Commodore Reference Diary 1985

Jim Butterfield

Keeping up with you

Personal Information

Name _____

Address _____

_____

Telephone Home _____

Business _____

Commodore Dealer Name _____

Telephone _____

User Group _____

Telephone _____

Bulletin Board Systems

Name	Telephone	Hours

Serial Numbers

Item	No

Commodore Reference Diary 1985

Jim Butterfield

Containing: Reference Material for PET/CBM, VIC-20, COMMODORE 64, Commodore 'B' Series, Commodore Plus/4

Pitman

Pitman Publishing Pty Ltd
(Incorporated in Victoria)

158 Bouverie Street
Carlton
Victoria 3053

Level 12
Town Hall House
452-462 Kent Street
Sydney
New South Wales 2000

9th Floor
National Bank Building
420 George Street
Brisbane
Queensland 4000

Associated Companies

Pitman Publishing Ltd
London

Copp Clark Pitman
Toronto

Pitman Publishing New
Zealand Ltd
Wellingon

ISBN 0 85896 196 2

Author's Note:
Some of the reference material in this diary has been previously published in The Transactor in expanded form. I wish to thank Karl Hildon, the Managing Editor of The Transactor, for his help in coordinating republication.

Design by Lorraine Hulme

Printed in Australia
by Brown Prior Anderson

Contents

Commodore User Groups

New South Wales

Sydney
Sydney Users Association
GPO Box 4721
Sydney 2006
(02) 922 6700

Mona Vale
Sydcom
PO Box 586
Mona Vale 2103
(02) 99 3370

Wentworthville
Vic 20 User Group
53 Lytton St
Wentworthville 2145
(02) 265 2320

Greystanes
Vic 20 User Group
1 Gipps Rd
Greystanes 2145
(02) 636 2080

Sydney
Telecom User Group
4th Floor
229 Castlereagh St
Sydney 2000
(02) 267 3300

Hornsby
Hornsby User Group
PO Box 1661
Northgate
Hornsby Heights 2077
(02) 476 4391

Casula
Southern Districts User Group
3 Lucille Cres
Casula 2170
(02) 602 8691

Cumberland
Cumberland Micro User Group
Granville Library

Newcastle
The Compu-Tech Computer Club
PO Box 43
Islington 2296
(049) 43 3352

Mayfield
Newcastle Computer User Group
136 Maitland Rd
Mayfield 2304
(049) 67 5700

Hawkesbury-Richmond
12 Inverary Drv
Kurmond 2757

Wollongong
Wollongong Users Group
155 Jacaranda Ave
Figtree 2525
(042) 28 8580

Deniliquin
Deniliquin User Group
139 Davidson St
Deniliquin 2710
(058) 81 3014

Hobartville
RAAF Base Richmond
Squadron 486
45 Luttrell St
Hobartville 2753

Jindera
Albury/Woolonga User Group
Drumwood Rd
Jindera 2642
(060) 26 3357

Australian Capital Territory

Belconnen
ACT Commodore Computer User Group
PO Box 455
Belconnen 2617
(062) 412 1211

Watson
Watson User Group
25 Kerford St
Watson 2602
(062) 41 2316

Victoria

Horsham-Wimmera
Commodore User Group
PO Box 4082
Horsham 3401

Laverton
Laverton User Group
School of Radio
RAAF Laverton 3027
(03) 368 2569

Bendigo
Bendigo User Group
5 Board Ct
Kangaroo Flat
(045) 47 7593

Mornington
Mornington District Users Group
Tanti Park Community Centre
Mornington
(03) 783 3007

Northcote
Melbourne VIC 20 Users Group
PO Box 252
Northcote 3070

Victoria Sub Groups

Doncaster
Athenaeum Hall
Doncaster Rd
Doncaster 3108
(03) 419 1924

Knox
Coonara Community House
20 Willow Rd
Upper Fern Tree Gully 3156
(03) 419 1924

Springvale
Dingley Community Centre
Marcus Rd
Dingley 3172
(03) 419 1924

Prahran
Prahran Primary School
67 High St
Prahran 3181
(03) 419 1924

Waverley
Alvie Hall
Alvie Rd
Mt Waverley 3149
(03) 419 1924

Western Suburbs
Please ring 369 4068 for waiting list

Machine Code Group
Blackburn Scout Hall
Koonung Rd
Blackburn Nth 3130
(03) 419 1924

Queensland

Southport
Southport User Group
Scarborough St
Southport 4215
(075) 32 0061

Springwood
Qld Commodore/PET User Group
PO Box 274
Springwood 4127
(07) 209 0899

Coolangatta (Dealer Based)
Tedita Pty Ltd
115 Griffith St
Coolangatta 4225
(075) 36 6722

Mount Isa
Commodore Computer Users Club
3 Crystal Street
Mount Isa 4825

Townsville
Townsville Users Group
1 Paxton St
Townsville 4810
(077) 72 6454

Ipswich
Commodore User Group
PO Box 298
Ipswich 4305
(07) 201 8118

Rockhampton
Commodore User Group
50 Ocean Pde
Yeppoon 4703

South Australia

Nth Adelaide
SA Commodore User Group
PO Box 427
Nth Adelaide 5006
(08) 255 4044

Clarence Gardens
Commodore Computer Users Assoc of SA
PO Box 60
Clarence Gardens 5039

Northern Territory

Darwin
Northern Territory Commodore User Group
349 McMillans Rd
Anula
Darwin 5793
(089) 27 9208

Nhulunbuy
Northern Territory User Group
PO Box 1195
Nhulunbuy 5797

Western Australia

Leederville
WACCUA
PO Box 31
Leederville 6007
(09) 381 4398

Lesmurdie
VIC Ups
4 Shield Rd
Lesmurdie 6076
(09) 451 4629

Kalgoorlie
VIC Ups
28 Dart St
Boulder 6432

Rockingham
VIC Chips
48 Hercules St
Rockingham Park 6168
(095) 27 3954

December 1984

31 Monday

1 January *New Year's Day* Tuesday

2 Wednesday

Thursday 3

Friday 4

Saturday 5

Sunday 6

January 1985

7 Monday

8 Tuesday

9 Wednesday

Thursday 10

Friday 11

Saturday 12

Sunday 13

January 1985

14 Monday

15 Tuesday

16 Wednesday

Thursday 17

Friday 18

Saturday 19

Sunday 20

January 1985

21 Monday

22 Tuesday

23 *Hobart Cup (South Tasmania)* Wednesday

Thursday 24

Friday 25

Saturday 26

Sunday 27

January 1985

28 *Australia Day* Monday

29 Tuesday

30 Wednesday

January 1985

Thursday 31

Friday February 1

Saturday 2

Sunday 3

February 1985

4 Monday

5 Tuesday

6 Wednesday

Thursday 7

Friday 8

Saturday 9

Sunday 10

February 1985

11 Monday

12 *Regatta Day (South Tasmania)* Tuesday

13 Wednesday

Thursday 14

Friday 15

Saturday 16

Sunday 17

February 1985

18 Monday

19 Tuesday

20 Wednesday

Thursday 21

Friday 22

Saturday 23

Sunday 24

February 1985

25 Monday

26 Tuesday

27 *Launceston Cup (North Tasmania)* Wednesday

Thursday 28

Friday March 1

Saturday 2

Sunday 3

March 1985

4 *Eight Hour Day (Tasmania)* Monday
Labour Day (WA)

5 Tuesday

6 Wednesday

Thursday 7

Friday 8

Saturday 9

Sunday 10

March 1985

11 *Labour Day (Vic)* Monday

12 Tuesday

13 Wednesday

Thursday 14

Friday 15

Saturday 16

Sunday 17

March 1985

18 *Canberra Day (ACT)* Monday

19 Tuesday

20 Wednesday

Thursday 21

Friday 22

Saturday 23

Sunday 24

March 1985

25 Monday

26 Tuesday

27 Wednesday

Thursday 28

Friday 29

Saturday 30

Sunday 31

April 1985

1 Monday

2 Tuesday

3 Wednesday

Thursday 4

Friday *Good Friday* 5

Saturday *Easter Saturday* 6

Sunday 7

April 1985

8 *Easter Monday* Monday

9 *Easter Tuesday (Vic)* Tuesday

10 Wednesday

Thursday 11

Friday 12

Saturday 13

Sunday 14

April 1985

15 Monday

16 Tuesday

17 Wednesday

Thursday 18

Friday 19

Saturday 20

Sunday 21

April 1985

22 Monday

23 Tuesday

24 Wednesday

Thursday *Anzac Day* 25

Friday 26

Saturday 27

Sunday 28

April 1985

29 Monday

30 Tuesday

1 May Wednesday

Thursday 2

Friday 3

Saturday 4

Sunday 5

May 1985

6 *May Day (NT)*
Labour Day (Qld) Monday

7 Tuesday

8 Wednesday

Thursday 9

Friday 10

Saturday 11

Sunday 12

May 1985

13 Monday

14 Tuesday

15 Wednesday

Thursday 16

Friday 17

Saturday 18

Sunday 19

May 1985

20 *Adelaide Cup Day (SA)* Monday

21 Tuesday

22 Wednesday

Thursday 23

Friday 24

Saturday 25

Sunday 26

May 1985

27 Monday

28 Tuesday

29 Wednesday

Thursday 30

Friday 31

Saturday June 1

Sunday 2

June 1985

3 *Foundation Day (WA)* Monday

4 Tuesday

5 Wednesday

Thursday 6

Friday 7

Saturday 8

Sunday 9

June 1985

10 *Queen's Birthday (Not WA)* Monday

11 Tuesday

12 Wednesday

Thursday 13

Friday 14

Saturday 15

Sunday 16

June 1985

17 Monday

18 Tuesday

19 Wednesday

Thursday 20

Friday 21

Saturday 22

Sunday 23

June 1985

24 Monday

25 Tuesday

26 Wednesday

Thursday 27

Friday 28

Saturday 29

Sunday 30

July 1985

1 Monday

2 Tuesday

3 Wednesday

Thursday 4

Friday 5

Saturday 6

Sunday 7

July 1985

8 Monday

9 Tuesday

10 Wednesday

...day 11

Friday 12

Saturday 13

Sunday 14

July 1985

15 Monday

16 Tuesday

17 Wednesday

Thursday 18

Friday 19

Saturday 20

Sunday 21

July 1985

22 Monday

23 Tuesday

24 Wednesday

Thursday 25

Friday 26

Saturday 27

Sunday 28

July 1985

29 Monday

30 Tuesday

31 Wednesday

August 1985

Thursday 1

Friday 2

Saturday 3

Sunday 4

August 1985

5 *Bank Holiday (NSW)*
Picnic Day (NT)
Monday

6 Tuesday

7 Wednesday

Thursday 8

Friday 9

Saturday 10

Sunday 11

August 1985

12 Monday

13 Tuesday

14 *Exhibition Day (Brisbane)* Wednesday

Thursday 15

Friday 16

Saturday 17

Sunday 18

August 1985

19 Monday

20 Tuesday

21 Wednesday

Thursday 22

Friday 23

Saturday 24

Sunday 25

August 1985

26 Monday

27 Tuesday

28 Wednesday

Thursday 29

Friday 30

Saturday 31

Sunday September 1

September 1985

2 Monday

3 Tuesday

4 Wednesday

Thursday 5

Friday 6

Saturday 7

Sunday 8

September 1985

9 Monday

10 Tuesday

11 Wednesday

Thursday 12

Friday 13

Saturday 14

Sunday 15

September 1985

16 Monday

17 Tuesday

18 Wednesday

Thursday 19

Friday 20

Saturday 21

Sunday 22

September 1985

23 Monday

24 Tuesday

25 Wednesday

Thursday *Show Day (Melbourne)* 26

Friday 27

Saturday 28

Sunday 29

September 1985

30 Monday

1 October Tuesday

2 Wednesday

Thursday 3

Friday 4

Saturday 5

Sunday 6

October 1985

7 *Labour Day (ACT) (NSW)* Monday

8 Tuesday

9 Wednesday

Thursday 10

Friday 11

Saturday 12

Sunday 13

October 1985

14 *Labour Day (SA)*
Queen's Birthday (WA)
Monday

15 Tuesday

16 Wednesday

Thursday 17

Friday 18

Saturday 19

Sunday 20

October 1985

21 Monday

22 Tuesday

23 Wednesday

Thursday 24

Friday 25

Saturday 26

Sunday 27

October 1985

28 Monday

29 Tuesday

30 Wednesday

Thursday 31

Friday November 1

Saturday 2

Sunday 3

November 1985

4 *Recreation Day (North Tasmania)* Monday

5 *Cup Day (Melbourne)* Tuesday

6 Wednesday

Thursday 7

Friday 8

Saturday 9

Sunday 10

November 1985

11 Monday

12 Tuesday

13 Wednesday

Thursday 14

Friday 15

Saturday 16

Sunday 17

November 1985

18 Monday

19 Tuesday

20 Wednesday

Thursday 21

Friday 22

Saturday 23

Sunday 24

November 1985

25 Monday

26 Tuesday

27 Wednesday

Thursday 28

Friday 29

Saturday 30

Sunday December 1

December 1985

2 Monday

3 Tuesday

4 Wednesday

Thursday 5

Friday 6

Saturday 7

Sunday 8

December 1985

9 Monday

10 Tuesday

11 Wednesday

Thursday 12

Friday 13

Saturday 14

Sunday 15

December 1985

16 Monday

17 Tuesday

18 Wednesday

Thursday 19

Friday 20

Saturday 21

Sunday 22

December 1985

23 Monday

24 Tuesday

25 *Christmas Day* Wednesday

Thursday *Boxing Day (Not SA)* 26

Friday *Additional Day (NT)* 27

Saturday 28

Sunday 29

December 1985

30 *Proclamation Day (SA)* Monday

31 Tuesday

Computers

PET/CBM

Commodore's first computer. Built-in screen and keyboard. Screen 40 or 80 columns by 25 rows. RAM memory varying from 4K (rare) to 32K, not including 1K-2K for screen RAM. Interfaces: two cassette tapes; IEEE for disk, printer, and other devices; user port and memory bus connections available for custom add-on. Two character sets: text and graphics. No high resolution graphics or software customized characters; no color. Later models had built-in sound device. Keyboards vary from early "chiclet" type to full business with numeric pad.
2001: First machine with tiny keyboard, built-in cassette.
4016, 4032: More memory, better file interface logic.
8032: 80 columns, business keyboard.

SUPERPET

SUPER/PET

Partitioned: one option is an 8032-style computer, with an extra 64K of memory "banked" in 4K blocks at address hexadecimal 9000. The other option is a 6809 processor supported by a set of language interpreters from Waterloo University – Assembler, Structured BASIC, Fortran, Pascal, Cobol, and APL. RS-232 interface is built in.

CBM 8096

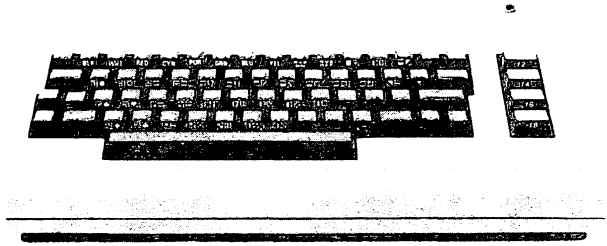
An 8032-style computer, with an extra 64K of memory which "twins" the main computer memory. The extra memory may be switched in blocks of 16K "banks". A large amount of business software has been developed for this machine. Recently, the 8096 has been re-released with new packaging similar to the "B" series (below).

Commodore 16

The enclosure is similar to the C64. Functionally equivalent to the Plus/4 except fitted with only 16K RAM and does not have provision for built in software.

VIC-20

A popular economical computer. Built-in keyboard; screen connects to television or monitor. Screen 22 columns by 23 rows. RAM memory is 5K including 506 bytes for screen RAM; additional memory may be fitted using cartridges. Interfaces: cassette tape; "serial bus" for disk, printer, and other devices; user port which is also available as RS-232 communications port, and a cartridge port (essentially a memory bus expansion) for extra RAM and application ROM cartridges. Two character sets: text and graphics; other character sets may be created in software. High resolution graphics may be achieved using customized characters. Eight colors. Built-in sound with four voices (three tones and noise). Full business keyboard; no numeric pad; four function keys, which may be used with the shift key to allow eight program-detectable functions.



Commodore 64

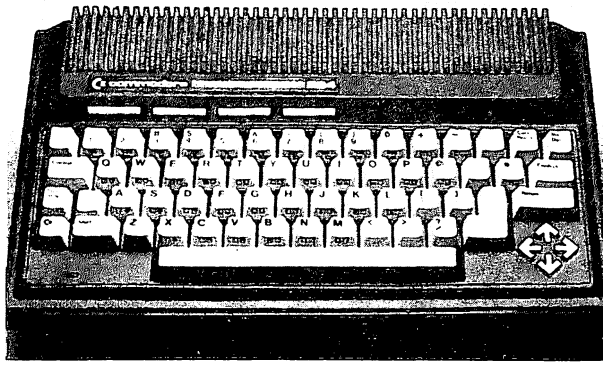
Commodore 64

A versatile large capacity computer. Built-in keyboard; screen connects to television or monitor. Screen 40 columns by 25 rows. RAM memory is 64K including 1K bytes for screen RAM; about 39K is available to BASIC. Interfaces: cassette tape; "serial bus" for disk, printer, and other devices; user port which is also available as RS-232 communications port, and a cartridge port (essentially a memory bus expansion) for application ROM cartridges. Two character sets: text and graphics; other character sets may be created in software. High resolution graphics may be readily achieved; "sprites" allow easy animation of objects across a background. Sixteen colors. Built-in sound with three high-quality voices, any of which may be given highly shaped waveforms, including noise. Full business keyboard; no numeric pad; four function keys, which may be used with the shift key to allow eight program-detectable functions.

B Series

B Series

Also called Series 700 in Europe. Built-in keyboard; adjustable screen built into some models. Screen 80 columns by 25 rows. RAM memory varying from 128K to 256K, possibly more in the future. Interfaces: cassette tape; RS-232 communications; IEEE for disk, printer, and other devices; user port connection available for custom add-on. Memory bus accessible via plug-in cartridge. Two character sets: text and graphics. No high resolution graphics or software customized characters; no color. Built-in sound device. Advanced business keyboard with numeric pad. BASIC language greatly expanded.



Commodore Plus/4

Commodore Plus/4

An efficient application-oriented computer. Built-in keyboard; screen connects to television or monitor. Screen 40 columns by 25 rows. RAM memory is 64K including 2K bytes for color and screen RAM; about 60K is available to BASIC. Interfaces: cassette tape; "serial bus" for disk, printer, and other devices; user port which is also an efficient RS-232 communications port, and a cartridge port (essentially a memory bus expansion) for application ROM cartridges. Four integrated programs built in: File Manager, Spread sheet, Word processing, Graphics. Two character sets: text and graphics; other character sets may be created in software. High resolution graphics may be easily achieved by means of numerous new BASIC graphics commands. Sixteen colors, each with eight "hues", or shades. Built-in sound with three voices. Full business keyboard; numeric pad on a companion machine (the 364). Four function keys, which may be used with the shift key to allow eight functions; functions are pre-defined at start-up and may be redefined by the user. Advanced BASIC with structured features, graphics, error trapping, etc.

Peripherals

Disk units

Two basic styles of 5¼ inch "floppy disk" devices are available: 35 track (4040, 2031, 1541); and 77-track (8050, 8250). All disk are "intelligent"; there is no need to load a "DOS" system into the computer's memory.

35 track Data capacity is approximately 170K bytes (683 blocks), of which 166K bytes (664 blocks) is available for data; the remainder is used for BAM (Block Availability Map) and directory storage. Maximum directory size: 144 entries. Number of sectors per track varies from 17 (track 35) to 21 (track 1). All disks in this series are read compatible, but slight format differences make it advisable to write on the model of machine on which the disk was formatted.

2040, 3040 Early version of the 4040 dual disk. May be upgraded with new ROMs to a 4040.

4040 Dual disk; 35 tracks; dual processor; connects to computer via IEEE cable for moderately fast data transfer. Capable of simultaneously handling 6 sequential files or 3 relative files.

2031 Single disk; 35 tracks; single processor; connects to computer via IEEE cable for moderately fast data transfer. Capable of simultaneously handling 3 sequential files or 1 relative file.

1540, 1541 Single disk; 35 tracks; single processor; connects to computer via Serial bus for fairly slow data transfer. Capable of simultaneously handling 2 sequential files or 1 relative file.

SFS/481 Fast disk with capability of connecting directly to the Plus/4 cartridge port. Disks compatible with 1541; no technical information currently available.

77 track Data capacity of the 8050 is approximately 521K bytes (2083 blocks), of which 513K bytes (2052 blocks) is available for data; on the double-sided 8250, data capacity is 1,041K bytes (4166 blocks), with 1,033K (4133 blocks) available for data. Maximum directory size: 224 entries. Number of sectors per track varies from 23 (track 77) to 29 (track 1). The 8250 is capable of reading 8050 disks; utility programs should be used for copying data.

8050 Dual disk; 77 tracks; dual processor; connects to computer via IEEE cable for moderately fast data transfer. Capable of simultaneously handling 6 sequential files or 3 relative files.

8250 Dual disk; writes 77 tracks on each side of disk with a total of 154 tracks; dual processor; connects to computer via IEEE cable for moderately fast data transfer. Capable of simultaneously handling 6 sequential files or 3 relative files.

Other Peripherals

Printers

Commodore has produced a wide spectrum of printers; space does not allow detailing of characteristics. All Commodore printers recognize Commodore's special "PETSCII" code (interface translation is needed for other printers); most have the ability to perform PRINT/USING type functions, print graphics characters, and other functions.

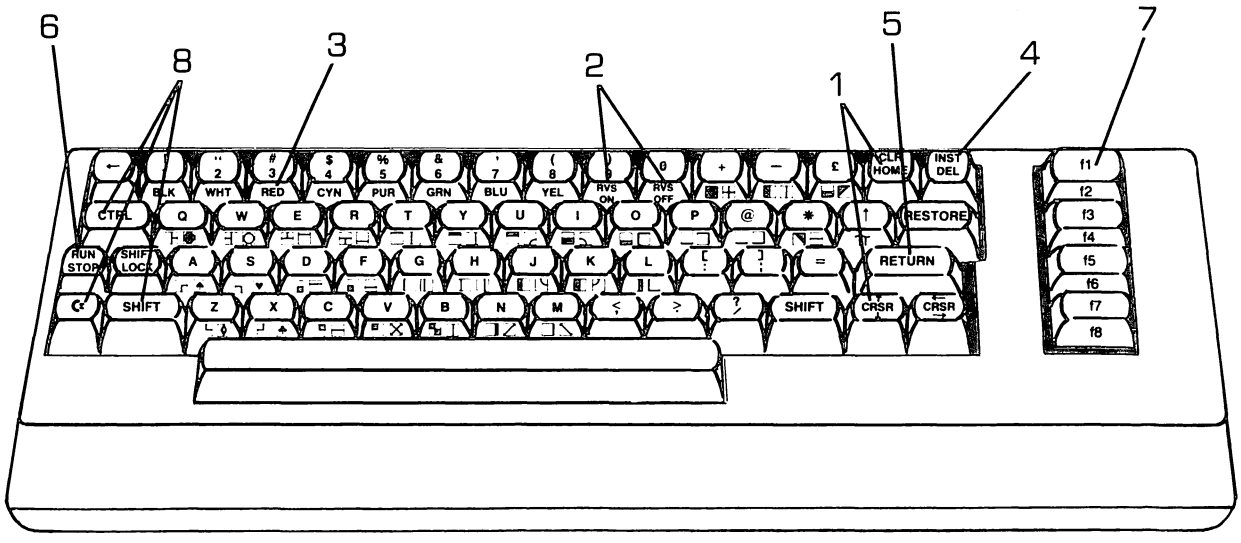
Modems

A wide variety of modem devices has been produced by Commodore and others for communications purposes. On PET/CBM, the modems are usually linked either by the IEEE bus or fitted directly to the user port with appropriate driving software. On SuperPET, B series, VIC-20, Commodore 64, and Commodore Plus/4, an RS-232 port is furnished for easier modem connection.

Others

Joysticks, paddles, graphics tablets, light pens and speech synthesizers may be readily fitted to the VIC-20 and Commodore 64 series. Commodore cassette tape units ("datasettes") may be attached for storage of programs and sequential data files.

Keyboard Layout



Important keys

1. **Cursor positioning and screen clear** The cursor up, down, left and right keys, together with the home and clear keys are used for screen convenience. "Screen editing" allows mistakes to be corrected by moving the cursor to the offending line, correcting the error, and pressing RETURN.

2. **Reverse, Reverse-Off** When RVS is pressed, following text will be typed in reverse font until RVS-OFF or until a new line is started.

3. **Colors** Pressing the appropriate key combination will cause following text to be typed in the selected color.
Note on Programmed Cursor: The above keys behave differently if pressed within quotation marks or as part of an "insert" sequence. Instead of being actioned immediately, they are recorded as programmed characters, and appear as reverse font characters. Such programmed characters will normally generate the appropriate action only when the line is actioned, say, by a RUN statement. If they appear at an unwanted time, the line may be exited by holding down SHIFT and pressing RETURN; programmed cursor mode will be terminated.

4. **Insert, Delete [INST/DEL]** Insert will open up space to the right of the cursor to allow insertions. The user may type Insert several times to allow several characters to be inserted. Delete moves the cursor left; it carries with it all text to the right of the cursor.

5. **RETURN** causes the computer to leave the current line and execute whatever commands were on that line. SHIFT-RETURN causes the computer to leave the current line with no other action.

6. **STOP** causes a BASIC program, listing, or disk load to stop. The computer will often report BREAK. RUN (Shifted STOP) will cause the computer to seek to load a program from tape or disk and run it.

7. **Function Keys** on VIC-20 and Commodore 64 are detected but normally cause no action unless a program is running and tests for these keys. On B systems and Commodore Plus/4 they are predefined with useful functions, and may be redefined if desired.

8. **Shift, Control [CTRL] and Commodore** keys are used in combination with other keys to allow selection of certain characters or functions. The RESTORE key on VIC-20 and Commodore 64 will usually cause the machine to reset to the ready state if pressed with the RUN/STOP key held down.

9. **ESC** (escape) is used on business keyboards only to cancel special modes, such as insert mode, or quotes mode. TAB may be used to tabulate to a given screen position; when shifted, it sets and resets tab stops.

BASIC Structure

BASIC ("Beginner's All-Purpose Symbolic Instruction Code") arranges instructions as a series of lines. Each line is numbered; when a new line is entered it will take its place within the program at the appropriate number sequence. If a previous line bears the same number, the new line will replace it. When the command RUN is given, the first line in sequence is executed, then the next, and so on. Certain instructions (GOTO, FOR ... NEXT, etc) may change the order in which lines are executed.
BASIC statements and commands may often be given without a line number prefix. These are "direct" statements; they are executed immediately instead of being stored. They are a good way to test an instruction to see how it works.

BASIC Values

Basic uses two general types of data: **constants**, such as 23.5 or "HELLO", which do not change; and **variables**, such as X or N3$, whose values may be changed if desired. If a constant is used often in a program, it may be useful to define it within a variable, even though it will not change: variables are usually more compact and faster than constants.

BASIC Variables

Variables may be defined or changed by two general methods: **assignment** ("LET X = 18.5 + 5") or **input** ("READ N3$"). A variable name may have several characters, but only the first two are significant; the user should avoid conflict with BASIC keywords ("SNIFF" contains keyword "IF"). There are three types of BASIC variable:

1. **Floating Point**, which will hold numeric values to about ten places of accuracy; very large and very small values, including fractions, can be held in this format.
2. **Integer**, which will hold whole numbers ranging between −32,767 and +32,767; integer variables are identified by a % symbol following the name.
3. **String**, which will hold words or collections of characters up to a maximum of 255 characters. String variables are identified by a $ symbol following the name.

BASIC Arrays

Collections of variables ("lists" or "tables") may be held as an array. The specific item within the list is designated by one or more **subscripts**. Thus, M$(4) is the fourth item in a list of strings called M$( ); T(3,6) is the number at row three, column six of a table of values called T( ). Strings may be floating point, integer, or string. It is wise to define the size of the array by using a DIM (dimension) statement near the start of a program; otherwise, BASIC will size the array dimension to ten units. A subscript of zero is valid in all arrays.

Special Variables

Certain "permanent" variables contain system values:

ST Status; 0 indicates that the last file input or output occurred uneventfully; 64 indicates that the last file input was the end of the data; anything else signals a problem.

TI Time in "jiffies" – a jiffy is 1/60 second. Resets when 24 hours is reached. (Not available on B series.)

TI$ Same time as TI, but expressed as a string in hours, minutes and seconds. This variable may be set by program or direct statement.

On 4.0, B series and Plus/4 only:

DS Disk status (numeric). Zero if the last disk activity was uneventful. Affected only by the Basic DOS commands (e.g., DOPEN, PRINT#, etc.).

DS$ Disk status (string). Descriptive disk status message.

On B series and Commodore Plus/4 only:

EL, ER, ERR$ Used with error trapping. EL is error line, ER is error type, and ERR$ is an error function.

BASIC Commands

Most BASIC keywords may be used within a program or as part of a direct statement. Those most often used with a program are called **statements**; those most often in direct statements are called **commands**. The keywords which must be used as part of a statement are called **operators** or **functions**.

LIST	Display the BASIC lines being held in memory
NEW	Clear all BASIC lines from memory
CLR	Scrap all program variables
RUN	Execute the BASIC lines in memory
CONT	Continue the program after a STOP or BREAK

LOAD Bring in a new program from disk or tape
SAVE Store a program to disk or tape
VERIFY Compare a program on disk or tape to memory

Extra commands

4.0 PET/CBM BASIC, the B series, and the Plus/4 have additional disk (DOS) commands: see the following section, "Disk Commands" for these. The B series and Plus/4 also have extra editing and control commands, not given here.

BASIC Statements

Control statements: These change the order in which instructions are executed.

FOR .. NEXT Statements between FOR and NEXT are repeated until the variable named has stepped through its specified value range.

IF .. THEN .. If the conditional statement following IF is true, perform the statements following THEN; otherwise pass to the next line.

IF .. GOTO .. If the conditional statement following IF is true, go to the specified line; otherwise pass to the next line. THEN may be used instead of GOTO in this sequence.

GOSUB .. RETURN Go to the specified line and perform the statements there. When RETURN is encounterd, come back to the statement following GOSUB.

ON .. GOSUB Based on the value following ON (which must be in the range 0 to 255), call one of the subroutines following GOSUB.

ON .. GOTO Based on the value following ON (which must be in the range 0 to 255), go to one of the lines named following GOTO.

GOTO ... Go to the line specified and continue execution there.

STOP, END Stop the program. If STOP, print the line number.

SYS Call a machine language subroutine at the address given.

WAIT Pause execution until a specified memory location achieves a particular value.

I/O Statements: These receive or transmit DATA to or from the program.

INPUT Receive information from a line on the screen.

GET Receive one character of information from the keyboard buffer. If no character is waiting, receive a null string.

READ Take information from the next DATA item within the program.

RESTORE Reset back to the first DATA statement within the program.

PRINT Send information to the screen.

OPEN Give details on a logical file: a device we will want to work with.

CMD Switch all output from the screen to the logical file specified. Cancelled by any "#" command, below.

INPUT# Receive from the specified logical file until a RETURN character is seen.

GET# Receive one character from the specified logical file. If the character is binary zero, receive a null string.
PRINT# Send information to the specified logical file.
CLOSE Wind up the logical file.
Other Statements:
LET Sets a variable to a given value. The word LET may always be omitted.
POKE Change a memory location to a given value.
DIM Sets the size of an array.
REM Does nothing; used for a programmer's remarks.
DEF Define a user function.
DATA Does nothing; provides DATA for READ usage.

BASIC Operators and Functions

+ − * / add, subtract, multiply, divide, power
= < > equals, less than, greater than - may be used in combination
AND, OR, NOT logical relationships
+ (strings) concatenation, joining together

Mathematical Functions

ABS( ) Absolute (unsigned) value
ATN( ) Arctan, inverse tangent
COS( ) Cosine (radians)
EXP( ) Exponential, e to the power
INT( ) Integer
LOG( ) Logarithm, base e
RND(1) Random fraction between 0 and 1
SGN( ) Sign: −1, 0 or +1
SIN( ) Sine (radians)
SQR( ) Square Root
TAN( ) Tangent (radians)

String Functions

ASC( ) ASCII number of the first character in the string
CHR$( ) Character which has the given ASCII number
VAL( ) Numeric value of the string
STR$( ) String representing a numeric value
LEN( ) Length of the string
LEFT$( ) Characters from the left of the string
RIGHT$( ) Characters from the right of the string
MID$( ) Characters from within the string

Special Functions

FRE(0) Amount of memory available to BASIC
PEEK( ) Contents of specified memory location
FNx( ) User function previously defined with DEF
USR( ) Machine language function evaluation

Disk Commands

There are three ways of sending commands to disk: **simply**, to secondary address 15 using a PRINT# statement; by means of a small program called **"DOS wedge"**, where commands are preceded by symbol > or @; or by means of **"advanced"** BASIC commands available in PET 4.0, B systems, and Commodore Plus/4.
Examples of all three methods will be shown here. PRINT# commands should be preceded with an OPEN 15,8,15 statement and followed by CLOSE 15. '@' commands require the preloading of the DOS wedge program.

Examining the disk directory

Simple LOAD "$0",8 followed by LIST (this destroys any BASIC program in memory)
Wedge @$0
Advanced DIRECTORY or CATALOG

Interrogating a disk error

Simple Write a small program:
```
1 OPEN 15,8,15: INPUT#15,E,
E$,E1,E2: CLOSE 15:
PRINT E; E$;E1;E2: STOP.
```
When the program is entered, RUN.
Wedge @
Advanced PRINT DS$

Initializing a diskette (not usually necessary)

Simple PRINT#15,"I0"
Wedge @I0
Advanced no command available.

Formatting a new disk

Simple PRINT#15,"N0:
DISK NAME,ID"
Wedge @N0:DISK NAME,ID
Advanced HEADER D0,
"DISK NAME",ID

Clearing a disk

Simple PRINT#15,
"N0: NEW NAME"
Wedge @N0:NEW NAME
Advanced HEADER D0,
"NEW NAME"

Validating (rebuilding the BAM)

Simple PRINT#15,"V0"
Wedge @V0
Advanced COLLECT D0

Scratching (removing) a file

Simple PRINT#15,"S0:OLDFILE"
Wedge @S0:OLDFILE
Advanced SCRATCH
"OLDFILE"
Note: Don't scratch an "unclosed" file, one which shows an asterisk near its name. Use "Validate" to remove it.

Renaming a file

Simple PRINT#15,"R0:
NEWNAME=OLDNAME"
Wedge @R0:
NEWNAME=OLDNAME
Advanced RENAME
"OLDNAME" TO "NEWNAME"

Copying a file [example shows dual disk]

```
Simple  PRINT#15,"C1:
 FILE=0:FILE"
Wedge   @C1:FILE=0:FILE
Advanced  COPY "FILE" TO
 "FILE",D1
```

Combining files [example shows single disk]

```
Simple  PRINT#15,"C0:
 FILE1=0:FILE1,0:FILE2"
Wedge   @C0:
 FILE1=0:FILE1,0:FILE2
Advanced  CONCAT "FILE2" TO
 "FILE1"
```

Disk Backup/Duplicate [dual disk only]

```
Simple  PRINT#15,"D1=0"
Wedge   @D1=0
Advanced  BACKUP D0 TO D1
```

Loading a program

```
Simple  LOAD "0:PROGNAME",8
Wedge   /0:PROGNAME
Advanced  DLOAD
 "PROGNAME"
```

Saving a program

```
Simple  SAVE "0:PROG2"
Wedge     0 = PROGNAME
Advanced  DSAVE "PROG2"
```

Opening an input file

```
Simple  OPEN 1,8,2,"FILENAME"
Advanced
 DOPEN#1,"FILENAME"
```

Reading from an input file

```
(string) INPUT#1,A$
(number) INPUT#1,A
```

Opening an output file

```
Simple  OPEN 1,8,2,"0:
 NEWFILE,S,W"
Advanced
 DOPEN#1,"NEWFILE",W
```

Writing to an output file

```
(string) PRINT#1,A$
(number) PRINT#1,A
```

Closing a file

```
Simple  CLOSE 1
Advanced  DCLOSE (closes all
 files)
```

Simple Techniques and Handy Locations

To switch to text mode (upper/lower case): PET/CBM: POKE 59468,14. 4.0 PET/CBM and all subsequent machines: PRINT CHR$(14).

To switch to graphics mode: PET/CBM: POKE 59468,12. 4.0 PET/CBM and all subsequent machines: PRINT CHR$(142).

To lock out inadvertent keyboard switching between text and graphics via the keyboard (VIC-20, C64 and subsequent): PRINT CHR$(8). To restore keyboard control: PRINT CHR$(9).

Disabling the RUN/STOP key: This can be done on most machines by increasing the contents of the interrupt vector by a value of three; it has the side effect of also disabling the jiffy (TI) clock. Original ROM PET: POKE 537,139. Upgrade ROM PET: POKE 144,49. 4.0 ROM PET: POKE 144,88. VIC-20: POKE 788,194. C64: POKE 788,52. Restore by replacing the original value.

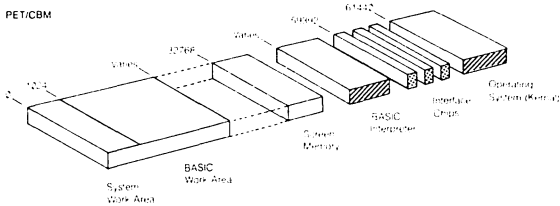
Disabling RUN/STOP and RESTORE: On VIC-20 and Commodore 64, this can be done by decreasing the contents of the STOP vector by three; it has the side effect of scrambling the LIST command. VIC-20: POKE 808,109. C64: POKE 808,234.

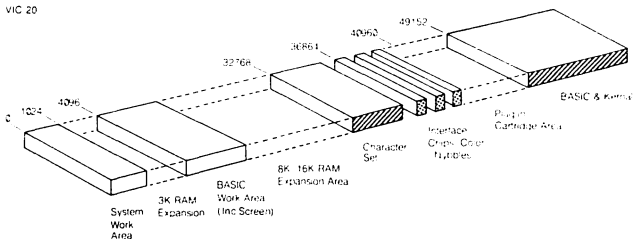
Cold start: All Commodore computers can be reset to power-up state with SYS(PEEK(65532) + PEEK(65533) * 256). B system will require a preceding BANK 15 statement. If desired, the numeric value for the SYS can be worked out for each machine.

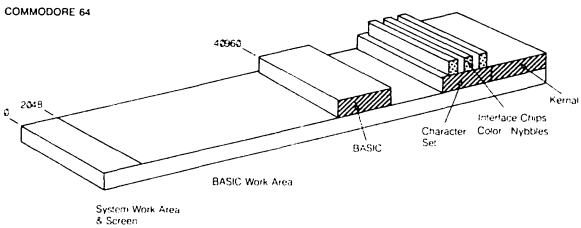
Obstructing LIST: On many Commodore computers, inserting a line containing REM followed by Shift-L and/or Shift-[(left square bracket) will cause LIST to abort at this line. This can be defeated by removing the line, but inserting several such lines makes the job harder.

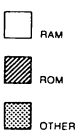
Omitting line numbers: A curious POKE causes line numbers to be dropped during a LIST activity. This can be used when listing to the printer to achieve a "poor man's word processor". Original PET: POKE 65,78. Other PET/CBMs: POKE 19,32. VIC-20 and Commodore 64: POKE 22,35.

Memory Architectures



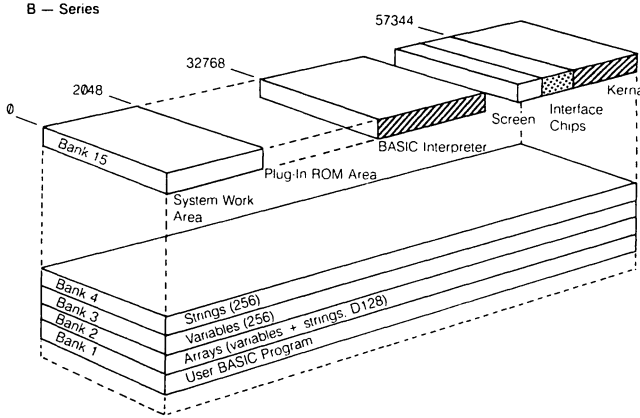




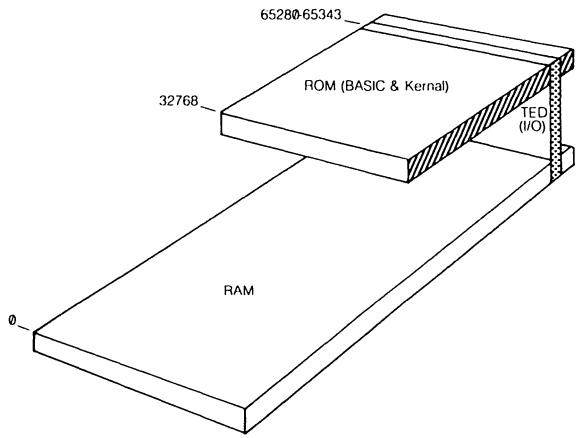


(Not drawn to scale.)

B — Series



Commodore Plus/4 Series



RAM

ROM

OTHER

(Not drawn to scale)

Useful Memory Locations

UPGRADE and BASIC 4.0 SYSTEMS

Where Upgrade ROM differs from 4.0, an asterisk (*) is shown and the 4.0 value is given. There are some differences in usage between the 40- and 80-column machines.

Hex	Decimal	Description
0000-0002	0-2	USR jump
0011-0012	17-18	Integer value (for SYS, GOTO and so on)
0028-0029	40-41	Pointer: start-of-BASIC
002A-002B	42-43	Pointer: start-of-variables
002C-002D	44-45	Pointer: start-of-arrays
002E-002F	46-47	Pointer: end-of-arrays
0030-0031	48-49	Pointer: string-storage (moving down)
0034-0035	52-53	Pointer: limit-of-memory
003C-003D	60-61	Current DATA line number
003E-003F	62-63	Current DATA address
005E	94	Accum#1: exponent
005F-0062	95-98	Accum#1: mantissa
0063	99	Accum#1: sign
0065	101	Accum#1 hi-order (overflow)
0066-006B	102-107	Accum#2: exponent, and so on
006C	108	Sign comparison, Acc#1 versus #2
0070-0087	112-135	CHRGET subroutine; get BASIC character
0077-0078	119-120	BASIC pointer (within subroutine)
008D-008F	141-143	Jiffy clock for TI and TI$
0090-0091	144-145	IRQ vector
0092-0093	146-147	BRK interrupt vector
0094-0095	148-149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key down: 255 = no key
0098	152	Shift key: 1 if depressed
009B	155	Keyswitch PIA: STOP and RVS flags
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00C4-00C5	196-197	Pointer to screen line
00C6	198	Position of cursor on above line
00D1	209	Number of characters in file name
00D2	210	Current file logical address

Hex	Decimal	Description
00D3	211	Current file secondary address
00D4	212	Current file device number
00D8	216	Line where cursor lives
00DA-00DB	218-219	File name pointer
00E3	227	*(80-column) Limit of keyboard buffer
00E4	228	*(80-column) Key repeat flag
00E9-00EA	233-234	*(80-column) Input vector
00EB-00EC	235-236	*(80-column) Output vector
00F9-00FA	249-250	Cassette status, #1 and #2
0251-025A	593-602	File logical address table
025B-0264	603-612	File device number table
0265-026E	613-622	File secondary address table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 input buffer
033A-03F9	826-1017	Tape#2 input buffer
033A-0380	826-896	*DOS work area
03EB	1003	(Fat 40) Keyboard buffer limit
03EE	1006	(Fat 40) Key repeat flag
03EE-03F7	1006-1015	(80-column) Tab stop table
03EF	1007	(Fat 40) Tab work value
03F0-9	1008-1017	(Fat 40) Tab stops
03FA-03FB	1018-1019	Monitor extension vector
03FC	1020	*IEEE timeout defeat
0400-7FFF	1024-32767	Available RAM including expansion
8000-83E7	32768-33767	(40-column) Video RAM
8000-87CF	32768-34767	*(80-column) Video RAM
9000-AFFF	36864-45055	Available ROM expansion area
B000-E7FF	45056-59391	BASIC ROM, part of kernal
E810-E813	59408-59411	PIA 1 - Keyboard I/O
E820-E823	59424-59427	PIA 2 - IEEE-488 I/O
E840-E84F	59456-59471	VIA - I/O and timers
E880-E881	59520-59521	(80-column and Fat 40) CRT controller
F000-FFFF	61440-65535	Kernal ROM

E810	Diagnostic Sense	IEEE EOI In	Cassette Sense #2	#1	Keyboard Row Select	PA	59408
E811	Tape #1 Input Flag		EOI Out	CA2	DDRA Access	Cassette #1 Read Control CA1	59409
E812	Keyboard Row Input						59410
E813	Retrace I Flag		Cassette #1 Motor Output	CB2	DDRB Access	Retrace Interrupt Control CB1	58411

E820	IEEE Input						59424
E821	ATN I Flag		IEEE NDAC Out	CA2	DDRA Access	IEEE ATN In Control CA1	59425
E822	IEEE Output						59426
E823	SRQ I Flag		IEEE DAV Out	CB2	DDRB Access	IEEE SRQ In Control CB1	59427

E840	DAV In	NRFD In	Retrace In	Cass. #2 Motor	Cassette Output	ATN Out	NRFD Out	NDAC In PB	59456
E841	Parallel User Port (PUP.) I/O with Handshake								59457
E842	Data Direction Register B (for E840)								59458
E843	Data Direction Register A (for E84F, PUP.)								59459
E844	Timer 1							L	59460
E845								H	59461
E846	Timer 1 Latch							L	59462
E847								H	59463
E848	Timer 2							L	59464
E849								H	59465
E84A	Shift Register								59466
E84B	T1 Control PB7 Out (one-shot / free-run)		T2 Ctrl PB6 Sense	Shift Register Control			PB, PA Latch Control		59467
E84C	CB2 (PUP. Pin M) Control In/Out			CB1 In, Cassette #2 Polarity	CA2 (Graphics, Lower Case) In/Out			CA1 In Polarity	59468
E84D	IRQ Status	T1 INT	T2 INT	CB1 Cass #2 INT	CB2 INT	SRQ INT	CA1 (PUPB) INT	CA2 INT	59469
E84E	Enable Clear/Set	T1 INT Enab	T2 INT Enab	CB1 INT Enab	CB2 INT Enab	SRQ INT Enab	CA1 INT Enab	CA2 INT Enab	59470
E84F	Parallel User Port I/O (PA)							PA	59471

6545 CRT Controller

D800 55296	D801 55297	Typical Value (Decimal)
0	Horizontal Total	108 or 126 or 127
1	Horizontal Char Displayed	80
2	Horizontal Sync Position	83 or 98 or 96
3	Sync Width V H	15 or 10
4	Vertical Total	25 or 31 or 38
5	Vert Total Adjust	3 or 6 or 1
6	Vertical Displayed	25
7	Vert. Sync Position	25 or 28 or 30
8	Mode	0
9	Scan Lines	13 or 7
10	Cursor Start	96 (blink) or 0 or 6 (underline)
11	Cursor End	13 or 7
12	Display Address H	0
13	Display Address L	0
14	Cursor Address H	Varies
15	Cursor Address L	Varies
16	Light Pen In H	0
17	Light Pen In L	0

Most Register are Write Only 14/15 are Read/Write
16/17 are Read Only
Registers 10, 14 and 15 change as the cursor moves

Hex	Decimal	Description
0000-0002	0-2	USR jump (VIC-20)
0000-0001	0-1	Memory map controls (C64)
0003-0004	3-4	Float-fixed vector
0005-0006	5-6	Fixed-float vector
0014-0015	20-21	Integer value
002B-002C	43-44	Pointer: start-of-BASIC
002D-002E	45-46	Pointer: start-of-variables
002F-0030	47-48	Pointer: start-of-arrays
0031-0032	49-50	Pointer: end-of-arrays
0033-0034	51-52	Pointer: string-storage (moving down)
0037-0038	55-56	Pointer: limit-of-memory
003F-0040	63-64	Current DATA line number
0041-0042	65-66	Current DATA address
0061	97	Accum#1: exponent
0062-0065	98-101	Accum#1: mantissa
0066	102	Accum#1: sign
0068	104	Accum#1 hi-order (overflow)
0069-006E	105-110	Accum#2: exponent, and so on
006F	111	Sign comparison, Acc#1 versus #2
0073-008A	115-138	CHRGET subroutine; get BASIC character
007A-007B	122-123	BASIC pointer (within subroutine)
0090	144	Status word ST
0091	145	Keyswitch PIA: STOP and RVS flags
0098	152	How many open files
0099	153	Input device, normally 0
009A	154	Output CMD device, normally 3
00A0-00A2	160-162	Jiffy Clock HML
00B7	183	Number of characters 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
00C0	192	Tape motor interlock
00C6	198	Number of characters in keyboard buffer
00CB	203	Which key: 64 if no key
00D1-00D2	209-210	Pointer to screen line
00D3	211	Position of cursor on above line
00D6	214	Row where cursor lives
0200-0258	512-600	BASIC input buffer
0259-0262	601-610	Logical file table
0263-026C	611-620	Device number table
026D-0276	621-630	Secondary address table
0277-0280	631-640	Keyboard buffer

Hex	Decimal	Description
0286	646	Current color code
0288	648	Screen memory page
0289	649	Maximum size of keyboard buffer
028A	650	Repeat all keys
0292	658	0 = scroll enable
0300-0301	768-769	Error message link
0302-0303	770-771	BASIC warm start link
0304-0305	772-773	Crunch BASIC tokens link
0306-0307	774-775	Print tokens link
0308-0309	776-777	Start new BASIC code link
030A-030B	778-779	Get arithmetic element link
0310-0312	784-786	USR Jump (C64 only)
0314-0315	788-789	IRQ vector (EABF)
0316-0317	790-791	Break interrupt vector (FED2)
0318-0319	792-793	NMI interrupt vector (FEAD)
031A-031B	794-795	OPEN vector (F40A)
031C-031D	796-797	CLOSE vector (F34A)
031E-031F	798-799	Set-input vector (F2C7)
0320-0321	800-801	Set-output vector (F309)
0322-0323	802-803	Restore I/O vector (F3F3)
0324-0325	804-805	INPUT vector (F20E)
0326-0327	806-807	Output vector (F27A)
0328-0329	808-809	Test-STOP vector (F770)
032A-032B	810-811	GET vector (F1F5)
032C-032D	812-813	Abort I/O vector (F3EF)
032E-032F	814-815	USR vector (FED2)
0330-0331	816-817	LOAD link
0332-0333	818-819	SAVE link
033C-03FB	828-1019	Cassette buffer

For more on the Commodore 64 see page 132.

VIC-20 Only:

Hex	Decimal	Description
0400-0FFF	1024-4095	3K RAM expansion area
1000-1FFF	4096-8191	Normal BASIC memory
2000-7FFF	8192-32767	Memory expansion area
8000-8FFF	32768-36863	Character bit maps (ROM)
9000-900F	36864-36879	Video interface chip (6560)
9110-912F	37136-37151	VIA (6522) interface - NMI
9120-912F	37152-37167	VIA (6522) interface - IRQ
9400-95FF	37888-38399	Alternate colour nybble area
9600-97FF	38400-38911	Main colour nybble area
A000-BFFF	40960-49151	Plug-in ROM area
C000-FFFF	49152-65535	ROM: BASIC and operating system

Commodore 64 Only:

Hex	Decimal	Description
0400-07FF	1024-2047	Screen memory
0800-9FFF	2048-40959	BASIC ROM memory
8000-9FFF	32768-40959	Alternative: ROM plug-in area
A000-BFFF	40960-49151	ROM: BASIC
A000-BFFF	49060-59151	Alternate: RAM
C000-CFFF	49152-53247	RAM memory, including alternative
D000-D02E	53248-53294	Video chip (6566)
D400-D41C	54272-54300	Sound chip (6581 SID)
D800-DBFF	55296-56319	Color nybble memory
DC00-DC0F	56320-56335	Interface chip 1, IRQ (6526 CIA)
DD00-DD0F	56576-56591	Interface chip 2, NMI (6526 CIA)
D000-DFFF	53248-53294	Alternative: character set
E000-FFFF	57344-65535	ROM: operating system
E000-FFFF	57344-65535	Alternative: RAM

6560 VIC Chip

9000	Interlace	Left Margin (=5)			36864
9001	Top Margin (=25)				36865
9002	Screen Ad Bit 9	Number of Columns (=22)			36866
9003	Bit 0	Number of Rows (=23)		Double Char	36867
9004	Input Raster Value: Bits 1-8				36868
9005	Screen Address Bits 13-10		Character Address Bits 13-10		36869
9006	Light Pen Input		Horizontal		36870
9007			Vertical		36871
9008	Paddle Input		X		36872
9009			Y		36873
900A	ON	Voice 1 Frequency			36874
900B	ON	Voice 2 Frequency			36875
900C	ON	Voice 3 Frequency			36876
900D	ON	Noise Frequency			36877
900E	Multi Colour Mode		Sound Amplitude		36878
900F	Background Colour		Foregnd/ Backgnd	Border Colour	36879

6522 VIA 1

9110	DSR In	CTS In		DCD* In	RI* In	DTR Out	RTS Out	Data In	37136
	RS-232 Interface or Parallel User Port								
9111	*Unused - see $911F								37137
9112	Data Direction Register B (for $9110)								37138
9113	Data Direction Register A (for $911F)								37139
9114	T1-L	RS 232 Send Speed; Tape Write Timing							37140
9115	T1-H								37141
9116	T1-Latch L								37142
9117	T1 Latch H								37143
9118	T2-L	RS 232 Input Timing							37144
9119	T2-H								37145
911A	Shift Register (* unused)								37146
911B	T1 Control		T2 Ctrl	Shift Register Control			PB LE	PA LE	37147
911C	CB2: RS 232 Send			CB1 Ctrl	CA2: Tape Motor Ctrl			CA1 Ctrl	37148
911D	NMI:	T1	T2	CB1: RS 232 In			CA1: [RESTORE]		37149
911E	NMI En.	T1 Enab	T2 Enab	CB1 En.			CA1 En.		37150
911F	ATN Out	Tape Sense	Joystick Fire	Left	Down	Up	Serial Data In	Serial Clock In	37151

6522 VIA 2

9120	Joystick Right				Tape Out				37152
	Keyboard Row Select								
9121	Keyboard Column Input								37153
9122	Data Direction Register B (for $9120)								37154
9123	Data Direction Register A (for $9121)								37155
9124	T1-L	Cassette Tape Read; Keyboard and Clock Interrupt Timing							37156
9125	T1-H								37157
9126	T1-Latch L								37158
9127	T1 Latch H								37159
9128	T2-L	Serial Bus Timing							37160
9129	T2-H	Tape R/W Timing							37161
912A	Shift Register (* unused)								37162
912B	T1 Control		T2 Ctrl	Shift Register Control			PB LE	PA LE	37163
912C	Serial Bus Data Out			CB1 Ctrl	Serial Clock Line Out			CA1 Ctrl	37164
912D	IRQ:	T1	T2	CB1: SRQ In			CA1: Tape In		37165
912E	IRQ En.	T1 Enab	T2 Enab	CB1 En.			CA1 En.		37166
912F	*Unused (see $9121)								37167

6566 Video Chip
C64 Control & Miscellaneous Registers

D011		Extended Clr. Mode	Display Map	Bit Enable	Row Select	Y-Scroll	53265
D012	Raster Register						53266
D013	Light Pen Input					X	53267
D014						Y	53268

D016	x	x	Reset	Multi Colour	Column Select	X-Scroll	53270

D018	Screen				Character Base			x	53272
	VM13	VM12	VM11	VM10	CB13	CB12	CB11		
D019	IRQ	**Interrupt Sense:**			Light Pen	Spr-Spr Collision	Spr-Back Collision	Raster	53273
D01A	**Interrupt Enable:**				Light Pen	Spr-Spr Collisions	Spr-Back Collisions	Raster	53274

Colour Registers

D020	X	Exterior Colour (Border)	53280
D021	X	Background Colour #0	53281
D022	X	Background Colour #1	53282
D023	X	Background Colour #2	53283
D024	X	Background Colour #3	53284
D025	X	Sprite MultiColour #0	53285
D026	X	Sprite MultiColour #1	53286

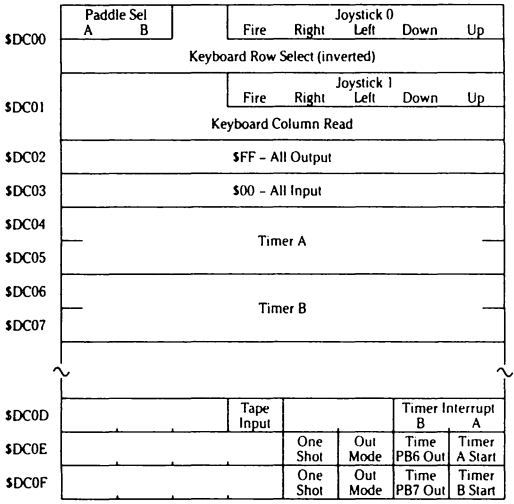
6566 Video Chip
C64 Sprite Registers

Sprite 0	Sprite 7		Sprite 0	Sprite 7
D000	D00E	X Position	53248	53262
D001	D00F	Y Position	53249	53263
D027	D02E	Colour	53287	53294

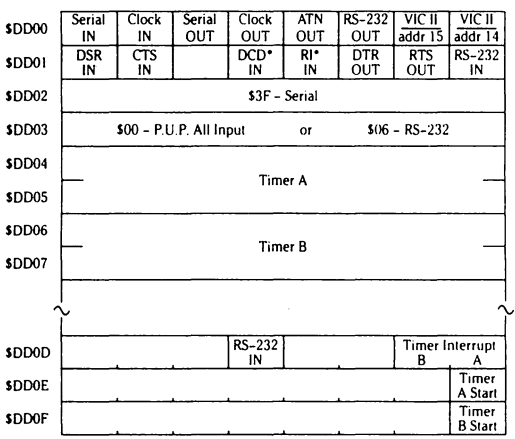
Bit For Sprite#:

	7	6	5	4	3	2	1	0	
D010	X-Position High								53264
D015	Sprite Enable Flags								53269
D017	Y-Expand								53271
D01B	Background Priority								53275
D01C	Sprite MultiColour Mode								53276
D01D	X-Expand								53277
D01E	Interrupt: Sprite Collision								53278
D01F	Interrupt: Background Collision								53279

CIA 1 (IRQ) (6526)



CIA 2 (NMI) (6526)



* Connected but not used by O.S.

Processor I/O Port (6510)

$0000	IN	IN	OUT	IN	OUT	OUT	OUT	OUT	DDR	0
$0001			Tape Motor	Tape Sense	Tape Write	D-ROM Switch	EF RAM Switch	AB RAM Switch	PR	1

SID (6581)

Voice 1	Voice 2	Voice 3			Voice 1	Voice 2	Voice 3
$D400	$D407	$D40E	Frequency	L	54272	54279	54286
$D401	$D408	$D40F		H	54273	54280	54287
$D402	$D409	$D410	Pulse Width	L	54274	54281	54288
$D403	$D40A	$D411	0 0 0 0	H	54275	54282	54289
$D404	$D40B	$D412	Voice Type: NSE, PUL, SAW, TRI	Key	54276	54283	54290
$D405	$D40C	$D413	Attack Time: 2ms - 8ms / Decay Time: 6ms - 24 sec		54277	54284	54291
$D406	$D40D	$D414	Sustain Level / Release Time: 6ms - 24 sec		54278	54285	54292

Voices (write only)

$D415	0 0 0 0 0	L	54293
$D416	Filter Frequency	H	54294
$D417	Resonance / Filter Voices: Ext, V3, V2, V1		54295
$D418	V3 off, Passband: HI, BP, LO / Master Volume		54296

Filter & Volume (write only)

$D419	Paddle X (A/D #1)	54297
$D41A	Paddle Y (A/D #2)	54298
$D41B	Noise 3 (random)	54299
$D41C	Envelope 3	54300

Sense (read only)

Note: Special Voice Features
(TEST, RING MOD, SYNC)
are omitted from the above diagram.

COMMODORE Plus/4

Preliminary

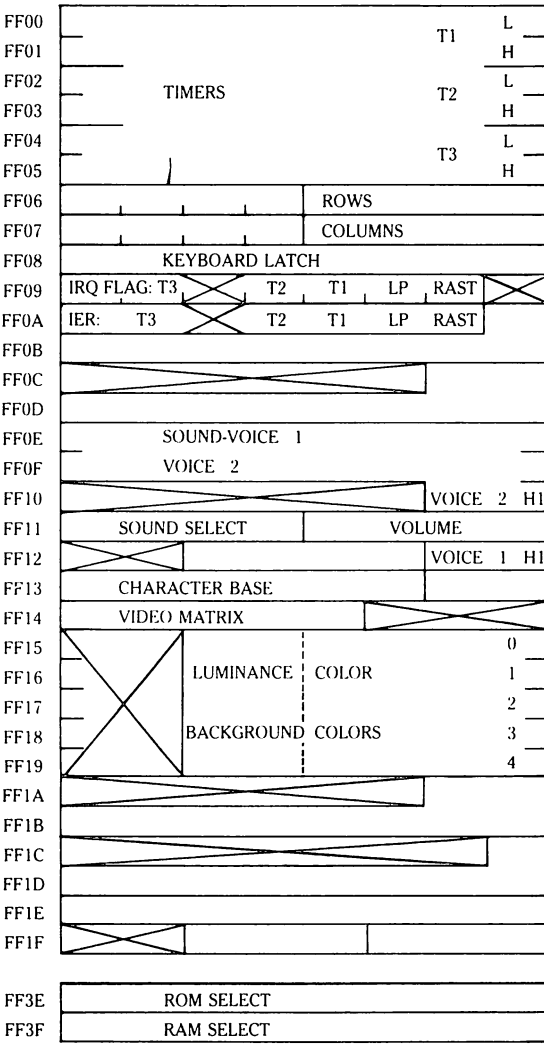
Most of zero page is the same as for the Commodore 64. Some differences, and other information:

Hex	Decimal	Description
0073-008A	115-138	CHRGET not present
0097	151	How many open files
0098	152	Input device, normally 0
0099	153	Output CMD device, normally 3
00AC	172	Current logical file
00AD	173	Current secondary address
00AE	174	Current device
00AF-00B0	175-176	Pointer to file name
00C8-00C9	200-201	Pointer to screen line
00CA	202	Position of cursor on above line
00CD	205	Row where cursor lives
00EF	239	Number of characters in keyboard buffer
0314-0315	788-789	IRQ vector (CE0E)
0316-0317	790-791	Break interrupt vector (F44B)
0318-0319	792-793	OPEN vector (EF53)

(Most other vectors are similar to the C64, but are two locations lower)

0500-0502	1280-1282	USR program jump
0509-0512	1289-1298	Logical file table
0513-051C	1299-1308	Device number table
051D-0526	1309-1318	Secondary address table
0527-0530	1319-1328	Keyboard buffer
0800-0BE7	2048-3047	Color memory
0C00-0FE7	3072-4071	Screen memory
1000-FFFF	4096-65535	BASIC RAM memory
8000-FFFF	32768-65535	ROM: BASIC
FF00-FF3F	65280-65343	TED I/O control chip

Commodore Plus/4
"TED" Chip — Preliminary



B SERIES (B128, B256)

All locations mapped here are in bank 15.

Hex	Decimal	Description
0002-0004	2-4	USR jump
001B-001C	27-28	Integer value
002D-002E	45-46	Start-of-BASIC pointer
002F-0030	47-48	End-of-BASIC pointer
0031-0032	49-50	Start-of-Variables pointer
0033-0034	51-52	End-of-Variables pointer
0035-0036	53-54	Start-of-Arrays pointer
0037-0038	55-56	End-of-Arrays pointer
0039-003A	57-58	Variable work pointer
003B-003C	59-60	Bottom-of-Strings pointer
003F-0041	63-65	Top of string memory pointer
0049-004A	73-74	Data line number
004B-004C	75-76	Data text pointer
0070	112	Accum string prefix
0071	113	Accum#1: exponent
0072-0075	114-117	Accum#1: mantissa
0076	118	Accum#1: sign
0078	120	Accum#1 hi order (overflow)
0079-007E	121-126	Accum#2
007F	127	Sign comparison, Acc#1 versus #2
0085-0087	133-135	BASIC text pointer
0090-0092	144-146	Pointer to file name
009C	156	Status word ST
009D	157	File name length
009E	158	Current logical file
009F	159	Current device
00A0	160	Current secondary address
00A1	161	Input device, normally 0
00A2	162	Output CMD device, normally 3
00A9	169	Keyswitch PIA: stop key, etc.
00C8-00C9	200-201	Pointer to screen line
00CA	202	Screen line number
00CB	203	Positition of cursor on line
00D1	209	Number of keys in keyboard buffer
00E1	225	Key pressed: 255 = no key
0280-0281	640-641	Error routine link [8555]
0282-0283	642-643	Warm start link [85CD]
0284-0285	644-645	Crunch token link [88C2]
0286-0287	646-647	List link [89F4]
0288-0289	648-649	Command dispatch link [8754]
028A-028B	650-651	Token evaluate link [96B1]
028C-028D	652-653	Expression eval link [95C4]

Hex	Decimal	Description
028E-028F	654-655	CHRGOT ink [BA2C]
0290-0291	656-657	CHRGET vector [BA32]
0292-0293	658-659	Float-fixed vector [BA1E]
0294-0295	660-661	Fixed-Float vector [9D39]
0296-0297	662-663	Error trap vector
0298-0299	664-665	Error line number
029A-029B	666-667	Error exit pointer
0300-0301	768-769	IRQ vector [FBE9]
0302-0303	770-771	BRK vector [EE21]
0304-0305	772-773	NMI vector [FCAA]
0306-033D	774-829	System vectors
0334-033D	820-829	File logical addresses table
033E-0347	830-839	File device table
0348-0351	840-849	File secondary address table
0360	864	Number of open files
03A1-03AA	929-938	Bit mapped tab stops
03AB-03B4	939-948	Keyboard input buffer
03F8-03F9	1016-1017	Restart vector
03FA-03FB	1018-1019	Restart test mask
0400-07FF	1024-2047	Free RAM (reserved for DOS)
0800-0FFF	2048-4095	Reserved for plug in RAM
1000-1FFF	4096-8191	Reserved for plug in DOS ROM
2000-7FFF	8192-23767	Reserved for cartridges
8000-BFFF	32768-49151	BASIC ROM
C000-CFFF	49152-53247	Unused
D000-D7CF	53248-55247	Screen RAM
D800-D801	55296-55297	Video controller 6545
DA00-DA1C	55808-55836	Sound interface device 6581
DB00-DB0F	56064-56079	Complex interface adaptor 6526
DC00-DC0F	56320-56335	Complex interface adaptor 6526
DD00-DD03	56576-56579	Asynchronous communications IA 6551
DE00-DE07	56832-56839	Tri Port Interface Adaptor 6525
DF00-DF07	57088-57095	Tri Port Interface Adaptor 6525
E000-FFFF	57344-65535	Kernal ROM

The above table shows contents for the link and vector addresses at $0280 to $0295; these are taken from a recent B-128.

6525 Tri Port

DE00	NRFD	NDAC	EOI	DAV	ATN	RFN			56832
DE01	Sense	Cassette Motor	Out	ARB	Network Rx	Tx	SRQ	IFC	56833
DE02									56834
DE03	Data Direction Register For DE00								56835
DE04	Data Direction Register For DE01								56836
DE05	IRQ:			ACIA	IP	ALM	IEEE	PWR	56837
DE06	CB		CA: Graphics				IRQ Stack On		56838
DE07	Active Interrupt Register								56839

6525 Tri Port 2

DF00	Keyboard		57088
DF01	Select		57089
DF02	CRT Mode	Keyboard Read	57090
DF03	Data Direction Register for DF00 (out)		57091
DF04	Data Direction Register for DF01 (out)		57092
DF05	Data Direction Register for DF02 (in)		57093
DF06	Unused		57094

6581 SID

DA01	Voice 1 Frequency High							55809
DA04			Saw Tooth		Ring Mod		Key	55812
DA05	Attack			Decay				55813
DA06	Sustain			Release				55814
DA0F	Voice 3 Modulating Freq Hi							55823
DA18				Volume				55832

6526 CIA 1

DB00	Inter-Processor Data						56064
DB01	X	IRQ Out	X	X	SEMAPH	Busy	56065
DB02	Data Direction Register For DB00						56066
DB02	Data Direction Register For DB01						56067
	Unused						
DB0D			IP Flag				56077
DB0E	Unused						56078
DB0F	Unused						56079

6526 CIA 2

DC00	IEEE Data In/Out	56320
DC01	User Port	56321
DC02	Data Direction Register For DC00	56322
DC02	Data Direction Register For DC01	56323
	Unused	
DC06	Timer B — L	56326
DC07	Timer B — H	56327
DC08	Time Of Day Clock (TOD) — 1/10 Sec.	56328
DC09	Time Of Day Clock (TOD) — Sec.	56329
DC0A	Time Of Day Clock (TOD) — Min.	56330
DC0B	Time Of Day Clock (TOD) — Hour	56331
DC0C	Unused	56332
DC0D	Alarm	56333
DC0E	Unused	56334
DC0F	TOD / Timer Force / Timer Start	56335

SuperChart: VIC 20 / Commodore 64

DECIMAL	HEX	ASCII	SCREEN	BASIC	DECIMAL	HEX	ASCII	SCREEN	BASIC
0	00		@	end-line	30	1E	green	↑	
1	01		A		31	1F	blue	←	
2	02		B		32	20	space	space	space
3	03	stop	C		33	21	!	!	!
4	04		D		34	22	"	"	"
5	05	white	E		35	23	#	#	#
6	06		F		36	24	$	$	$
7	07		G		37	25	%	%	%
8	08	lock	H		38	26	&	&	&
9	09	unlock	I		39	27	'	'	'
10	0A		J		40	28	(	(	(
11	0B		K		41	29	)	)	)
12	0C		L		42	2A	*	*	*
13	0D	car ret	M		43	2B	+	+	+
14	0E	text	N		44	2C	,	,	,
15	0F		O		45	2D	-	-	-
16	10		P		46	2E	.	.	.
17	11	cur down	Q		47	2F	/	/	/
18	12	reverse	R		48	30	0	0	0
19	13	cur home	S		49	31	1	1	1
20	14	delete	T		50	32	2	2	2
21	15		U		51	33	3	3	3
22	16		V		52	34	4	4	4
23	17		W		53	35	5	5	5
24	18		X		54	36	6	6	6
25	19		Y		55	37	7	7	7
26	1A		Z		56	38	8	8	8
27	1B		[		57	39	9	9	9
28	1C	red	\		58	3A	:	:	:
29	1D	cur right	]		59	3B	;	;	;

DECIMAL	HEX	ASCII	SCREEN	BASIC
60	3C	<	<	<
61	3D	=	=	=
62	3E	>	>	>
63	3F	?	?	?
64	40	@	@	@
65	41	A	a	A
66	42	B	b	B
67	43	C	c	C
68	44	D	d	D
69	45	E	e	E
70	46	F	f	F
71	47	G	g	G
72	48	H	h	H
73	49	I	i	I
74	4A	J	j	J
75	4B	K	k	K
76	4C	L	l	L
77	4D	M	m	M
78	4E	N	n	N
79	4F	O	o	O
80	50	P	p	P
81	51	Q	q	Q
82	52	R	r	R
83	53	S	s	S
84	54	T	t	T
85	55	U	u	U
86	56	V	v	V
87	57	W	w	W
88	58	X	x	X
89	59	Y	y	Y
90	5A	Z	z	Z
91	5B	[	[	[
92	5C	£	£	£
93	5D	]	]	]
94	5E	↑	↑	↑
95	5F	←	←	←
96	60			
97	61			
98	62			
99	63			
100	64			
101	65			
102	66			
103	67			
104	68			
105	69			
106	6A			
107	6B			
108	6C			
109	6D			
110	6E			
111	6F			
112	70			
113	71			
114	72			
115	73			
116	74			
117	75			
118	76			
119	77			
120	78			
121	79			
122	7A			

DECIMAL	HEX	ASCII	SCREEN	BASIC
123	7B			
124	7C			
125	7D			
126	7E			
127	7F			
128	80		@	END
129	81	orange	A	FOR
130	82		B	NEXT
131	83	load & run	C	DATA
132	84		D	INPUT#
133	85	f1	E	INPUT
134	86	f2	F	DIM
135	87	f3	G	READ
136	88	f4	H	LET
137	89	f5	I	GOTO
138	8A	f6	J	RUN
139	8B	f7	K	IF
140	8C	f8	L	RESTORE
141	8D	car ret	M	GOSUB
142	8E	graphics	N	RETURN
143	8F		O	REM
144	90	black	P	STOP
145	91	cur up	Q	ON
146	92	rvs off	R	WAIT
147	93	clear	S	LOAD
148	94	insert	T	SAVE
149	95	brown	U	VERIFY
150	96	lt. red	V	DEF
151	97	dk. grey	W	POKE
152	98	md. grey	X	PRINT#
153	99	lt. green	Y	PRINT
154	9A	lt. blue	Z	CONT
155	9B	lt. grey	[	LIST
156	9C	magenta	\	CLR
157	9D	cur left	]	CMD
158	9E	yellow	↑	SYS
159	9F	cyan	←	OPEN
160	A0			CLOSE
161	A1		!	GET
162	A2		"	NEW
163	A3		#	TAB(
164	A4		$	TO
165	A5		%	FN
166	A6		&	SPC(
167	A7		'	THEN
168	A8		(	NOT
169	A9		)	STEP
170	AA		*	+
171	AB		+	-
172	AC		,	*
173	AD		-	/
174	AE		.	↑
175	AF		/	AND
176	B0		0	OR
177	B1		1	>
178	B2		2	=
179	B3		3	<
180	B4		4	SGN
181	B5		5	INT
182	B6		6	ABS
183	B7		7	USR
184	B8		8	FRE
185	B9		9	POS

DECIMAL	HEX	ASCII	SCREEN	BASIC
186	BA	□,☑		SQR
187	BB	◧		RND
188	BC	◨		LOG
189	BD	◰		EXP
190	BE	◱		COS
191	BF	◲		SIN
192	C0	▤		TAN
193	C1	▩,a		ATN
194	C2	▥,b		PEEK
195	C3	▤,c		LEN
196	C4	▥,d		STR$
197	C5	□,e		VAL
198	C6	▤,f		ASC
199	C7	▥,g		CHR$
200	C8	▥,h		LEFT$
201	C9	▥,i		RIGHT$
202	CA	▥,j		MID$
203	CB	▥,k		GO
204	CC	□,l		
205	CD	☒,m		
206	CE	☒,n		
207	CF	□,o		
208	D0	□,p		
209	D1	▩,q		
210	D2	□,r		
211	D3	▩,s		
212	D4	□,t		
213	D5	☒,u		
214	D6	☒ v		
215	D7	□,w		
216	D8	▩,x		
217	D9	□,y		
218	DA	▩,z		
219	DB	⊞		
220	DC	◧		

Reverse of ASCII

DECIMAL	HEX	ASCII	SCREEN	BASIC
221	DD	▥		
222	DE	▥,▩		
223	DF	◥,▧		
224	E0		▩	
225	E1		▮	
226	E2		▬	
227	E3		▩	
228	E4		▩	
229	E5		▮	
230	E6		▩	
231	E7		▩	
232	E8		▩	
233	E9		◢,▨	
234	EA		▩	
235	EB		▩	
236	EC		▩	
237	ED		▩	
238	EE		▩	
239	EF		▩	
240	F0		▩	
241	F1		▩	
242	F2		▩	
243	F3		▩	
244	F4		▮	
245	F5		▮	
246	F6		▮	
247	F7		▬	
248	F8		▬	
249	F9		▬	
250	FA		▩ ▩	
251	FB		▛	
252	FC		▙	
253	FD		▩	
254	FE		▟	
255	FF		▜	π

Useful Programs

Detecting key hold

The GET command detects a key and extracts it from the keyboard buffer; but if the key is held down, GET won't see it again. All machines have a PEEK location that tells whether or not a key is being held down; but it won't tell you reliably which key. The two may be used together to give a continuous key effect. The PEEK location is: Original PET: 515. Other PET/CBM: 151. VIC-20 and Commodore 64: 203. If no key is being pressed, the PET/CBM locations will give a value of 255; the VIC-20 and Commodore 64 will give 64. The following VIC/64 program demonstrates typical handling: observe the order of tests carefully.

```
100 PRINT "KEY HOLD DEMO"
110 PRINT "HOLD DOWN H FOR HIGHER VALUE"
120 PRINT "HOLD DOWN L FOR LOWER VALUE"
130 PRINT "PRESS S TO STOP."
140 PRINT
150 V=500
160 PRINT CHR$(145);"VALUE =";STR$(V);" "
170 K=PEEK(203)
180 GET K$
190 IF K$="H" THEN X=1:GOTO 240
200 IF K$="L" THEN X=-1:GOTO 240
210 IF K$="S" THEN END
220 IF K<64 GOTO 240
230 X=0
240 V=V+X
250 GOTO 160
```

Simple PET Emulator

This brief program for the Commodore 64 will cause it to take on the PET/CBM's BASIC memory configuration. Subsequently loaded programs will relocate into PET-compatible memory space. Such programs may then be saved, and will load satisfactorily to the PET or any other machine. They may or may not work correctly in the new machine.

```
1 POKE 56576,5:POKE 53272,4:POKE 648,128: POKE 1024,0:
POKE 44,4:POKE 56,128: PRINT CHR$(147):NEW
```

To fit the above line onto the screen, it may be necessary to eliminate the spaces and abbreviate PRINT as "?".

Outputting in Columns

Don't use TAB( ) or POS( ) or "comma punctuation" within the PRINT statement; these will not transfer easily to a printer. Don't use the printer's built-in PRINT USING function; this won't allow you to debug to the screen or transfer to other printers. Use string functions, padding out with spaces and trimming to size. The following program illustrates:

```
100 DATA JOE, INCOMPREHENSIBLE,5
110 DATA ROCHESTER,CAT,134
120 DATA GLOOM,DOOM,-2
200 REM:  print in columns
210 S$ = " "
220 FOR J = 1 TO 3
230 READ A$,B$,N
240 PRINT LEFT$(A$ + S$,8);" ";LEFT$(B$ + S$,8);" ";
250 PRINT RIGHT$(S$ + STR$(N),6)
260 NEXT J
```

Numbers in Columns

This subroutine extends the above logic for numbers with fractional parts. Set V1 to equal the number of characters to be printed before the decimal point and V2 to the number of characters after. V1 + V2 should not exceed nine; BASIC's numeric accuracy starts to fall off at this point.

```
50000 REM:  ARRANGE IN COLUMNS
50010 REM:  V IS VALUE; V1 & V2 POSITIONS
50020 V4 = INT(V*10 ↑ V2 + .5)
50030 V$ = RIGHT$(" " + STR$(V4),V1 + V2 + 2)
50040 IF V2 < 1 GOTO 50080
50050 FOR V5 = V1 + 2 TO V1 + V2 + 1
50060 IF ASC(MID$(V$,V5)) < 48 THEN NEXT V5
50070 V6 = V5 - V1 - 1
50080 V$ = MID$(V$,V6,V1 + 1) + LEFT$(".00000",V6) + MID$(V$,V5)
50090 IF ASC(V$) > 47 THEN V$ = LEFT$("**********",
  V1 + V2 + 2 + (V2 = 0))
50100 RETURN
```

The following program will demonstrate the above subroutine. It prints each number in three different formats. One number won't fit.

```
100 REM:  'USING' ARRANGE IN COLUMNS
110 DATA 123,3.33333, 6.66666, -45.25555
120 DATA 9999, .01234, -.5678,0
```

```
130 READ V
140 V1 = 4: V2 = 0: GOSUB 50000: PRINT V$;" ";
150 V1 = 3: V2 = 1: GOSUB 50000: PRINT V$;" ";
160 V1 = 3: V2 = 3: GOSUB 50000:PRINT V$
170 IF V< >0 GOTO 130
180 END
```

Calendar Dates

This program will convert a date between 1901 and 2099 into a chronological number (days elapsed since October 31, 1899), or vice versa. Month codes: 1 = January, 2 = February, etc. Week-day codes: 1 = Sunday, 2 = Monday, etc.

```
100 REM: Test July 4, 1985
110 M = 7:D = 4:Y = 1985
120 GOSUB 300
130 GOSUB 200
140 REM: ..equals day 31294
150 M = 0:D = 0:Y = 0
160 GOSUB 400
170 GOSUB 200
190 END
200 REM: print result
210 PRINT "MONTH";M;"DAY"; D;Y
220 PRINT "IS DAY NUMBER";N
230 PRINT "AND IS WEEKDAY";W
240 PRINT
250 RETURN
300 REM: Convert Month, Day, Year
310 REM: ..to Numbered day
320 M1 = M + 1:M2 = INT(1/M1 + .7)
330 M3 = Y-M2-1900: M4 = M1 + 12*M2
340 N = INT(M4*30.6001) + INT(M3*365.25) + D
350 M6 = INT(N/7):W = N-7*M6 + 1
360 RETURN
400 REM: Convert Numbered day
410 REM: ..to Month, Day, Year
420 M3 = INT((N-122.1)/365.25)
430 M5 = N-INT(M3*365.25): M4 = INT(M5/30.6001)
440 D = M5-INT(M4*30.6001)
450 M1 = INT(M4/14): Y = M3 + M1 + 1900:
    M = M4-12*M1-1
460 M6 = INT(N/7):W = N-7*M6 + 1
470 RETURN
```

The program converts dates by changing the normal year to an "offset year" which starts on March 1. Variables: M, D, and Y are the month, day and year; N is the chronological day number; W is the weekday number. Within the calculation: M2 is 1 for January and February, zero otherwise. M3 is the adjusted year, and M4 the adjusted month (ranging from 4 for March to 15 for February). M5 is the number of days into the adjusted year, and M6 is the number of weeks from "zero date".

To find out how many days have elapsed between any two dates—run the program once with one date in line 110. Run the program a second time changing to the second date. The answer is the difference in the results of the two program runs.

To find out what date is 100 days after July 4, 1985, add line 145:

```
145 N = N + 100
```

File Inspection

A "conventional" sequential file may be viewed on the screen with the following program, which may be used with either tape or disk.

```
Tape: 100 OPEN 1
Disk: 100 OPEN 1,8,2,"0: FILENAME"
All:  110 INPUT#1,A$:PRINT A$: IF/ST=0 GOTO 110
      120 CLOSE 1
```

If it is desired to output to the printer, ST must be handled more carefully:

```
Tape: 100 OPEN 1
Disk: 100 OPEN 1,8,2,"0: FILENAME"
All:  110 OPEN 4,4
      120 INPUT#1,A$:S=ST: PRINT#4,A$:IF S=0 GOTO 110
      130 CLOSE 1
```

Where a file is suspected of containing "unusual" characters, the INPUT# statement must be abandoned in favor of GET#, and the PRINT must be dealt with in greater detail:

```
100 PRINT "FILE LISTER - JIM BUTTERFIELD"
110 OPEN 15,8,15
120 INPUT "NAME OF FILE";F$
130 OPEN 1,8,3,F$
140 INPUT#15,E,E$,E1,E2
150 IF E<>0 THEN PRINT E$:CLOSE 15:RUN
160 INPUT "TO PRINTER";P$
170 P=3:IF ASC(P$)=89 THEN P=4
180 OPEN 3,P
190 Z$=CHR$(0)
```

```
200 GET#1,A$:SW=ST:IF A$="" THEN A$=Z$
210 A=ASC(A$):B=A AND 127
220 IF A=34 THEN B=99
230 D$=A$
240 IF B<32 THEN D$=CHR$(18) +CHR$(A+64)+CHR$(146)
250 IF B>95 THEN D$=CHR$(18) +STR$(A)+CHR$(146)
260 PRINT#3,D$;:IF B=13 THEN PRINT#3
270 IF SW=0 GOTO 200
280 CLOSE 1
290 PRINT#3:CLOSE 3
300 CLOSE 15
```

Converting numbers to and from hexadecimal

```
100 INPUT "DECIMAL NUMBER"; N
110 IF N<>INT(N) THEN PRINT "NO FRACTIONS": GOTO 100
120 IF N<0 OR N>65535 THEN PRINT "RANGE!": GOTO 100
130 D=4:F=4096
140 IF N>255 GOTO 160
150 D=2:F=16
160 V=N/F
170 FOR J=1 TO D
180 V%=V:V=(V-V%)*16
190 IF V%>9 THEN V%=V%+7
200 PRINT CHR$(V%+48);
210 NEXT J
220 PRINT
400 INPUT "HEXADECIMAL NUMBER";H$
410 V=0:FOR J=1 TO LEN(H$)
420 D=ASC(MID$(H$,J))
430 IF D<58 THEN D=D-48
440 IF D>64 THEN D=D-55
450 IF D<0 OR D>15 THEN PRINT "?": GOTO 400
460 V=V*16+D
470 NEXT J
480 PRINT "DECIMAL =";V
```

Rounding:

To **round down** a value V to the next lower integer, use INT(V).

To **round** a value V to the nearest integer, use INT(V + .5).

To **round up** a value V to the next higher integer, use INT(V + .9999).

Musical Notes:

Values given here for VIC-20 and Commodore-64 are valid for North American machines. Because of different clock frequencies, they must be modified for Europe.

To create sound on the VIC, turn up the volume (POKE 36878,15) and then POKE to the appropriate voice address: 36874, 36875, 36876, and 36877 for noise. The value to be POKEd should be taken from the table below; the specific octave will vary with the voice. To create sound on the C64, turn up the volume (POKE 54296,15), set the voice's ADSR characteristics (POKE 54277,0; POKE 54278,240 for voice 1), POKE the appropriate set of values from the table below to the frequency registers (POKE 54272,xx; POKE 54273,xx for voice 1). Finally, POKE the waveform value plus 1 to the waveform location (POKE 54276,17) to activate the sound; then POKE the waveform value to the same location to deactivate the sound.

Waveform table

	Activate	Deactivate
Triangle (soft)	17	16
Sawtooth (sharp)	33	32
Pulse (thin or brassy)	65	64
Noise (crashes, etc)	129	128

The pulse waveform requires you to set the pulse width.

Frequency Table

Note	VIC	C64		Note	VIC	C64	
		LOW	HIGH			LOW	HIGH
C-2	135	48	4	C-4	225	195	16
C#-2	143	112	4	C#-4	227	195	17
D-2	147	180	4	D-4	228	209	18
D#-2	151	251	4	D#-4	229	239	19
E-2	159	71	5	E-4	231	31	21
F-2	163	152	5	F-4	232	96	22
F#-2	167	237	5	F#-4	233	181	23
G-2	175	71	6	G-4	235	30	25
G#-2	179	167	6	G#-4	236	156	26
A-2	183	12	7	A-4	237	49	28
A#-2	187	119	7	A#-4	238	223	29
B-2	191	233	7	B-4	239	165	31
C-3	195	97	8	C-5	240	135	33
C#-3	199	225	8	C#-5	241	134	35
D-3	201	104	9	D-5		162	37
D#-3	203	247	9	D#-5	242	223	39
E-3	207	143	10	E-5	243	62	42
F-3	209	48	11	F-5		193	44
F#-3	212	218	11	F#-5	244	107	47
G-3	215	143	12	G-5	245	60	50
G#-3	217	78	13	G#-5		57	53
A-3	219	24	14	A-5	246	99	56
A#-3	221	239	14	A#-5		190	59
B-3	223	210	15	B-5	247	75	63

Sound may be generated on the PET/CBM machines by POKE 59467,16 (turn on sound); POKE 59466,15 (set tone); and POKE 52464,90 (set frequency). Sound must be turned off with POKE 59467,0 before performing input/output.

Color Codes

	VIC-20	C64	Plus/4	ASCII
Black	0	0	1	144
White	1	1	2	5
Red	2	2	3	28
Cyan	3	3	4	159
Purple	4	4	5	156
Green	5	5	6	30
Blue	6	6	7	31
Yellow	7	7	8	158
Orange	8*	8	9	129
Brown		9	10	149
Light Orng	9*			
Pink	10*	10	12	150

	VIC-20	C64	Plus/4	ASCII
Yellow/Grn			11	150
Dark Grey		11		151
Light Cyan	11*			
Med Grey		12		152
Lt Purple	12*			
Blue/Green			13	152
Lt Green	13*	13	16	153
Lt Blue	14*	14	14	154
Dark Blue			15	154
Lt Grey		15		155
Lt Yellow	15*			

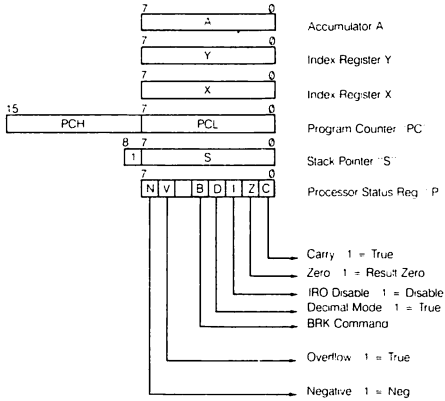
* Not available for characters

Numbers given for VIC and C64 are POKE values; those for Plus/4 are used in COLOR commands. ASCII values may be used to change color by means of PRINT CHR$(30) type statements.

Checklist for sprite generation

1. If necessary, set the 16K video slice with POKE 56576.
2. Sketch out the sprite on a 24x21 grid. Select a 64-byte "drawing" slot and POKE the 63 bytes of your sketch there. (FOR J=832 TO 894: POKE J,15:NEXT J).
3. Signal in the screen area what drawing the selected sprite is to use. (POKE 2040,13).
4. Give the sprite a color (POKE 53287,7) and X- and Y- positions (POKE 53248,50: POKE 53249,60).
5. If desired, set X- or Y-expansion (POKE 53271,1: POKE 53277,1), background priority, and multicolor.
6. Enable the sprite and it will appear (POKE 53269,1).
7. Remember to turn the sprite off before loading another program or doing other I/O.

Processor Programming Model



Mde	IMM	ZPg	Z,X	(I,X)	(I),Y	ABS	A,X	A,Y
Byts	2	2	2	2	2	3	3	3
ORA	09	05	15	01	11	0D	1D	19
AND	29	25	35	21	31	2D	3D	39
EOR	49	45	55	41	51	4D	5D	59
ADC	69	65	75	61	71	6D	7D	79
STA		85	95	81	91	8D	9D	99
LDA	A9	A5	B5	A1	B1	AD	BD	B9
CMP	C9	C5	D5	C1	D1	CD	DD	D9
SBC	E9	E5	F5	E1	F1	ED	FD	F9
Op Code ends in -1, -5, -9, or -D								

Mde	IMM	ZPg	Z,X	ABS	A,X
Byts	2	2	2	3	3
BIT		24		2C	
STY		84	94	8C	
LDY	A0	A4	B4	AC	BC
CPY	C0	C4		CC	
CPX	E0	E4		EC	
Op Code ends in -0, -4, or -C					

Branches -0			
BPL	10	BMI	30
BVC	50	BVS	70
BCC	90	BCS	B0
BNE	D0	BEQ	F0

Jumps		
Mde	ABS	(IND)
JSR	20	
JMP	4C	6C

Mde	IMM	ZPg	Z,X	Z,Y	ABS	A,X	A,Y
Byts	2	2	2	2	3	3	3
ASL		06	16		0E	1E	
ROL		26	36		2E	3E	
LSR		46	56		4E	5E	
ROR		66	76		6E	7E	
STX		86		96	8E		
LDX	A2	A6		B6	AE		BE
DEC		C6	D6		CE	DE	
INC		E6	F6		EE	FE	
Op Code ends in -2, -6, or -E							

Single Byte Op Codes (* Accumulator Mode)															
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	BRK				RTI		RTS									
-8	PHP	CLC	PLP	SEC	PHA	CLI	PLA	SEI	DEY	TYA	TAY	CLV	INY	CLD	INX	SED
-A	ASL*		ROL*		LSR*		ROR*		TXA	TXS	TAX	TSX	DEX		NOP	

MCS65XX Microprocessor
Instruction Set Alphabetic Sequence

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 overflow 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 acculmulator 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.
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.

Addressing Modes

Accumulator Addressing - This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

Immediate Addressing - In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing - In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65k bytes of addressable memory.

Zero Page Addressing - The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instructions and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Zero Page Addressing - (X, Y Indexing) - This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing - (X, Y Indexing) - This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes on the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing - In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing - Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.
The second byte of the instruction becomes the operand which is an "offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

Indexed Indirect Addressing - In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of the addition points to a memory location on page zero whose contents is the low order eights bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

Indirect Indexed Addressing - In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

Absolute Indirect - The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address which is loaded into the sixteen bits of the program counter.

Important Kernal Subroutines

The Commodore Reference Guides have an overwhelming list of kernal subroutines. The beginning user should start with three BASIC ones:

CHROUT: JSR $FFD2 will cause the ASCII character in the A register to be delivered to the output (normally the screen). All data registers (X, Y, and A) will be unchanged by the subroutine. Cursor movement and control characters will behave in the usual way.

GETIN: JSR $FFE4 will cause the input channel (normally the keyboard buffer) to be interrogated for an ASCII character; the character will appear in the A register. If no character is available when interrogating keyboard or RS-232, a binary zero will be delivered to A. All data registers (X, Y, and A) may be changed.

STOP: JSR $FFE1 will cause a test of the STOP key. If it is being held down at that instant, the computer will set the Z flag; if not, it will clear the Z flag. Thus, BEQ will detect stop key activity. Data register A will be changed.

When Input/Output become important, the user will expand to three more subroutines. Files may still be OPENed and CLOSEd in BASIC; the following routines will switch input or output to a previously opened logical file so that GETIN or CHROUT will receive from the designated file:

CHKIN: LDX (logical file number): JSR $FFC6 will switch the input channel from its normal default (keyboard) to the specified logical file. Subsequent calls to GETIN will get from this file until a call to CLRCHN (below) restores I/O to normal. The contents of the A register will be affected.

CHKOUT: LDX (logical file number): JSR $FFC9 will switch the output channel from its normal default (screen) to the specified logical file. Subsequent calls to CHROUT will send to this file until a call to CLRCHN (below) restores I/O to normal. The contents of the A register will be affected.

CLRCHN: JSR $FFCC will restore input and output channels to their normal default settings (keyboard in, screen out). It does not close the file; it just disconnects. There should be one CLRCHN to match each CHKIN and CHKOUT call. The contents of the A and X registers will be affected.

File Opening and Closing

It is usually much more convenient to do this in BASIC. On occasion, if it is necessary to open and close in machine language, VIC, C64 and recent machines work well with the following sequences (no details are given here): SETLFS ($FFBA), SETNAM ($FFBD), followed by OPEN ($FFC0); CLOSE ($FFC3). The PET/CBM differs in its use of these routines; check carefully if moving from one machine to another.

Hexadecimal Conversion Chart

Hex	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	-00	-000
0-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1-	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2-	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3-	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4-	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5-	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6-	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7-	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8-	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9-	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A-	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B-	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C-	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D-	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E-	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F-	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

Bit Values

Bit	Dec	Hex
0	1	$0001
1	2	0002
2	4	0004
3	8	0008
4	16	0010
5	32	0020
6	64	0040
7	128	0080
8	256	0100
9	512	0200
10	1024	0400
11	2048	0800
12	4096	1000
13	8192	2000
14	16384	4000
15	32768	8000

CALENDAR 1985

JANUARY

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

FEBRUARY

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

MARCH

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

APRIL

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

MAY

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

JUNE

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

JULY

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

AUGUST

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

SEPTEMBER

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

OCTOBER

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

NOVEMBER

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

DECEMBER

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

CALENDAR 1986

JANUARY

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

FEBRUARY

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

MARCH

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

APRIL

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	39	30			

MAY

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

JUNE

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

JULY

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

AUGUST

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

SEPTEMBER

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

OCTOBER

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

NOVEMBER

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

DECEMBER

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Commodore State Offices

New South Wales (Head Office)
5 Orion Road
Lane Cove 2066

Phone: (02) 427 4888

Victoria
Unit 13/1 663 Victoria Street
Abbotsford 3067

Phone: (03) 429 9855

Queensland
991 Stanley Street
East Brisbane 4169

Phone: (07) 393 0300

Western Australia
198 Daly Street
Belmont 6104

Phone: (09) 478 1744

Pitman
ISBN 0 85896 196 2