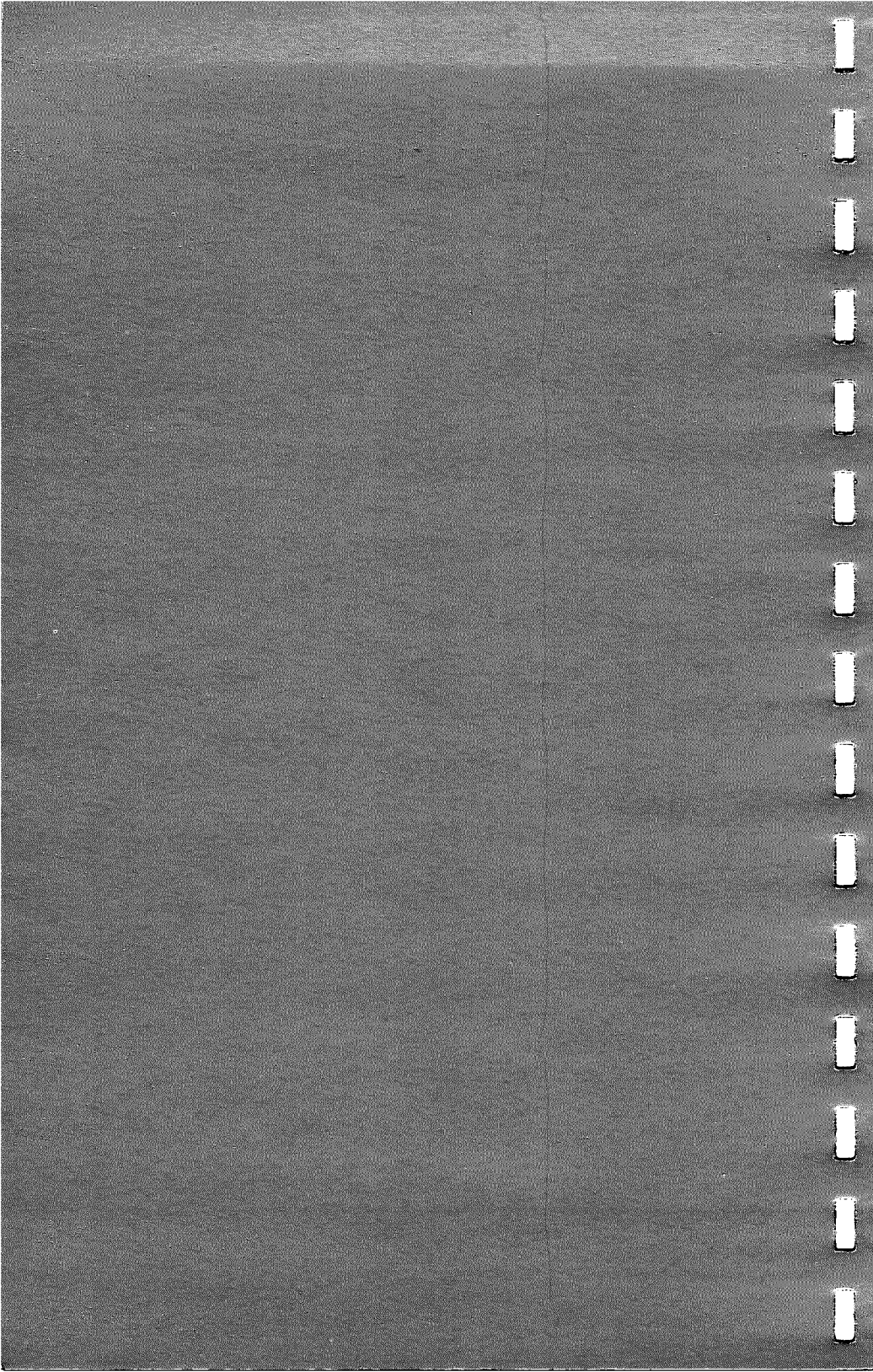
C-128 KERNAL ROUTINES

The Kernal ROM is a table of pre-defined subroutines. These addresses act as pointers to all the components needed to complete each routine. If Commodore changes or updates the operating system, these locations stay unchanged. Programmers can use this table, knowing a program will run on future models of the C-128.

HEX	DECIMAL	LABEL	ACTION
\$FF4D	65357	C64 MODE	Go to 64 mode, no return to 128
\$FF50	65360	DMA-CALL	Initialize external RAM
\$FF53	65363	BOOTCALL	Boot the disk at startup/insert
\$FF56	65366	PHOENIX	Cold start in C-128 mode
\$FF59	65369	LKUPLA	Search for logical file number
\$FF5C	65372	LKUPSA	Search for secondary address
\$FF5F	65375	SWAPPER	Swap from 40 to 80 columns
\$FF62	65378	DLCHR	Move char. set for 40/80 column
\$FF65	65381	PFKEY	Redefine function keys,SHIFT/ RUN-STOP,HELP
\$FF68	65384	SETBNK	Define mem. bank for disk drive
\$FF6B	65387	GETCONF	Get configuration byte
\$FF6E	65390	JSRFRAR	Jump to subroutine in any bank
\$FF71	65393	JMPFAR	Jump to any bank
\$FF74	65396	INDFET	Get byte from any bank
\$FF77	65399	INDSTA	Store accumulator in any bank
\$FF7A	65402	INDCMP	Compare accumulator with memory in any bank
\$FF7D	65405	PRIMM	Output text to current device
\$FF81	65409	CINIT	Initialize screen editor
\$FF84	65412	IOINIT	Initialize input/output
\$FF87	65415	RAMTAS	Initialize RAM/tape buffer, set screen
\$FF8A	65418	RESTOR	Restore default input/output vectors
\$FF8D	65421	VECTOR	Read/set vectored input/output
\$FF90	65424	SETMSG	Enable/disable DOS messages
\$FF93	65427	SECND	Send secondary addr. to LISTEN
\$FF96	65430	TKSA	Send secondary address to TALK
\$FF99	65433	MEMTOP	Read and set top of memory
\$FF9C	65436	MEMBOT	Read and set bottom of memory

HEX	DECIMAL	LABEL	ACTION
\$FF9F	65439	KEY	Scan the keyboard
\$FFA2	65442	SETTMO	Set time out flag for IEEE
\$FFA5	65445	ACPTR	Get a byte from the serial bus
\$FFA8	65448	CIOUT	Output a byte to the serial bus
\$FFAB	65451	UNTLK	Command serial bus to UNTALK
\$FFAE	65454	UNLSN	Command serial bus to UNLISTEN
\$FFB1	65457	LISTN	Command serial bus to LISTEN
\$FFB4	65460	TALK	Command the serial bus to TALK
\$FFB7	65463	READST	Read the status of input/output
\$FFBA	65466	SETLFS	Set file parameters
\$FFBD	65469	SETNAM	Set file name parameters
\$FFC0	65472	OPEN	Open a logical file
\$FFC3	65475	CLOSE	Close a logical file
\$FFC6	65478	CHKIN	Define a file as input
\$FFC9	65481	CKOUT	Define a file as output
\$FFCC	65484	CLRCH	Close input/output channels
\$FFCF	65487	BASIN	Get char. from input channel
\$FFD2	65490	BSOUT	Output char. to output channel
\$FFD5	65493	LOADSP	Load a file into memory
\$FFD8	65496	SAVESP	Save memory to a file
\$FFDB	65499	SETTIM	Set the system clock
\$FFDE	65502	RDTIM	Read the system clock
\$FFE1	65505	STOP	Scan the STOP key
\$FFE4	65508	GETIN	Get a character from a device
\$FFE7	65511	CLALL	Close all channels and files
\$FFEA	65514	UDTIM	Increment the system clock
\$FFED	65517	SCRORG	Get size of current window
\$FFF0	65520	PLOT	Read and set the X,Y cursor position
\$FFF3	65523	IOBASE	Return the base address of input/output device





Introduction	120
Transient utility commands	121
COPYSYS	121
DATE	121
DEVICE	121
DIR	121
DUMP	123
ED	123
ERASE	125
GENCOM	125
GET	125
HELP	125
HEXCOM	126
INITDIR	126
LIB	126
LINK	127
MAC	129
PATCH	130
PIP	130
PUT	131
RENAME	132
RMAC	132
SAVE	132
SET	132
SETDEF	134
SHOW	134
SID	134
SUBMIT	136
TYPE	136
USER	136
XREF	136
Resident utility commands	137
DIR	137
DIRSYS	137
ERASE	137
RENAME	137
TYPE	137
USER	137
Assignment/logical devices	138
Reserved characters	138
Reserved file types	139
Control codes	139
Line editing with control	140

CP/M PLUS

The CP/M (Control Program for Microprocessors) Plus Version 3.0 operating system within the Commodore 128 is from Digital Research Inc. The system is loaded in to the computer from a disk drive. The software then takes control of the C-128 resources, such as memory, disk drive, and printer. The main component, of this mode is the Basic Disk Operating System (BDOS). CP/M Plus can run hundreds of programs, developed for CP/M 2.2. A large selection of software has been placed in the public domain. Ask your local user's group on how to contact one of the clubs interested in the field.

There are two types of commands used in the operation of CP/M Plus. Transient commands must be loaded from disk before use. Resident Utility commands are built-in the CP/M operating system. The following is a description of both types of commands. All syntax examples use the parameters set forth in the opening chart.

NOTE: <> Optional parameters.
[] Optional parameters, brackets mandatory
() Indicates a range of options, parenthesis must be typed also.
/ Separates choices for one or more options
^ or CTRL Control key.
... May be repeated over and over.
SYS System attribute.
* Wild card, replaces any/all of filename
? Wild card, replaces one char. of filename
sea Search
ins Insert
str String
del Delete

CP/M PLUS COMMANDS

TRANSIENT UTILITY COMMANDS

COPYSYS

FUNCTION: Copies CP/M Plus system to another disk.
FORMAT : COPYSYS

DATE

FUNCTION: Displays/enables setting of DATE and time.
FORMAT : DATE [CONTINUOUS/SET]
DATE [12/24/85 11:30:0]

DEVICE

FUNCTION: Displays or enables setting of DEVICE.
FORMAT : DEVICE <names/values/phys.-dev/log.-dev>
DEVICE log.-dev=phys.-dev<option>
 <,phys.-dev<option>,...>
DEVICE log.-dev = NULL
DEVICE phys.-dev <option>
DEVICE console[page/columns=#/lines=#]

OPTIONS : XON : Signals transmission on
XOFF : Signals transmission off
NOXON : Indicates no protocol
BAUD-RATE: Sets speed of drive. Rates are
as follows;

50,75,110,134,150,300,600,1200,
1800,2400,3600,4800,7200,9600,
19200

DIR

FUNCTION: Displays the names of files and attributes
associated with files.
FORMAT : DIR<d:/filespec/>[options]
DIRSYS<d:/filespec>

OPTIONS : ATT	: Exhibits user-definable keys.
DATE	: Exhibits files with date and time stamps.
DIR	: Exhibits files with DIR assigned to them.
DRIVE = ALL	: Exhibits all files on drive.
DRIVE = (A,B,...)	: Exhibits files on specified drives.
DRIVE = D	: Exhibits files only for specified drive.
EXCLUDE	: Shows files not specified in command line.
FF	: Sends a line-feed to printer if activated by CTRL-P. Otherwise, turns off default paged display.
FULL	: Exhibits everything connected to a file.
LENGTH = n	: Displays n lines before inserting a heading. n = range 5 to 65536
MESSAGE	: Shows names of specified drives and user# it's searching for.
NOPAGE	: Turns off paging.
NOSORT	: Exhibits files as listed in the directory
RO	: Shows files with Read-Only capabilities.
RW	: Shows files with Read-Write capabilities.
SIZE	: Shows filenames and size in kilobytes.
SYS	: Shows files that have SYS specified.
USER = ALL	: Shows all files with user numbers.

USER = n : Shows only files with
user number specified.
USER = (0,1,...): Shows only files with
user numbers specified.

DUMP

FUNCTION: To exhibit contents of a file in HEX/ASCII.
FORMAT : DUMP filename

ED

FUNCTION: Allows creation and editing of disk files.
FORMAT : ED <input-filename[d:/output-filename]>
COMMANDS: nA : Append n lines from disk
file to buffer.
0A : Append from disk file
until buffer is half full
B : Move cursor to top/buffer
-B : Move cursor to bottom of
buffer.
nC : Move cursor n characters
forward.
-nC : Move cursor n characters
backwards.
nD : Delete n characters from
the cursor position.
-nD : Delete n characters be-
fore the cursor position.
E : Save buffer/exit to CP/M3
Fstring(^z) : Search for string.
H : Save buffer, then use new
file as original.
I : Insert mode, use ^Z or
ESCape key to exit.
Istring(^z) : Insert string at cursor
position.
Jse str ^z : Look for string before
ins str^z place to insert new
del to str(^z) string. Then delete up to
string right after.

nK	: Delete n lines from cursor.
-nK	: Delete n line before cursor.
nL	: Move cursor n lines forward.
-nL	: Move cursor n lines backward.
L	: Skip to next line.
nMcommands	: Execute commands n times.
n	: Move cursor forward n lines and display
-n	: Move cursor backwards n lines and display.
n:	: Move cursor to line n.
:ncommands	: Execute command until line n.
Nstring(^z)	: Finds first line with string.
O	: Exit to original file.
nP	: Move cursor forward n lines/display n lines
-nP	: Move cursor backward n lines and display n lines
Q	: Exit to CP/M without saving buffer.
R(^z)	: Read X\$\$\$\$\$\$\$.LIB file into buffer.
Rfilename(^z)	: Read file into buffer.
Sdel str^zins str(^z)	: Deletes old string and inserts new string.
nT	: Displays n lines forward from cursor.
-nT	: Displays n lines backwards from cursor.
T	: Displays current line.
U	: Change to upper case.
-U	: Change to lower case.
V	: Line numbering ON.
-V	: Line numbering OFF.
OV	: Exhibit buffer space left

nW	: Enter n lines to update file.
nX<^z>	: Write or append n lines to X\$\$\$\$\$\$\$.LIB.
nXfilename<^z>	: Write n lines to file or append if previous X command applied to same file
OX<^z>	: Delete LIB file.
OXfilename<^z>	: Delete named file.
nZ	: Wait n seconds.

ERASE

FUNCTION: Deletes one or more files from disk.

FORMAT : ERASE <d:><filename><[CONFIRM]>

ERASE <d:><wildcard>

GENCOM

FUNCTION: Creates a COM file with attached RSX files.

FORMAT : GEMCOM COM-filename RSX-filename...<[LOADER / SCB=(offset,value)]>

GENCOM RSX-filename...<[NULL / SCB=(offset,value)]>

GENCOM filename[SCB=(offset,value)]

GENCOM filename

GET

FUNCTION: Tells CP/M to get input from CONSOLE/file

FORMAT : GET<CONSOLE INPUT FROM>filename <[options]>

GET<CONSOLE INPUT FROM>CONSOLE

OPTIONS : ECHO : Echos input to screen

NO ECHO : Turns off ECHO.

SYSTEM : Takes all SYSTEM input from file.

HELP

FUNCTION: Provides information files either supplied with CP/M or created by the user.

FORMAT : HELP<topic><subtopic1...subtopic8><[NO PAGE / LIST]>

HELP [EXTRACT]

HELP [CREATE]

COMMANDS: CREATE : Builds a revised HELP.HLP from
HELP.DAT

EXTRACT : Creates the HELP.DAT file. So you
can add your help files to the
master list.

LIST : Eliminates extra lines between
headings.

NO PAGE : Disables the paging option.

HEXCOM

FUNCTION: Generates a command file (filename.COM)
from a HEX file.

FORMAT : HEXCOM <d:>filename

INITDIR

FUNCTION: Initializes a disk directory for date and
time stamping.

FORMAT : INITDIR d:

LIB

FUNCTION: This utility is used to create, replace,
delete, select, and append LIB files.

FORMAT : LIB filename <[options]>

LIB filename <[options]>=filename<modifier>

OPTIONS : D : Displays contents of
object modules in ASCII.
I : Creates indexed library
file (filename.IRL).
M : Displays module names.
P : Displays module names and
public variables for
the new file.

MODIFIER: Delete <module=>

Deletes named module.

Replace <module=filename.REL>

Replaces module with filename.REL

Select <1st module-last module>

The selected modules (1st module thru last module) are saved as one file.

Select <module1,module2,...>

The selected modules (module1, module2, ...) are saved as one file.

LINK

FUNCTION: Combines relocatable objec modules such as produced by RMAC and PL/I-80 into a .COM file. These files can access modules in .LIB files.

FORMAT : LINK <d:filename,...>

LINK filename

LINK d:<filename,<[option]>=>filename
<[option]><,...>

OPTIONS : A : Writes temp. data to disk and reduces buffer space.
B : Aligns data segment on page border, puts length of code segment in header (default is .SPR).
Dxxxx : Sets memory address for common and data area.

common and data area.

Gn : Set start address to n.
Lxxxx : Change default load add. of module to xxxx (default = 0100H).
Mxxxx : Define free memory for MP/M modules.
NL : Disable symbol table listing at console.
NR : Do not create a symbol table.

OC	: Create a .COM command file (default).
OP	: Create a .PRL file for execution under MP/M in relocatable segment.
OR	: Create a .RSP file for execution under MP/M.
OS	: Create a .SPR file for execution under MP/M.
Pxxxx	: Change default program load address to xxxx (default = 0100H).
Q	: List symbols table with leading question mark.
S	: Search preceding file as library file.
\$Cd	: Send console messages to d, where d can be one of the following: X=console, Y=printer, and Z=no output (default = X).
\$Id	: Source of files is d, d can be drives A thru P (default = current drive)
\$Ld	: Source of library is d, where d can be drives A thru P (default =
\$Od	: Destination of symbol file, where d can be one of the following: Y=printer, Z=no output or disk drives A thru P(default = same drive as first file in LINK-80 command).
\$Sd	: Destination of symbole file, where d can be one of tje following: Y=printer, Z=no output or disk drives A thru P(default = same drive as first file in LINK-80 command).

MAC

FUNCTION: Reads .ASM file, produces three output files/filetypes of .HEX, .PRN, >SYM (all filenames are the same).

FORMAT : MAC filename <\$options>

OPTIONS : A : source drive for .ASM file (A-O)
H : Destination drive for HEX file (A-O,Z).
L : Source drive for .LIB files called by MACLIB statement.
+L : List input from .LIB files
-L : Turn off list from input .LIB file (default).
+M : Display all macro lines as they are processed.
-M : Disable display of macro lines processed.
*M : Display only hex generated by macro expansions.
P : Destination drive for .PRN file(s) (A-O,X,P,Z)
+Q : Displays LOCAL Symbols in symbols list
-Q : Disable display of LOCAL symbols in list (default)
S : Destination drive for .SYM file(s) (A-O,X,P,Z).
+S : Appends symbol file to print file.
-S : Disables creation of symbol file.
+1 : Produces a pass 1 listing for debugging .PRN file.
-1 : Disables pass 1 listing (default).

PATCH

FUNCTION: Displays or installs patch number n to CP/M system or CP/M command files.

FORMAT : PATCH filename<filetype><n>

PIP

FUNCTION: Performs the following: copy files, combine files, transfer files between drive printer, console, etc..

FORMAT : PIP

PIP d:<[Gn]>=filename<options>

PIP filename<[Gn]>=filename<options><,...>

PIP filename<[Gn]>/d:=filename<options>/d:

OPTIONS :

A	: Copy files not modified since the last copy.
C	: Ask for confirmation before performing copy.
Dn	: Delete characters past column n.
E	: Echo at console.
F	: Disables form-feeds from source data.
Gn	: Tells PIP where to get/place source data (n=user number, range 0 to 15).
H	: Hex data transfer. The console displays error messages.
I	: Ignore 00 records in hex transfer.
L	: Translates upper to lower case in destination file.
N	: Adds line numbers to destination file.
N2	: Adds leading 0's to line #s, inserts a tab after # in destination file.
O	: Tells PIP you are combining object code files.

Pn	: Sets page length, adds page eject every n lines in destination file. (default = 60).
Qs	: Stop copying from source drive after string s. String argument MUST end with ^z.
R	: Read .SYS files.
Ss	: Begin copying at string s from source device. String argument MUST end with ^z.
Tn	: Expanded tabs, positions next character in column dividable by n.
U	: Translates lower to upper case in destination file.
V	: Verify, to compare destination to source.
W	: Writes over files with RO attribute.
Z	: Zero parity bit.

PUT

FUNCTION: Directs console or printer output to one of the following; CONSOLE, PRINTER, or FILE.

FORMAT : PUT CONSOLE<OUTPUT>filename[<option>]
PUT CONSOLE<OUTPUT>CONSOLE
PUT PRINTER<OUTPUT>filename[<option>]
PUT PRINTER<OUTPUT>PRINTER

OPTIONS : ECHO : Echo output to console
(default).
FILTER : Translate control char.
to printable characters.
NO ECHO : Turn echo off.
NO FILTER : Turn filter off (default)
SYSTEM : Write output to specified
file.

RENAME

FUNCTION: Renames a file on disk.

FORMAT : RENAME <new-filename = old-filename>

RMAC

FUNCTION: Assembles files (.ASM) into files (.REL) that can be linked to create .COM files.

FORMAT : RMAC filename <options>

OPTIONS : d : Where d equals destination of output. Drives A thru O
P = printer, X = console,
Z = no output.

Pd : Output for .PRN file, d =
A-O, P, X, Z.

Rd : Output for .REL file, d =
A-O, Z.

Sd : Output for .SYM file, d =
A-O, P, X, Z.

SAVE

FUNCTION: Saves the contents of mem. to a file. Asks for filename, starting/ending add. To use command, first give the SAVE command next run your program that alters memory/loads info into mem. If program exits, SAVE command takes control.

FORMAT : SAVE

SET

FUNCTION: The SET command performs the following; initiates protection/time stamping of files sets file, drive attributes RW, RO, DIR, SYS, label a disk/password protection on disk labels.

FORMAT : SET <d:> [options/option=modifier]
SET filename [options/option=modifier]

PASSWORD

OPTIONS : **PROTECT=ON** : Enables password protection for files on disk.

PROTECT=OFF : Disable password protection for all files on disk

PASSWORD=password : Assigns a password to a file.

PROTECT=READ : Password for read, write copy, rename, delete file

PROTECT=WRITE : Password for write, rename deleting a file.

PROTECT=DELETE : Password for deleting or renaming a file.

PROTECT=NONE : No password needed to modifications to the file

FILE

OPTIONS : **ARCHIVE=ON** : File has been backed-up.

ARCHIVE=OFF : File is not backed-up.

DIR : Assigns .DIR to file.

F1=ON : User-definable file on.

F1=OFF : User-definable file off.

F2=ON : User-definable file on.

F2=OFF : User-definable file off.

F3=ON : User-definable file on.

F3=OFF : User-definable file off.

F4=ON : User-definable file on.

F4=OFF : User-definable file off.

RO : Sets file to Read Only.

RW : Sets file to Read Write.

SYS : Assigns .SYS to file.

TIME-STAMP

OPTIONS : **ACCESS=ON** : Enables recording of last access of file.

ACCESS=OFF : Stops recording of last access of file.

CREATE=ON : Enables recording of creation time of file.

CREATE=OFF : Disables recording of creation time of file.

UPDATE=ON	: Enables recording of last modification of file.
UPDATE=OFF	: Disables recording of last modification.

SETDEF

FUNCTION: To Display/disk, search order, temp. drive, filetype search order.

FORMAT : SETDEF

SETDEF <[DISPLAY/NO DISPLAY/PAGE/NOPAGE]>

SETDEF <d:<,d:<,d:<,d:>>>><[TEMPORARY=d:] /
[ORDER=(type<,type>)]>

SHOW

FUNCTION: Displays information from diskette in drive

FORMAT : SHOW <d:><[SPACE/LABEL/USERS/DIR/DRIVE]>

SID

FUNCTION: Allows debugging of M/L programs.

FORMAT : SID <filename.type><,filename.SYM>

COMMANDS:

As	: Assemble code starting at add. s
Cs<b<,d>>	: Identifies mem. location s for SID, b= value of BC register pair, d = DE register pair.
D<W><s><,e>	: Shows mem. in HEX/ASCII W specifies 16-bit format s = starting address, e = ending address.
Efilename.type	: Load program/symbol table for execution. <,filename.SYM>
E*filename.SYM	: Load symbol table only.
Fs,e,d	: Fill mem. Where s = starting add., e = end add., d = 8-bit data item.

G<s><,a<,b>>	: Begin execution at s (starting address), a = 1st temporary breakpoint, b = 2nd temp. breakpoint.
GO	: Exits SID by a warm start
H	: Shows symbols/addresses in hex.
Ha	: Displays hex, dec., ASCII values of a.
Ha,b	: Displays hex sum, difference of a and b.
Icommand tail	: Enter CCP command.
L<s><,e>	: List 8080 instructions, s = starting address, e = ending address.
Ms,e,n	: Move a section of memory, s = starting address, e = ending address, n = new starting address.
P<p<,x>>	: Pass point set, reset, display. Where p = permanent breakpoint address x = value of pass counter
Rfilename<,x>	: Read code/symbols from file, x = offset to each address.
S<W>s	: Set memory values, s = address where value is sent, W = 16-bit word.
T<n<,x>>	: Turn trace on, n = # of program steps to trace, x = utility entry address
T<W><n<,x>>	: Turn trace on without call, W = SID no trace subroutines. n = # of program steps to trace, x = utility entry address
U<W><n<,x>>	: Trace off, W = SID not to trace subroutines, n = # of program steps to trace x = utility entry address

V : Shows value of next available location in memory.

Wfilename<,s,e> : Write a segment of memory to filename, s = starting address, e = ending add..

X<f><r> : Examine or alter CPU. f = flag bit for one of the following; C, E, I, M, or Z. r is one of the following registers; A, B, D, H, P, or S.

SUBMIT

FUNCTION: Executes a group of commands in a .SUB file. Information may be passed to a .SUB file by using the argument option. Up to nine pieces of info maybe passed in this manner.

FORMAT : SUBMIT <filename><argument1,...argument9>

TYPE

FUNCTION: Displays contents of a ASCII file to screen

FORMAT : TYPE <filename[options]>>

OPTIONS : PAGE : Turn paging on.

NO PAGE : Turn paging off.

USER

FUNCTION: Enables setting of current USER number.

User numbers are in the range of 0 thru 15.

FORMAT : USER <number>

XREF

FUNCTION: Provides a cross-reference of variables used in a program. XREF requires the .PRN and .SYM files created by MAC or RMAC for the specified file.

FORMAT : XREF <d:> filename <option>

OPTIONS : \$P : Sends output to printer.

RESIDENT UTILITY COMMANDS

DIR

FUNCTION: Display files on current disk drive, except files marked with .SYS.

FORMAT : DIR

DIRSYS

FUNCTION: Displays all files marked with .SYS

FORMAT : DIRSYS

ERASE

FUNCTION: Deletes a file from the current disk drive, releases the allocated space for that file.

FORMAT : ERASE filename

RENAME

FUNCTION: Renames a file on the current disk drive.

FORMAT : RENAME <new-filename=old-filename>

TYPE

FUNCTION: Displays the contents of a ASCII file on the screen.

FORMAT : TYPE filename

USER

FUNCTION: Enables setting of current USER number.

User numbers are in the range of 0 thru 15.

FORMAT : USER <number>

ASSIGNMENT/LOGICAL DEVICES

The minimum equipment requirement, for the C-128 CP/M 3.0 system is a console (keyboard plus monitor) and a disk drive. Other devices may be added to the group as your system grows. The following chart, lists the names of logical devices and their assignment.

LOGICAL DEVICE NAME	DEVICE TYPE	PHYSICAL DEVICE ASSIGNMENT
CONIN:	Console input	Keyboard
CONOUT:	Console output	80 column screen
AUXIN:	Auxillary input	Null
AUXOUT:	Auxillary output	Null
LST:	List output	PTR1 or PTR2

RESERVED CHARACTERS

The characters in the following table are used by the CP/M operating system. They are not to be used except where indicated.

CHARACTER	DEFINITION
< \$, !] > []	File specification delimiters
tab space	" " "
carriage return	" " "
:	Drive delimiter in file specification
.	Filetype delimiter in file spec.
;	Password delimiter in file spec.
;	Comment delimiter at start of a command line
* ?	Wildcard characters
<> & !] -	Option list delimiters
[]	List option delimiters, global and local options
()	Delimiters for multiple modifiers inside [], options that have modifiers
/ \$	Option delimiters in a command line

RESERVED FILE TYPES

A three character file type, may be added to the name of a file. A number of categories, have been defined by the developer and are listed below.

FILE TYPE	DESCRIPTION
ASM	Assembler source file
BAS	BASIC source program
COM	Z80 or equivalent machine language program
HEX	Output file from MAC (used by HEXCOM)
HLP	HELP message file
\$\$\$	Temporary file
PRN	Print file from MAC or RMAC
REL	Output file from RMAC (used by LINK)
SUB	List of commands, executed by SUBMIT
SYM	Symbol file from MAC, RMAC or LINK
SYS	System file

CONTROL CODES

CP/M Plus 3.0 supports a number, of different CONTROL key functions. To implement one of the following operations, press the CONTROL key plus a designated key at the same time.

CONTROL	Plus UPPER cursor key, to scroll right to view entire screen in 40 column mode.
CONTROL S	Pause scrolling when viewing the screen.
CONTROL Q	To restart scroll after it has been stopped by CONTROL command.
CONTROL P	This commands send all output from the console to the printer. Press the keys again to stop output to the printer.
CONTROL DIR	To list a directory of files from the disk to the printer.
CONTROL H	Use this command or INST/DEL key to correct typing errors.
CONTROL C	To end program run/reset the disk system.

LINE-EDITING with CONTROL

KEY	ACTION
CONTROL A	Moves the cursor one space to the left.
CONTROL B	Moves the cursor to start of line, if cursor is at start, moves to the end.
CONTROL E	Forces a physical return without command to CP/M 3.0. Allows user to enter code and not be executed until user is ready.
CONTROL F	Cursor on character to the right.
CONTROL G	Moves character on the right one space left, deletes left character as it moves.
CONTROL H	Deletes character to left of cursor, characters to right move one space left.
CONTROL I	Moves cursor to the next tab stop.
CONTROL J	Has the same effect as a carriage return or CONTROL M
CONTROL K	Deletes to the end of line from cursor.
CONTROL M	Has the same effect as a carriage return or CONTROL J.
CONTROL R	Duplicates current line of code.
CONTROL U	Deletes current line of code, prints # at cursor position.
CONTROL W	Recalls previous line.
CONTROL X	Deletes all characters left of cursor.





1541 Specifications	142
Diskette Format	142
Sector Distribution	143
1541 BAM	144
1541 Directory Header	144
Directory Format	144
Error Checking	145
Change Device Number	145
DOS Wedge Commands	146
1541 File Types	146
1541 File Formats	147
PRG File Format	148
SEQ File Format	148
REL File Format	148
REL Side Sector Format	149
File Commands	150
SEQ File Commands	150
REL File Commands	151
Direct Access Commands	152
Internal Disk Commands	152
GCR Conversion	154
BIT Values	154
BASIC 2.0 Commands	155
1541 DOS Error Messages	157
1541 RAM	160
Stack Area	166
Parser Tables	168
Output Buffers	169
Data Buffers	169
Serial I/O	170
Disk Controller	170

1541 SPECIFICATIONS

Drives per unit	1
Heads per drive	1

FORMATTED STORAGE	
Capacity per unit	170 K
Max # SEQ files	168 K
Max # REL files	167 K
Disk system buffer	2 K

DISK FORMAT	
Tracks	35
Sectors per track	17-21
Bytes per sector	256
Blocks free per unit	664

ACCESS TIMES (milli-seconds)	
Track-to track	30
Average track	360
RPM	300

PHYSICAL DIMENSIONS		ELECTRICAL	
Height (inches)	3	Power (watts)	35
Width (inches)	7	Voltage	110-120
Depth (inches)	13		VAC. 60 hz.
Weight (pounds)	10		

DISKETTE FORMAT

The diskettes you purchase are totally blank. This is so they are generic to many types of disk drive. Each buyer must format the disks for their own unit. When in C-64 mode, the command is the NEW command. As the format process takes place, the diskette is sectioned into tracks and made ready to receive information. A directory (DIR) and Block Availability Map (BAM) are created on track 18 of the diskette. The directory is much like the table of contents of a book. Everytime

a file is recorded to the disk, it is recorded in the directory. When the disk is accessed, the drive looks at the DIR as a road map of the diskette. The BAM is another kind of map of the diskette. It indicates the blocks that have been used, and ones that are vacant.

The command to format a diskette is as follows:

```
OPEN 15,device #,15,"NEW drive#:diskette name,ID"  
CLOSE 15
```

The diskette name is your choice, up to 16 character spaces. The ID can be any two characters, so long as they do not form a key-word. It is best, that all of your ID's be different, so as to not confuse the disk drive. If you were to attempt to move a file from one disk to another, and both had the same ID's, the disk drive would not know where the file should go.

The following charts show the layout of the diskette after the format command has taken place. Track one is located at the outer edge of the diskette, track 35 at the center.

Number of tracks per diskette	35
Number of blocks per diskette	683
Number of blocks reserved (BAM/DIR)	19
Number of blocks available to user	664

SECTOR DISTRIBUTION

TRACK NUMBER	NUMBER of SECTORS
1 - 17	21
18 - 24	19
25 - 30	18
31 - 35	17

1541 BAM
(Block Availability Map)

Track 18, Sector 0		
BYTES	CONTENTS	DEFINITION
0	18	Pointer to the track of the next directory entry.
1	01	Pointer to the sector of the next directory entry.
2	65	ASCII character A indicating 1541 format.
3		Flag, double-sided disk not used in 1541 mode
4 - 143		Block Availability Map

1541 DIRECTORY HEADER

Track 18, Sector 0		
BYTES	CONTENTS	ACTION
144-159		Diskette name padded with shifted spaces.
160-161	160	Shifted spaces.
162-163		Diskette ID
164	160	Shifted space.
165-166	50/65	DOS version/format type.
167-170	160	Shifted spaces.
171-255	00	Unused.

DIRECTORY FORMAT

The identifying name, type, location, and length of individual files on a disk, are listed on track 18 as shown below.

Track 18, Sector 1

BYTE	DEFINITION
0,1	Track and sector of next directory block.
2-31	File number one
34-63	File number two
66-95	File number three
98-127	File number four
130-159	File number five
162-191	File number six
194-223	File number seven
226-255	File number eight

ERROR CHECKING

When a disk error happens, the light on the front of the disk drive will flash red. To identify the problem, type in the following:

EXAMPLE:

```
10 OPEN15,8,15
20 INPUT#15,E1$,E2$,E3$,E4$
30 PRINT E1$,E2$,E3$,E4$
40 CLOSE15
```

CHANGE DEVICE NUMBER

To make a temporary change of device number, type the following:

```
PRINT#15,"M-W"CHR$(119)CHR$(0)CHR$(2)
CHR$(new drive#+32)
```

DOS WEDGE COMMANDS

The DOS SUPPORT or WEDGE program offers the user many short cuts to disk operation. First LOAD and RUN the program. You may now use the commands found below in-place of conventional BASIC 2.0 disk commands.

@ \$ or > \$	To display the directory
@ or >	Disk status, reads error channel
^ filename	Load and automatic run
/ filename	Load program
% filename	Load file at it's own address
filename	Save file
@ NO:disk name,ID	To format a disk
@ NO:disk name	To rename the directory
@ SO:name of file	Scratch or remove file
@ VO	To validate the disk
@ RO:old name=0:new name	To rename a file

1541 FILE TYPES

There are several different types of files that may be recorded to disk. Each appears in the directory by a specific name. If you go a bit further and examine the first two bytes of each file, the second identifies the file type in hexadecimal.

FILE TYPE	DIRECTORY APPEARS	HEX	ASCII
Scratched	not shown	\$00	0
Deleted (DEL)	DEL	\$80	128
Sequential (SEQ)	SEQ	\$81	129
Program (PRG)	PRG	\$82	130
User (USR)	USR	\$83	131
Relative (REL)	REL	\$84	132
Open DEL	not shown	\$00	0
Open SEQ	*SEQ	\$01	1
Open PRG	*PRG	\$02	2

Open USR	*USR	\$03	3
Open REL	not apply	\$04	4
DEL @ replacement	DEL	\$A0	160
SEQ @ replacement	SEQ	\$A1	161
PRG @ replacement	PRG	\$A2	162
USR @ replacement		\$A3	163
REL @ replacement	not apply	\$A4	164
Locked DEL	DEL<	\$C0	192
Locked SEQ	SEQ<	\$C1	193
Locked PRG	PRG<	\$C2	194
Locked USR	USR<	\$C3	195
Locked REL	REL<	\$C4	196

1541 FILE FORMATS

The three primary types of files stored upon disk are PROGRAM (PRG), SEQUENTIAL (SEQ), and RELATIVE (REL). The PRG format is generally used for programs that do not store user input to the disk. The SEQ data file is an efficient method of information storage. This type of files are referred to as sequential, because pieces of information are stored one after another. When data is stored in this manner, the complete file must be brought into the computer memory in entirety to be adjusted. The REL data file is well suited for extensive amounts of information and allows for fast access of data. Individual records from within the file, may be brought into memory. Each type has it's advantages. As each file type handles data in a different manner, the disk format must also be unique to the type of file.

PROGRAM (PRG) FILE FORMAT

BYTE	DEFINITION
SECTOR 1	
0,1	Track and sector of next block in file 1.
2,3	Load address of the program
4-255	Program information as in memory.
REMAINING SECTORS	
0,1	Track and sector of next block in file 1.
2-255	Program information as in memory.
LAST SECTOR	
0,1	Null-followed by # of data bytes in sector
2-??	Last bytes of information as in memory. End of file is marked by 3 zero bytes.

SEQUENTIAL (SEQ) FILE FORMAT

BYTE	DEFINITION
ALL EXCEPT LAST SECTOR	
0,1	Track and sector of next SEQ data block.
2-255	254 bytes of data.
LAST SECTOR	
0,1	Null-followed by # of data bytes in sector
2-??	Last bytes of data.

RELATIVE (REL) FILE FORMAT

BYTE	DEFINITION
DATA BLOCK	
0,1	Track and sector of next data block.
2-255	254 bytes of data. The first byte of an unused record contains \$FF followed by \$00 to the end of record. Partial records are completed with zeros.

RELATIVE FILE, SIDE SECTOR FORMAT

0,1	Track and sector of next side sector block
2	Side sector number (0-5).
3	Record length.
4-5	Track & sector pointer of first (#0) side sector.
6-7	Track & sector pointer of second (#1) side sector.
8-9	Track & sector pointer of third (#2) side sector.
10-11	Track & sector pointer of fourth (#3) side sector.
12-13	Track & sector pointer of fifth (#4) side sector.
14-15	Track & sector pointer of sixth (#5) side sector.
16-255	Track & sector pointers to 120 data blocks

FILE COMMANDS

Data files store information to be used by a program. SEQuential or RELative files, save the information to a diskette. Both are generic in that they may be read by many different programs. For an example, data that is saved by a word processor, may be stored, as a SEQ file and read by several brands of program.

SEQ files are more efficient in the use of disk space than REL files. Data stored in SEQ files is stored in order received, one after the other. Where as, with a REL file, you can tell the disk drive, where to place the information. To retrieve stored information, the entire SEQ file must be moved into memory. Access to a single record, within a file is possible with a REL file. The pros and cons go on, choice is dependent on the intended use.

The commands below, are used to create SEQ, and REL files.

SEQ FILE COMMANDS

APPEND 350 OPEN#1,8,1,"data file,s,a"
Open a SEQ file for appending to.

GET# 385 GET#1,a\$
Used to read 1 byte from SEQ file set up for read.

INPUT# 390 INPUT#1,a\$
Used to read 1 record from SEQ file set up for read.

M mode 395 open1,8,2,"data 1,s,M"
Enables recovery of data from open SEQ file.

PRINT# 380 PRINT#1,a\$
Writes to already open SEQ file set for write.

READ 360 open#1,8,1,"data file,s,R"
Open a SEQ file for reading.

WRITE 370 open#1,8,1,"date file,s,W"
Open a SEQ file for writing.

REL FILE COMMANDS

OPEN 350 OPEN#1,"data",1[record len],D0,U8
Open a REL file or create one.

GET# 370 GET#1,a\$
Used to read 1 byte from a REL file.

INPUT# 375 INPUT#1,a\$
Used to read 1 record from a REL file.

P 395 print#15,"P"+chr\$(2)+chr\$(rl)+chr\$(rh)
Positions the drive at the right record.
NOTE: The P command should be used twice.

PRINT# 400 PRINT#1,a\$
Used to write to an opened REL file.

DIRECT ACCESS COMMANDS

The DIRECT ACCESS commands read or write data to specific sectors. These commands are entirely controlled by the user. These are excellent when used in utility programs to view or alter locations on a diskette.

BLOCK-ALLOCATE 390 print#15,"B-A";0;1;1
Marks a sector as used.

BLOCK-FREE 400 print#15,"B-F";0;1;1
Used to free a sector.

BLOCK-READ 360 print#15,"B-R";2;0;10;12
Moves 1 block of data into selected buffer.

BLOCK-WRITE 370 print#15,"B-W";2;0;10;12
Writes 1 block of data to the selected disk sector.

BUFFER-POINTER 380 print#15,"B-P";2;1
Points to where next READ/WRITE begin in disk buffer

OPEN# 350 OPEN#2,8,2,"#"
Open a channel and assign any buffer.

U1 360 print#15,"U1";2;0;10;12
Moves 1 block of data into selected buffer. More reliable than the 'BLOCK-READ' command.

U2 370 print#15,"U2";2;0;10;12
Writes 1 block of data to the selected disk sector. More reliable than the 'BLOCK-WRITE' command.

INTERNAL DISK COMMANDS

The following commands can alter routines that reside within the 1541 disk drive. An accomplished programmer can write machine language programs directly into disk memory. A memory map in the back of this section identifies many locations useful with these commands.

BLOCK-EXECUTE 420 print#15,"B-E";3;18;0
Rarely used, allows loading a sector into disk memory and execution.

MEMORY-EXECUTE 410 print#15,"M-E"
 chr\$(L add)chr\$(H add)
Allows execution of any routine in disk memory, RAM or ROM. L = low memory add., H = high memory add.

MEMORY-READ 390 print#15,"M-R"
 chr\$(L add)chr\$(H add)[chr\$(bytes)]
Allows selection of which byte/bytes to read from disk memory.

MEMORY-WRITE 400 PRINT#15,"M-W"
 chr\$(L add)chr\$(H add)
Allows writing up to 34 bytes at once into disk mem.

User Commands	Function
U0	Restores default user jump table.
U1 or UA	Block read replacement.
U2 or UB	Block write replacement.
U3 or UC	Jump to \$0500.
U4 or UD	Jump to \$0503.
U5 or UE	Jump to \$0506.
U6 or UF	Jump to \$0509.
U7 or UG	Jump to \$050C.
U8 or UH	Jump to \$050F.
U9 or UI	Jump to (\$FFFA) reset tables.
U: or UJ	Power up vector.

EXAMPLE: 430 print#15,"Ucharacter";

GCR CONVERSION

GCR is the method in which data is magnetically written to the diskette. Encoding involves breaking up each 8 bit byte (binary) into two 4-bit nibbles. The 5-bit equivalent for each nibble can then be found by looking at the table below. The resulting 10 bits are stored in two consecutive memory locations.

HEX	DECIMAL	BINARY	GCR
\$0000	0	0000	01010
\$0001	1	0001	01011
\$0002	2	0010	10010
\$0003	3	0011	10011
\$0004	4	0100	01110
\$0005	5	0101	01111
\$0006	6	0110	10110
\$0007	7	0111	10111
\$0008	8	1000	01001
\$0009	9	1001	11001
\$000A	10	1010	11010
\$000B	11	1011	11011
\$000C	12	1100	01101
\$000D	13	1101	11101
\$000E	14	1110	11110
\$000F	15	1111	10101

BIT VALUES

BIT	DECIMAL	HEX	BIT	DECIMAL	HEX
0	1	\$0001	8	256	\$0100
1	2	\$0002	9	512	\$0200
2	3	\$0003	10	1024	\$0400
3	4	\$0004	11	2048	\$0800
4	5	\$0005	12	4096	\$1000
5	6	\$0006	13	8192	\$2000
6	7	\$0007	14	16384	\$4000
7	8	\$0008	15	32768	\$8000

BASIC 2.0 COMMANDS

The following are the commands from BASIC 2.0 that pertain to the 1541 disk drive.

CLOSE PRINT#15:CLOSE15
To close channel that has been opened to receive data

CMD CMD1
To redirect screen output to a disk file.

COPY PRINT#15,"C:new file=old file"
To make a copy (duplicate) of a file.

GET #
To read one byte from a record.

INITIALIZE PRINT#15,"I"
Returns the disk drive to power-up condition.

INPUT# GET#5,a\$
Reads a whole record from the disk.

LOAD

LOAD"file name",8	Load a BASIC program.
LOAD"***",8	Load first program of DIR.
LOAD"file name",8,1	Load machine language program.
LOAD"file name",8,3	Automatic load and run.
LOAD"\$",8	Loads directory of disk.
LOAD"A\$",8	Loads contents of variable (A\$)
LOAD"T*",8	Loads first file beginning with 'T'.
LOAD"JAN*",8	Loads first file beginning with 'JAN'.

NEW

PRINT#15,"N:name,ID" To NEW a disk (format).
PRINT#15,"N:name" To clear a directory.

OPEN OPEN 15,8,15
Opens a file or channel for input or output.

PRINT# PRINT#5,p\$
To output data to file or channel.

RENAME PRINT#15,"R:new name=old name"
To rename a file on a diskette.

SAVE SAVE"file name",8
To SAVE a file in memory to disk.

SAVE/REPLACE SAVE"@:file name",8
To SAVE a new program and replace old.

SCRATCH PRINT#15,"S:name"
To erase (SCRATCH) a file.

VALIDATE PRINT#15,"V"
To re-organize, collect scattered blocks and make
space available on the diskette.

VERIFY VERIFY"file name",8
To compare a saved file with one in memory

1541 DOS ERROR MESSAGES

Errors are evaluated by the DOS when come upon during a read or write function. What follows are error messages that are generated if an error is found. Don't panic, evaluate the situation carefully. Some errors may also be caused by equipment failure. Check all of the cable connections. Also check your electrical outlet grounded, it MUST be grounded.

ERROR #	MESSAGE and EFFECT
0	Everything is OK, no error exists
1	This is not an error, denotes file scratched
2-19	Unused error messages, ignore
20	READ ERROR, block header of requested block not found. Cause, either illegal block number or destroyed header.
21	READ ERROR, sync mark missing. Cause, read-write head out of alignment, no disk in drive, disk not formatted.
22	READ ERROR, data failed to match against a preset data block identifier.
23	READ ERROR, checksum error in data block did not match checksum read from the disk.
24	READ ERROR, byte decoding error. Hardware error created due to bad bit pattern in the data byte.
25	WRITE ERROR, write verify error. There is a mismatch between data just written and data in RAM when compared.
26	WRITE PROTECT ON, a write protect tab is over the notch in the side of the disk. Remove to write to the disk.
27	READ ERROR, checksum error in header. There is an error in the header of the requested data block.
28	WRITE ERROR, long data block. A sync mark has not been found, cause bad format.
29	DISK ID MISMATCH, a comparison of ID in a header block has been made, it does not match the master copy of the disk ID.

ERROR	MESSAGE and EFFECT
30	SYNTAX ERROR, the DOS cannot understand the command sent to the command channel.
31	SYNTAX ERROR, the DOS cannot recognize the command. Command must begin in the first space
32	SYNTAX ERROR, invalid command. A command has been sent that is longer than the allowable 58 characters.
33	SYNTAX ERROR, invalid file name. Pattern matching may not be used with the OPEN or SAVE command.
34	SYNTAX ERROR, no file given. The file name has been left out of the command.
39	SYNTAX ERROR, invalid command. Command is not recognized by the DOS.
50	RECORD NOT PRESENT, the disk has been ask by an INPUT#, OR GET# to read past the last record.
51	OVERFLOW IN RECORD, PRINT# statement exceeds the defined size.
52	FILE TOO LARGE, record position within a REL file indicates that disk overflow will result.
60	WRITE FILE OPEN, attempt made to open a file that is already open.
61	FILE NOT OPEN, attempt made to access a file that is not open.
62	FILE NOT FOUND, the file does not exist on the indicated drive. Common cause, in-correct file name.
63	FILE EXISTS, attempt made to save a file by a name that already exists on the disk.
64	FILE TYPE MISMATCH, requested file type does not match type recorded on the directory.
65	NO BLOCK, block has been previously allocated.
66	ILLEGAL TRACK AND SECTOR, attempt made to access a track or sector that does not exist in the format being used.
67	ILLEGAL SYSTEM TRACK OR SECTOR, indicates an illegal track or sector.

ERROR #	MESSAGE and EFFECT
70	NO CHANNEL, requested channel is not available , channel may be in use. SEQ files are limited to three files open at one time. Direct access channels are limited to six.
71	DIRECTORY ERROR, BAM (block availability map) does not match internal count.
72	DISK FULL, all blocks have been used or the directory has reach capacity (144 entries).
73	DOS MISMATCH, The diskette has been written by another DOS. You may read the disk but not write to it.
74	DRIVE NOT READY, there is no diskette in the drive or the drive is not on line.

1541 RAM

HEX	LABEL	FUNCTION
\$0000	JOBS	Job Que: Buffer 0
\$0001		Buffer 1
\$0002		Buffer 2
\$0003		Buffer 3
\$0004		Buffer 4
\$0005		Buffer 5
\$0006	HDRS	Job headers: Buffer 0-low
\$0007		Buffer 0-high
\$0008		Buffer 1-low
\$0009		Buffer 1-high
\$000A		Buffer 2-low
\$000B		Buffer 2-high
\$000C		Buffer 3-low
\$000D		Buffer 3-high
\$000E		Buffer 4-low
\$000F		Buffer 4-high
\$0010		Buffer 5-low
\$0011		Buffer 5-high
\$0012	DSKID	Master copy disk ID: Drive 0
\$0013		Drive 0
\$0014		Un-used drive 1
\$0015		Un-used drive 1
\$0016	HEADER	Image last header: ID byte 1
\$0017		ID byte 2
\$0018		Track
\$0019		Sector
\$001A		Checksum
\$001B	ACTJOB	Controllers active job
\$001C	WPSW	WP change flag: Drive 0
\$001D		Un-used Drive 1
\$001E	LWPT	Last state of WP switch: Drive 0
\$001F		Un-used Drive 1
\$0020	DRVST	Drive current status: Drive 0
\$0021		Un-used
\$0022	DRVTRK	Drive track number: Drive 0
\$0023		Un-used Drive 1
\$0024	STAB	Storage table GCR conversion

HEX	LABEL	FUNCTION
\$0025		Continued: storage table GCR
\$0026		Continued: storage table GCR
\$0027		Continued: storage table GCR
\$0028		Continued: storage table GCR
\$0029		Continued: storage table GCR
\$002A		Continued: storage table GCR
\$002B		Continued: storage table GCR
\$002C		Continued: storage table GCR
\$002D		Continued: storage table GCR
\$002E	SAVPNT	Temporary save pointer location
\$002F		
\$0030	BUFPNT	Pointer to active buffer
\$0031		Cont: pointer to active buffer.
\$0032	HDRPNT	Header pointer: track
\$0033		Header pointer: sector
\$0034	GCRPNT	GCR pointer
\$0035	GCRERR	Indicates GCR decode error
\$0036	BYTCNT	Byte counter GCR/binary conver.
\$0037	BITCNT	Bit counter
\$0038	BID	Data block ID character
\$0039	HBID	Header block ID character
\$003A	CHKSUM	Data or header checksum
\$003B	HINIB	Un-used
\$003C	BYTE	Un-used
\$003D	DRIVE	Drive number, \$00=1541
\$003E	CDRIVE	Current active drive number
\$003F	JOB	Current job number
\$0040	TRACC	Track-internal storage location
\$0041	NXTJOB	Next job
\$0042	NXTRK	Next track to move head to
\$0043	SECTR	Sectors per track for formatting
\$0044	WORK	Temporary work storage location
\$0045	JOB	Temporary storage of job type
\$0046	CTRAK	Un-used
\$0047	DBID	Data block ID
\$0048	ACLTIM	Timer for acceleration of head
\$0049	SAVSP	Temporary save of stack pointer
\$004A	STEPS	# steps to move head to track
\$004B	TMP	Temporary storage location

HEX	LABEL	FUNCTION
\$004C	CSECT	Current sector
\$004D	NEXTS	Next sector
\$004E	NXTBF	Pointer to next GCR source buffer
\$004F	NXTPNT	Ptr/next byte location in buffer
\$0050	GCRFLG	GCR/binary flag in active buffer
\$0051	FTNUM	Current format flag
\$0052	BTAB	Binary table:GCR/binary work area
\$0053		Continued: binary table
\$0054		Continued: binary table
\$0055		Continued: binary table
\$0056	GTAB	GCR table: GCR/binary work
\$0057		Continued: GCR table
\$0058		Continued: GCR table
\$0059		Continued: GCR table
\$005A		Continued: GCR table
\$005B		Continued: GCR table
\$005C		Continued: GCR table
\$005D		Continued: GCR table
\$005E	AS	# of steps to used to accel. head
\$005F	AF	Acceleration factor
\$0060	ACLSTP	Steps to go before complete
\$0061	RSTEPS	Number of run steps left
\$0062	NXTST	Pointer to stepping rtn-\$FA05
\$0063		Continued from \$0062
\$0064	MINSTP	Minimum steps required to accel.
\$0065	VNMI	Indirect for NMI-\$EB22
\$0066		
\$0067	NMIFLG	NMI in progress flag
\$0068	AUTOFG	Auto drive initialization flag
\$0069	SECINC	Sector increment for sequential
\$006A	RECCNT	Error recovery counter
\$006B	USRJMP	User jump table pointer-\$FFEA
\$006C		Continued from \$006B
\$006D	BMPNT	Bit map pointer
\$006E		Continued from \$006D
\$006F	T0	Temporary work space
\$0070	T1	Temporary work space
\$0071	T2	Temporary work space
\$0072	T3	Temporary work space

HEX	LABEL	FUNCTION
\$0073	T4	Temporary work space
\$0074		Temporary work space
\$0075	IP	Indirect pointer variable
\$0076		
\$0077	LSNADR	Listen address: device # + \$20
\$0078	TLKADR	Talker address: device # + \$40
\$0079	LSNACT	Active listener flag
\$007A	TLKACT	Active talker flag
\$007B	ADRSED	Addressed flag
\$007C	ATNPND	Attention pending flag
\$007D	ATNMOD	6502 in attention mode
\$007E	PRGTRK	Last program accessed
\$007F	DRVNUM	Current drive number
\$0080	TRACK	Current track number
\$0081	SECTOR	Current sector number
\$0082	LINDX	Logical index
\$0083	SA	Current secondary address
\$0084	ORGSA	Original secondary address
\$0085	DATA	Temporary data byte
\$0086	R0	Temporary work area
\$0087	R1	Temporary work area
\$0088	R2	Temporary work area
\$0089	R3	Temporary work area
\$008A	R4	Temporary work area
\$008B	RESULT	Result of multiply/divide rtns
\$008C		Result location
\$008D		Result location
\$008E		Result location
\$008F	ACCUM	Remainder of multiply/divide rtns
\$0090		Accumulator
\$0091		Accumulator
\$0092		Accumulator
\$0093		Accumulator
\$0094	DIRBUF	Pointer to directory buffer
\$0095		Continued from \$0094
\$0096	ICMD	IEEE command in:un-used
\$0097	MYPA	MY PA flag: UN-USED
\$0098	CONT	Serial bit counter
\$0099	BUFTAB	Buffer byte pointer:buffer 0 low

HEX	LABEL	FUNCTION
\$009A		:buffer 0 high
\$009B		:buffer 1 low
\$009C		:buffer 0 high
\$009D		:buffer 2 low
\$009E		:buffer 0 high
\$009F		:buffer 3 low
\$00A0		:buffer 0 high
\$00A1		:buffer 4 low
\$00A2		:buffer 0 high
\$00A3		:CMD buffer low
\$00A4		:CMD buf. high
\$00A5		:error buf. low
\$00A6		:error buf. high
\$00A7	BUFO	Inactive flags for buffers
\$00A8		Cont.: inactive flags for buffers
\$00A9		Cont.: inactive flags for buffers
\$00AA		Cont.: inactive flags for buffers
\$00AB		Cont.: inactive flags for buffers
\$00AC		Cont.: inactive flags for buffers
\$00AD		Cont.: inactive flags for buffers
\$00AE	BUF1	Active flags for buffers
\$00AF		Cont.: active flags for buffers
\$00B0		Cont.: active flags for buffers
\$00B1		Cont.: active flags for buffers
\$00B2		Cont.: active flags for buffers
\$00B3		Cont.: active flags for buffers
\$00B4		Cont.: active flags for buffers
\$00B5	RECL	Low record # find Relative file
\$00B6		Cont.: low record # find REL file
\$00B7		Cont.: low record # find REL file
\$00B8		Cont.: low record # find REL file
\$00B9		Cont.: low record # find REL file
\$00BA		Cont.: low record # find REL file
\$00BB	RECH	High record # to find REL file
\$00BC		Cont.: high rec. # find REL file
\$00BD		Cont.: high rec. # find REL file
\$00BE		Cont.: high rec. # find REL file
\$00BF		Cont.: high rec. # find REL file
\$00C0		Cont.: high rec. # find REL file

HEX	LABEL	FUNCTION
\$00C1	NR	Next record table
\$00C2		Cont.: next record table
\$00C3		Cont.: next record table
\$00C4		Cont.: next record table
\$00C5		Cont.: next record table
\$00C6		Cont.: next record table
\$00C7	RS	Relative record size table
\$00C8		Cont.: relative record size table
\$00C9		Cont.: relative record size table
\$00CA		Cont.: relative record size table
\$00CB		Cont.: relative record size table
\$00CC		Cont.: relative record size table
\$00CD	SS	Side sector table
\$00CE		Cont.: side sector table
\$00CF		Cont.: side sector table
\$00D0		Cont.: side sector table
\$00D1		Cont.: side sector table
\$00D2		Cont.: side sector table
\$00D3	F1PTR	File stream 1 pointer
\$00D4	RECTR	1st byte wanted from REL record
\$00D5	SSNUM	Side sector number of REL file
\$00D6	SSIND	Index to side sector
\$00D7	RELPTR	Ptr/1st byte wanted in REL file
\$00D8	ENTSEC	Sector of directory entries
\$00D9		Cont.: sector of dir. entries
\$00DA		Cont.: sector of dir. entries
\$00DB		Cont.: sector of dir. entries
\$00DC		Cont.: sector of dir. entries
\$00DD	ENTIND	Index of directory entries
\$00DE		Cont.: index of directory entries
\$00DF		Cont.: index of directory entries
\$00E0		Cont.: index of directory entries
\$00E1		Cont.: index of directory entries
\$00E2	FILDRV	Default flag, drive number
\$00E3		Cont.: default flag, drive number
\$00E4		Cont.: default flag, drive number
\$00E5		Cont.: default flag, drive number
\$00E6		Cont.: default flag, drive number
\$00E7	PATTYP	Pattern,renew,closed-flags,type

HEX	LABEL	FUNCTION
\$00E8		Cont.: pat.,renew,close-flag,type
\$00E9		Cont.: pat.,renew,close-flag,type
\$00EA		Cont.: pat.,renew,close-flag,type
\$00EB		Cont.: pat.,renew,close-flag,type
\$00EC	FILTYP	Channel file type
\$00ED		Cont.: channel file type
\$00EE		Cont.: channel file type
\$00EF		Cont.: channel file type
\$00F0		Cont.: channel file type
\$00F1		Cont.: channel file type
\$00F2	CHNRDY	Channel status
\$00F3		Continued: channel status
\$00F4		Continued: channel status
\$00F5		Continued: channel status
\$00F6		Continued: channel status
\$00F7		Continued: channel status
\$00F8	EOIFLG	Temporary EOI
\$00F9	JOBNUM	Current job number
\$00FA	LRUTBL	Least recently used buffer table
\$00FB		Cont.:least recent used buf. tab.
\$00FC		Cont.:least recent used buf. tab.
\$00FD		Cont.:least recent used buf. tab.
\$00FE		Cont.:least recent used buf. tab.
\$00FF	NODRV	No drive flag: drive 0
\$0100		drive 1, un-used
\$0101	DSKVER	DOS version from 18,0
\$0102		Cont.: DOS version from 18,0
\$0103	ZPEND	Un-used

STACK AREA \$0104-\$01FF

\$0200-	CMDBUF	Command buffer
\$0229		
\$022A	CMDNUM	Command number
\$022B-	LINTAB	Secondary address: index table
\$020D		
\$023E-	CHNDAT	Channel data byte
\$0243		
\$0244-	LSTCHR	Channel last character ptr.

HEX	LABEL	FUNCTION
\$0249		
\$024A	TYPE	Active file type
\$024B	STRSIZ	String size in command buffer
\$024C	TEMPSA	Temporary secondary address
\$024D	CMD	Temporary job command
\$024E	LSTSEC	Last sector
\$024F	BUFUSE	Buffer allocation
\$0251	MDIRTY	Bam dirty flag: drives 0 and 1
\$0252		Cont.: bam dirty flag
\$0253	ENTFND	Directory entry found flag
\$0254	DIRLST	Directory listing flag
\$0255	CMDWAT	Command waiting flag
\$0256	LINUSE	Logical index (lindx) use word
\$0257	LBUSED	Last buffer used
\$0258	REC	Record size
\$0259	TRKSS	Track of side sector
\$025A	SECSS	Sector of side sector
\$025B	LSTJOB	Last job
\$025C		Continued: last job
\$025D		Continued: last job
\$025E		Continued: last job
\$025F		Continued: last job
\$0260	DSEC	Sector of directory entry
\$0261		Cont.: sector of directory entry
\$0262		Cont.: sector of directory entry
\$0263		Cont.: sector of directory entry
\$0264		Cont.: sector of directory entry
\$0265		Cont.: sector of directory entry
\$0266	DIND	Index of directory entry
\$0267		Cont.: index of directory entry
\$0268		Cont.: index of directory entry
\$0269		Cont.: index of directory entry
\$026A		Cont.: index of directory entry
\$026B		Cont.: index of directory entry
\$026C	ERWORD	Error word for recovery
\$026D	ERLED	Error led mask for flashing
\$026E	PRGDRV	Last program drive
\$026F	PRGSEC	Last program sector
\$0270	WLINDX	Write logical index

HEX	LABEL	FUNCTION
\$0271	RLINDX	Read logical file
\$0272	NBTEMP	Number blocks temporary
\$0273		Cont.: number blocks temporary
\$0274	CMDSIZ	Command string size
\$0275	CHAR	Character under parser
\$0276	LIMIT	Pointer limit in compare
\$0277	F1CNT	File stream 1 count
\$0278	F2CNT	File stream 2 count
\$0279	F2PTR	File stream 2 pointer

PARSER TABLES \$027A-\$0289

\$027A	F1LTBL	Filename pointer
\$027B		Continued: filename pointer
\$027C		Continued: filename pointer
\$027D		Continued: filename pointer
\$027E		Continued: filename pointer
\$027F		Continued: filename pointer
\$0280	F1LTRK	1st file link (track)
\$0281		Continued: 1st file link (track)
\$0282		Continued: 1st file link (track)
\$0283		Continued: 1st file link (track)
\$0284		Continued: 1st file link (track)
\$0285	F1LSEC	1st file link (sector)
\$0286		Continued: 1st file link (sector)
\$0287		Continued: 1st file link (sector)
\$0288		Continued: 1st file link (sector)
\$0289		Continued: 1st file link (sector)
\$028A	PATFLG	Pattern presence flag
\$028B	IMAGE	File stream image
\$028C	DRVCNT	Number of drive searches
\$028D	DRVFLG	Drive search flag
\$028E	LSTDRV	Last drive without error
\$028F	FOUND	Found flag in directory searches
\$0290	DIRSEC	Directory sector
\$0291	DELSEC	Sector of 1st available entry
\$0292	DELIND	Index of 1st available entry
\$0293	LSTBUF	=0 if last block
\$0294	INDEX	Current index in buffer

HEX	LABEL	FUNCTION
\$0295	FILCNT	Counter of file entries
\$0296	TYPFLG	Match by type flag
\$0297	MODE	Active mode (r,w)
\$0298	JOBRTN	Job return flag
\$0299	EPTR	Pointer for recovery
\$029A	TOFF	Total track offset
\$029B	UBAM	Last bam update pointer
\$029C		Cont.: last bam update pointer
\$029D	TBAM	Track number of bam image
\$029E		Cont.: track number of bam image
\$02A1-	BAM	BAM images
\$02B0		Continued: BAM images
OUTPUT BUFFERS \$02B1-\$02F8		
\$02B1-	NAMBUF	Directory buffer
\$02D4		
\$02D5-	ERRBUF	Error message buffer
\$02F8		
\$02F9	WBAM	Don't write bam flag
\$02FA	NDBL	Blocks free low byte, drive 0 & 1
\$02FB		Cont.: block free low byte
\$02FC	NDBH	Blocks free high byte, drive 0 & 1
\$02FD		Cont.: blocks free high byte
\$02FE	PHASE	Phase offset
\$02FF		Continued: phase offset
DATA BUFFERS \$0300-\$06FF		
\$0300-	BUFF0	Buffer #0
\$03FF		Continued: Buffer #0
\$0400-	BUFF1	Buffer #1
\$04FF		Continued: Buffer #1
\$0500-	BUFF2	Buffer #2
\$05FF		Continued: Buffer #2
\$0600-	BUFF3	Buffer #3
\$06FF		Continued: Buffer #3
\$0620	CNT	Error counter: decrements from 10

HEX	LABEL	FUNCTION
\$0620	FMTVAR	Format variable
\$0621	NUM	Number between sync & non-sync
\$0623	TRYS	Number of tries in verify
\$0624	TRAL	
\$0625		
\$0626	DTRCK	Distance to track
\$0627	REMDR	Remainder of size
\$0628	SECT	Sector number counter

SERIAL I/O 6522 \$1800-\$180F

\$1800	PB	Data port b
\$1801	PA1	Data port a: UN-USED
\$1802	DDRB1	Data direction register port b
\$1803	DDRA1	Data direction register port a
\$1804	T1LC1	Timer 1 low count
\$1805	T1HC1	Timer 1 high counter
\$1805	TIMER1	Timer 1 counter
\$1806	T1LL1	Timer 1 low latch
\$1807	T1HL1	Timer 1 high latch
\$1808	T2LC1	Timer 2 low counter
\$1809	T2HC1	Timer 2 high counter
\$180A	SR1	Shift register
\$180B	ACR1	Auxillary control register
\$180C	PCR1	Peripheral control register
\$180D	1FR1	Interrupt flag register
\$180E	1ER1	Interrupt enable register

DISK CONTROLLER 6522 \$1C00-\$1C0F

\$1C00	DSKCNT	Disk controller I/O control line bit 0: step head in bit 1: step head out bit 2: motor on (1), off (0) bit 3: drive active LED on/off bit 4: write protect sense bit 5: density select 0 bit 6: density select 1 bit 7: sync detect
--------	--------	---

HEX	LABEL	FUNCTION
\$1C01	DATA2	Data port a
\$1C02	DDRB2	Data direction for port b
\$1C03	DDRA2	Data direction for port a
\$1C04	T1LC2	Timer 1 low counter
\$1C05	T1HC2	Timer 1 high counter
\$1C06	T1LL2	Timer 1 low counter
\$1C07	T1HL2	Timer 1 high counter
\$1C08	T2LL2	Timer 2 low counter
\$1C09	T2LH2	Timer 2 high counter
\$1C0A	SR2	Shift register
\$1C0B	ACR2	Auxillary control register
\$1C0C	PCR2	Peripheral control register
\$1C0D	1FR2	Interrupt flag register
\$1C0E	1ER2	Interrupt enable register







1571 DISK DRIVE

YELLOW

1571 Specifications	174
MFM Format	175
GCR Format	175
1571 Diskette Format	176
1571 Sector Distribution	177
1571 BAM	178
1571 Directory Header	178
DOS Shell	179
Language Choice	179
40 OR 80 Column	179
Main Menu	179
Disk/Printer Setup	179
Run a Program	180
Format a Disk	180
Cleanup a Disk	180
Copy a Disk	181
Copy Files	181
Delete Files	182
Restore Files	182
Rename Files	182
Reorder Directory	182
1571 File Types	183
1571 File Formats	184
PRG File Format	184
SEQ File Format	184
REL File Format	185
REL Side Sector Format	185
File Commands	186
SEQ File Commands	186
REL File Commands	187
Internal Disk Commands	188
Direct Access Commands	189
Related Kernal Routines	190
Error Checking	190
Memory Locations	191
Change Device Number	191
BASIC 7.0 Disk Commands	192
1571 DOS Error Messages	193

1571 SPECIFICATIONS

	SINGLE SIDED	DOUBLE SIDED
Drives per unit	1	1
Heads per drive	2	2
FORMATTED STORAGE		
Capacity per unit	170 K	341 K
Max # files per diskette	144	144
Max # SEQ file size	164 K	329 K
Max # REL file size	163 K	163 K
DISK FORMAT		
Tracks	35	70
Sectors per track	17-21	17-21
Bytes per sector	256	256
Blocks free per unit	664	1328
PHYSICAL DIMENSIONS		ELECTRICAL
Height (inches)	3	Power (watts) 35
Width (inches)	7	Voltage 110-120
Depth (inches)	13	VAC. 60 hz.
Weight (pounds)	10	

The 1571 disk drive, can record data in two different formats, MFM and GCR. The MFM format, is double density, this allows access to CPM software libraries of other computers. The GCR format, is used in both the C-128 and C-64 modes.

MFM FORMAT

Total unformatted capacity	250000 bytes per side
Total formatted capacity	
Sector size 128	133120 bytes per side
Sector size 256	163840 bytes per side
Sector size 512	184320 bytes per side
Sector size 1024	204800 bytes per side
Maximum tracks per side	40 x 2 sides
Sectors per track	
Sector size 128	26
Sector size 256	16
Sector size 512	9
Sector size 1024	5

1571 GCR FORMAT

	SINGLE SIDED	DOUBLE SIDED
Unformatted capacity	252019 bytes	252019*2 bytes
Formatted capacity	174848 bytes	349696 bytes
Max SEQ file size	168656 bytes	337312 bytes
Max REL file size	167132 bytes	167132 bytes
Records per file	65535	65535
Max files per disk	144	144
Tracks per disk	35	70
Max sectors per track	17-21	17-21
Max Sectors per disk	683	1366
Reserved sectors	19	38
Available sectors	664	1328
Max bytes per sector	256	256

1571 DISKETTE FORMAT

The diskettes you purchase are totally blank. This is so they are generic to many types of disk drive. Each buyer must format new disks for their own disk unit. The command to format a diskette is called the HEADER command. As the format process takes place, the diskette is divided into tracks and made ready to receive information. A directory (DIR) and Block Availability Map (BAM) are created on track 18 of the diskette. A directory is much like an index or table of contents of a book. Everytime a file is added to the disk, it is recorded in the directory. When a disk is accessed the drive looks at the DIR as a road map to find it's way around the diskette. The BAM is another kind of map of the diskette. It shows blocks that have been used, and ones that are vacant.

The command to format a diskette is as follows:

C-128 mode:

```
HEADER"diskette name"Id,Ddrive#[,Udevice#]
```

The diskette name is your choice, up to 16 character spaces. The ID can be any two characters, so long as they do not form a key-word. It is best, that all of your ID's be different, so as to not confuse the disk drive. If you were to attempt to move a file from one disk to another, and both had the same ID's, the disk drive would not know where the file should go.

The following charts show the layout of the diskette after the format command has taken place. Track one is on the outer edge of the disk and track 35 at the center. The back side of a double sided diskette, is a continuation of the front (tracks 35 to 70).

	SINGLE SIDED	DOUBLE SIDED
Number of tracks per diskette	35	70
Number of blocks per diskette	683	1366
Number of blocks reserved (BAM/DIR)	19	38
Number of blocks available to user	664	1328

1571 SECTOR DISTRIBUTION

TRACK NUMBER	SECTOR NUMBER	SINGLE SIDED	DOUBLE SIDED
1 to 17	0 through 20	21	21
18 to 24	0 through 18	19	19
25 to 30	0 through 17	18	18
31 to 35	0 through 16	17	17
36 to 52	0 through 20		21
53 to 59	0 through 18		19
60 to 65	0 through 17		18
66 to 70	0 through 16		17

1571 BAM

(Block Availability Map, Track 18, Sector 0)

BYTES	CONTENTS	DEFINITION
0	18	Pointer to the track of the next directory entry.
1	01	Pointer to the sector of the next directory entry.
2	65	ASCII character A indicating 1541 format.
3		Flag, double-sided disk \$80 = Double-sided, \$00 = Single-sided.
4 - 143		Block Availability Map
221 - 237		Block Availability Map
238		Usable SECTORS TRACK 53=0
239 - 255		SECTORS ARE PRE-ALLOCATED. Block Availability map

Track 53, Sector 0

0 - 104		BAM for tracks 36 thru 70.
105 - 255	00	Unused

1571 DIRECTORY HEADER

	Track 18, Sector 0	
BYTES	CONTENTS	ACTION
144-159		Diskette name padded with shifted spaces.
160-161	160	Shifted spaces.
162-163		Diskette ID
164	160	Shifted space.
165-166	50/65	DOS version/format type.
167-170	160	Shifted spaces.
171-220	00	Unused.

DOS SHELL

The program DOS shell can be found on the disk titled 'TEST/DEMO DISKETTE' that was provided with the 1571. The shell provides a substitute method for entree of disk drive commands. To load the DOS shell; insert the diskette while the computer is turned off. When the computer is switched on (or reset), the program will automatically load and run. To enter or exit the DOS shell, press the F1 key. The file is menu driven for quick identification of operations.

LANGUAGE CHOICE

All information may be displayed in one of four languages: English, French, German, and Italian. To adopt a preference, press the SPACE BAR as the one you want appears or wait until the next is shown.

40 OR 80 COLUMN MODE

The shell may be used in either 40 or 80 column mode. The choice must be made before a selection is made from the main menu.

MAIN MENU

To make a selection from the menu, use the CRSR key to move the cursor up or down to the wanted function. Then press the SPACE BAR to activate the operation.

DISK/PRINTER SETUP

There are two logical drives (A and B), that live within the shell. They may be set to represent a single disk drive, two individual drive units, or a dual drive.

EXAMPLE: If you have one disk drive the setting would be A = device #8 and B = drive #0. These are the settings when you enter the shell. If you have a single drive unit, everything is ready to go. When using two single drives, or a dual drive, the device number of one unit must be changed.

To make the desired changes, move the cursor over the proper choice, then depress the SPACE BAR. Next press F7 to lock in a setting. The F7 key will cancel your choice, if you should change your mind. The printer device #'s may also be changed in the same manner.

RUN A PROGRAM

CRSR keys	To choose a program
SPACE BAR	To lock in choice
F5	To un-select a program
F7	To load and run choosen file
STOP key	To cancel function
F3	To print out file list

FORMAT A DISK

Place the diskette to be formatted in the disk drive. Next press the SPACE BAR. The shell will check to see the disk is blank or contains files you might wish to save. If everything is the way you intend, enter the new disk name and press RETURN. The shell generates a new disk ID, or you may enter one, by adding a coma, plus a two digit number after the name of the disk.

CLEANUP A DISK

The 'CLEANUP A DISK' command, is the same as the VALIDATE function. The Block Availability Map (BAM) is recalculated, counting only closed files or programs. All un-used sectors, or improperly closed files, are made ready for use.

It is good practice to use this function when a diskette has had many alterations or many files have been scratched. Often the blocks free shown will increase. increase.

COPY A DISK

This function, is for making a entire copy of a disk. All files will be moved from the old disk, to the new one.

DUAL DRIVE

CRSR keys	To choose from which drive to copy from/to
SPACE BAR	To lock in choice of drives
STOP key	To cancel function
SPACE BAR or F5	To re-enter, if an error is made

SINGLE DRIVE

WHEN THIS FUNCTION IS USED, ALL FILES IN MEMORY WILL BE LOST. Press the STOP or F5 key cancel or the SPACE BAR to continue. The copy is made by swapping disks in the drive, from one to four times, as prompted by the computer. The new disk is automatically formatted and a new ID assigned. To fall back to the previous disk swap, press either the SPACE BAR or F5. The STOP key will cancel the copy function completely.

COPY FILES

This mode enables the user to move single files from one disk to another. If the destination diskette is not formatted, you may do so during this process. The shell automatically checks to see, if there is space for the new file on your diskette. You will be warned if there are not enough blocks free.

CRSR keys	To choose a file
SPACE BAR	To lock in choice of file
F5	To un-select a file
F7	To continue function
STOP key	To cancel function

DELETE FILES

CRSR keys	To choose a file
SPACE BAR	To lock in choice
F7	Double check by the computer answer 'Y' or 'N'
F5	To restart file selection
STOP	To cancel function

RESTORE FILES

This function can be used to restore files.

CRSR keys	To choose a file or file type
SPACE BAR	To lock in choice of file
F5	To un-select a file
F7	To continue function

RENAME FILES

This function allows the user to rename files.

CRSR keys	To choose a file to rename
SPACE BAR	To lock in choice of file
F5	To return to file selection
F7	To complete rename function

REORDER DIRECTORY

This function allows the user to rearrange or alphabetize the directory of a disk.

CRSR keys	To choose a file or move
SPACE BAR	To lock in choice of file or lock in move
F7	To end file selection
F5	To restart file selection

1571 FILE TYPES

There are several different types of files that may be recorded to disk. Each appears in the directory by a specific name. If you go a bit further and examine the BAM (block allocation map, track 18, sector 1), byte two identifies the file type in hexadecimal.

FILE TYPE	DIRECTORY APPEARS	HEX	ASCII
Scratched	not shown	\$00	0
Deleted (DEL)	DEL	\$80	128
Sequential (SEQ)	SEQ	\$81	129
Program (PRG)	PRG	\$82	130
User (USR)	USR	\$83	131
Relative (REL)	REL	\$84	132
Open DEL	not shown	\$00	0
Open SEQ	*SEQ	\$01	1
Open PRG	*PRG	\$02	2
Open USR	*USR	\$03	3
Open REL	not apply	\$04	4
DEL @ replacement	DEL	\$A0	160
SEQ @ replacement	SEQ	\$A1	161
PRG @ replacement	PRG	\$A2	162
USR @ replacement		\$A3	163
REL @ replacement	not apply	\$A4	164
Locked DEL	DEL<	\$C0	192
Locked SEQ	SEQ<	\$C1	193
Locked PRG	PRG<	\$C2	194
Locked USR	USR<	\$C3	195
Locked REL	REL<	\$C4	196

1571 FILE FORMATS

The three primary types of files stored upon disk are PROGRAM (PRG), SEQUENTIAL (SEQ), and RELATIVE (REL). The PRG format is generally used for programs that do not store user input to the disk. The SEQ file is an efficient method of information storage. This type of files are referred to as sequential, because pieces of information are stored in them one after another. The typical REL file is well suited for extensive amounts of information and allows for fast access of data. As they all handle data in a different manner, the disk format must also be unique to the type of file.

PROGRAM (PRG) FILE FORMAT

BYTE	DEFINITION
SECTOR 1	
0,1	Track and sector of next block in file 1.
2,3	Load address of the program
4-255	Program information as in memory.
REMAINING SECTORS	
0,1	Track and sector of next block in file 1.
2-255	Program information as in memory.
LAST SECTOR	
0,1	Null-followed by # of data bytes in sector
2-??	Last bytes of information as in memory. End of file is marked by 3 zero bytes.

SEQUENTIAL (SEQ) FILE FORMAT

BYTE	DEFINITION
ALL EXCEPT LAST SECTOR	
0,1	Track and sector of next SEQ data block.
2-255	254 bytes of data.

LAST SECTOR

0,1	Null-followed by # of data bytes in sector
2-??	Last bytes of data. any other bytes are garbage.

RELATIVE (REL) FILE FORMAT

BYTE

DEFINITION

DATA BLOCK

0,1	Track and sector of next data block.
2-255	254 bytes of data. The first byte of an unused record contains \$FF followed by \$00 to the end of record. Partial records are completed with zeros.

SIDE SECTOR BLOCK

0,1	Track and sector of next side sector block
2	Side sector number (0-5).
3	Record length.
4-5	Track & sector of first (#0) side sector.
6-7	Track & sector of second (#1) side sector.
8-9	Track & sector of third (#2) side sector.
10-11	Track & sector of fourth (#3) side sector.
12-13	Track & sector of fifth (#4) side sector.
14-15	Track & sector of sixth (#5) side sector.
16-255	Track & sector pointers to 120 data blocks

FILE COMMANDS

Data files store information to be used by a program. SEQuential or RELative files, save the information to a diskette. Both are generic in that they may be read by many different programs. For an example, data that is saved by a word processor, may be stored, as a SEQ file and read by several brands of program.

SEQ files are more efficient in the use of disk space than REL files. Data stored in SEQ files is stored in order recieved, one after the other. Where as, with a REL file, you can tell the disk drive, where to place the information. To retrieve stored information, the entire SEQ file must be moved into memory. Access to a single record, within a file is possible with a REL file. The pros and cons go on, choice is dependent on the intended use.

The commands below, are used to create SEQ, and REL files.

SEQ FILE COMMANDS

APPEND 350 OPEN#1,8,1,"data file,s,a"
Open a SEQ file for appending to.

GET# 385 GET#1,a\$
Used to read 1 byte from SEQ file set up for read.

INPUT# 390 INPUT#1,a\$
Used to read 1 record from SEQ file set up for read.

M mode 395 open1,8,2,"data 1,s,M"
Enables recovery of data from open SEQ file.

PRINT# 380 PRINT#1,a\$
Writes to already open SEQ file set for write.

READ 360 open#1,8,1,"data file,s,R"
Open a SEQ file for reading.

WRITE 370 open#1,8,1,"date file,s,W"
Open a SEQ file for writing.

REL FILE COMMANDS

DOPEN 350 DOPEN#1,"data",1[record len],D0,U8
Open a REL file or create one.

GET# 370 GET#1,a\$
Used to read 1 byte from a REL file.

INPUT# 375 INPUT#1,a\$
Used to read 1 record from a REL file.

PRINT# 400 PRINT#1,a\$
Used to write to an opened REL file.

RECORD 360 RECORD#1,20,0:RECORD#1,20,0
Positions the drive at RECORD #20. Note: the RECORD
command should be used twice

INTERNAL DISK COMMANDS

BLOCK-EXECUTE 420 print#15,"B-E";3;18;0

This rarely used command allows loading of a sector into disk memory and execution of it.

MEMORY-EXECUTE 410 print#15,"M-E"
chr\$(L add)chr\$(H add)

Allows execution of any routine in disk memory, RAM or ROM. L = low memory add., H = high memory add.

MEMORY-READ 390 print#15,"M-R"
chr\$(L add)chr\$(H add)[chr\$(bytes)]

Allows you to select which byte or bytes to read in from disk memory.

MEMORY-WRITE 400 PRINT#15,"M-W"
chr\$(L add)chr\$(H add)

Allows you to write up to 34 bytes at a time into disk memory.

User Commands

Function

U0		Restores default user jump table.	
U1	or	UA	Block read replacement.
U2	or	UB	Block write replacement.
U3	or	UC	Jump to \$0500.
U4	or	UD	Jump to \$0503.
U5	or	UE	Jump to \$0506.
U6	or	UF	Jump to \$0509.
U7	or	UG	Jump to \$050C.
U8	or	UH	Jump to \$050F.
U9	or	UI	Jump to (\$FFFA) reset tables.
U:	or	UJ	Power up vector.

EXAMPLE: 430 print#15,"Ucharacter";

DIRECT ACCESS COMMANDS

The DIRECT ACCESS commands read or write data to specific sectors. These commands are entirely controlled by the user. These are excellent when used in utility programs to view or alter locations on a diskette.

BLOCK-ALLOCATE 390 print#15,"B-A";0;1;1
Marks a sector as used.

BLOCK-FREE 400 print#15,"B-F";0;1;1
Used to free a sector.

BLOCK-READ 360 print#15,"B-R";2;0;10;12
Moves 1 block of data into selected buffer.

BLOCK-WRITE 370 print#15,"B-W";2;0;10;12
Writes 1 block of data to the selected disk sector.

BUFFER-POINTER 380 print#15,"B-P";2;1
Points to where next READ/WRITE begin in disk buffer

OPEN# 350 OPEN#2,8,2,"#"
Open a channel and assign any buffer.

U1 360 print#15,"U1";2;0;10;12
Moves 1 block of data into selected buffer. More reliable than the 'BLOCK-READ' command.

U2 370 print#15,"U2";2;0;10;12
Writes 1 block of data to the selected disk sector. More reliable than the 'BLOCK-WRITE' command.

RELATED KERNAL SUBROUTINES

The following list of subroutines are from the kernal ROM of the C-128. These may be of useful when writing programs that interact with the disk drive.

ADDRESS	LABEL	FUNCTION
\$FFBA	SETLFS	Set logical, first & second add.
\$FFBD	SETNAM	Set length & address of filename
\$FFC0	OPEN	Open a logical file
\$FFC3	CLOSE	Close a logical file
\$FFC6	CHKIN	Select a channel for input
\$FFC9	CHKOUT	Select a channel for output
\$FFCC	CLRCHN	Clear all channels & restore default I/O
\$FFCF	CHRIN	Get byte from current input device
\$FFD2	CHROUT	Output a byte to current output device

ERROR CHECKING

When a disk error happens, the light on the front of the disk drive will flash green. To identify the problem, type the following:

PRINT DS\$

An error message will be displayed to the screen, the error will be cleared from disk memory. This command may be used to read the error channel at any time.

PRINT DS

DS (Disk Status numeric value only) The value returned will be zero if the last disk access was OK.

MEMORY LOCATIONS

The following are memory locations from the 1571.

0000-00FF	Zero page work area, job que, variables
0100-01FF	GCR overflow area/stack (1571 mode BAM side one)
0200-02FF	Command buffer, parser, tables, variables
0300-07FF	5 data buffers, 0-4, one is used for BAM
1800-65C22A	Serial, controller ports
1C00-65C22A	Controller ports
8000-FFE5	32K byte ROM, DOS/controller routines
FFE6-FFFF	JMP table, user command vectors

CHANGE DEVICE NUMBER

HARDWARE CHANGE

On the back of the 1571 disk drive, are two switches. Below is a chart that shows the different settings to change the device number of the drive.

LEFT	RIGHT	DEVICE #
UP	UP	8
DOWN	UP	9
UP	DOWN	10
DOWN	DOWN	11

SOFTWARE CHANGE

To make a temporary change of device number, type the following:

```
PRINT#15,"U0>" + CHR$(n)
```

The letter may be a number from 8 to 30. The change is effective until changed again or the 1571 is reset.

BASIC 7.0 DISK COMMANDS

The commands listed below are used with the 1571 disk drive in C-128 mode.

COMMAND	USE
APPEND	Append data to file
BLOAD	Load a binary file starting at specific memory location
BOOT	Load and execute program
BSAVE	Save a binary file to a specified memory location
CATALOG	Display contents of dir. on monitor
CMD	Redirect screen output to disk file
COLLECT	Free inaccessible disk space
CONCAT	Concatenate two data files
COPY	Copy files between devices
DCLEAR	Clear open channels on disk drives
DCLOSE	Close logical disk file
DIRECTORY	Display contents of dir. on monitor
DLOAD	Load a BASIC program from disk
DOPEN	Open disk file for read/write
DSAVE	Save a BASIC program to disk
DVERIFY	Compare file in mem to file on disk
GET#	Read one byte from a record
HEADER	To format a disk, make ready to use
INITIALIZE	Returns drive to power-up condition
INPUT#	Read whole record from disk
PRINT#	Output data to a file
RECORD	Position REL file pointers
RENAME	Change name of a file on disk
RUN	Execute BASIC program from disk
SCRATCH	Erase file(s) from a disk
VERIFY	Compare file in mem to one on disk

1571 DOS ERROR MESSAGES

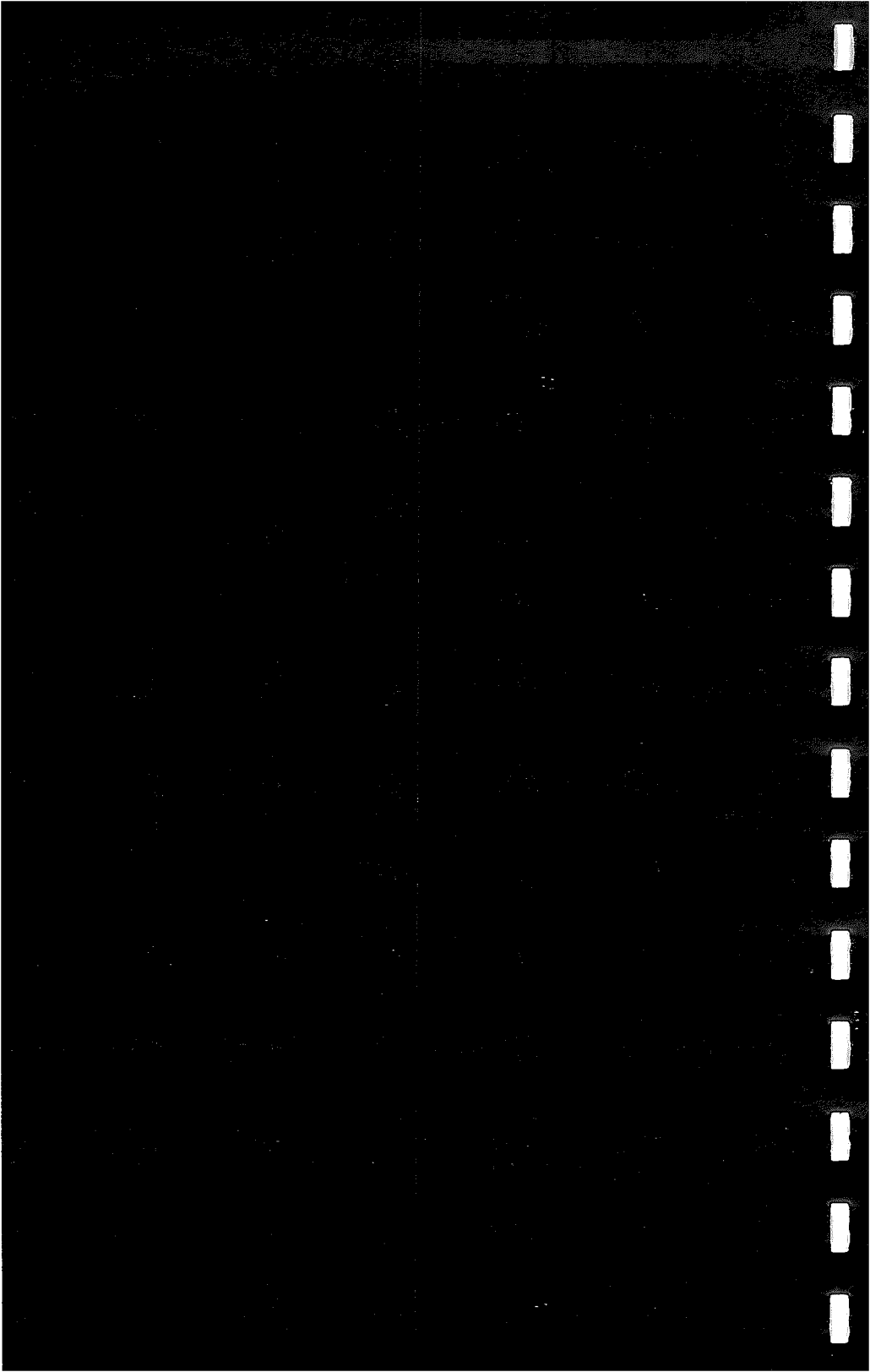
Errors are evaluated by the DOS when encountered during a read or write function. The error messages that follow, may be generated if an error is found. Don't panic, evaluate the situation carefully. Some errors may also be caused by equipment failure. Check cable connections and be sure the system is grounded.

ERROR #	MESSAGE and EFFECT
0	Everything is OK, no error exists
1	This is not an error, denotes file scratched
2-19	Unused error messages, ignore
20	READ ERROR, block header of requested block not found. Cause, either illegal block number or destroyed header.
21	READ ERROR, sync mark missing. Cause, read-write head out of alignment, no disk in drive, disk not formatted.
22	READ ERROR, data failed to match against a preset data block identifier.
23	READ ERROR, checksum error in data block did not match checksum read from the disk.
24	READ ERROR, byte decoding error. Hardware error created due to bad bit pattern in the data byte.
25	WRITE ERROR, write verify error. There is a mismatch between data just written and data in RAM when compared.
26	WRITE PROTECT ON, a write protect tab is over the notch in the side of the disk. Remove to write to the disk.
27	READ ERROR, checksum error in header. There is an error in the header of the requested data block.
28	WRITE ERROR, long data block. A sync mark has not been found, cause bad format.
29	DISK ID MISMATCH, a comparison of ID in a header block has been made, it does not match the master copy of the disk ID.

ERROR	MESSAGE and EFFECT
30	SYNTAX ERROR, the DOS cannot understand the command sent to the command channel.
31	SYNTAX ERROR, the DOS cannot recognize the command. Command must begin in the first space
32	SYNTAX ERROR, invalid command. A command has been sent that is longer than the allowable 58 characters.
33	SYNTAX ERROR, invalid file name. Pattern matching may not be used with the OPEN or SAVE command.
34	SYNTAX ERROR, no file given. The file name has been left out of the command.
39	SYNTAX ERROR, invalid command. Command is not recognized by the DOS.
50	RECORD NOT PRESENT, the disk has been ask by an INPUT#, OR GET# to read past the last record.
51	OVERFLOW IN RECORD, PRINT# statement exceeds the defined size.
52	FILE TOO LARGE, record position within a REL file indicates that disk overflow will result.
60	WRITE FILE OPEN, attempt made to open a file that is already open.
61	FILE NOT OPEN, attempt made to access a file that is not open.
62	FILE NOT FOUND, the file does not exist on the indicated drive. Common cause, in-correct file name.
63	FILE EXISTS, attempt made to save a file by a name that already exists on the disk.
64	FILE TYPE MISMATCH, requested file type does not match type recorded on the directory.
65	NO BLOCK, block has been previously allocated.
66	ILLEGAL TRACK AND SECTOR, attempt made to access a track or sector that does not exist in the format being used.
67	ILLEGAL SYSTEM TRACK OR SECTOR, indicates an illegal track or sector.

ERROR #	MESSAGE and EFFECT
70	NO CHANNEL, requested channel is not available , channel may be in use. SEQ files limited to three open at a time. Direct access channels limited to six.
71	DIRECTORY ERROR, BAM (block availability map) does not match internal count.
72	DISK FULL, all blocks have been used or the directory has reached limit (144 entries).
73	DOS MISMATCH, The diskette has been written by another DOS. You may read the disk but not write to it.
74	DRIVE NOT READY, there is no diskette in the drive or drive is not on.





GENERAL

ORANGE

Numeric Conversion Chart	
Decimal-Hex-Octal-Binary	198
ASCII Conversion Chart	205
Multiplication Table	209
HEX Addition Table	210
HEX Multiplication Table	210
Super Help Chart	211
MCS6510 Instruction Set	218
Symbols & Operators	220
Screen Format Tables	221
80 Column Screen	221
40 Column Screen	222
25 Row Screen Format	223
Standard Abbreviations	224
Cassette Commands	229

NUMERIC CONVERSION TABLE

DECIMAL	HEX	OCTAL	BINARY
00000	\$0000	0	00000000
00001	\$0001	1	00000001
00002	\$0002	2	00000010
00003	\$0003	3	00000011
00004	\$0004	4	00000100
00005	\$0005	5	00000101
00006	\$0006	6	00000110
00007	\$0007	7	00000111
00008	\$0008	10	00001000
00009	\$0009	11	00001001
00010	\$000A	12	00001010
00011	\$000B	13	00001011
00012	\$000C	14	00001100
00013	\$000D	15	00001101
00014	\$000E	16	00001110
00015	\$000F	17	00001111
00016	\$0010	20	00010000
00017	\$0011	21	00010001
00018	\$0012	22	00010010
00019	\$0013	23	00010011
00020	\$0014	24	00010100
00021	\$0015	25	00010101
00022	\$0016	26	00010110
00023	\$0017	27	00010111
00024	\$0018	30	00011000
00025	\$0019	31	00011001
00026	\$001A	32	00011010
00027	\$001B	33	00011011
00028	\$001C	34	00011100
00029	\$001D	35	00011101
00030	\$001E	36	00011110
00031	\$001F	37	00011111
00032	\$0020	40	00100000
00033	\$0021	41	00100001
00034	\$0022	42	00100010
00035	\$0023	43	00100011
00036	\$0024	44	00100100

DECIMAL	HEX	OCTAL	BINARY
00037	\$0025	45	00100101
00038	\$0026	46	00100110
00039	\$0027	47	00100111
00040	\$0028	50	00101000
00041	\$0029	51	00101001
00042	\$002A	52	00101010
00043	\$002B	53	00101011
00044	\$002C	54	00101100
00045	\$002D	55	00101101
00046	\$002E	56	00101110
00047	\$002F	57	00101111
00048	\$0030	60	00110000
00049	\$0031	61	00110001
00050	\$0032	62	00110010
00051	\$0033	63	00110011
00052	\$0034	64	00110100
00053	\$0035	65	00110101
00054	\$0036	66	00110110
00055	\$0037	67	00110111
00056	\$0038	70	00111000
00057	\$0039	71	00111001
00058	\$003A	72	00111010
00059	\$003B	73	00111011
00060	\$003C	74	00111100
00061	\$003D	75	00111101
00062	\$003E	76	00111110
00063	\$003F	77	00111111
00064	\$0040	100	01000000
00065	\$0041	101	01000001
00066	\$0042	102	01000010
00067	\$0043	103	01000011
00068	\$0044	104	01000100
00069	\$0045	105	01000101
00070	\$0046	106	01000110
00071	\$0047	107	01000111
00072	\$0048	110	01001000
00073	\$0049	111	01001001
00074	\$004A	112	01001010
00075	\$004B	113	01001011

DECIMAL	HEX	OCTAL	BINARY
00076	\$004C	114	01001100
00077	\$004D	115	01001101
00078	\$004E	116	01001110
00079	\$004F	117	01001111
00080	\$0050	120	01010000
00081	\$0051	121	01010001
00082	\$0052	122	01010010
00083	\$0053	123	01010011
00084	\$0054	124	01010100
00085	\$0055	125	01010101
00086	\$0056	126	01010110
00087	\$0057	127	01010111
00088	\$0058	130	01011000
00089	\$0059	131	01011001
00090	\$005A	132	01011010
00091	\$005B	133	01011011
00092	\$005C	134	01011100
00093	\$005D	135	01011101
00094	\$005E	136	01011110
00095	\$005F	137	01011111
00096	\$0060	140	01100000
00097	\$0061	141	01100001
00098	\$0062	142	01100010
00099	\$0063	143	01100011
00100	\$0064	144	01100100
00101	\$0065	145	01100101
00102	\$0066	146	01100110
00103	\$0067	147	01100111
00104	\$0068	150	01101000
00105	\$0069	151	01101001
00106	\$006A	152	01101010
00107	\$006B	153	01101011
00108	\$006C	154	01101100
00109	\$006D	155	01101101
00110	\$006E	156	01101110
00111	\$006F	157	01101111
00112	\$0070	160	01110000
00113	\$0071	161	01110001
00114	\$0072	162	01110010

DECIMAL	HEX	OCTAL	BINARY
00115	\$0073	163	01110011
00116	\$0074	164	01110100
00117	\$0075	165	01110101
00118	\$0076	166	01110110
00119	\$0077	167	01110111
00120	\$0078	170	01111000
00121	\$0079	171	01111001
00122	\$007A	172	01111010
00123	\$007B	173	01111011
00124	\$007C	174	01111100
00125	\$007D	175	01111101
00126	\$007E	176	01111110
00127	\$007F	177	01111111
00128	\$0080	200	10000000
00129	\$0081	201	10000001
00130	\$0082	202	10000010
00131	\$0083	203	10000011
00132	\$0084	204	10000100
00133	\$0085	205	10000101
00134	\$0086	206	10000110
00135	\$0087	207	10000111
00136	\$0088	210	10001000
00137	\$0089	211	10001001
00138	\$008A	212	10001010
00139	\$008B	213	10001011
00140	\$008C	214	10001100
00141	\$008D	215	10001101
00142	\$008E	216	10001110
00143	\$008F	217	10001111
00144	\$0090	220	10010000
00145	\$0091	221	10010001
00146	\$0092	222	10010010
00147	\$0093	223	10010011
00148	\$0094	224	10010100
00149	\$0095	225	10010101
00150	\$0096	226	10010110
00151	\$0097	227	10010111
00152	\$0098	230	10011000
00153	\$0099	231	10011001

DECIMAL	HEX	OCTAL	BINARY
00154	\$009A	232	10011010
00155	\$009B	233	10011011
00156	\$009C	234	10011100
00157	\$009D	235	10011101
00158	\$009E	236	10011110
00159	\$009F	237	10011111
00160	\$00A0	240	10100000
00161	\$00A1	241	10100001
00162	\$00A2	242	10100010
00163	\$00A3	243	10100011
00164	\$00A4	244	10100100
00165	\$00A5	245	10100101
00166	\$00A6	246	10100110
00167	\$00A7	247	10000111
00168	\$00A8	250	10101000
00169	\$00A9	251	10101001
00170	\$00AA	252	10101010
00171	\$00AB	253	10101011
00172	\$00AC	254	10101100
00173	\$00AD	255	10101101
00174	\$00AE	256	10101110
00175	\$00AF	257	10101111
00176	\$00B0	260	10110000
00177	\$00B1	261	10110001
00178	\$00B2	262	10110010
00179	\$00B3	263	10110011
00180	\$00B4	264	10110100
00181	\$00B5	265	10110101
00182	\$00B6	266	10110110
00183	\$00B7	267	10110111
00184	\$00B8	270	10111000
00185	\$00B9	271	10111001
00186	\$00BA	272	10111010
00187	\$00BB	273	10111011
00188	\$00BC	274	10111100
00189	\$00BD	275	10111101
00190	\$00BE	276	10111110
00191	\$00BF	277	10111111
00192	\$00C0	300	11000000

DECIMAL	HEX	OCTAL	BINARY
00193	\$00C0	301	11000001
00194	\$00C2	302	11000010
00195	\$00C3	303	11000011
00196	\$00C4	304	11000100
00197	\$00C5	305	11000101
00198	\$00C6	306	11000110
00199	\$00C7	307	11000111
00200	\$00C8	310	11001000
00201	\$00C9	311	11001001
00202	\$00CA	312	11001010
00203	\$00CB	313	11001011
00204	\$00CC	314	11001100
00205	\$00CD	315	11001101
00206	\$00CE	316	11001110
00207	\$00CF	317	11001111
00208	\$00D0	320	11010000
00209	\$00D1	321	11010001
00210	\$00D2	322	11010010
00211	\$00D3	323	11010011
00212	\$00D4	324	11010100
00213	\$00D5	325	11010101
00214	\$00D6	326	11010110
00215	\$00D7	327	11010111
00216	\$00D8	330	11011000
00217	\$00D9	331	11011001
00218	\$00DA	332	11011010
00219	\$00DB	333	11011011
00220	\$00DC	334	11011100
00221	\$00DD	335	11011101
00222	\$00DE	336	11011110
00223	\$00DF	337	11011111
00224	\$00E0	340	11100000
00225	\$00E1	341	11100001
00226	\$00E2	342	11100010
00227	\$00E3	343	11100011
00228	\$00E4	344	11100100
00229	\$00E5	345	11100101
00230	\$00E6	346	11100110
00231	\$00E7	347	11100111

DECIMAL	HEX	OCTAL	BINARY
00232	\$00E8	350	11101000
00233	\$00E9	351	11101001
00234	\$00EA	352	11101010
00235	\$00EB	353	11101011
00236	\$00EC	354	11101100
00237	\$00ED	355	11101101
00238	\$00EE	356	11101110
00239	\$00EF	357	11101111
00240	\$00F0	360	11110000
00241	\$00F1	361	11110001
00242	\$00F2	362	11110010
00243	\$00F3	363	11110011
00244	\$00F4	364	11110100
00245	\$00F5	365	11110101
00246	\$00F6	366	11110110
00247	\$00F7	367	11110111
00248	\$00F8	370	11111000
00249	\$00F9	371	11111001
00250	\$00FA	372	11111010
00251	\$00FB	373	11111011
00252	\$00FC	374	11111100
00253	\$00FD	375	11111101
00254	\$00FE	376	11111110
00255	\$00FF	377	11111111

ASCII CONVERSION CHART

				EVEN PARITY		ODD PARITY		EBC- DIC
DEC	HEX	CBM	TRUE	DEC	HEX	DEC	HEX	
0	00		NUL	0	00	128	80	00
1	01		SOH	129	81	1	01	01
2	02		STX	130	82	2	02	02
3	03		ETX	3	03	131	83	03
4	04		EOT	132	84	4	04	37
5	05		ENQ	5	05	133	85	2D
6	06		ACK	6	06	134	86	2E
7	07		BEL	135	87	7	07	2F
8	08		BS	136	88	8	08	16
9	09		HT	9	09	137	89	05
10	0A		LF	10	0A	138	8A	25
11	0B		VT	139	8B	11	0B	0B
12	0C		FF	12	0C	140	8C	0C
13	0D		CR	141	8D	13	0D	0D
14	0E		SO	142	8E	14	0E	0E
15	0F		SI	15	0F	143	8F	0F
16	10		DLE	144	90	16	10	10
17	11		DC1	17	11	145	91	11
18	12		DC2	18	12	146	92	12
19	13		DC3	147	93	19	13	13
20	14		DC4	20	14	148	94	14
21	15		NAK	149	95	21	15	3D
22	16		SYN	150	96	22	16	32
23	17		ETB	23	17	151	97	26
24	18		CAN	24	18	152	98	18
25	19		EM	153	99	25	19	19
26	1A		SUB	154	9A	26	1A	3F
27	1B		ESC	27	1B	155	9B	27
28	1C		FS	156	9C	28	1C	22
29	1D		GS	29	1D	157	9D	
30	1E		RS	30	1E	158	9E	35
31	1F		US	159	9F	31	1F	
32	20			160	A0	32	20	40

EVEN PARITY				ODD PARITY				EBC- DIC
DEC	HEX	CBM	TRUE	DEC	HEX	DEC	HEX	
33	21	!	!	33	21	161	A1	5A
34	22	"	"	34	22	162	A2	7F
35	23	#	#	163	A3	35	23	7B
36	24	\$	\$	36	24	164	A4	5B
37	25	%	%	165	A5	37	25	6C
38	26	&	&	166	A6	38	26	50
39	27	'	'	39	27	167	A7	7D
40	28	(	(	40	28	168	A8	4D
41	29	)	)	169	A9	41	29	5D
42	2A	*	*	170	AA	42	2A	5C
43	2B	+	+	43	2B	171	AB	4E
44	2C	,	,	172	AC	44	2C	6B
45	2D	-	-	45	2D	173	AD	60
46	2E	.	.	46	2E	174	AE	4B
47	2F	/	/	175	AF	47	2F	61
48	30	0	0	48	30	176	B0	F0
49	31	1	1	177	B1	49	31	F1
50	32	2	2	178	B2	50	32	F2
51	33	3	3	51	33	179	B3	F3
52	34	4	4	180	B4	52	34	F4
53	35	5	5	53	35	181	B5	F5
54	36	6	6	54	36	182	B6	F6
55	37	7	7	183	B7	55	37	F7
56	38	8	8	184	B8	56	38	F8
57	39	9	9	57	39	185	B9	F9
58	3A	:	:	58	3A	186	BA	7A
59	3B	;	;	187	BB	59	3B	5E
60	3C	<	<	60	3C	188	BC	4C
61	3D	=	=	189	BD	61	3D	7E
62	3E	>	>	190	BE	62	3E	6E
63	3F	?	?	63	3F	191	BF	6F
64	40	@	@	192	C0	64	40	7C
65	41	a	A	65	41	193	C1	C1
66	42	b	B	66	42	194	C2	C2
67	43	c	C	195	C3	67	43	C3

EVEN PARITY				ODD PARITY				EBC- DIC
DEC	HEX	CBM	TRUE	DEC	HEX	DEC	HEX	
68	44	d	D	68	44	196	C4	C4
69	45	e	E	197	C5	69	45	C5
70	46	f	F	198	C6	70	46	C6
71	47	g	G	71	47	199	C7	C7
72	48	h	H	72	48	200	C8	C8
73	49	i	I	201	C9	73	49	C9
74	4A	j	J	202	CA	74	4A	D1
75	4B	k	K	75	4B	203	CB	D2
76	4C	l	L	204	CC	76	4C	D3
77	4D	m	M	77	4D	205	CD	D4
78	4E	n	N	78	4E	206	CE	D5
79	4F	o	O	207	CF	79	4F	D6
80	50	p	P	80	50	208	D0	D7
81	51	q	Q	209	D1	81	51	D8
82	52	r	R	210	D2	82	52	D9
83	53	s	S	83	53	211	D3	E2
84	54	t	T	212	D4	84	54	E3
85	55	u	U	85	55	213	D5	E4
86	56	v	V	86	56	214	D6	E5
87	57	w	W	215	D7	87	57	E6
88	58	x	X	216	D8	88	58	E7
89	59	y	Y	89	59	217	D9	E8
90	5A	z	Z	90	5A	218	DA	E9
91	5B	[	[	219	DB	91	5B	NA
92	5C	*	*	92	5C	220	DC	E0
93	5D	]	]	221	DD	93	5D	NA
94	5E	^	^	222	DE	94	5E	NA
95	5F	*	*	95	5F	223	DF	6D
96	60		*	96	60	224	E0	79
97	61		a	225	E1	97	61	81
98	62		b	226	E2	98	62	82
99	63		c	99	63	227	E3	83
100	64		d	228	E4	100	64	84
101	65		e	101	65	229	E5	85
102	66		f	102	66	230	E6	86

				EVEN PARITY		ODD PARITY		
DEC	HEX	CBM	TRUE	DEC	HEX	DEC	HEX	EBC- DIC
103	67		g	231	E7	103	67	87
104	68		h	232	E8	104	68	88
105	69		i	105	69	233	E9	89
106	6A		j	106	6A	234	EA	91
107	6B		k	235	EB	107	6B	92
108	6C		l	108	6C	236	EC	93
109	6D		m	237	ED	109	6D	94
110	6E		n	238	EE	110	6E	95
111	6F		o	111	6F	239	EF	96
112	70		p	240	F0	112	70	97
113	71		q	113	71	241	F1	98
114	72		r	114	72	242	F2	99
115	73		s	243	F3	115	73	A2
116	74		t	116	74	244	F4	A3
117	75		u	245	F5	117	75	A4
118	76		v	246	F6	118	76	A5
119	77		w	119	77	247	F7	A6
120	78		x	120	78	248	F8	A7
121	79		y	249	F9	121	79	A8
122	7A		z	250	FA	122	7A	A9
123	7B		*	123	7B	251	FB	C0
124	7C		*	252	FC	124	7C	6A
125	7D			125	7D	253	FD	D0
126	7E			126	7E	254	FE	A1
127	7F		DEL	255	FF	127	7F	07

MULTIPLICATION TABLE

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32
3	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48
4	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
5	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
6	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96
7	7	14	21	28	35	42	49	56	63	70	77	84	91	98	105	112
8	8	16	24	32	40	48	56	64	72	80	88	96	104	112	120	128
9	9	18	27	36	45	54	63	72	81	90	99	108	117	126	135	144
10	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160
11	11	22	33	44	55	66	77	88	99	110	121	132	143	154	165	176
12	12	24	36	48	60	72	84	96	108	120	132	144	156	168	180	192
13	13	26	39	52	65	78	91	104	117	130	143	156	169	182	195	208
14	14	28	42	56	70	84	98	112	126	140	154	168	182	196	210	224
15	15	30	45	60	75	90	105	120	135	150	165	180	195	210	225	240
16	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256

HEX ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

HEX MULTIPLICATION TABLE

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	48	5A	69	78	87	96	A5	B4	C3	D2	E1

SUPER HELP CHART

DEC	HEX	BASIC	6510	ASCII	SCREEN
			OPS CODES		
000	00	END	BRK		@
001	01		ORA(\$ZZ,X)	A	a
002	02			B	b
003	03		stop	C	c
004	04			D	d
005	05		ORA \$ZZ	E	e
006	06		ALS \$ZZ	F	f
007	07			G	g
008	08		PHP	H	h
009	09		ORA\$\$ii	I	i
010	0A		ASL	J	j
011	0B			K	k
012	0C			L	l
013	0D		ORA\$nnnn	M	m
014	0E		ASL\$nnnn	N	n
015	0F			O	o
016	10		BPL\$nnnn	P	p
017	11		ORA(\$zz),Y	Q	q
018	12			R	r
019	13			S	s
020	14			T	t
021	15		ORA\$zz,X	U	u
022	16		ASL\$zz,X	V	v
023	17			W	w
024	18		LLC	X	x
025	19		ORA\$nnnn,Y	Y	y
026	1A			Z	z
027	1B			[	
028	1C			E	
029	1D		ORA\$nnnn,X	]	
030	1E		ASL\$nnnn,X	^	
031	1F			←	
032	20	space	JSR\$nnnn	space	space
033	21	!	AND(\$zz,X)	!	!
034	22	"		"	"
035	23	#		#	#

DEC	HEX	BASIC	6510 OPS CODES	ASCII	SCREEN
036	24	\$	BIT\$zz	\$	\$
037	25	%	AND\$zz	%	%
038	26	&	RDL\$zz	&	&
039	27	'		'	'
040	28	(	PLP	(	(
041	29	)	AND\$\$ii	)	)
042	2A	*	ROL	*	*
043	2B	+		+	+
044	2C	,	BIT\$nnnn	,	,
045	2D	-	AND\$nnnn	-	-
046	2E	.	ROL\$nnnn	.	.
047	2F	/		/	/
048	30	0	BMI\$nn	0	0
049	31	1	AND(\$zz),Y	1	1
050	32	2		2	2
051	33	3		3	3
052	34	4		4	4
053	35	5	AND\$zz,X	5	5
054	36	6	ROL\$zz,X	6	6
055	37	7		7	7
056	38	8	SEC	8	8
057	39	9	AND\$nnnn,Y	9	9
058	3A	:		:	:
059	3B	;		;	;
060	3C	<		<	<
061	3D	=	AND\$nnnn,X	=	=
062	3E	>	ROL\$nnnn,X	>	>
063	3F	?		?	?
064	40	@	RTI	@	graphics
065	41	A	EOR(\$zz,X)	A	A
066	42	B		B	B
067	43	C		C	C
068	44	D		D	D
069	45	E	EOR\$zz	E	E
070	46	F	LSR\$zz	F	F
071	47	G		G	G
072	48	H	PHA	H	H
073	49	I	EOR#\$ii	I	I

DEC	HEX	BASIC	6510 OPS CODES	ASCII	SCREEN
074	4A	J	LSR	J	" J
075	4B	K		K	" K
076	4C	L	JMP\$nnnn	L	" L
077	4D	M	EOR\$nnnn	M	" M
078	4E	N	LSR\$nnnn	N	" N
079	4F	O		O	" O
080	50	P	BVC\$nn	P	" P
081	51	Q	EOR(\$zz),Y	Q	" Q
082	52	R		R	" R
083	53	S		S	" S
084	54	T		T	" T
085	55	U	EOR\$zz,X	U	" U
086	56	V	LSR#zz,X	V	" V
087	57	W		W	" W
088	58	X	CLI	X	" X
089	59	Y	EOR\$zz,Y	Y	" Y
090	5A	Z		Z	" Z
091	5B	a		left bracket	
092	5C	b		english pound	
093	5D	c	EOR\$nnnn,X	^	graphics
094	5E	d	LSR\$nnnn,X	^	" "
095	5F	e		⌞	" "
096	60	f	RTS	graphics	space
097	61	g	ADC(\$zz),X	"	graphics
098	62	h		"	" "
099	63	i			" "
100	64	j			
101	65	k	ADC\$zz		
102	66	l	ROR\$zz		
103	67	m			
104	68	n	PLA		
105	69	o	ADC#\$ii		
106	6A	p	ROR		
107	6B	q			
108	6C	r	JMP(\$nnnn)		
109	6D	s	ADC\$nnnn		
110	6E	t	ROR\$nnnn		
111	6F	u			

DEC	HEX	BASIC	6510 OPS CODES	ASCII
112	70	v	BVS\$nn	
113	71	w	ADC(\$zz),y	
114	72	x		
115	73	y		
116	74	z		
117	75		ADC\$zz,X	
118	76		RDR\$zz,X	
119	77			
120	78		SEI	
121	79		ADC\$nnnn,Y	
122	7A			
123	7B			
124	7C			
125	7D		ADC\$nnnn,X	
126	7E		ROR\$nnnn,X	
127	7F			
128	80	END		
129	81	FOR	STA(\$zz,X)	orange
130	82	NEXT		
131	83	DATA		
132	84	INPUT#	STA\$zz	
133	85	INPUT	STA\$zz	F1
134	86	DIN	STX\$zz	F3
135	87	READ		F5
136	88	LET	DEY	F7
137	89	GOTO		F2
138	8A	RUN	TXA	F4
139	8B	IF		F6
140	8C	RESTORE	STY\$nnnn	F8
141	8D	GOSUB	STA\$nnnn	SHIFT/RETURN
142	8E	RETURN	STX\$nnnn	upper case
143	8F	REM		
144	90	STOP	BCC\$nn	black
145	91	ON	STA(\$zz),Y	cursor up
146	92	WAIT		ReVerSe off
147	93	LOAD		CLear HOME
148	94	SAVE	STY\$zz,x	INSeRT DELeTe
149	95	VERIFY	STA\$zz,X	brown

DEC	HEX	BASIC	6510 OPS CODES	ASCII
150	96	DEF	STX\$zz,Y	light red
151	97	POKE		light gray
152	98	PRINT #	TYA	med. gray
153	99	PRINT	STA\$nnnn,Y	light green
154	9A	CONT	TXS	light blue
155	9B	LIST		dark gray
156	9C	CLR		purple
157	9D	CMD	STA\$nnnn,x	cursor left
158	9E	SYS		yellow
159	9F	OPEN		cyan
160	A0	CLOSE	LDY#\$ii	space
161	A1	GET	LDA(\$zz,X)	graphics
162	A2	NEW	LDX#\$ii	"
163	A3	TAB(		"
164	A4	TO	LDY\$zz	"
165	A5	FN	LDA\$zz	"
166	A6	SPC(	LDX\$zz	"
167	A7	THEN		"
168	A8	NOT	TAY	"
169	A9	STEP	LDA#\$ii	"
170	AA	+	TAX	"
171	AB	-		"
172	AC	\$	LDY\$nnnn	"
173	AD	/	LDA\$nnnn	"
174	AE	^	LDX\$nnnn	"
175	AF	AND		"
176	B0	OR	BCS\$nn	"
177	B1	>	LDA(\$zz),Y	"
178	B2	=		"
179	B3	<		"
180	B4	SGN	LDY\$ZZ,X	"
181	B5	INT	LDA\$ZZ,X	"
182	B6	ABS	LDX\$ZZ,Y	"
183	B7	USR		"
184	B8	FRE	CLV	"
185	B9	POS	LDA\$nnnn,Y	"
186	BA	SQR	TSX	"
187	BB	RND		"

DEC	HEX	BASIC	6510 OPS CODES	ASCII
188	BC	LOG	LDY\$nnnn,X	"
189	BD	EXP	LDA\$nnnn,X	"
190	BE	COS	LDX\$nnnn,Y	"
191	BF	SIN		"
192	C0	TAN	CPY#\$ii	
193	C1	ATN	CMP(zz,X)	
194	C2	PEEK		
195	C3	LEN		
196	C4	STR\$	CPY\$zz	
197	C5	VAL	CMP\$zz	
198	C6	ASC	DEC\$zz	
199	C7	CHR\$		
200	C8	LEFT\$	INY	
201	C9	RIGHT\$	CMP\$zz	
202	CA	MID\$	DEX	
203	CB	GO		
204	CC		CPY\$nnnn	
205	CD		CMP\$nnnn	
206	CE		DEC\$nnnn	
207	CF			
208	D0		BNE\$nn	
209	D1		CMP\$(zz),Y	
210	D2			
211	D3			
212	D4			
213	D5		CMP\$zz,X	
214	D6		DEC\$ZZ,X	
215	D7			
216	D8		CLD	
217	D9		CMP\$nnnn,Y	
218	DA			
219	DB			
220	DC			
221	DD		CMP\$nnnn,X	
222	DE		DEC\$nnnn,X	
223	DF			
224	E0		CPX#\$ii	graphics
225	E1		SBC\$(zz,X)	"

<-- GENERAL MODE ORANGE -->

DEC	HEX	BASIC	6510 OPS CODES	ASCII
226	E2			"
227	E3			"
228	E4		CPX\$zz	"
229	E5		SBC(\$zz,X)	"
230	E6		INC\$zz	"
231	E7			"
232	E8		INX	"
233	E9		SBC\$zz	"
234	EA		NOP	"
235	EB			"
236	EC		COX\$nnnn	"
237	ED		SBC\$nnnn	"
238	EE		INC\$nnnn	"
239	EF			"
240	F0		BEQ\$nn	"
241	F1		SBC(\$zz),Y	"
242	F2			"
243	F3			"
244	F4			"
245	F5		SBC\$zz,X	"
246	F6		INC\$zz,X	"
247	F7			"
248	F8		SED	"
249	F9		SBC\$nnnn,Y	"
250	FA			"
251	FB			"
252	FC			"
253	FD		SBC\$nnnn,X	"
254	FE		INC\$nnnn,X	"
255	FF			"

MCS6510 INSTRUCTION SET

MNEMONIC	DEFINITION
ADC	Add memory to accumulator with carry
AND	AND memory with accumulator
ASL	Shift left one bit (memory or accumulator)
BCC	Branch on carry clear
BCS	Branch on carry set
BEQ	Branch on result zero
BIT	Test bits in memory with accumulator
BMI	Branch on result minus
BNE	Branch on result not zero
BPL	Branch on result plus
BRK	Force break
BVC	Branch on overflow clear
BVS	Branch on overflow set
CLC	Clear carry flag
CLD	Clear decimal mode
CLI	Clear interrupt disable bit
CLV	Clear carry flag
CMP	Compare memory and accumulator
CPX	Compare memory and index X
CPY	Compare memory and index Y
DEC	Decrement memory by one
DEX	Decrement index X by one
DEY	Decrement index Y by one
EOR	Exclusive OR memory with accumulator
INC	Increment memory by one
INX	Increment index X by one
INY	Increment index Y by one
JMP	Jump to new location
JSR	Jump to new location saving return address
LDA	Load accumulator with memory
LDX	Load index X with memory
LDY	Load index Y with memory
LSR	Shift right one bit (memory or accumulator)
NOP	No operation

MNEMONIC	DEFINITION
ORA	OR memory with accumulator
PHA	Push accumulator on stack
PHP	Push processor status on stack
PLA	Pull accumulator from stack
PLP	Pull processor status from stack
ROL	Rotate one bit left (memory or accumulator)
ROR	Rotate one bit right (memory or accumulator)
RTI	Return from interrupt
RTS	Return from subroutine
SBC	Subtract memory from accumulator with borrow
SEC	Set carry flag
SED	Set decimal mode
SEI	Set interrupt disable status
STA	Store accumulator in memory
STX	Store index X in memory
STY	Store index Y in memory
TAX	Transfer accumulator to index X
TAY	Transfer accumulator to index Y
TSX	Transfer stack pointer to index X
TXA	Transfer index X to accumulator
TXS	Transfer index X to stack pointer
TYA	Transfer index Y to accumulator

SYMBOLS & OPERATORS

SYMBOL	FUNCTION
+	Arithmetic addition, plus
-	Arithmetic subtraction, minus
*	Arithmetic multiplication
/	Arithmetic division
^	Exponentiation (raise to the tenth power)
=	Relationship testing, equal to
<	Relationship testing, less than
>	Relationship testing, greater than
<= or =<	Less than or equal to
>= or =>	Greater than or equal to
<>	Not equal to
?	Used inplace of the keyword PRINT
:	The end of a BASIC statement
;	Suppress TAB after PRINT
,	TAB to next column
.	Decimal point, may be used inplace of zero
"	Identifies a character string
\$	Identifies a string variable
%	Identifies a floating point variable
()	Top priority in order of operation
#	Signifies the word number
&	And, also, in addition, to join together

SCREEN FORMAT TABLES

80 COLUMN SCREEN FORMAT TABLE

To center screen text horizontally, (from side to side of the screen); If lines of text equal xx then enter TAB(xx) to center.

TEXT LENGTH	TAB XX	TEXT LENGTH	TAB XX
Text = 1 then	TAB(39)	Text = 32 then	TAB(24)
Text = 2 then	TAB(39)	Text = 33 then	TAB(23)
Text = 3 then	TAB(38)	Text = 34 then	TAB(23)
Text = 4 then	TAB(38)	Text = 35 then	TAB(22)
Text = 5 then	TAB(37)	Text = 36 then	TAB(22)
Text = 6 then	TAB(37)	Text = 37 then	TAB(21)
Text = 7 then	TAB(36)	Text = 38 then	TAB(21)
Text = 8 then	TAB(36)	Text = 39 then	TAB(20)
Text = 9 then	TAB(35)	Text = 40 then	TAB(20)
Text = 10 then	TAB(35)	Text = 41 then	TAB(19)
Text = 11 then	TAB(34)	Text = 42 then	TAB(19)
Text = 12 then	TAB(34)	Text = 43 then	TAB(18)
Text = 13 then	TAB(33)	Text = 44 then	TAB(18)
Text = 14 then	TAB(33)	Text = 45 then	TAB(17)
Text = 15 then	TAB(32)	Text = 46 then	TAB(17)
Text = 16 then	TAB(32)	Text = 47 then	TAB(16)
Text = 17 then	TAB(31)	Text = 48 then	TAB(16)
Text = 18 then	TAB(31)	Text = 49 then	TAB(15)
Text = 19 then	TAB(30)	Text = 50 then	TAB(15)
Text = 20 then	TAB(30)	Text = 51 then	TAB(14)
Text = 21 then	TAB(29)	Text = 52 then	TAB(14)
Text = 22 then	TAB(29)	Text = 53 then	TAB(13)
Text = 23 then	TAB(28)	Text = 54 then	TAB(13)
Text = 24 then	TAB(28)	Text = 55 then	TAB(12)
Text = 25 then	TAB(27)	Text = 56 then	TAB(12)
Text = 26 then	TAB(27)	Text = 57 then	TAB(11)
Text = 27 then	TAB(26)	Text = 58 then	TAB(11)
Text = 28 then	TAB(26)	Text = 59 then	TAB(10)
Text = 29 then	TAB(25)	Text = 60 then	TAB(10)
Text = 30 then	TAB(25)	Text = 61 then	TAB(9)
Text = 31 then	TAB(24)	Text = 62 then	TAB(9)

TEXT LENGTH	TAB XX	TEXT LENGTH	TAB XX
Text = 63 then	TAB(8)	Text = 72 then	TAB(4)
Text = 64 then	TAB(8)	Text = 73 then	TAB(3)
Text = 65 then	TAB(7)	Text = 74 then	TAB(3)
Text = 66 then	TAB(7)	Text = 75 then	TAB(2)
Text = 67 then	TAB(6)	Text = 76 then	TAB(2)
Text = 68 then	TAB(6)	Text = 77 then	TAB(1)
Text = 69 then	TAB(5)	Text = 78 then	TAB(1)
Text = 70 then	TAB(5)	Text = 79 then	TAB(0)
Text = 71 then	TAB(4)	Text = 80 then	TAB(0)

40 COLUMN SCREEN FORMAT TABLE

To center screen text horizontally, (from side to side of the screen); If lines of text equal xx then enter TAB(xx) to center.

TEXT LENGTH	TAB XX	TEXT LENGTH	TAB XX
Text = 1 then	TAB(19)	Text = 21 then	TAB(9)
Text = 2 then	TAB(19)	Text = 22 then	TAB(9)
Text = 3 then	TAB(18)	Text = 23 then	TAB(8)
Text = 4 then	TAB(18)	Text = 24 then	TAB(8)
Text = 5 then	TAB(17)	Text = 25 then	TAB(7)
Text = 6 then	TAB(17)	Text = 26 then	TAB(7)
Text = 7 then	TAB(16)	Text = 27 then	TAB(6)
Text = 8 then	TAB(16)	Text = 28 then	TAB(6)
Text = 9 then	TAB(15)	Text = 29 then	TAB(5)
Text = 10 then	TAB(15)	Text = 30 then	TAB(5)
Text = 11 then	TAB(14)	Text = 31 then	TAB(4)
Text = 12 then	TAB(14)	Text = 32 then	TAB(4)
Text = 13 then	TAB(13)	Text = 33 then	TAB(3)
Text = 14 then	TAB(13)	Text = 34 then	TAB(3)
Text = 15 then	TAB(12)	Text = 35 then	TAB(2)
Text = 16 then	TAB(12)	Text = 36 then	TAB(2)
Text = 17 then	TAB(11)	Text = 37 then	TAB(1)
Text = 18 then	TAB(11)	Text = 38 then	TAB(1)
Text = 19 then	TAB(10)	Text = 39 then	TAB(0)
Text = 20 then	TAB(10)	Text = 40 then	TAB(0)

25 ROW SCREEN FORMAT TABLE

To center text vertically (from top to bottom): if lines of text equal xx, then begin the text on row.

TEXT	LINES	BEGIN	TEXT	LINES	BEGIN
Text =	1	then ROW 13	Text =	14	then ROW 7
Text =	2	then ROW 13	Text =	15	then ROW 6
Text =	3	then ROW 12	Text =	16	then ROW 6
Text =	4	then ROW 12	Text =	17	then ROW 5
Text =	5	then ROW 11	Text =	18	then ROW 5
Text =	6	then ROW 11	Text =	19	then ROW 4
Text =	7	then ROW 10	Text =	20	then ROW 4
Text =	8	then ROW 10	Text =	21	then ROW 3
Text =	9	then ROW 9	Text =	22	then ROW 3
Text =	10	then ROW 9	Text =	23	then ROW 2
Text =	11	then ROW 8	Text =	24	then ROW 2
Text =	12	then ROW 8	Text =	25	then ROW 1
Text =	13	then ROW 7			

STANDARD ABBREVIATIONS

A

ACIA	Asynchronous Communications Interface Adapter
ADC	Advanced Data Communication
ADLC	Advanced Data Link Controller
ALGOL	ALGOrithmic Language
ALU	Arithmetic Logic Unit
ANSI	American National Standards Institute
APT	Automatically Programmed Tools
AQL	Acceptable Quality Level
ARQ	Automatic Request for Repeat
ARU	Audio-response Unit
ASCII	American Standard Code for Information Interchange
ATE	Automatic Test Equipment
ATG	Automatic Test Generation

B

BASIC	Beginner's All-purpose Symbolic Instruction Code
BCS	Basic Control System
BCD	Binary Coded Decimal
BOS	Basic operating System
BPS	Bits Per Second
BSC	Binary Synchronous Communications

C

CAD	Computer Aided Design
CAI	Computer Assisted Instruction
CAM	Computer Aided Manufacturing
CCD	Charge Coupled Device
CLM	Current Mode Logic
CMD	CoMmanD
COBOL	Common Business Oriented Language
COGO	COordinate GeOmetry
COM	Computer Output Microfilm

CNC	Computerized Numeric Control
CPC	Computer Power Center
COMSAT	COMMunications SATellite
CPE	Central Processing Element
CPM	Control Program for Microprocessors
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CROM	Control Read Only Memory
CRT	Cathode Ray Tube

D

DAR	Data Access Register
DAC	Data Acquisition and Control
DAC	Digital Analog Converter
DB	Data Base
DCTL	Direct Coupled Transistor Logic
DED	Data Element Dictionary
DIB	Data Input Bus
DIR	DIRectory
DMA	Direct Memory Access
DNC	Direct Numerical Control
DOS	Disk Operating System
DPMA	Data Processing Management Association
DSW	Device Status Word
DTE	Data Terminal Equipment
DTL	Diode Transistor Logic
DUV	Data Under Voice

E

EAM	Electrical Accounting Machine
EAROM	Electrically Alterable Read Only Memory
EBCDIC	Extended Binary Coded Decimal Interchange Code
ECDs	ElectroChromic Displays
ECL	Emitter Coupled Logic
EEPROM	Electrically Erasable Programmable Read Only Memory
EFTS	Electronic Funds Transfer Systems
EIA	Electronic Industries Association

EOF	End Of File
EOM	End Of Message
EOT	End Of Transmission
ENQ	ENquiry Character
EPROM	Electrically Programmable Read Only Memory
EXEC	EXECutive statement or EXECutive system

F

FIFO	First In First Out
FORTRAN	FORMula TRANslator
FP	Floating Point
FPLA	Field Programmable Logic Array
FSK	Frequency Shift Keying

G

GIGO	Garbage In Garbage Out
------	------------------------

I

IC	Integrated Circuit or Instruction Counter
IEEE	Institute of Electrical and Electronics Engineers
IO	Input Output
IPL	Initial Program Loader
ISAM	Indexed Sequential Access Method
ISO	International Organization for Standardization

K

KSAM	Keyed Sequential Access Method
KSR	Keyboard Send/Receive
KWIC	KeyWord In Context

L

LCD	Liquid Crystal Display
LIFO	Last In First Out
LSI	Large Scale Integration

M

MICR Magnetic Ink Character Recognition
MIS Metal Insulator Silicon
MKS Meter Kilogram Second
ML Machine Language
MOS Metal Oxide Semiconductor
MSI Medium Scale Integration
MTBF Mean Time Between Failure
MTF Mean Time to Failure
MPS MicroProcessor System
MPU MicroProcessor Unit

N

NMI Non Maskable Interrupt
NRZ1 NonReturn to Zero one

O

OR OperRator

P

P Power
PAL Programmable Array Logic
PC Programmable Controller
PERT Program Evaluation and Review Technique
PIA Peripheral Interface Adapter
PLA Programmable Logic Array
PPL Phase Locked Loop
POS Point Of Sale
PROM Programmable Read Only Memory
PRT Program Reference Table
PSU Program Storage Unit
PSW Program Status Word

Q

QAM Quadrature Amplitude Modulation

R

RAM Random Access Memory
RCTL Resistor Capacitor Transistor Logic
ROM Read Only Memory
RTC Relative Time Clock
RTE Real Time Executive
RTL Resistor Transistor Logic
RZ Return to Zero

S

SI SuperImpose
SIO Serial Input/Output
SOS Silicon On Sapphire
STRESS StructUral Enginerrng Systems Solver
SYSGEN SYStems GENeration

T — U

TCAM Telecommunications Access Method
TDM Time Division Multiplexer
TTL Transistor-Transistor Logic
UART Universal Asynchronous Receiver/Transmitter
UHF UltraHigh Frequencies
ULA Uncommon Logic Arrays
UBC Universal Block Channel
UPS Uninterruptible Power Supply

V

VAB Voice Answer Back
VDG Video Display Generator
VRAM Video RAM
VRC Vertical Redundancy Checking
VS Virtual Storage
VTAM Vortex TelecommunicAtions Method

W

WP Word Processing
WPM Words Per Minute
WS Working Storage

CASSETTE COMMANDS

LOAD.....

LOAD Load next program on the tape

LOAD"*" Load first program on the tape

LOAD"name",1 Load machine language program

LOAD"name",1,1 Load to proper place in mem.

LOAD"T" Load first file starting with T

LOAD"Jan" Load first file starting with Jan

SAVE To save file in memory to tape

SAVE"Jan" To save program named Jan by name

VERIFY Compare file with one in memory

VERIFY"JAN" Compare prog. named Jan to memory

OPEN 1,1,1 File open for recording info

OPEN 2,1,1 Opens file 2 for recording SEQ file

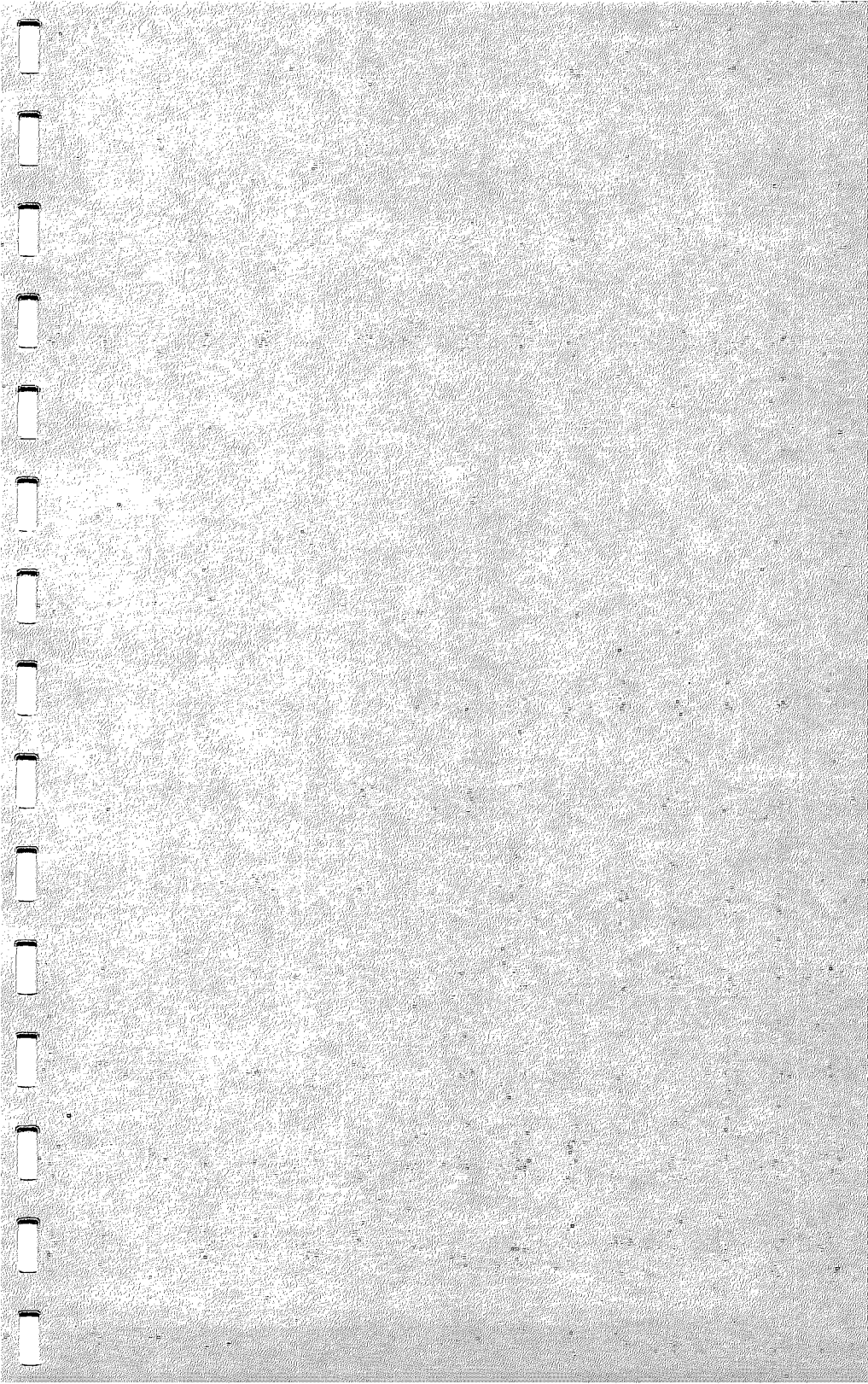
OPEN 3,1,0,"Jan" Open file 3 for recall/SEQ

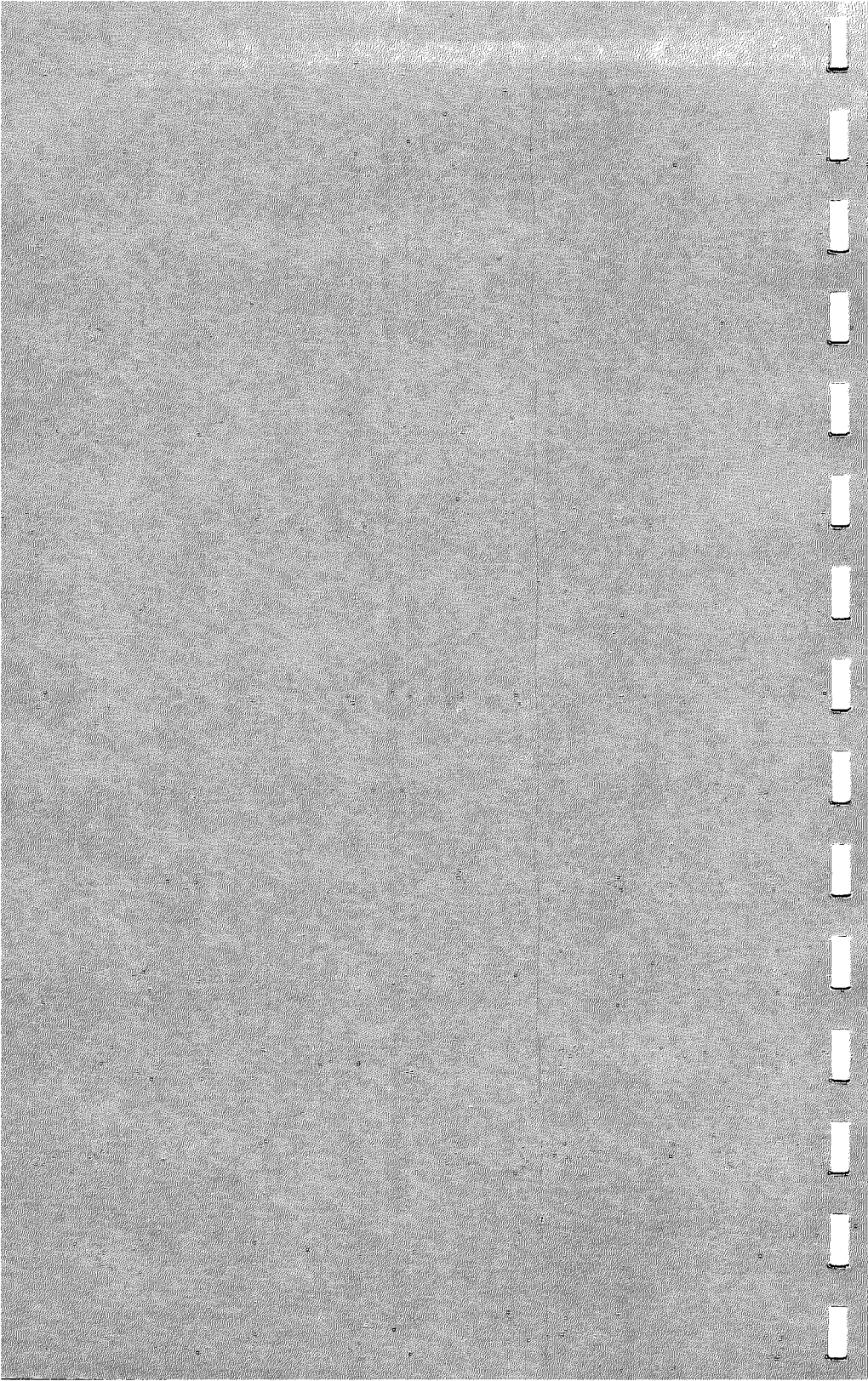
OPEN 1,1 Open file 1 for recall next SEQ file

OPEN 2,1,2,"end of tape" Open file 2 for
recording file called END OF TAPE

OPEN 1,1,2:CLOSE 1 End of tape marker







EQUIPMENT PURCHASES	231
EQUIPMENT REPAIR	237
SUPPLIES	243
DISK ID's	247
BULLETIN BOARDS LOCAL	252
BULLETIN BOARDS NON-LOCAL	256

This section is intended to record personal data such as: serial and model numbers of equipment, what store has the best buy on printer ribbon, your favorite BBS number, all the things you were always going to write down but never have. The pages labeled DISK ID's contain, a set of pre-assigned disk ID numbers. Several months ago I was looking through my software and discovered, a number of disks, with the same ID. The numerals 64 seemed to be very popular. The problem did not sink in until I attempted to move a file from one disk to another. The drive immediately became muddled over the duplication of ID's. I promptly made a chart of new ID's and assigned each disk a number. Now when a disk needs to be formatted I check the chart to see what the ID should be.



EQUIPMENT PURCHASES

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASES

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASES

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASES

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASED _____

PURCHASED FROM _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

DATE BOUGHT _____

WARRANTY EXPIRES _____

SERIAL NUMBER _____

EQUIPMENT PURCHASES

EQUIPMENT PURCHASED

PURCHASED FROM

ADDRESS:
STREET

CITY & STATE

TELEPHONE NUMBER

DATE BOUGHT

WARRANTY EXPIRES

SERIAL NUMBER

EQUIPMENT PURCHASED

PURCHASED FROM

ADDRESS:
STREET

CITY & STATE

TELEPHONE NUMBER

DATE BOUGHT

WARRANTY EXPIRES

SERIAL NUMBER

EQUIPMENT PURCHASES

EQUIPMENT PURCHASED

PURCHASED FROM

ADDRESS:
STREET

CITY & STATE

TELEPHONE NUMBER

DATE BOUGHT

WARRANTY EXPIRES

SERIAL NUMBER

EQUIPMENT PURCHASED

PURCHASED FROM

ADDRESS:
STREET

CITY & STATE

TELEPHONE NUMBER

DATE BOUGHT

WARRANTY EXPIRES

SERIAL NUMBER

EQUIPMENT REPAIR

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIR

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIR

EQUIPMENT REPAIRED

SHOP NAME

HOURS

ADDRESS:
STREET

CITY & STATE

TELEPHONE NUMBER

CHARGES

WARRANTY

COMMENTS

EQUIPMENT REPAIRED

SHOP NAME

HOURS

ADDRESS:
STREET

CITY & STATE

TELEPHONE NUMBER

CHARGES

WARRANTY

COMMENTS

EQUIPMENT REPAIR

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS:
STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIR

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIR

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

EQUIPMENT REPAIRED _____

SHOP NAME _____

HOURS _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

CHARGES _____

WARRANTY _____

COMMENTS _____

SUPPLIES

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

SUPPLIES

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

SUPPLIES

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

SUPPLIES

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

ITEM _____ PRICE _____

PURCHASE FROM _____

ADDRESS: _____

STREET _____

CITY & STATE _____

TELEPHONE NUMBER _____

COMMENTS _____

DISK ID's

The ID of a disk is an important factor in the format process. During any disk operation, the drive checks for a unique identification. If two diskettes were to have the same IDs, and you wanted to move a file from one to the other, the disk drive would become confused by the duplication. Once the program was in memory the drive would not know where you wanted the file to go. The following list of pre-assigned ID's, are all different. Next time you want to format a disk, just allocate one from the list.

Some letter combinations have been left out, because of possible conflict with keywords and disk utility instructions.

AA_____	AN_____	BB_____	BO_____	CC_____	CP_____
AB_____	AO_____	BC_____	BP_____	CD_____	CR_____
AC_____	AP_____	BD_____	BQ_____	CE_____	CS_____
AD_____	AQ_____	BE_____	BR_____	CF_____	CT_____
AE_____	AR_____	BF_____	BS_____	CG_____	CV_____
AF_____	AS_____	BG_____	BT_____	CH_____	CW_____
AG_____	AT_____	BH_____	BV_____	CI_____	CX_____
AH_____	AV_____	BI_____	BW_____	CJ_____	CY_____
AI_____	AW_____	BJ_____	BX_____	CK_____	CZ_____
AJ_____	AX_____	BK_____	BY_____	CL_____	DA_____
AK_____	AY_____	BL_____	BZ_____	CM_____	DB_____
AL_____	AZ_____	BM_____	CA_____	CN_____	DC_____

<-- PERSONAL DATA GOLD -->

AM.....	BA.....	BN.....	CB.....	CO.....	DD.....
DE.....	EB.....	EY.....	FV.....	GR.....	HO.....
DF.....	EC.....	EZ.....	FW.....	GS.....	HP.....
DG.....	ED.....	FA.....	FX.....	GT.....	HR.....
DH.....	EE.....	FB.....	FY.....	GV.....	HS.....
DI.....	EF.....	FC.....	FZ.....	GW.....	HT.....
DJ.....	EG.....	FD.....	GA.....	GX.....	HV.....
DK.....	EH.....	FE.....	GB.....	GY.....	HW.....
DL.....	EI.....	FF.....	GC.....	GZ.....	HX.....
DM.....	EJ.....	FG.....	GD.....	HA.....	HY.....
DN.....	EK.....	FH.....	GE.....	HB.....	HZ.....
DO.....	EL.....	FI.....	GF.....	HC.....	IA.....
DP.....	EM.....	FJ.....	GG.....	HD.....	IB.....
DQ.....	EN.....	FK.....	GH.....	HE.....	IC.....
DR.....	EO.....	FL.....	GI.....	HF.....	ID.....
DS.....	EP.....	FM.....	GJ.....	HG.....	IE.....
DT.....	EQ.....	FN.....	GK.....	HH.....	IF.....
DV.....	ER.....	FO.....	GL.....	HI.....	IG.....
DW.....	ES.....	FP.....	GM.....	HJ.....	IH.....
DX.....	ET.....	FQ.....	GN.....	HK.....	IJ.....
DY.....	EV.....	FR.....	GO.....	HL.....	IK.....
DZ.....	EW.....	FS.....	GP.....	HM.....	IL.....

<-- PERSONAL DATA GOLD -->

EA_____	EX_____	FT_____	GQ_____	HN_____	IM_____
IN_____	JN_____	KK_____	LI_____	MF_____	NC_____
IP_____	JO_____	KL_____	LJ_____	MG_____	ND_____
IQ_____	JP_____	KM_____	LK_____	MH_____	NE_____
IR_____	JQ_____	KN_____	LL_____	MI_____	NF_____
IS_____	JR_____	KO_____	LM_____	MJ_____	NG_____
IT_____	JS_____	KP_____	LN_____	MK_____	NH_____
IV_____	JT_____	KR_____	LO_____	ML_____	NI_____
IW_____	JV_____	KS_____	LP_____	MM_____	NJ_____
IX_____	JW_____	KT_____	LQ_____	MN_____	NK_____
IY_____	JX_____	KV_____	LR_____	MO_____	NL_____
IZ_____	JY_____	KW_____	LS_____	MP_____	NM_____
JA_____	JZ_____	KX_____	LT_____	MQ_____	NN_____
JB_____	KA_____	KY_____	LV_____	MR_____	NO_____
JC_____	KB_____	KZ_____	LW_____	MS_____	NP_____
JD_____	KC_____	LA_____	LX_____	MT_____	NQ_____
JE_____	KD_____	LB_____	LY_____	MV_____	NR_____
JF_____	KE_____	LC_____	LZ_____	MW_____	NS_____
JG_____	KF_____	LD_____	MA_____	MX_____	NT_____
JH_____	KG_____	LE_____	MB_____	MY_____	NV_____
JK_____	KH_____	LF_____	MC_____	MZ_____	NW_____
JL_____	KI_____	LG_____	MD_____	NA_____	NX_____

<-- PERSONAL DATA GOLD -->

JM_____	KJ_____	LH_____	ME_____	NB_____	NY_____
NZ_____	OK_____	PT_____	RP_____	SN_____	TL_____
OA_____	OY_____	PS_____	RQ_____	SO_____	TM_____
OB_____	OZ_____	PV_____	RS_____	SP_____	TN_____
OC_____	PA_____	PW_____	RT_____	SQ_____	TP_____
OD_____	PB_____	PX_____	RV_____	SR_____	TQ_____
OE_____	PC_____	PY_____	RW_____	SS_____	TR_____
OF_____	PD_____	PZ_____	RX_____	ST_____	TS_____
OG_____	PE_____	RA_____	RY_____	SV_____	TT_____
OH_____	PF_____	RB_____	RZ_____	SW_____	TV_____
OI_____	PG_____	RC_____	SA_____	SX_____	TX_____
OJ_____	PH_____	RD_____	SB_____	SY_____	TY_____
OK_____	PI_____	RE_____	SC_____	SZ_____	TZ_____
OL_____	PJ_____	RF_____	SD_____	TA_____	VA_____
OM_____	PK_____	RG_____	SE_____	TB_____	VB_____
ON_____	PL_____	RH_____	SF_____	TC_____	VC_____
OP_____	PM_____	RI_____	SG_____	TD_____	VD_____
OQ_____	PN_____	RJ_____	SH_____	TE_____	VE_____
OR_____	PO_____	RK_____	SI_____	TF_____	VF_____
OS_____	PP_____	RL_____	SJ_____	TG_____	VG_____
OT_____	PQ_____	RM_____	SK_____	TH_____	VH_____
OV_____	PR_____	RN_____	SL_____	TJ_____	VI_____

<-- PERSONAL DATA GOLD -->

OW_____	PS_____	RO_____	SM_____	TK_____	VJ_____
VK_____	WH_____	XE_____	YB_____	YY_____	ZV_____
VL_____	WI_____	XF_____	YC_____	YZ_____	ZW_____
VM_____	WJ_____	XG_____	YD_____	ZA_____	ZX_____
VN_____	WK_____	XH_____	YE_____	ZB_____	ZY_____
VO_____	WL_____	XI_____	YF_____	ZC_____	ZZ_____
VP_____	WM_____	XJ_____	YG_____	ZD_____	A1_____
VQ_____	WN_____	XK_____	YH_____	ZE_____	A2_____
VR_____	WO_____	XL_____	YI_____	ZF_____	A3_____
VS_____	WP_____	XM_____	YJ_____	ZG_____	A4_____
VT_____	WQ_____	XN_____	YK_____	ZH_____	A5_____
VV_____	WR_____	XO_____	YL_____	ZI_____	A6_____
VW_____	WS_____	XP_____	YM_____	ZJ_____	A7_____
VX_____	WT_____	XQ_____	YN_____	ZK_____	A8_____
VY_____	WV_____	XR_____	YO_____	ZL_____	A9_____
VZ_____	WW_____	XS_____	YP_____	ZM_____	A0_____
WA_____	WX_____	XT_____	YQ_____	ZN_____	B1_____
WB_____	WY_____	XV_____	YR_____	ZO_____	B2_____
WC_____	WZ_____	XW_____	YS_____	ZP_____	B3_____
WD_____	XA_____	XX_____	YT_____	ZQ_____	B4_____
WE_____	XB_____	XY_____	YV_____	ZR_____	B5_____
WF_____	XC_____	XZ_____	YW_____	ZS_____	B6_____

BULLETIN BOARDS-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BULLETIN BOARDS-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BULLETIN BOARDS-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BULLETIN BOARDS-LOCAL

BOARD NAME

HOURS

TELEPHONE NUMBER

CHARGES Y/N

PASSWORD

COMMENTS

BOARD NAME

HOURS

TELEPHONE NUMBER

CHARGES Y/N

PASSWORD

COMMENTS

BOARD NAME

HOURS

TELEPHONE NUMBER

CHARGES Y/N

PASSWORD

COMMENTS

BULLETIN BOARDS NON-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BULLETIN BOARDS NON-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BULLETIN BOARDS NON-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BULLETIN BOARDS NON-LOCAL

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

BOARD NAME _____

HOURS _____

TELEPHONE NUMBER _____

CHARGES Y/N _____

PASSWORD _____

COMMENTS _____

REFERENCES

Commodore Business Machines
COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE
Commodore Bus. Machines & Howard W. Sams & Co., Inc.

Data Becker
COMMODORE 128 INTERNALS
Abacus Software, Grand Rapids, MI

Stan Krute
COMMODORE 64 GRAPHICS & SOUND PROGRAMMING
Tab Books Inc., Blue Ridge Summit, PA

Sheldon Leemon
MAPPING THE COMMODORE 64
Compute! Publications, Inc., Greensboro, NC

Tim Onosko
COMMODORE 64 GETTING THE MOST FROM IT
Prentice-Hall Publishing, Bowie, MD

Jim Heilborn & Ran Talbott
YOUR COMMODORE 64
Osborne/Mc Graw Hill, Berkley, CA

Paul Pavelko & Tim Kelly
THE MASTER MEMORY MAP
Prentice-Hall Co., Reston, VA

COMMODORE 1571 DISK DRIVE USER'S GUIDE
Commodore Electronics, Ltd.

COMMODORE 128 PERSONAL COMPUTER SYSTEM GUIDE
Commodore Electronics, Ltd.

COMMODORE 1541 DISK DRIVE
Commodore Business Machines Electronics Ltd.

M.A.P.S.
THE BRAIN, A PROGRAMMER'S ASSISTANT
Value-Soft Inc., Portland, OR

M.A.P.S.

KISS, A REFERENCE HANDBOOK FOR THE COMMODORE 64
Value-Soft Inc., Portland, OR

Shirley A. & Paul P. Damon
REFERENCE CARD FOR THE COMMODORE 64 MICROCOMPUTER
Nanos Systems Corp.

Digital Research
CP/M PLUS, USER'S GUIDE
Digital Research, PO Box 579, 160 Central Ave.
Pacific Grove, CA

Digital Research
CP/M PLUS, PROGRAMMER'S GUIDE
Digital Research, PO Box 579, 160 Central Ave.
Pacific Grove, CA

Digital Research
CP/M PLUS, SYSTEM GUIDE
Digital Research, PO Box 579, 160 Central Ave.
Pacific Grove, CA

Karl J.H. Hildon
THE COMPLETE COMMODORE INNER SPACE ANTHOLOGY
Transactor Publishing Inc., 500 Steeles Ave.
Milton, Ontario, L9T 3P7, Canada

PERIODICALS

THE TRANSACTOR
Transactor Publishing Inc., 500 Steeles Ave.
Milton, Ontario, L9T 3P7, Canada

RUN
CW COMMUNICATIONS/PETERBOROUGH, INC.
80 PINE ST., PETERBOROUGH, NH, 03458





The C-128 is one of the finest computers Commodore has ever offered. The Black Book of C-128 is a vast authoritative reference source, to assist programmers of all levels, from the beginning BASIC student, to the experienced machine language writer. Think of it as a dictionary, ready to answer your questions. The aim of the book, is to support the user, not teach BASIC programming.

The BLACK BOOK of C-128 tells you everything you need to know about your Commodore 128, 1541, and 1571 disk drive. More than 100 easy to read charts plus numerous examples. All the topics vital to the operation of the C-128 at your finger tips.
