PEEK A BYTE™ 64

AN ESSENTIAL DISK & MEMORY UTILITY
FOR THE COMMODORE 64™ & DRIVE

By

Dr. Philip A. Slaymaker

QUANTUM SOFTWARE
P.O. BOX 12716
LAKE PARK, FL 33403

[illegible]

PEEK A BYTE™ 64

AN ESSENTIAL DISK & MEMORY UTILITY FOR THE COMMODORE 64™ & DRIVE

By

Dr. Philip A. Slaymaker

© Copyright 1984 Slaymaker

All Rights Reserved

QUANTUM SOFTWARE
P.O. BOX 12716
LAKE PARK, FL 33403

Commodore 64 is a registered trademark of Commodore Electronics Ltd.
PEEK A BYTE is a trademark of Quantum Software

COPYRIGHT NOTICE

The Peek A Byte software and instruction manual are copyright by Quantum Software and Philip A. Slaymaker. The original and backup copies of this product are intended for your personal use in connection with a single computer. You may not copy and give away or sell any part of this product without express written permission of Quantum Software and Philip A. Slaymaker. For use on multiple computers, please contact Quantum Software to make such arrangements.

LIMITATION ON WARRANTIES AND LIABILITY

Quantum Software and the program author have endeavored to ensure that this program works as specified in this manual. Nevertheless, there may be errors in the program or its documentation. We would appreciate receiving notice of any errors you may find. If the original program diskette becomes unreadable, the distributor will replace it upon return of the defective diskette within 30 days of the date of delivery.

Quantum Software and the program author shall have no liability or responsibility to the purchaser or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by this software, including, but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of this software. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

TABLE OF CONTENTS

Peek A Byte 64 Manual
Copyright 1984 Slaymaker

PEEK A BYTE 64 INTRODUCTION

Peek A Byte 64 is the finest disk and memory utility available for the Commodore 64 computer. With this utility you can read or write data from any sector on a disk, edit the data, change a program name, disk ID or name, correct a program or directory, read the 64 memory, modify values in the BASIC memory, etc. The data can be displayed as HEX values, screen or PETASCII characters, or disassembled to machine code.

Peek A Byte also will do a fast compare of disk sectors, read the disk drive memory, convert between HEX and decimal, and recover or un-new Basic programs. A monitor program or user supplied machine language routine may be executed from Peek A Byte. This feature is unique and allows for additional versatility and future expansion capability. Compatablity with the DOS Wedge 5.1 is maintained, allowing it to be used after exiting Peek A Byte. Many Basic programs can also be run while Peek A Byte is resident in memory. Many uses for Peek A Byte will be examined throughout this manual.

Equipment Required

Peek A Byte is designed to work with the Commodore 64 computer, 1541 single drive floppy disk, and an optional printer. Other compatible serial bus drives should work with all drive commands except disk Verify. Peek A Byte will produce a character screen dump to a serial bus printer such as the VIC 1515 or VIC 1525 if connected. Other serial bus printers may work if designed to emulate the above or if they accept standard ASCII and the printer interface has a transparent mode.

Loading Peek A Byte

Peek A Byte is provided on disk only and can be loaded from a 1541 drive assigned a unit number of either 8 or 9, which may be software or hardware assigned. The program should be loaded from the power-up or reset state. To be sure, either turn the computer off and back on, or to reset, type:

```
          SYS 64738   <RETURN>
```

To load, type from BASIC:

```
          LOAD "PEEK*",8,1  <RETURN>
```

It is not necessary to type the full name. A unit of 9 should be used for a drive assigned unit 9.

Alternately, you may load the directory by typing:

```
LOAD "$",8   <RETURN>
```

LIST, then move the cursor home and press <RETURN>.

The copyright notice should appear, the drive will load the rest of the program, and then the HELP page will appear.

If a loader error occurs, a message will appear at the bottom of the screen:

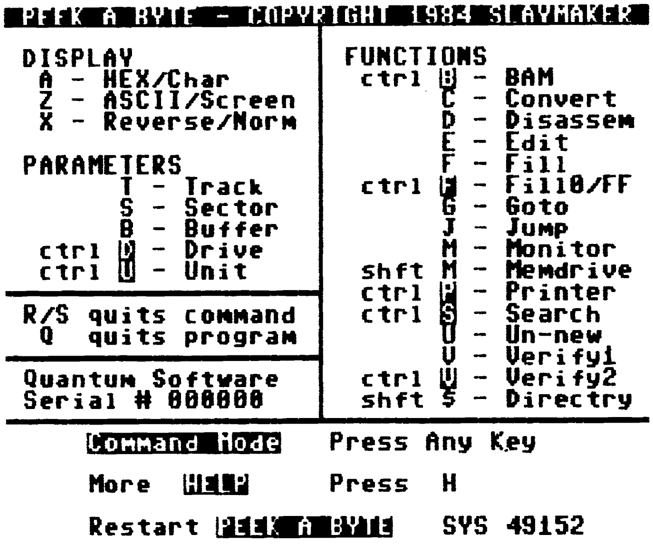
```
ERROR-REDO
```

If this occurs, remove the disk from the drive and reset the computer by turning it off and then back on. Then try reloading Peek A Byte. If it still won't load, check other disks known to be good in order to check the drive.

Remove your Peek A Byte disk from the drive. DO NOT put any disk in the drive without an opaque write protect tab until told to do so!

GETTING STARTED - HELP AND DISPLAY COMMANDS

If everything has gone correctly (it has, I trust), you should be looking at the first HELP page which summarizes display, parameter, and function commands. In the HELP mode, pressing "H" will show the other HELP page. It shows the Read/Write and Sector/Buffer Commands. Any other key returns you to the Command mode. The two HELP pages are shown below.



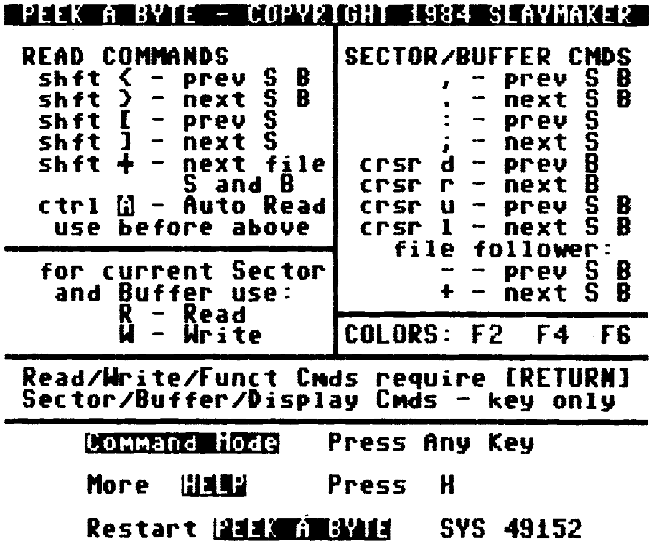
```
PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER
DISPLAY                   FUNCTIONS
 A - HEX/Char              ctrl B - BAM
 Z - ASCII/Screen               C - Convert
 X - Reverse/Norm               D - Disassem
                                E - Edit
PARAMETERS                      F - Fill
       T - Track           ctrl F - Fill0/FF
       S - Sector               G - Goto
       B - Buffer               J - Jump
 ctrl  D - Drive                M - Monitor
 ctrl  U - Unit            shft M - Memdrive
                           ctrl P - Printer
R/S quits command          ctrl S - Search
 Q  quits program               U - Un-new
                                V - Verify1
Quantum Software           ctrl W - Verify2
Serial # 000000            shft $ - Directry

      Command Mode    Press Any Key

      More  HELP      Press  H

      Restart PEEK A BYTE   SYS 49152
```

Figure 1. HELP Screen 1



```
PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER
READ COMMANDS            SECTOR/BUFFER CMDS
 shft < - prev S B              , - prev S B
 shft > - next S B              . - next S B
 shft [ - prev S                : - prev S
 shft ] - next S                ; - next S
 shft + - next file       crsr d - prev B
          S and B         crsr r - next B
 ctrl A - Auto Read       crsr u - prev S B
  use before above        crsr l - next S B
                            file follower:
 for current Sector             - - prev S B
  and Buffer use:               + - next S B
        R - Read
        W - Write         COLORS: F2  F4  F6

Read/Write/Funct Cmds require [RETURN]
Sector/Buffer/Display Cmds - key only

      Command Mode    Press Any Key

      More  HELP      Press  H

      Restart PEEK A BYTE   SYS 49152
```

Figure 2. HELP Screen 2

One very important key is the <STOP> key. It will ALWAYS abort, terminate, or quit any function or command. In the Command mode it does nothing (almost nothing - the jiffy clock changes). If <STOP> doesn't work (very rare, indeed), then <STOP> and <RESTORE> will do an NMI (nonmaskable interrupt) to BASIC. This causes the screen to reset to blue and resets most vectors such as the monitor break address. The DOS wedge is not restored either.

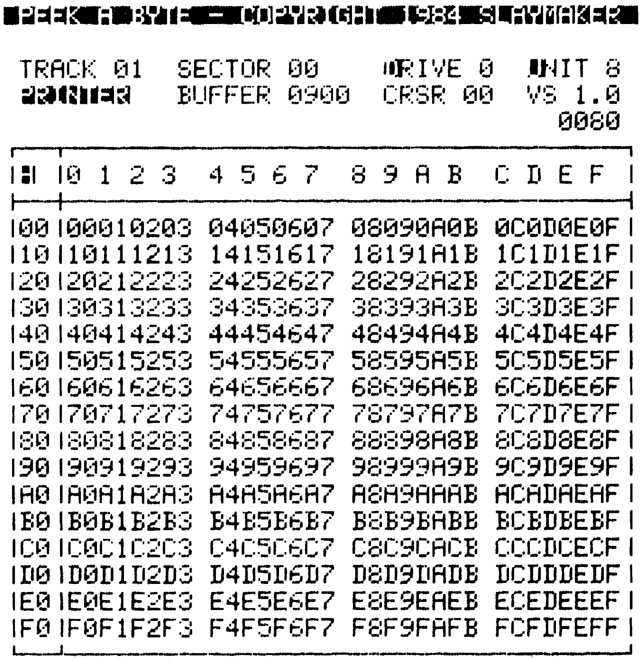
The other command is the "Q" key, for Quit. In the Command mode, pressing "Q" and <RETURN> keys will do a warm start to BASIC and maintain the screen colors, vector addresses, and, if present, the DOS wedge. To restart Peek A Byte, type:

```
SYS 49152   <RETURN>
```

In the Command mode, the display consists of a rectangular array of 256 values for each memory location in the memory page indicated by the Buffer. (Just a minute now, and I'll explain what that means). The initial display shows all zeros, 00. The Display commands only affect how the data are displayed. The "A" key toggles between HEX values and characters. "Z" toggles between PETASCII and screen code (see Appendix A and B in the Commodore 64 manual). "X" turns inverse or reverse characters on or off. The square in the upper left corner indicates the display mode and will contain an A for PETASCII, S for Screen code, and an H for HEX. The adjacent letter I indicates that inverse or reverse characters will be displayed, while N means only the normal character equivalent is shown.

The HEX values are shown in order from left to right, top to bottom. The vertical axis shows the high order nybbles and the horizontal axis the low order nybbles. All numerical values in Peek A Byte are shown in HEX, rather than decimal, because it is easier to describe the computer memory and organization of the I/O ports. HEX stands for hexadecimal - the number base is 16 instead of 10. Thus, a two digit number can range from 0 to 16*16-1 or 255. In hexadecimal only two digits are required for every value. A further discussion will be deferred to the section describing the CONVERT function.

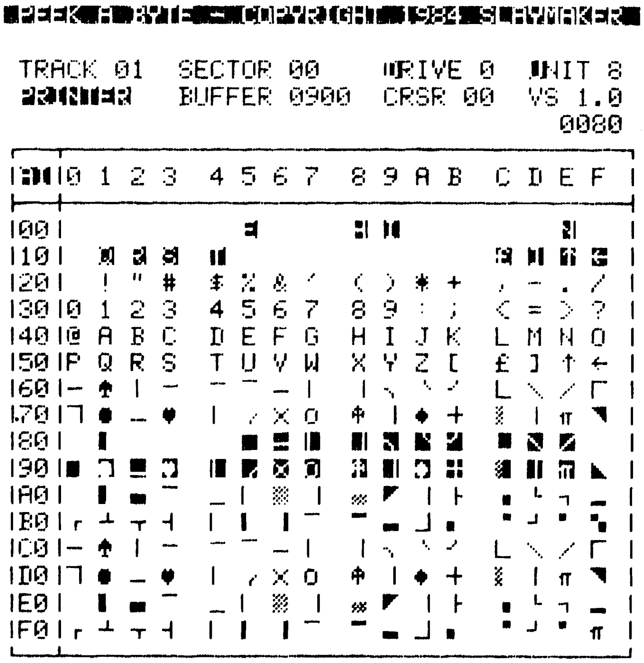
Let's try a new function and then you can play with the display keys. Press "ctrl-F" (hold both down, pressing ctrl first) and then <RETURN>. This will FILL the buffer with values from 00 to FF. Note: Often HEX values are designated by $, eg. $FF. In this manual $ will be omitted since (almost) all values are HEX. By toggling the display, the corresponding character codes can be seen. Note - some values do not have a displayable PETASCII character. See the figures below for two command screens for this example.



```
 PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

 TRACK 01   SECTOR 00    DRIVE 0   UNIT 8
 PRINTER    BUFFER 0900  CRSR 00   VS 1.0
                                     0080
|H |0 1 2 3  4 5 6 7  8 9 A B  C D E F |
|00|00010203 04050607 08090A0B 0C0D0E0F|
|10|10111213 14151617 18191A1B 1C1D1E1F|
|20|20212223 24252627 28292A2B 2C2D2E2F|
|30|30313233 34353637 38393A3B 3C3D3E3F|
|40|40414243 44454647 48494A4B 4C4D4E4F|
|50|50515253 54555657 58595A5B 5C5D5E5F|
|60|60616263 64656667 68696A6B 6C6D6E6F|
|70|70717273 74757677 78797A7B 7C7D7E7F|
|80|80818283 84858687 88898A8B 8C8D8E8F|
|90|90919293 94959697 98999A9B 9C9D9E9F|
|A0|A0A1A2A3 A4A5A6A7 A8A9AAAB ACADAEAF|
|B0|B0B1B2B3 B4B5B6B7 B8B9BABB BCBDBEBF|
|C0|C0C1C2C3 C4C5C6C7 C8C9CACB CCCDCECF|
|D0|D0D1D2D3 D4D5D6D7 D8D9DADB DCDDDEDF|
|E0|E0E1E2E3 E4E5E6E7 E8E9EAEB ECEDEEEF|
|F0|F0F1F2F3 F4F5F6F7 F8F9FAFB FCFDFEFF|
```

Figure 3. HEX Numbers 0 - FF



```
 PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

 TRACK 01   SECTOR 00    DRIVE 0   UNIT 8
 PRINTER    BUFFER 0900  CRSR 00   VS 1.0
                                     0080
|HA|0 1 2 3  4 5 6 7  8 9 A B  C D E F |
|20|  ! " #  $ % & '  ( ) * +  , - . / |
|30|0 1 2 3  4 5 6 7  8 9 : ;  < = > ? |
|40|@ A B C  D E F G  H I J K  L M N O |
|50|P Q R S  T U V W  X Y Z [  £ ] ↑ ← |
```

Figure 4. Corresponding PETASCII Characters

Loading the DOS Wedge

The memory from CC00 to CFFF is reserved for the DOS or C-64 Wedge program supplied on the VIC-1541 Test/Demo Disk, or any other user supplied ML program. After exiting Peek A Byte, load the DOS Wedge. After the copyright notice appears, the program must be moved to a storage location under the Basic Rom. Enter the following command from Basic:

```
SYS 49155   <RETURN>
```

Peek A Byte may then be restarted. Normally exiting the program, using the JMP command or the MONITOR command will restore the DOS wedge and its CHRGET routine.

PARAMETERS

The top few lines show the track, sector, drive, unit, buffer, edit cursor location, and the version number. Below the version (VS 1.0) is a four digit checksum for the buffer page shown. These will be discussed in the following paragraphs. Note: as a general rule, commands that only affect the display mode require only a single keystroke. If buffer memory values are affected or the disk drive is being accessed, then a <RETURN> is required. A command which requires a <RETURN> will be aborted if the key is pressed again immediately. <STOP> will terminate any command.

Track "T"

The current track value is shown in HEX, not decimal. Pressing "T" will allow entry of a new value which must be followed by a <RETURN> before it will be accepted. Invalid track values will not be accepted. See the section on entering values.

Sector "S"

Pressing "S" will allow entering a new sector value as done for the track. If a track value is entered which causes an illegal sector to be displayed, the cursor will shift to the sector value so that it may be corrected.

Drive "ctrl D"

The drive value is normally 0. A toggle option for drive 1 is included for possible use with dual drives. Using drive 1 with a 1541 drive results in a DRIVE NOT READY error. The command is "ctrl D" <RETURN>.

Unit "ctrl U"

The unit may be toggled between 8 and 9 by pressing "ctrl-U" and <RETURN>. Only typically used values are entered by this command. Other values require editing memory location 0343. A DEVICE NOT PRESENT error will result if the correct unit is not connected.

Buffer "B"

The buffer is the page in memory that is being examined. The Commodore 64 is arranged in 256 pages of 256 bytes (or FF+1 pages of FF+1 values). The pages 0000-0300 are reserved for the operating system, Peek A Byte, and BASIC system variables. Pages 0400-0700 are normally used for the screen. Pages 0800-9F00 are the normal space for BASIC programs. Buffers from 0800 to 9F00 may be edited. A sector may be read into any buffer in this range except 0800 since this could overwrite the beginning of BASIC. If a program such as a monitor lowers the top of Basic, these pages will be protected as well. Pages A000-FF00 contain the BASIC and Kernal ROMs, the I/O ports, and the Peek A Byte program. Editing non-BASIC memory is not allowed to prevent the user from "bombing" the program. (You wouldn't do that, would you?) An exception is that buffer 0200 may be edited since it contains the SEARCH string, to be described later. A second exception will be discussed in the Advanced Users section.

The buffer is changed by pressing "B" and entering the page number in HEX. Only the first two digits are required plus a <RETURN>. A number of Sector/Buffer commands are available which change the buffer. Two affect only the buffer and are useful for scanning the computer memory. Two cursor keys, "crsr d" (down) and "crsr r" (right) repeatedly decrease or increase the buffer by one page, respectively.

Try setting the buffer to A000, the display mode to screen characters, and to lower case (use shift and Commodore key). You are looking at the beginning of the BASIC ROM and the BASIC command words. Try scanning forward a few pages and you'll see the error messages. Now try pressing "D" for Disassemble and you'll see the ROM disassembled (more later).

ENTERING VALUES AFTER A COMMAND

When a command is initiated which requires values to be entered, the command name appears in reverse characters and the blinking cursor appears at the number to be entered. Two digits may be entered in sequence. The cursor will shift back to the beginning allowing the digits to be reentered if needed. The "crsr r" key will move the cursor without affecting the digit displayed. When the value is OK, press <RETURN> and it will be accepted. Pressing <STOP> will abort the input command.

Note: only digits from 0 to 9 and A to F are accepted. Other values produce a BEEP! (which you will soon learn to hate, but you can always turn the volume down).

EDIT AND GOTO "E", "G" and "ctrl G"

The Edit command is invoked by pressing "E" and <RETURN>. The cursor will appear after the CRSR. The memory location should be entered, eg. D7, and the cursor will move to location D7. If the display is in HEX mode, then values are input as discussed above. A <RETURN> must be pressed to accept the value and move the cursor to the next position. The "crsr" keys will move the cursor up, down, left, or right to any memory location within the displayed page without entering or affecting the displayed values. Alternately, "ctrl G" will move the cursor so that a new memory location may be entered.

Characters may be edited in the PETASCII mode simply by typing the character. The shift, ctrl, and Commodore keys all work, so normal graphic or control characters may be entered. Reverse screen code characters must be entered in HEX, in general. The graphics symbol for any of the crsr, color, function, or edit keys will appear if those keys are entered. The crsr keys normally control the cursor position and cannot be entered using the EDIT command. When a key is pressed, the cursor continues to blink over the new value. If the value is correct, press <RETURN> and the cursor will move to the next location. If "ctrl G" is pressed instead of <RETURN>, no value is entered and a new cursor position may be entered.

The GOTO command is identical to EDIT except that the "crsr" keys do not control the cursor position in ASCII mode, allowing "crsr" keys to be entered as values. "G" and <RETURN> start the GOTO edit mode.

The EDIT and GOTO commands can only be exited by pressing <STOP>. <STOP> and <RESTORE> quits the program and does a warm start to BASIC (unless you've changed the NMI vector at 0318).

READ/WRITE COMMANDS

The basic read command is "R" ⟨RETURN⟩. The current track and sector will be read into the displayed buffer. A BEEP will occur if the buffer is not valid for reading. If an error occurs, the message will be displayed below the command. Press any key to remove the message.

The only write command is "W" ⟨RETURN⟩. Any buffer can be written to any track and sector on the disk. Multiple writes are not implemented to avoid accidentally wiping out a disk. A WRITE PROTECT TAB should always be placed over the notch on the disk to prevent accidentally writing to the disk. Remove the tab only when specifically writing to the disk. Use an opaque or silvered tab. Masking tape DOES NOT work with the 1541 drive because the LED light can go thru it.

Now lets try the read command. Set the track to 12 and the sector to 00. Place a disk in the drive and press "R" ⟨RETURN⟩. Any valid buffer is OK. This is the sector with the BAM (block allocation map), disk name, and ID. You can read the disk name in ASCII mode. Now press "ctrl B" and a map of the BAM will be displayed. More later.

ERROR MESSAGES

By now you may have gotten error messages and BEEPs. There are several types of messages. If an invalid command key or input key is pressed, you will get a BEEP. If the device is not connected or is not working properly, you will get a DEVICE NOT PRESENT error. This applies to a disk drive or printer. Pressing any key clears this error message. Try setting the unit number to one not connected and press "R" ⟨RETURN⟩. Then change it back to the correct value.

During a read or write command a more serious error may occur. If you forgot the disk, you'll get a 21, READ ERROR and a lot of chattering from the drive. The data in the drive buffer is read even if there is an error. Usually it is from the last sector read. Do a write with a write protect tab on and 26, WRITE PROTECT ON appears. The track and sector listed in the message are in decimal, not HEX, because the disk drive generates them that way, and because the author is too lazy to convert them for you.

Other disk errors may occur and may indicate problems with the drive or disk. The errors and possible difficulties are listed in the 1541 drive manual in appendix B.

The errors may also have been intentionally written to the disk for copy protection. Peek A Byte may be used to scan a disk for read errors. The normal drive read routine will cause a head reset and chatter, however.

READ/SECTOR/BUFFER COMMANDS

A series of commands are available to change the sector and/or buffer before reading a sector. These are summarized in the Commands summmary and on the HELP screen. The sector/buffer commands are ",.:;+". The read equivalents perform the sector/buffer command, then read the disk. The read commands are all the shifted keys (hold shift down first) for the sector/buffer commands and are "<>[]+". The READ command will appear with the key symbol. A <RETURN> is required to execute the read command.

Sector and Buffer Changes

The "shft >" is normally used for scanning a disk by sectors and for storing the sector data in sequential buffers - the sector and buffer are both incremented. "." or "crsr l" will display the next buffer and increment the sector. A series of sectors can be written using both "." and "W" <RETURN>. The corresponding backwards scanning commands are "shft< ", ",", and "crsr u".

Sector Only Changes

The "shft [" or "shft]" only change the sector and are used to read sectors when the data is only to be examined and not saved in different buffers. The sector may be decremented or incremented by pressing ":" or ";" respectively.

File Follower "shft +"

The most useful read command is the "shft +" file follower command. This reads the next sector in a file using the file sector link. Byte 00 is the track and byte 01 is the sector. If byte 00 is 00, then byte 01 is the length of the file on that sector. Typically the value 4B appears in byte 00 of a newly formatted disk indictating that it is not part of a file. User files (such as on some protected disks) may not follow this convention. A BEEP will occur if the sector is the last sector in a file or is invalid.

File Scanning "+" and "-"

The "+" and "-" keys will scan thru a range of sectors previously read into the buffers using the "shft +" command. Note that file sectors are not numbered sequentially. This is done to improve SAVE and LOAD times. The first sector will be "remembered" for only one sequence or file.

Auto "ctrl A"

The file sectors can be read in automatically using the AUTO flag. The first sector should be read using "R". AUTO is activated by "ctrl A". Any shift read command will be performed sequentially until either the last file sector is read, the last valid buffer for reading is reached, or you press <STOP>. Any other command will be executed normally and will deactivate the AUTO flag. Once the sectors are in the buffer memory, they may be scanned using the sector/buffer commands, edited, and/or rewritten to the original disk or another disk.

FUNCTIONS

All Functions require a ⟨RETURN⟩. Pressing any other key after the reverse command name appears aborts the command. The only exceptions are BAM and DISASSEMBLE which do not require the ⟨RETURN⟩.

DIRECTORY "shft $"

The disk directory sectors will be loaded into the computer memory starting at the current buffer. The command is "shft $". Character mode is automatically selected and the display will be in PETASCII. The file loader is used, so there will be a BEEP when loading is finished. The "-" and "+" commands can be used to move from one buffer to the next while keeping the sectors correct. Observe that the directory sector number initially increases by 3, not 1. This spacing of sectors on the disk requires less time to load the directory.

BAM (Block Allocation Map) "ctrl B"

The BAM and directory header are located on sector 00 of track 12. This can be read directly or read using the "shft $" directory command. The BAM shows whether a given sector is used or not. The BAM function "ctrl B" will display the BAM as a map with the tracks running across the bottom of the screen and the sectors running vertically. Note that the values are in HEX (are you getting the hang of it now?). An asterisk (*) means the sector is used while a period (.) means it is free. The BAM values are taken from the current buffer, so if a non-BAM sector is displayed a row of question marks (?) will appear across the top of the display. The (?) means that the number of sectors free for a given track does not match the map for that track. If the BAM sector is displayed and one or more (?) appear, it means the BAM is "screwed-up".

The BAM starts with byte 04 and 4 bytes are used to map each track. The first byte for each track is the number of sectors free. The second thru fourth bytes map each sector starting with bit 0 for sector 0. The sector is free if the bit is 1 and used if the bit is zero. Peek A Byte computes the number of sectors free for each track and compares it to the number recorded on the BAM. If the values do not match, all the sectors are assumed to be used and are not included in the SECTORS FREE in the upper right corner of the display. All of track 12 sectors (the directory sectors) are assumed to be used.

The BAM can be modified and rewritten to the disk. This should only be attempted on a BACKUP disk after you understand what the change will do. It is very easy to "bomb" the BAM of a disk which can cause data to be lost.

If a printer is connected, "ctrl P" will dump the BAM display to the printer.

Disk Name

The disk name is located on the BAM sector starting with byte 90. The name is sixteen (10 HEX) characters long. The name may be changed using the edit command. This change is only made to the buffer until the buffer is rewritten to the disk. The safest way is to read track 12, sector 0 into two buffers and change the name in only one buffer. Be sure that nothing else is changed, then write that buffer back to the disk.

Disk ID

The disk ID is written to the BAM and every sector on the disk when it is formatted using the 1541 "NEW" command. The ID is located at A2 and A3 of the BAM sector and may be changed using the same procedure as for the disk name. This only changes the ID displayed. To change the ID written to each sector, the disk must be reformatted. However, since this erases all the data, a backup disk should be formatted and the data copied to it.

Directory Names

The directory sectors normally start at track 12, sector 1. The file type is located at byte 02 of each file, with two rows for each file. This value is 00 if the file is deleted, 81 for sequential, 82 for program, 83 for user, 84 for relative; other values are not currently defined. If the file was improperly closed, the 8 will be a zero. A file which has been just deleted may be undeleted by changing this value from 00 to the correct value, usually 82 for program. The file should be reloaded and saved back to the disk under a different name to be sure the BAM is updated correctly. Then delete the old file.

If other files have been written to the disk after deleting the file of interest, the file may have been overwritten. The file sectors can be traced using Peek A Byte, as discussed below, to be sure the sectors are intact. If some have been overwritten, the file is essentially unrecoverable.

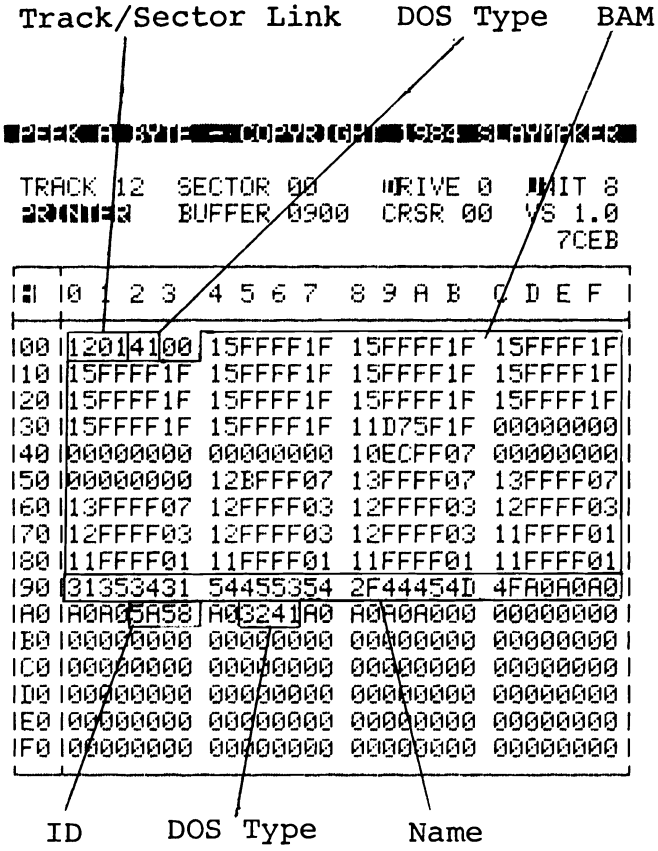
A file may be scratch protected or locked by changing the 8 to a C, thus setting bit 6 as a lock flag. For a program file byte 02 would read C2 instead of 82. When the directory is listed, locked files appear with a < symbol after the file type. To unlock the file just change the byte back to 82.

Bytes 03 and 04 are the beginning track and sector of the file. These values can be used with the file follower "shft +" to load file sectors into sequential buffers. These values should not be changed unless the sector data has been moved to a different sector using Peek A Byte. Use the "+" and "-" commands to scan thru the file and be sure it is correct.

The file name starts at byte 05 and continues thru byte 14, or until the first reverse space, A0, is encountered. Add 20 for each additional file - this is HEX, remember. Look at the vertical axis beside the file name. Peek A Byte can read all characters of a name including normally illegal control characters. A favorite character is 14 or delete, commonly used for copy protection. The shifted space or A0 signifies the end of the name. At 1E and 1F the file length is given as low byte, high byte (in HEX, of course). Additional parameters appear for relative files.

The name may be changed by reading the directory sector, using the edit command to change the name in the buffer, and writing the name back to the disk. Any PETASCII character can be used in ASCII mode, or HEX mode can be used. Be very careful when writing to the disk since the directory can be "bombed" by writing the wrong data back to the disk.

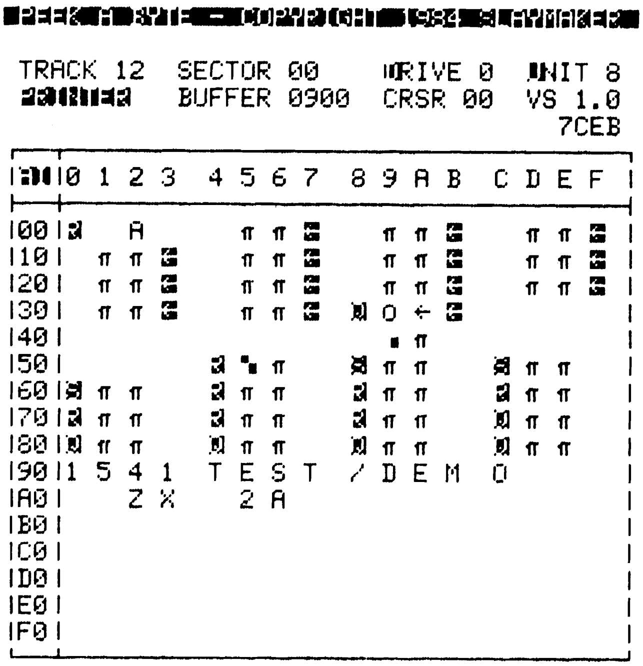
Examples of the BAM sector, BAM (block or sector allocation map), and directory sectors are shown using the VIC 1541 Test/Demo Disk (which you should have).



```
 PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

 TRACK 12   SECTOR 00     DRIVE 0  UNIT 8
 PRINTER    BUFFER 0900   CRSR 00  VS 1.0
                                     7CEB
|  |0 1 2 3  4 5 6 7  8 9 A B  C D E F |
|00|12014100 15FFFF1F 15FFFF1F 15FFFF1F|
|10|15FFFF1F 15FFFF1F 15FFFF1F 15FFFF1F|
|20|15FFFF1F 15FFFF1F 15FFFF1F 15FFFF1F|
|30|15FFFF1F 15FFFF1F 11D75F1F 00000000|
|40|00000000 00000000 10ECFF07 00000000|
|50|00000000 12BFFF07 13FFFF07 13FFFF07|
|60|13FFFF07 12FFFF03 12FFFF03 12FFFF03|
|70|12FFFF03 12FFFF03 12FFFF03 11FFFF01|
|80|11FFFF01 11FFFF01 11FFFF01 11FFFF01|
|90|31353431 54455354 2F44454D 4FA0A0A0|
|A0|A0A05A58 A03241A0 A0A0A000 00000000|
|B0|00000000 00000000 00000000 00000000|
|C0|00000000 00000000 00000000 00000000|
|D0|00000000 00000000 00000000 00000000|
|E0|00000000 00000000 00000000 00000000|
|F0|00000000 00000000 00000000 00000000|
```

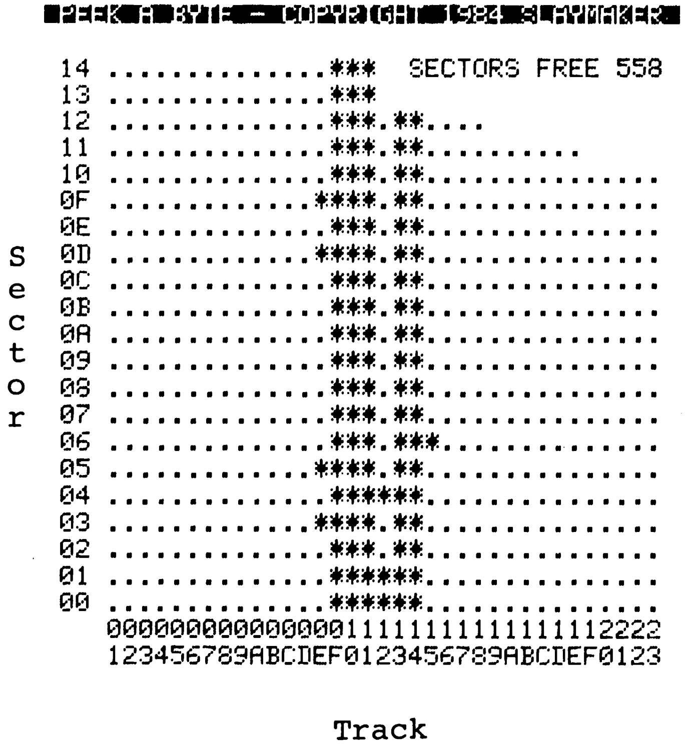
Figure 5. BAM Sector - HEX



```
 PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

 TRACK 12   SECTOR 00     DRIVE 0  UNIT 8
 PRINTER    BUFFER 0900   CRSR 00  VS 1.0
                                     7CEB
|  |0 1 2 3  4 5 6 7  8 9 A B  C D E F |
|00|▒   A      π π ▒    π π ▒    π π ▒ |
|10|  π π ▒    π π ▒    π π ▒    π π ▒ |
|20|  π π ▒    π π ▒    π π ▒    π π ▒ |
|30|  π π ▒    π π ▒  ▒ O ← ▒          |
|40|                    ▪ π            |
|50|         ▒ ▘ π    ▒ π π    ▒ π π   |
|60|▒ π π    ▒ π π    ▒ π π    ▒ π π   |
|70|▒ π π    ▒ π π    ▒ π π    ▒ π π   |
|80|▒ π π    ▒ π π    ▒ π π    ▒ π π   |
|90|1 5 4 1  T E S T  / D E M  O       |
|A0|    Z X    2 A                     |
|B0|                                   |
|C0|                                   |
|D0|                                   |
|E0|                                   |
|F0|                                   |
```

Figure 6. BAM Sector - PETASCII

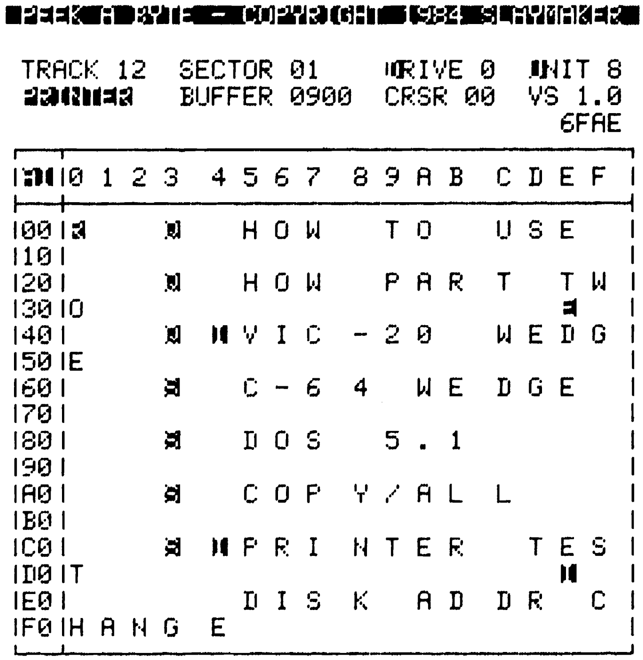


```
PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

14 ..............***  SECTORS FREE 558
13 ..............***
12 ..............***.**....
11 ..............***.**..........
10 ..............***.**...............
0F .............****.**...............
0E ..............***.**...............
0D .............****.**...............
0C ..............***.**...............
0B ..............***.**...............
0A ..............***.**...............
09 ..............***.**...............
08 ..............***.**...............
07 ..............***.**...............
06 ..............***.***..............
05 .............****.**...............
04 ..............******...............
03 .............****.**...............
02 ..............***.**...............
01 ..............******...............
00 ..............******...............
   00000000000000011111111111111112222
   123456789ABCDEF0123456789ABCDEF0123
```

Figure 7. Block Allocation Map

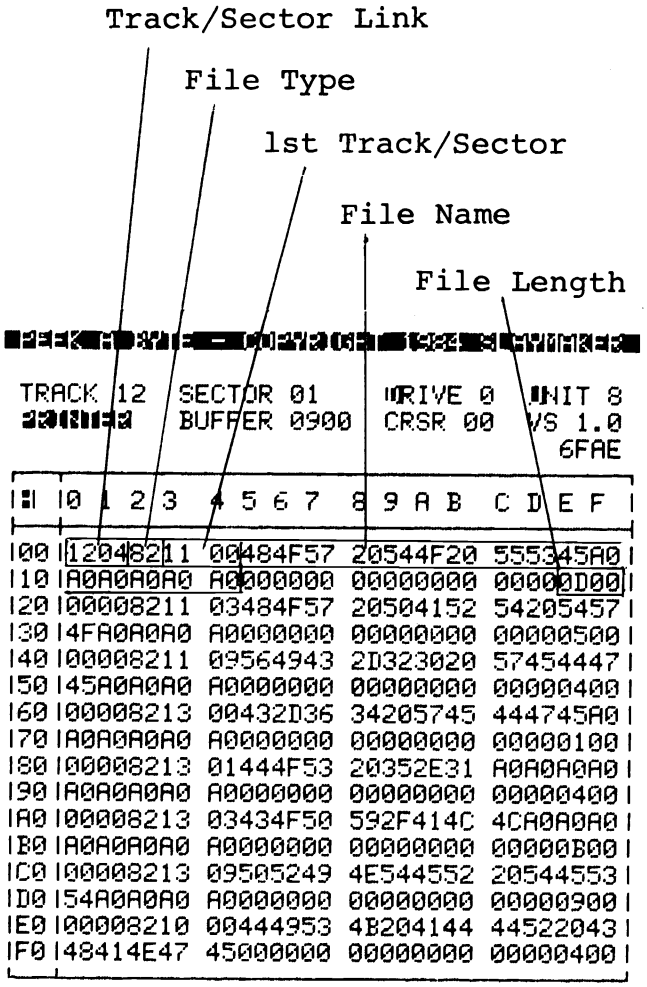
* Sectors Used
. Sectors Free



```
PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

 TRACK 12   SECTOR 01    DRIVE 0   UNIT 8
 PRINTER    BUFFER 0900  CRSR 00   VS 1.0
                                     6FAE

|■ |0 1 2 3  4 5 6 7  8 9 A B  C D E F |
|00|■     ■    H O W    T O    U S E   |
|10|                                   |
|20|      ■    H O W    P A R  T   T W |
|30|O                              ■   |
|40|      ■  ■ V I C  - 2 0    W E D G |
|50|E                                  |
|60|      ■    C - 6  4   W E  D G E   |
|70|                                   |
|80|      ■    D O S    5 . 1          |
|90|                                   |
|A0|      ■    C O P  Y / A L  L       |
|B0|                                   |
|C0|      ■  ■ P R I  N T E R    T E S |
|D0|T                              ■   |
|E0|           D I S  K   A D  D R   C |
|F0|H A N G  E                         |
```



```
PEEK A BYTE - COPYRIGHT 1984 SLAYMAKER

 TRACK 12   SECTOR 01    DRIVE 0   UNIT 8
 PRINTER    BUFFER 0900  CRSR 00   VS 1.0
                                     6FAE

|■ |0 1 2 3  4 5 6 7  8 9 A B  C D E F |
|00|12048211 00484F57 20544F20 555345A0|
|10|A0A0A0A0 A0000000 00000000 00000D00|
|20|00008211 03484F57 20504152 54205457|
|30|4FA0A0A0 A0000000 00000000 00000500|
|40|00008211 09564943 2D323020 57454447|
|50|45A0A0A0 A0000000 00000000 00000400|
|60|00008213 00432D36 34205745 444745A0|
|70|A0A0A0A0 A0000000 00000000 00000100|
|80|00008213 01444F53 20352E31 A0A0A0A0|
|90|A0A0A0A0 A0000000 00000000 00000400|
|A0|00008213 03434F50 592F414C 4CA0A0A0|
|B0|A0A0A0A0 A0000000 00000000 00000B00|
|C0|00008213 09505249 4E544552 20544553|
|D0|54A0A0A0 A0000000 00000000 00000900|
|E0|00008210 00444953 4B204144 44522043|
|F0|48414E47 45000000 00000000 00000400|
```

Figure 8. Directory Sector - PETASCII Figure 9. Directory Sector - HEX

CONVERT "C"

This mode allows you to convert between hex and decimal numbers. After the screen clears, a blinking cursor and five zeros will appear. The normal input routine is used to accept five values instead of two. The "crsr r" key will skip over any digits. A <RETURN> must be entered to accept the number and convert it. For this function only, a "$" must be entered first if the number is in hex. A "$" in any other position will be treated as a "4". The hex answer is listed first. A decimal number shorter than five digits may be right adjusted or terminated with a hex digit "A - F". Numbers larger than 65535 will be converted after subtracting 65536.

Use the <STOP> key to return to the command mode. The CONVERT screen cannot be printed.

DISASSEMBLE "D"

This command was introduced previously. "D" will disassemble the data in any buffer or page. Repeated use of "D" will continue to disassemble code past the end of the current buffer. A <RETURN> will turn on the cursor at the BUFFER address allowing a new address to be input in groups of two hex digits. Repeated use of <RETURN> continues disassembly at the address shown. <STOP> terminates disassembly, as will any other key if the input cursor is off.

If character mode is in effect, the character for each memory value will be shown on the right side of the disassembly. This is very useful for determining what is op code versus character data. If a printer is connected, "ctrl P" will cause the screen to be dumped to the printer and the next lines of code disassembled.

FILL COMMANDS "ctrl F" and "F"

The current buffer can be filled with a test pattern of values. The values 00 to FF are written if "ctrl F" <RETURN> is used. This is useful for identifying screen or PETASCII characters.

A specific value can be written to the buffer using the "F" <RETURN> command. The cursor will turn on allowing a value other than 00 to be entered. This command is useful for clearing memory or for writing a single value to a particular sector, for example, to erase data from a deleted file.

The Fill commands only work for buffers from 0900 to 9F00 to prevent overwriting the operating system or Peek A Byte. Other buffers will produce a BEEP.

HELP "H"

You've already met the HELP command. Just press "H" when in the command mode and help is on the way - no waiting!

JMP "J"

JMP FE66 is the initial response to pressing "J". Any memory address may be entered (which should be a valid ML subroutine). If <RETURN> is pressed Peek A Byte will clear the screen and jump to whatever machine language routine is at the displayed address. FE66 is the BASIC warm restart vector and behaves the same as Quit: "Q" <RETURN>, except that screen and vector parameters are reset. This warm restart to BASIC places the values 66 FE at memory locations 32E and 32F, which are a user-defined vector.

JMP with User Routines

An example of a user routine is to change the disk unit number toggle from 8 or 9 to A or B (decimal 10 or 11). This is necessary if the 1541 drives are hardwired for these unit numbers. Let's assemble a short routine at $6000 using a monitor:

```
A 6000 LDA #$0A  ; Unit 10 decimal
A 6002 STA $0343 ; Unit location
A 6005 RTS
```

Invoke the JMP command and enter 60 <RETURN> 00 <RETURN> and it will jump to your routine and set the unit number to toggle between A and B. It will be reset to 8 if the Peek A Byte is exited and reentered, however.

User routines should either finish with an RTS or JMP $C000. An RTS will not reinitialize variables. The JMP $C000 should be used if the routine alters memory values listed in the Memory Map section. The routine should not alter the current stack values.

MONITOR "M"

The MONITOR command is designed for use with a machine language monitor such as Supermon by J. Butterfield or Micromon by B. Yee for COMPUTE! The monitor used must not be located above A000 or use memory locations there (I/O is OK). Similarly, page 300 should not be used since Peek A Byte stores its variables there (and wipes out any sprites stored there). Some monitors, such as the one provided with the Commodore 64 Macro Assembler Development System alter BASIC variables and will not work with Peek A Byte.

The monitor should be loaded after loading Peek A Byte. The monitor should place the monitor entry address at 0316 and 0317, the BRK vector. To reenter Peek A Byte from the monitor use

```
          G C000  <RETURN>   (same as SYS 49152)
```

With Micromon, do not exit the monitor using the "X" command if the monitor was entered from Peek A Byte. The monitor may try to return to Peek A Byte, but will not reinitialize the variables used by Peek A Byte and the computer may hang.

The MONITOR command SHOULD be used with the HESMON cartridge, however you may wish to change the background color by alterring memory location D021. DO NOT enter HESMON by hitting the <RESTORE> key from Peek A Byte since this can hang the computer. The XC and X commands are used to exit from HESMON to BASIC. The DOS wedge will only be restored if the MONITOR and X commands are used, however.

MEMDRIVE "shft M"

This command will read the disk drive memory in full pages into the computer memory starting at the current buffer, if valid for reading. The default drive pages are 00 thru 07, the only RAM in the 1541 drive. The page numbers (high byte of address) may be entered at the blinking cursor and accepted by hitting <RETURN>. <STOP> will abort the command.

The 6522 VIA registers are at 1800 to 180F and 1C00 to 1C0F, with images appearing from 1800 to 1FFF. The ROM appears from C000 to FFFF. Other disk drives may have different addresses.

PRINTER "ctrl P"

If a VIC 1515 or 1525 compatible printer is connected, a screen dump may be obtained of the command screen by using "ctrl P" <RETURN>. The BAM, COMMAND, DISASSEM, SEARCH, and VERIFY screens can be printed using just "ctrl P". The HELP and CONVERT screens can't be printed. If the printer is not connected, a DEVICE NOT PRESENT error will occur - press any key to remove the message. If the printer is connected, but not properly initialized, Peek A Byte may "hang". If this happens, press <STOP> and <RESTORE> and reset the printer. This problem usually only occurs when using an emulator interface and the printer has been turned on after the computer (check the procedure in your manual).

Peek A Byte sends PETASCII characters to the printer and uses graphics mode for the quote character. Emulators may not correctly print the graphics characters or character spacing, thereby producing a funny output. Also, if the printer is a VIC, a few graphics characters are not identical to those on the C-64 screen. The default printer parameters which are user setable from Peek A Byte are listed below with the memory location.

Unit	02E0	04
Secondary Address	02E1	07
Margin spaces	02E2	0A
CBM Printer Flag	02E3	00

The secondary address is normally used for cursor down mode. If transparent mode to a non-CBM printer is desired, the proper secondary address should be put in 02E1, the flag at 02E3 set to 80, and the computer put into cursor down mode. Only standard ASCII characters will be sent to the printer, no graphics characters or commands. The printout will have upper and lowercase letters reversed and some other characters may be different from the screen. These parameters must be reentered if Peek A Byte is restarted.

⟨STOP⟩ will terminate printing, however, the printer buffer may still contain characters.

The PRINTER command may be executed from Basic by using:

```
SYS 49158
```

The parameters must be set as in Peek A Byte and the printer MUST be connected, otherwise the program will hang and the computer must be reset.

SEARCH "ctrl S" or ⟨HOME⟩

Peek A Byte will search the memory from 0000 thru FFFF with the BASIC and Kernal ROMs and I/O selected. The number of characters to be searched for is entered at 2F0, byte F0 of buffer 0200. The search string starts at byte F1 of buffer 0200. The number should be entered in HEX while the string may be entered either in HEX or ASCII mode. Buffer 0200 contains the normal BASIC input buffer. If you've just entered Peek A Byte from BASIC you'll see the tokenized command at 200 for SYS 49152.

The search command is "ctrl S" ⟨RETURN⟩ or ⟨HOME⟩ ⟨RETURN⟩. All locations will be printed to the screen. The listing will pause at the bottom to allow either examination or printing of the memory addresses. To print the screen, type "ctrl P". To abort the search command before finishing the search, use the ⟨STOP⟩ key.

As an example, enter at 2F0 the value 05 for the number of characters and in ASCII, the word BASIC starting at 2F1. Press <HOME> <RETURN>. The addresses that should appear are

```
          02F1 A007 E461 E48B
```

Examine these locations using the commands that you've learned.

A second example is to search for the address of CHROUT, the Kernal print routine, at FFD2. At 2F0 enter 02 D2 FF and use the search command (addresses are almost always low byte, high byte).

UN-NEW "U"

A Basic program which has been lost by using the Basic NEW command can be recovered using "U" <RETURN>. This resets the line link at 0801 (or different start of Basic if changed) and the start of BASIC variables pointer at 002D. The pointers will be reset properly only if the NEW command was used and the program was not overwritten. If the computer was reset, the start of the Basic program must be 0801 as determined by the pointer at 002B.

VERIFY1 "V"

This command reads each sector on a disk starting with the displayed track. A checksum is computed and stored by the computer for each sector for use in comparisons between disks. The sectors are compared to the current buffer displayed. The VERIFY display format is the same as for the BAM except that a period (.) means that the sector checksums are the same and asterisk (*) means they are different.

The VERIFY command may be aborted by pressing <STOP>. Any other key will temporarily halt the program and restart it if pressed again. A printer dump can be obtained by pressing "ctrl P" assuming the printer is connected.

The VERIFY command is very fast because the sectors are read in a staggered order and because the checksum is computed in the disk drive. For this reason it will not work with a non-1541 drive. A "*" will always be reported for that comparison. If a read error occurs, the last digit of the error code will be displayed on the screen instead for any drive. A read error will cause the drive head to reset, therefore disks with extensive errors should not be checked using this command.

VERIFY2 "ctrl V"

This command is used to check a second disk against the checksums stored for the first disk. The second disk's checksums are not stored, allowing many disks to be checked against the original. See the section on checksums for a discussion of their limitations.

ADVANCED USER'S GUIDE

The author originally wrote Peek A Byte for the purpose of reconstructing a "blown" directory on the only disk owned at that time. The culprit was the 1541 save with replace command, which didn't always work properly. A blown directory can also result if two disks are formatted with the same ID and a file is saved to one disk after the other has been read by the disk drive. A full backup with a different ID, maybe two backups, is now made of every disk the author creates - this is just good practice. Also, a write protect tab is good practice, except when specifically writing to the disk. Hopefully, if you follow this advice, you won't HAVE to use Peek A Byte. You can spend your time learning about the computer and disk drive.

The best advanced use of Peek A Byte is to fix disks that have had a few bytes of data scrambled on them. Two sectors may be compared from different disks by direct examination after using the VERIFY functions or checksum in the upper right corner. Alternately, the data on a sector may be disassembled or displayed to determine if it makes sense and corrected if it doesn't. Corrections may be rewritten to the disk.

The Checksum

The checksum is designed to check for differences of one or two bytes or for two characters which have been transposed. That two checksums are the same does not guarantee that the buffers examined are the same. Changes of several bytes can result in an identical checksum. Try filling a buffer with all 01's or 03's and the checksum will still be 0000. Use the checksum as a tool, not as the only test.

Blown disks

A disk with a blown directory may be reconstructed by scanning the disk and using the file follower command "shft +" to load in what appear to be files. When the start track and sector and length have been determined along with the file type, a directory entry can be edited into a buffer and written to the disk when complete. A direct backup disk should be used to avoid further loss of data. The BAM can then be recreated using the 1541 Validate command (except random files).

If a sector sometimes gives an error when read, it is usually possible to read the sector until read correctly and to then write it back to the disk. This works if the disk formatting is OK but the data was written with a drive which was misaligned. If a large number of sectors have occasional read errors, then a full disk backup should be made with a disk copy program such as J. Butterfield's Copyall or 1541 Backup or one of the many commercially available programs.

Copy Protection

The author of this program does not believe that utilities such as Peek A Byte should be copy protected. Quantum Software has kept the purchase price of Peek A Byte low to encourage purchasing. Also, the Peek A Byte program and its documentation are copyrighted and copying is illegal for any reason other than a backup copy for the use of the purchaser.

With that said, it is obvious that Peek A Byte can be used to either protect or remove protection from some programs. Illegal BAM or directory entries can effect some protection and can be both read and written using Peek A Byte. The disk errors written to a protected disk can be determined using Peek A Byte by scanning the disk using one of the read commands. However, Peek A Byte has no provision for writing errors to the disk. (In fact, some of those "error" sectors actually contain data that can't normally be read by the 1541 drive.)

Peek A Byte can be used to read normally formatted data from any sector on a disk, disassemble it, and modify it to remove copy protection routines. Care should be exercised, however, since it is certainly possible to overwrite data on a disk sector and make the program unusable. It is beyond the scope of this manual to discuss how to trace such copy protection routines.

A few aspects of disk copy protection can be determined or changed using Peek A Byte and the MEMDRIVE function. The actual disk ID of a track is stored at 1541 drive locations 0016 and 0017, whereas the BAM ID is at 0012 and 0013. If these do not match, a disk ID error occurs. A short Basic program using the M-W command can be used to change the ID at 0012 and 0013 to match that read off the track. Peek A Byte may then be able to read that sector's data.

Software write protection of a disk is done by changing byte 02 of the BAM sector on track 12 from a 41 to something else, usually a 01 which is the code for CBM drive 2040 or 3040 format. This may also prevent a copy of the disk from being made. A short BASIC command can be used to change the code in the disk drive after the protected BAM sector is read. Exit Peek A Byte and enter in direct mode:

```
OPEN15,8,15:PRINT#15,"M-W"CHR$(1)CHR$(1)CHR$(1)CHR$(65):CLOSE15
```

Reenter Peek A Byte and change byte 02 of the BAM sector to 41 and rewrite the sector. DO THIS ON A BACKUP COPY FIRST!

Additional tips and programs concerning copy protection will be made available in the future to registered purchasers at nominal charge if interest and demand warrant.

SUMMARY OF PEEK A BYTE COMMANDS

DISPLAY MODES:

```
A - Toggle between HEX (H) and Character mode
Z - Toggle between PETASCII (A) and Screen (S) Characters
X - Toggle between Inverse (I) and Normal (N) Characters
```

PARAMETERS:

```
     T - Track:  input track value
     S - Sector: input sector value
     B - Buffer: input memory page value
ctrl D - Drive:  toggle between 0 and 1 (only dual drives)
ctrl U - Unit:   toggle between 8 and 9 drive unit
```

Note: All parameters require a <CR> to complete command

READ AND WRITE COMMANDS:

```
     R - Read current sector into current buffer
shft   - Read previous sector into previous buffer
shft   - Read next sector into next buffer
shft [ - Read previous sector into current buffer
shft ] - Read next sector into current buffer
shft + - Read next sector in file into next buffer
```

Note: All read and write commands require <CR> or <RETURN>

ctrl A - Auto: Continuous reading - use prior to shft Read commands

```
     W - Write current buffer to current sector
```

SECTOR AND BUFFER COMMANDS:

```
     , - Decrement sector and buffer
     . - Increment sector and buffer
     : - Decrement sector
     ; - Increment sector
     + - Next file sector and increment buffer
     - - Previous file sector and decrement buffer
crsr u - Decrement sector and buffer - repeats
crsr l - Increment sector and buffer - repeats
crsr d - Decrement buffer            - repeats
crsr r - Increment buffer            - repeats
```

FUNCTIONS:

ctrl B - BAM display of current buffer
C - Convert between hex and decimal
D - Disassemble starting at current buffer
E - Edit Mode
F - Fill buffer with specified HEX value
ctrl F - Fill buffer with 00 to FF
G - Goto - Edit command with crsr keys locked out
ctrl G - Goto cursor edit location - only effective in Edit or Goto mode
H - HELP, I need somebody, HELP ...
J - Jump to address specified by user - vector at 32E
M - Monitor - jumps to address specified by the BRK vector
shft M - Memdrive - reads specified disk drive memory pages
ctrl P - Print screen to printer
Q - Quit to BASIC
<STOP> - Abort Command
ctrl S - Search or hunt for string located at $2F1. Number of characters in HEX at $2F0
U - Un-new recovers Basic programs lost due to NEW
V - Verify1 - stores disk checksums and compares to buffer
ctrl V - Verify2 - compare 2nd disk to 1st
shft $ - Directory sectors loaded into buffers

Note: All commands require <CR> except BAM and Disassemble.

MEMORY MAP

A limited memory map is provided to assist the advanced user in maintaining compatiblity with Basic, monitor, or ML programs. Basic system memory locations, if used, must be restored by the user so that Basic will operate correctly.

0 Page Locations

02	Character mode
39	Temp
3A	"
3F	"
40	"
45	"
46	"
FB	"
FC	"
FD	"
FE	"

most Kernal variables

Page 200 Locations

2E0	Printer Unit #
2E1	Printer Secondary address
2E2	Printer Margin spaces
2E3	Printer Flag - 00 for CBM, 80 for ASCII only
2E4	Edit Flag - FF allows extended edit

2F0	Search string length
2F1-2FF	Search string beginning - upto fifteen values

Page 300 Locations

All of page 3 is reserved for Peek A Byte variables. Sprites normally stored here will be overwritten.

Screen 400 - 7FF

Sprite pointers normally stored from 7E8 - 7FF are erased by Peek A Byte.

Locations 800 - 9FFF

This space may be used for user supplied Basic or ML programs. If data is read into the computer from either disk or the drive, this is where it is stored. The user is responsible for ensuring any such programs are not overwritten when using Peek A Byte.

Locations A000 - FFFF

This space is reserved for the Peek A Byte program, much of which is stored in RAM under the ROM. NO Pokes or other changes should be made in this memory range. Exceptions are as follows:

Locations CC00 - CFFF

User supplied routines, such as the DOS or C-64 wedge, which run independently of Peek A Byte may be used here. Load the routine after Peek A Byte and use SYS 49155 to store it. Peek A Byte uses the same memory locations and will copy itself here and move the user routine back when exited.

Locations D000 - DFFF

The user may use the normal I/O registers in this range as long as it does not alter the Basic and Kernal operating system. Peek A Byte will reset other values, such as color, when restarted.

				UP		DOWN		
HEX	DEC	HIGH DEC	BINARY	ASCII	SCN	ASCII	SCN	BASIC TOKEN
$00	0	0	00000000		@		@	
$01	1	256	00000001		A		a	
$02	2	512	00000010		B		b	
$03	3	768	00000011		C		c	
$04	4	1024	00000100		D		d	
$05	5	1280	00000101	▮	E	▮	e	
$06	6	1536	00000110		F		f	
$07	7	1792	00000111		G		g	
$08	8	2048	00001000	▮	H	▮	h	
$09	9	2304	00001001	▮	I	▮	i	
$0A	10	2560	00001010		J		j	
$0B	11	2816	00001011		K		k	
$0C	12	3072	00001100		L		l	
$0D	13	3328	00001101		M		m	
$0E	14	3584	00001110	▮	N	▮	n	
$0F	15	3840	00001111		O		o	
$10	16	4096	00010000		P		p	
$11	17	4352	00010001	▮	Q	▮	q	
$12	18	4608	00010010	▮	R	▮	r	
$13	19	4864	00010011	▮	S	▮	s	
$14	20	5120	00010100	▮	T	▮	t	
$15	21	5376	00010101		U		u	
$16	22	5632	00010110		V		v	
$17	23	5888	00010111		W		w	
$18	24	6144	00011000		X		x	
$19	25	6400	00011001		Y		y	
$1A	26	6656	00011010		Z		z	
$1B	27	6912	00011011		[		[	
$1C	28	7168	00011100	▮	£	▮	£	
$1D	29	7424	00011101	▮	]	▮	]	
$1E	30	7680	00011110	▮	↑	▮	↑	
$1F	31	7936	00011111	▮	←	▮	←	
$20	32	8192	00100000					
$21	33	8448	00100001	!	!	!	!	
$22	34	8704	00100010	"	"	"	"	
$23	35	8960	00100011	#	#	#	#	
$24	36	9216	00100100	$	$	$	$	
$25	37	9472	00100101	%	%	%	%	
$26	38	9728	00100110	&	&	&	&	
$27	39	9984	00100111	'	'	'	'	
$28	40	10240	00101000	(	(	(	(	
$29	41	10496	00101001	)	)	)	)	
$2A	42	10752	00101010	*	*	*	*	
$2B	43	11008	00101011	+	+	+	+	
$2C	44	11264	00101100	,	,	,	,	
$2D	45	11520	00101101	-	-	-	-	
$2E	46	11776	00101110	.	.	.	.	
$2F	47	12032	00101111	/	/	/	/	
$30	48	12288	00110000	0	0	0	0	
$31	49	12544	00110001	1	1	1	1	
$32	50	12800	00110010	2	2	2	2	
$33	51	13056	00110011	3	3	3	3	
$34	52	13312	00110100	4	4	4	4	
$35	53	13568	00110101	5	5	5	5	
$36	54	13824	00110110	6	6	6	6	
$37	55	14080	00110111	7	7	7	7	
$38	56	14336	00111000	8	8	8	8	
$39	57	14592	00111001	9	9	9	9	
$3A	58	14848	00111010	:	:	:	:	
$3B	59	15104	00111011	;	;	;	;	
$3C	60	15360	00111100	<	<	<	<	
$3D	61	15616	00111101	=	=	=	=	
$3E	62	15872	00111110	>	>	>	>	
$3F	63	16128	00111111	?	?	?	?	

				UP		DOWN		
HEX	DEC	HIGH DEC	BINARY	ASCII	SCN	ASCII	SCN	BASIC TOKEN
$40	64	16384	01000000	@	─	@	─	
$41	65	16640	01000001	A	♠	a	A	
$42	66	16896	01000010	B	│	b	B	
$43	67	17152	01000011	C	─	c	C	
$44	68	17408	01000100	D	─	d	D	
$45	69	17664	01000101	E	▔	e	E	
$46	70	17920	01000110	F	─	f	F	
$47	71	18176	01000111	G	│	g	G	
$48	72	18432	01001000	H	│	h	H	
$49	73	18688	01001001	I	╮	i	I	
$4A	74	18944	01001010	J	╰	j	J	
$4B	75	19200	01001011	K	╯	k	K	
$4C	76	19456	01001100	L	└	l	L	
$4D	77	19712	01001101	M	╲	m	M	
$4E	78	19968	01001110	N	╱	n	N	
$4F	79	20224	01001111	O	┌	o	O	
$50	80	20480	01010000	P	┐	p	P	
$51	81	20736	01010001	Q	●	q	Q	
$52	82	20992	01010010	R	─	r	R	
$53	83	21248	01010011	S	♥	s	S	
$54	84	21504	01010100	T	│	t	T	
$55	85	21760	01010101	U	╭	u	U	
$56	86	22016	01010110	V	╳	v	V	
$57	87	22272	01010111	W	○	w	W	
$58	88	22528	01011000	X	♣	x	X	
$59	89	22784	01011001	Y	│	y	Y	
$5A	90	23040	01011010	Z	♦	z	Z	
$5B	91	23296	01011011	[	┼	[	┼	
$5C	92	23552	01011100	£	▒	£	▒	
$5D	93	23808	01011101	]	│	]	│	
$5E	94	24064	01011110	↑	π	↑	▒	
$5F	95	24320	01011111	←	◥	←	▧	
$60	96	24576	01100000	─		─		
$61	97	24832	01100001	♠	▌	A	▌	
$62	98	25088	01100010	│	▄	B	▄	
$63	99	25344	01100011	─	▔	C	▔	
$64	100	25600	01100100	─	▁	D	▁	
$65	101	25856	01100101	▔	▏	E	▏	
$66	102	26112	01100110	─	▒	F	▒	
$67	103	26368	01100111	│	▕	G	▕	
$68	104	26624	01101000	│	▒	H	▒	
$69	105	26880	01101001	╮	◤	I	▧	
$6A	106	27136	01101010	╰	▕	J	▕	
$6B	107	27392	01101011	╯	├	K	├	
$6C	108	27648	01101100	└	▗	L	▗	
$6D	109	27904	01101101	╲	└	M	└	
$6E	110	28160	01101110	╱	┐	N	┐	
$6F	111	28416	01101111	┌	▂	O	▂	
$70	112	28672	01110000	┐	┌	P	┌	
$71	113	28928	01110001	●	┴	Q	┴	
$72	114	29184	01110010	─	┬	R	┬	
$73	115	29440	01110011	♥	┤	S	┤	
$74	116	29696	01110100	│	▎	T	▎	
$75	117	29952	01110101	╭	▍	U	▍	
$76	118	30208	01110110	╳	▐	V	▐	
$77	119	30464	01110111	○	▔	W	▔	
$78	120	30720	01111000	♣	▀	X	▀	
$79	121	30976	01111001	│	▃	Y	▃	
$7A	122	31232	01111010	♦	┘	Z	✓	
$7B	123	31488	01111011	┼	▖	┼	▖	
$7C	124	31744	01111100	▒	▝	▒	▝	
$7D	125	32000	01111101	│	┘	│	┘	
$7E	126	32256	01111110	π	▘	▒	▘	
$7F	127	32512	01111111	◥	▚	▧	▚	

				UP		DOWN		
HEX	DEC	HIGH DEC	BINARY	ASCII	SCN	ASCII	SCN	BASIC TOKEN
$80	128	32768	10000000		@		@	END
$81	129	33024	10000001	▌	A	▌	a	FOR
$82	130	33280	10000010		B		b	NEXT
$83	131	33536	10000011		C		c	DATA
$84	132	33792	10000100		D		d	INPUT#
$85	133	34048	10000101	■	E	E	e	INPUT
$86	134	34304	10000110	■	F	F	f	DIM
$87	135	34560	10000111	■	G	G	g	READ
$88	136	34816	10001000	■	H	H	h	LET
$89	137	35072	10001001	■	I	I	i	GOTO
$8A	138	35328	10001010	■	J	J	j	RUN
$8B	139	35584	10001011	■	K	K	k	IF
$8C	140	35840	10001100	■	L	L	l	RESTORE
$8D	141	36096	10001101	■	M	M	m	GOSUB
$8E	142	36352	10001110	■	N	N	n	RETURN
$8F	143	36608	10001111		O		o	REM
$90	144	36864	10010000	■	P	P	p	STOP
$91	145	37120	10010001	■	Q	Q	q	ON
$92	146	37376	10010010	■	R	R	r	WAIT
$93	147	37632	10010011	■	S	S	s	LOAD
$94	148	37888	10010100	■	T	T	t	SAVE
$95	149	38144	10010101	■	U	U	u	VERIFY
$96	150	38400	10010110	■	V	V	v	DEF
$97	151	38656	10010111	■	W	W	w	POKE
$98	152	38912	10011000	■	X	X	x	PRINT#
$99	153	39168	10011001	■	Y	Y	y	PRINT
$9A	154	39424	10011010	■	Z	Z	z	CONT
$9B	155	39680	10011011	∷	[	∷	[	LIST
$9C	156	39936	10011100	■	£	■	£	CLR
$9D	157	40192	10011101	■	]	■	]	CMD
$9E	158	40448	10011110	π	↑	✧	↑	SYS
$9F	159	40704	10011111	◣	←	▒	←	OPEN
$A0	160	40960	10100000		█		█	CLOSE
$A1	161	41216	10100001	▌	!	▌	!	GET
$A2	162	41472	10100010	▄	"	▄	"	NEW
$A3	163	41728	10100011	▔	#	▔	#	TAB(
$A4	164	41984	10100100	▁	$	▁	$	TO
$A5	165	42240	10100101	▏	%	▏	%	FN
$A6	166	42496	10100110	▒	&	▒	&	SPC(
$A7	167	42752	10100111	▕	'	▕	'	THEN
$A8	168	43008	10101000	▒	(	▒	(	NOT
$A9	169	43264	10101001	◤	)	▨	)	STEP
$AA	170	43520	10101010	▕	*	▕	*	+
$AB	171	43776	10101011	├	+	├	+	-
$AC	172	44032	10101100	▗	,	▗	,	*
$AD	173	44288	10101101	└	-	└	-	/
$AE	174	44544	10101110	┐	.	┐	.	↑
$AF	175	44800	10101111	▁	/	▁	/	AND
$B0	176	45056	10110000	┌	0	┌	0	OR
$B1	177	45312	10110001	┴	1	┴	1	>
$B2	178	45568	10110010	┬	2	┬	2	=
$B3	179	45824	10110011	┤	3	┤	3	<
$B4	180	46080	10110100	▏	4	▏	4	SGN
$B5	181	46336	10110101	▎	5	▎	5	INT
$B6	182	46592	10110110	▐	6	▐	6	ABS
$B7	183	46848	10110111	▔	7	▔	7	USR
$B8	184	47104	10111000	▀	8	▀	8	FRE
$B9	185	47360	10111001	▃	9	▃	9	POS
$BA	186	47616	10111010	┘	:	✓	:	SQR
$BB	187	47872	10111011	▖	;	▖	;	RND
$BC	188	48128	10111100	▝	<	▝	<	LOG
$BD	189	48384	10111101	┘	=	┘	=	EXP
$BE	190	48640	10111110	▘	>	▘	>	COS
$BF	191	48896	10111111	▚	?	▚	?	SIN

				UP		DOWN		
HEX	DEC	HIGH DEC	BINARY	ASCII	SCN	ASCII	SCN	BASIC TOKEN
$C0	192	49152	11000000	─	█	─	█	TAN
$C1	193	49408	11000001	♠	█	A	█	ATN
$C2	194	49664	11000010	│	█	B	█	PEEK
$C3	195	49920	11000011	─	█	C	█	LEN
$C4	196	50176	11000100	─	█	D	█	STR$
$C5	197	50432	11000101	─	█	E	█	VAL
$C6	198	50688	11000110	─	█	F	█	ASC
$C7	199	50944	11000111	│	█	G	█	CHR$
$C8	200	51200	11001000	│	█	H	█	LEFT$
$C9	201	51456	11001001	╮	█	I	█	RIGHT$
$CA	202	51712	11001010	╰	█	J	█	MID$
$CB	203	51968	11001011	╯	█	K	█	
$CC	204	52224	11001100	└	█	L	█	
$CD	205	52480	11001101	╲	▧	M	█	
$CE	206	52736	11001110	╱	▨	N	█	
$CF	207	52992	11001111	┌	█	O	█	
$D0	208	53248	11010000	┐	█	P	█	
$D1	209	53504	11010001	●	█	Q	█	
$D2	210	53760	11010010	─	█	R	█	
$D3	211	54016	11010011	♥	█	S	█	
$D4	212	54272	11010100	│	█	T	█	
$D5	213	54528	11010101	╭	█	U	█	
$D6	214	54784	11010110	╳	▨	V	█	
$D7	215	55040	11010111	○	█	W	█	
$D8	216	55296	11011000	♣	█	X	█	
$D9	217	55552	11011001	│	█	Y	█	
$DA	218	55808	11011010	♦	█	Z	█	
$DB	219	56064	11011011	┼	▚	┼	▚	
$DC	220	56320	11011100	▒	▓	▒	▓	
$DD	221	56576	11011101	│	█	│	█	
$DE	222	56832	11011110	π	█	▒	█	
$DF	223	57088	11011111	◥	◣	▨	▓	
$E0	224	57344	11100000		█		█	
$E1	225	57600	11100001	▌	▐	▌	▐	
$E2	226	57856	11100010	▄	▀	▄	▀	
$E3	227	58112	11100011	▔	█	▔	█	
$E4	228	58368	11100100	▁	█	▁	█	
$E5	229	58624	11100101	▏	█	▏	█	
$E6	230	58880	11100110	▒	▒	▒	▒	
$E7	231	59136	11100111	▕	█	▕	█	
$E8	232	59392	11101000	▒	▓	▒	▓	
$E9	233	59648	11101001	◤	◢	▨	▓	
$EA	234	59904	11101010	▕	█	▕	█	
$EB	235	60160	11101011	├	█	├	█	
$EC	236	60416	11101100	▗	▛	▗	▛	
$ED	237	60672	11101101	└	█	└	█	
$EE	238	60928	11101110	┐	█	┐	█	
$EF	239	61184	11101111	▂	█	▂	█	
$F0	240	61440	11110000	┌	█	┌	█	
$F1	241	61696	11110001	┴	█	┴	█	
$F2	242	61952	11110010	┬	█	┬	█	
$F3	243	62208	11110011	┤	█	┤	█	
$F4	244	62464	11110100	▎	█	▎	█	
$F5	245	62720	11110101	▍	█	▍	█	
$F6	246	62976	11110110	▐	█	▐	█	
$F7	247	63232	11110111	▔	█	▔	█	
$F8	248	63488	11111000	▀	█	▀	█	
$F9	249	63744	11111001	▃	▀	▃	▀	
$FA	250	64000	11111010	┘	█	✓	█	
$FB	251	64256	11111011	▖	▜	▖	▜	
$FC	252	64512	11111100	▝	▙	▝	▙	
$FD	253	64768	11111101	┘	█	┘	█	
$FE	254	65024	11111110	▘	▟	▘	▟	
$FF	255	65280	11111111	π	▞	▒	▞	