COMMODORE 64

BusCard®

BATTERIES INCLUDED
TORONTO, ONT. CANADA

The Batteries Included BusCard

by Jim Law

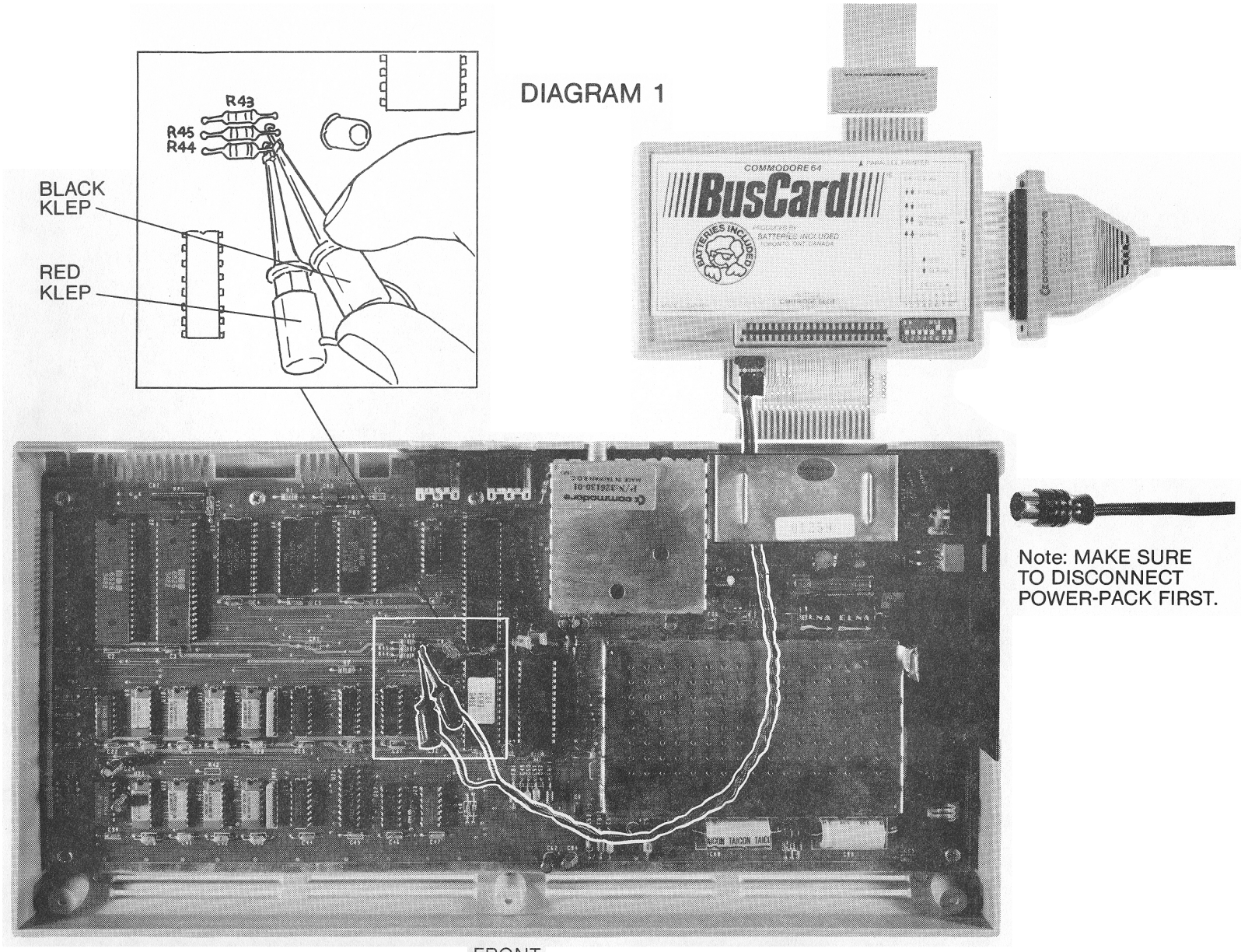
DIAGRAM 1
R43
R45
R44
BLACK
KLEP
RED
KLEP
COMMODORE 64
BusCard
BATTERIES INCLUDED
Note: MAKE SURE
TO DISCONNECT
POWER-PACK FIRST.
FRONT

1.Ø - Installing your BusCard

1.1 - Unpacking

Before installing your BusCard in your Commodore 64 check the package you received to make sure you have everything listed below.

-The BusCard interface
-BusCard cable with 2 clips at one end
-Warranty registration card

In addition, you should have on hand the following items:

-small phillips (cross) head screwdriver for opening your C-64
-PET to IEEE-488 cable to connect any IEEE peripherals to the BusCard. Available from any Commodore dealer.
-parallel printer cable to connect any parallel printer to the BusCard. This is available as an option with the BusCard or as a separate item at the dealer where you purchased the BusCard. (Part # BC-par)

1.2 - Installation

Make sure that the power for your Commodore 64 computer is disconnected and that any cartridges and connectors are removed. Open your computer by removing the three phillips head screws under the bottom front of the machine. Hinge the top up and back to remove the top of the computer. It will still be attached by the keyboard and power light wires.

Just left of the centre line of the computer board and about half way back are two cylindrical devices (resistors) labelled R44 and R45. Locate these and clip one of the two connectors on the supplied cable to the right end of each of the two resistors. The

red clip should attach to R44 and the black one to R45. For best connections, hook the clips under the resistor leads from front to back and lay the clips down toward the front of the machine. (see diagram 1)

Route the cable to the back right of the machine and pass the four place connector out of the machine under the metal shield surrounding the cartridge port connector.

Re-assemble the computer, making sure that the case is fully closed and that the clips are laying down towards the front.

Place the BusCard interface behind the computer and push the cable connector hanging out of the cartridge slot onto the corresponding right angle connector at the front of the BusCard. The connector is keyed to ensure correct installation but check that all the pins are engaged.

Insert the BusCard into the cartridge slot of your computer. If you have to remove the BusCard temporarily from your computer, the cable may be disconnected and left installed; it will not interfere with normal operation of the machine.

Reconnect only the power connector to the C-64 and turn the unit on. It should power up and display the normal sign-on message. Turn the unit off.

Connect any IEEE-488 peripherals to the connector at the right side of the BusCard. Pet to IEEE cables are usually keyed and are inserted label side up.

Connect the parallel printer (if any) to the printer port at the back of the BusCard. This cable is optional. The other end of the cable should plug directly into the parallel input connector on your printer.

2.Ø - Setting the Switches

The switches which are located on the top of the BusCard are used to tell the C-64 where to look to find a particular device. With these switches it is possible to have a mixed system consisting of some devices on the IEEE-488 bus, some on the Commodore Serial bus, and one on the BusCard parallel printer port. These switches are generally set once for your particular system and then left. The settings on the switches are only checked when the machine is powered ON (reset) or when a soft-rest (RUN-STOP/RESTORE) is performed.
This means that if you change the switches you must do one of the above before the new switch settings will be recognized by the BusCard.

2.1 - Device 4 (printer)

Switches 1 and 2 on the top of the BusCard control the allocation of device 4. These switches control what port will be used if a program talks to device 4. This is usually the printer and the BusCard allows the following options in connecting your printer.

Type of device	sw1	sw2	where to connect
IEEE-488 printer (2Ø22,4Ø22/3,8Ø23 etc.)	ON	OFF	BusCard (at right)
Commodore Serial printer (1515,1525)	ON	ON	C-64 Serial connector
Parallel printer (Epson,Centronics etc.)	OFF	*	BusCard (at back)

* If switch 1 is OFF, configuring device 4 for a parallel printer, then switch 2 enables or disables automatic character conversions from PET ASCII (PETSCII) to true ASCII (as used by the printer). If switch 2 is OFF, the conversions are disabled; if

switch 2 is ON, conversions will be performed to make the upper and lower cases on the printer and C-64 screen agree. This is useful if a BASIC program which assumes that you have a CBM printer is required to print to a printer which is true ASCII. Most word processors will require that this switch be OFF: they perform their own character conversions.

2.2 - Devices 5-1Ø (including disk)

Switches 3 to 8 on the top of the BusCard control the device allocation for devices 5 to 1Ø respectively. If a switch is ON then that device is allocated to the IEEE-488 bus on the BusCard (at right). If a switch is OFF then that device is assumed to be on the Commodore Serial bus, available on the back of the C-64.

Device #	Switch #
5	3
6	4
7	5
8	6 (disk)
9	7
1Ø	8

2.3 - Devices 11 and up

Devices 11 and up have no switches and are permanently allocated to the IEEE-488 bus on the BusCard.

2.4 - General Configuration

You need only set the switches for the devices you attach. Other switches may be ignored and the system will react with the normal "DEVICE NOT PRESENT" error message if they are accessed.

Some users may already have interfaces from Commodore Serial or IEEE-488 buses to their parallel printers. These interfaces may be connected to the bus they normally use or the printer may be connected directly to the BusCard parallel port. Set the switches to allocate the printer device (4) to the appropriate port, not necessarily the parallel port.

If you have an IEEE disk drive and a 1541 Serial disk drive and wish to connect both of them to your C-64, there are two options to consider:

You may leave both drives as device 8, one on the Serial bus and one on the IEEE-488. This requires you to switch between the two buses by changing switch 6 and resetting the machine (turning it OFF and back ON) or performing a RUN-STOP/RESTORE to re-read the switches.

Alternately, you may change the device number of one of the disk drives to 9. This allows you to set up device 8 on one bus and device 9 on the other. Both Serial and IEEE disk drives may have their device numbers changed by cutting a trace in the drive. See your dealer for information on this. The IEEE disk drive models 4040, 8050, 8250 and the hard disk drives 9060 and 9090 may be temporarily changed to device 9 with the following short program:

```
10 open1,8,15
20 print#1,"m-w"chr$(12)chr$(0)chr$(2)chr$(41)chr$(73)
```

... this program may be cancelled by resetting the drive.

2.5 - Re-reading the switches

If you change the switches because of some change in device allocation, the changes will not affect the current allocation until you force the C-64 to re-read the switches. This is done automatically on power up. If you do not wish to turn OFF the machine, you may cause the switches to be re-read by performing a RUN-STOP/RESTORE by pressing and holding the 'RUN-STOP' key and then striking the 'RESTORE' key.

3.0 - Basic 4.0

Basic 4.0 is a set of additional commands added to the regular set of commands available in the standard Basic 2.0 language. This section is intended to outline and explain in simple terms the additional commands available with the Basic 4.0 language. It is not intended to teach programming or explain in great detail the techniques which can utilize these commands. For more comprehensive information the following sources may be useful:

User's Reference Manual - Commodore BASIC Version 4.0 - Part# 321604
Commodore Business Machines, Inc - check with your local dealer

User's Manual for CBM 5 1/4-inch Dual Floppy Disk Drives - Part# 320899
Commodore Business Machines, Inc - check with your local dealer

PET PERSONAL COMPUTER GUIDE - Adam Osbourne, Jim Strasma, Ellen Strasma
copyright 1982 - Osbourne/McGRaw-Hill - ISBN 0-931988-76-4

The following convention has been used for the Basic 4.0 command descriptions. Commands are in UPPER case, variable data is in lower case. Square brackets [and] indicate an optional parameter. Quotes "" indicate that a string constant or variable is required. Angle brackets <> indicate that a constant or variable may be used. In most cases when a variable is used it must be enclosed in brackets () to avoid a SYNTAX error.

3.1 - Turning Basic 4.0 ON and OFF

When the C-64 is first powered on, the Basic 4.0 language extension is not enabled. Enable it with the following command: **SYS 61000** . To disable the Basic 4.0 language and return operation to the standard C-64 Basic 2.0 language enter the following command: **SYS 61003** . Do NOT enable or disable the Basic 4.0 language from within a Basic program - only from the keyboard in immediate mode.

The Basic 4.0 does use some memory when enabled. Whenever the Basic 4.0 is enabled all variables are cleared and the top of memory is moved down to provide the memory Basic 4.0 needs. If some other utility has already moved the top-of-memory pointers then Basic 4.0 will adjust itself to use memory just below that area. When disabled, Basic 4.0 will adjust the top-of-memory pointers back to the normal position only if the pointers had not been altered since Basic 4.0 turn on. In the case where another program has altered the top-of-memory pointers, Basic 4.0 will not readjust them when disabled.

3.2 - DOS file names

The Commodore DOS systems allow considerable flexibility in specifying which files some commands will act upon. For example, if you wished to scratch (delete) a file or group of files, you could specify the file name in many different ways:

```
fred      -the single file 'fred'
fred*     -any and all files starting with the
           characters 'fred'
fr?d      -any and all files matching 'fr?d', where
           '?' is any character
fr?d*     -any and all files matching 'fr?d', with
           any (or no) characters following
Ø:fred    -the file 'fred' on drive Ø
1:fred*   -any and all files on drive 1 starting
           with 'fred'
fred=prg  -the program file 'fred'
Ø:f*=seq  -any and all sequential files on drive Ø
           which start with 'f'
1:*=rel   -any and all relative files on drive 1
```

Many of the commands which the disk drive is capable of performing can use some or all of the methods shown above to specify files. Others will require a specific format as described in the individual command summaries below.

3.3 - DS and DS$ (Disk Status) and ST (I/O Status)

Because the disk drive controller is a separate computer, any errors it encounters during operation must be manually retrieved from the drive to the computer. Error messages are usually indicated by a RED centre light on dual slot drives, or a rapidly flashing RED light on single slot drives. Basic 4.0 can retrieve and display an error message from the disk drive simply by refering to the pseudo-variables DS or DS$. DS and DS$ are used as if they were regular Basic variables, except that you cannot assign values to them. Instead, when you refer to either DS or DS$ after issuing a disk related command (such as DSAVE), the computer will fetch the current error status of the disk unit last referred to and store the resulting string in DS$. The first number in the DS$ string is stored in DS - this is known as the error code. It is important to note that DS and DS$ are only updated when you refer to them AND a disk related command has been issued since the last reference to DS or DS$. If you refer to DS or DS$ and a disk command has not been issued since the last reference, then the current values are reused and the drive is not queried. This allows programming of the form:

```
IF DS <> 0 THEN PRINT DS$ : STOP
```

There are three messages which are not errors (and do not light the error lamp) which may be retrieved by DS and DS$.

73, cbm dos v.....,00,00 - This message waits in the drive after power-up until a command is executed.

00, ok,00,00 - This is the message returned if the previous command was executed with no errors. It is also returned when there was an error, DS or DS$ was referenced, and for some reason the drive was queried a second time without any disk command being issued (this normally shouldn't happen since DS and DS$ keep track of disk commands).

01, files scratched,xx,00 - This message is only returned after a SCRATCH command is executed by the drive. The two digits 'xx' indicate the number of files actually scratched by the drive - it will be zero if no files were scratched.

Note1: If no device responds to Basic 4.0's request for the disk message then DS will have a value of 0 to 20. DS$ will have a length of zero.
Note2: The act of fetching DS or DS$ will cause ST to be changed - therefore you must save ST before checking DS and DS$ if ST is needed later in the program.

3.31 STATUS - (ST) - result of last I/O operation

ST is used as if it was a regular Basic variable, except that you cannot assign a value to it. It is used as a variable which contains the result of the last attempted I/O operation. The value may vary from 0 to 255 depending on the success of the I/O with the various bits indicating exact nature of the problem.

Bit position	ST numeric value	IEEE meaning
0	1	time out on write
1	2	time out on read
2	4	none
3	8	none
4	16	none
5	32	none
6	64	EOI (end or identify)
7	-128	device not present

Status values of 1 and 2 indicate that the device you are trying to communicate with is not accepting or sending data. For instance, if in reading from a sequential file the program attempts to read more data than exists in the file, the drive will TIME OUT ON READ and status will equal 2. A status value of 64 on reading from a file indictes that the last data item fetched is the last data item in the file and there are no more to follow. The computer often generates a status of 64 when it is sending data to the disk drive or printer. If status is -128, then the device number you are trying to communicate with does not exist.

3.4 - Basic 4.0 commands

3.41 - APPEND

Format - APPEND# <file number> , "<name>" [,D<x>] [ON U<y>]

Use - To add additional data to the end of a sequential disk file.

Notes - APPEND is used like a DOPEN command but can only be used to add data to an existing sequential disk file. APPEND opens the specified data file for write and positions the DOS pointers to the current end of the file and new data can be added. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0.

Example - x=1 : APPEND#1, "data file", D(x) ON U 9
open a file with a logical number of 1 called "data file" on drive 1 of unit 9.

3.42 - BACKUP

Format - BACKUP D<x> to D<y> [ON U<z>]

Use - To create an exact duplicate of a diskette using a dual drive

Notes - Not applicable for single slot drives.
Both drive numbers must be specified and must be 0 and 1. Unit defaults to device 8. Any variable or evaluated expressions must be enclosed in parentheses.

Example - BACKUP D0 to D1

3.43 - CATALOG or DIRECTORY

Format - CATALOG ["<pattern>",] [D<x>] [ON U<y>]
DIRECTORY ["<pattern>",] [D<x>] [ON U<y>]

Use - Displays on the current output device (usually the screen) the disk directory from the specified drive. If no drive is specified both drives are searched in a dual drive. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8. Pattern matching as described in Section 3.2, Page 7 may be used to view limited segments of the directory. Pressing the SPACE bar will cause the scroll of the display to pause, pressing it again will continue the display. The STOP key terminates the listing.

Example - CATALOG D0 ON U 9
DIRECTORY D0, "fred*"

3.44 - COLLECT

Format - COLLECT [D<x>] [ON U <y>]

Use - Causes the disk drive to review the block allocations for all files in the directory, deleting all incorrectly closed files (those marked with an asterisk in the directory). The block count and allocation map are updated to correspond correctly to the files currently on the disk. Space allocated by the BLOCK-ALLOCATE command is freed since it is not linked to the directory. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0.

Example - COLLECT D0

3.45 - CONCAT

Format - CONCAT [D<x>,] <"namea"> TO [D<y>,] "nameb" [ON U<z>]

Use - Causes the sequential file "nameb" to have the data in "namea" appended to it. The data in the file "namea" is unaltered. Can only be used with sequential data files. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0.

Example - CONCAT "two" TO "one" ,D1
creates a file named "one" which contains "one" + "two"

3.46 - COPY

Format - COPY [D<x>,] "<namea>" TO [D<y>,] "<nameb>" [ON U<z>]
or - COPY D<x> TO D<y> [ON U<z>]

Use - causes the disk drive to copy a file or files from one place to another within a disk unit. It can be used to copy data from one disk to another in a dual drive and to copy a file to another place on the same diskette. Pattern matching as described in Section 3.2, Page 7 may be used to copy multiple files. If the file is to be copied to the same diskette then "nameb" must differ from "namea". COPY without file names copies all files from one diskette to the other in a dual drive without erasing those already there. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8.

Example - COPY D0 to D1
COPY D1, "temp" TO "tempb" :copies "temp" to "tempb" on drive 1

3.47 - DCLOSE

Format - DCLOSE [#<x>] [ON U<y>]

Use - Close one or many currently open disk files. DCLOSE will only close those disk files which were currently open in the computer's internal file table. If a logical file number <x> is specified then only that file will be closed. If only the unit number <y> is specified then all files on that unit will be closed. If both logical and unit numbers are specified then the unit number is ignored. If no parameters are given then all disk files will be closed. Relative record files may have more records generated at close time than were actually written in order to fill out the sectors in use. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8.

Example - DCLOSE :closes all disk files currently open in the computer
DCLOSE#2 :close the file associated with logical number 2

3.48 - DLOAD

Format - DLOAD "<name>" [,D<x>] [ON U<y>]

Use - Load a Basic program from disk, relocating it if necessary. Any variable or evaluated expressions must be enclosed in parentheses. Pattern matching as described in Section 3.2, Page 7 may be used. If an asterisk (*) is used as the filename then the last accessed program on the disk will be loaded. If the last file accessed was not a program file then the first program file in the disk directory will be loaded. Unit defaults to device 8. If no drive is specified both drives are searched in a dual drive. If used in program mode the subsequently loaded program will begin execution with the first line.

Example - DLOAD "first file" ,D1

3.49 - DOPEN

Format - DOPEN# <a> , "<name>" [,L<y>] [,D<x>] [ON U<z>] [,W]

Use - Open a sequential or random access file for read or write. <a> is the logical file number. Logical file numbers greater than 127 cause each PRINT# statement to transmit a LINE FEED character and a CARRIAGE RETURN character. The L<y> parameter only needs to be specified when opening a relative record file for the first time. Relative record files are always opened for both read and write. If not specified, sequential files are oped for read. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0. Both drives in a dual drive will be searched if all the optional parameters are missing. If the first character of the filename is the @ (at) symbol, it is removed and the specified file is opened replacing any existing file with that name provided that the file type is the same.

Example - DOPEN# 5 , "fred" , L 27 , D1
open a relative file, record length of 27 bytes, on drive 1.

DOPEN# 2 , "@seq file", W
open a sequential file on drive 0 for write with a logical file # of 2. If "seq file" previously existed it has now been replaced.

3.50 - DSAVE

Format - DSAVE "<name>" [,D<x>] [ON U<y>]

Use - Save a Basic program to disk. The "name" parameter can be up to 16 characters long. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0. If the first character of the filename is the @ (at) symbol, it is removed and the specified file replaces any existing file with that name provided that the file type is the same.

Example - DSAVE "my program" ,D1
DSAVE "@freds prog" : save "freds prog" on drive 0 replacing any previous program file named "freds prog".

3.51 - HEADER

Format - HEADER "<disk name>" [,D<x>] [I<zz>] [ON U<y>]

Use - To format (new) a new disk or erase the contents of an old one. The <disk name> may be up to 16 characters long. A 'STRING TO LONG error will occur if the <disk name> is greater than 16 characters long. If the I<zz> parameter is missing then the drive will attempt to reformat only the directory track. The <zz> parameter may be specified as a string variable. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0.
When used in immediate mode the question 'ARE YOU SURE?' will be asked before the command proceeds. Answer Y or YES to continue, N to abort. On completion the screen will display 'READY' if the disk was correctly formatted and 'BAD DISK' if any error was encountered. The actual error can be displayed by printing the contents of DS$.

CAUTION: The HEADER command will erase all information previously stored on the disk! Be careful!

Example - HEADER "my first disk" , D1 ON U 8 , Ibz
HEADER "next disk"

3.52 - RECORD

Format - RECORD# <logical file#> , <record#> [,<byte position>]

Use - Used before a GET#, INPUT# or PRINT# statement to position to the correct record number in a random access data file opened with the <logical file number>. <record#> must be between 1 and 65535 inclusive. <byte position> must be between 1 and 254 inclusive. <byte position> defaults to 1 if omitted. Any variable or evaluated expressions must be enclosed in parentheses.
If the <record#> causes the pointer to be positioned beyond the current end of file (as it would be when adding data to the file) the disk status (DS$) will show a RECORD NOT PRESENT error. If the intent is to add data to the file, then the next PRINT# statement will cause that record and any necessary intervening records to be generated. If INPUT# is executed then a null value is returned with STATUS (ST) set to EOI (64).

Example - RECORD# 1, 402 , 5 : positions to byte 5 of record 402

3.53 - RENAME

Format - RENAME [D<x>,] "<old name>" TO "<new name>" [ON U<y>]

Use - Changes the name of a specified disk file. The file must not be currently open. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0. It is not possible to RENAME a file to another name already existing on the disk.

Example - RENAME "freds file" TO "my file" , D1

3.54 - SCRATCH

Format - SCRATCH "<name>" [,D<x>] [ON U<y>]

Use - Erase a disk file or files. Any variable or evaluated expressions must be enclosed in parentheses. Unit defaults to device 8, drive 0. When used in immediate mode the question 'ARE YOU SURE?' will be asked before the command proceeds. Answer Y or YES to continue, N to abort. Multiple files may be scratched at one time by using the pattern matching as described in Section 3.2, Page 7.

Example - SCRATCH "my file" : erase only "my file" from drive 0
SCRATCH "fred*" ,d1
erase all files on drive 1 with names beginning with "fred"

3.60 - AUTO RUN (shift RUN/STOP)

Format - Press the RUN/STOP key while holding the SHIFT key

Use - Pressing the RUN/STOP key while holding the SHIFT key down will cause the computer to load and run the first program on drive 0 of device 8.

4.Ø - The Machine Language Monitor

It is beyond the scope of this manual to teach machine language programming so the next section is necessarily superficial and only describes how to use the included monitor, and not why.

The monitor prompts the user with a single dot when it is expecting a command.

4.1 - Entering the Monitor

The monitor is disabled on power-up and the following command must be issued to enable and enter it for the first time: **SYS 61ØØ6.** Once enabled, the monitor will be entered whenever the 65Ø2 executes a BRK instruction, such as when typing SYS 8. If a Run-stop/Restore is performed, the monitor will be disabled and must be SYS'ed again to restore it. It is, therefore, always a good idea to enter the monitor with SYS 61ØØ6 to avoid confusion.

4.2 - The Register Display (R)

Upon entering the monitor, and upon execution of the 'R' command, the MLM (machine language monitor) will display the contents of some pertinent CPU registers. The register names are given on one line with their corresponding contents shown below:

```
B*  PC   AC SR XR YR SP
.; Ø4F9 2A 3Ø FF 54 ED
```

PC - The 65Ø2 program counter when the BRK instruction was executed.
AC - The 65Ø2 .A register (accumulator) contents.
SR - The 65Ø2 status register (.P) register contents.
XR - The 65Ø2 .X index register contents.
YR - The 65Ø2 .Y index register contents.

SP - The bottom of the 65Ø2 hardware stack (in this case $Ø1ED).

All values are represented in hexidecimal. The contents of any of the registers may be changed by moving back over the line with the values on it, typing over the old contents and striking 'RETURN'.

4.3 - Memory Display and Modify (M)

The contents of any number of memory locations may be displayed and modified with the 'M' command. Typing 'M', followed by a space, followed by the starting address, followed by a space, followed by the ending address, will display as many lines of eight bytes as required to show the entire range. The starting and ending addresses are four hex digits and both must be present. The ending address must be higher than the starting address.

```
.M Ø3FØ Ø4Ø2
.: Ø3FØ 25 74 EE D4 6F AA ØØ Ø2
.: Ø3F8 AD CØ 14 15 85 51 EE 35
.: Ø4ØØ ØØ 12 Ø4 ØA ØØ 97 45 46
```

As with the register display, any of the displayed memory locations may be changed merely by typing over the two hex digits and striking 'RETURN'. All of the bytes displayed on that line are re-written to memory. In some instances, particularly when viewing the I/O chips, it may not be desirable to write to all of the bytes on a displayed line. In this case, simply write spaces over the displayed bytes which are not to be re-written when 'RETURN' is pressed: only the bytes remaining shown on the line will be written. If a '?' is displayed by the monitor after a memory byte when that line has just been re-written, it means that the previous byte was written, but when re-read to verify, was found to be in error; this may be acceptable depending on what is present at that address.

4.4 - Disassembling 650Ø2 Instructions (D)

A screen of 650Ø2 mnemonics may be displayed for a block of memory by typing 'D' followed by the starting address. The memory address, actual memory byte contents, and the mnemonic equivalent of the instruction will be displayed for 23 sequential processor instructions. Undecodable opcodes will be shown as '???'.

Instructions shown on the screen may only be modified by re-typing the byte content field of each line, and not the mnemonic field. Sequential pages of display are possible by striking 'D' 'RETURN' on the bottom line of suceeding pages.

4.5 - Simple Assembler (A)

Included in the monitor is a non-symbolic assembler which will translate 650Ø2 mnemonics and operand fields to equivalent 650Ø2 code bytes in memory. The format is:

.A XXXX III OOOO

...where XXXX is the intruction's address, III is the mnemonic, and OOOO is the (optional) operand field.

All values must be entered in hex and must be one or two bytes long and preceded by a '$'. Immediate operands would take the form: #$xx. Branch instructions have the destination address for an operand; the assembler will calculate the offset by itself. To obtain examples of the format of the assembler, try disassembling portions of the ROM code in the machine and examining the printed format; it is the same as that required by the assembler.

The assembler will automatically generate the address of the next instruction, so writing short segments of code is quite simple. For longer programs of course, a symbolic assembler is recomended.

4.6 - Saving Memory (S)

It is possible to save a block of memory to a storage device (disk or cassette) with the following command:

.S "name",aa,bbbb,cccc

...where 'name' is the file name including any drive number etc. 'aa' is two hex digits for the device number. 'bbbb' is the four hex digits of the starting address. 'cccc' is the four hex digits of the end of **block+1.**

4.7 - Loading Memory (L)

It is possible to load a block of memory from a storage device with the following command:

.L "name",aa

Where 'name' and 'aa' have the same meaning as above. Note that the load is non-relocating and correnspponds to the 'LOAD "name",8,1' of BASIC.

4.8 - Executing Code (G)

From within the monitor, you may directly pass control to a routine at some address with the Go command. If no address is given after the 'G' command, then the last address shown in the Register display of the program counter (PC) is the location where execution will resume. Alternately, you may specify a four hex digit address at which to start.

4.9 - Leaving the Monitor (X)

Typing 'X' followed by 'RETURN' will exit the monitor and return you to BASIC.

4.91 - Hunt memory (H)

A specified section of memory may be searched looking for a specified sequence of bytes. Type 'H' followed by a space, followed by the starting address, followed by a space, followed by the ending address, followed by a space, followed by the first byte in the search string, followed by a space, followed by the second byte, etc. The starting and ending addresses are four hex digits and both must be present. The ending address must be higher than the starting address. The hunt function is also capable of searching for a string of PET-ASCII characters. To search for a string, type a single quote character followed by the search string in place of the byte sequence above.

```
.H 040A 41BF 4C 00 A0
.04F3 05D2 215A 41A0

or...   .H 0803 105B 'BASIC
        .A002 010E
```

The address of each area of memory that contains the specified search string is displayed, followed by a space, followed by the next address, until all occurances of the search string have been noted.

4.92 - Printing disassembly (P)

The P command provides a continuous disassembly of memory from a starting address to an ending address. Output is of the same form as the D command, but does not stop until the ending address is reached. Before using this command it is neccessary to open the output channel from Basic. From Basic type:
OPEN 4,4 : CMD4 : SYS 61006

Once in the monitor, type 'P' followed by a space, followed by the starting address, followed by a space, followed by the ending address. The disassembly output will be directed to the printer. When the printing has stopped, exit the monitor by typing 'X' and 'RETURN'. Then type PRINT#4 and 'RETURN'.

```
.P C000 C504
.X
READY
PRINT#4
```

Printed output from the other functions of the monitor can be accomplished in the same way by typing the appropriate command in place of the 'P' instruction.

4.93 - Fill memory (F)

A specified section of memory may be filled with a specific byte. Type 'F' followed by a space, followed by the starting address, followed by a space, followed by the ending address, followed by a space, followed by the byte that memory is to be filled with. The starting and ending addresses are four hex digits and both must be present. The ending address must be higher than the starting address.

```
.F 050C 21CA FF
```

4.94 - Transfer memory (T)

A specified section of memory may be transferred to another area in memory with the 'T' command. Type 'T' followed by a space, followed by the starting address, followed by a space, followed by the ending address, followed by a space, followed by the new starting address. The starting and ending addresses are four hex digits and both must be present. The ending address must be higher than the starting address. The original area of memory is unchanged. Care must be taken when the new memory area overlaps the old area.

```
.T A000 BFFF 4000
```

Appendix A - Theory of operation

The BusCard operates by permanently mapping out a section of the internal KERNAL ROM in the C-64. All of the normal Serial bus routines are present in the segment of that ROM from $EC00 to $EFFF. The BusCard ROM replaces this segment of KERNAL in order to dispatch the low-level commands to the proper device handler depending on the switch allocation of that device number. The actual device handlers (IEEE, Serial bus, Parallel port) and the machine language monitor and DOS Wedge are switched into the memory space only when required. Their image appears in place of the BASIC ROM from $A000 to $BFFF. This means that the machine language monitor may not be used to examine memory in the BASIC ROM or in the RAM behind the BASIC or KERNAL ROMs.

The switch settings are read only on power up or RUN-STOP/RESTORE. The settings are stored in the time of day registers in the 6526 at $DD00, so these registers always should remain untouched.

If the KERNAL ROM image is transfered back into the RAM behind $E000-FFFF and the ROMs disabled, the ROMs are temporarily restored for any I/O operations by the BusCard.

used as a permanent variable location.

The input/output chip used by the BusCard is a 8255A-5, mapped into the I/O space from $DEC0 to $DEFF. The lower portion of this I/O space is available to any other device which might be plugged into the BusCard cartridge extension.

The two conections made using clips inside the C-64 are used to obtain the HIMEM and LOMEM control bits from the processor output port. These allow the complete invisibility of the BusCard by ensuring that the interface ROMs disappear and appear using the same commands as the internal ROMs.

Because of the ability of the BusCard to allow mixed device allocation, operations which cause a

device on any bus (IEEE, Serial, or parallel) to LISTEN (as after a CMD) and then perform some other bus action (usually a directory), will not have the transfer sent to the LISTENed device. If a listing of a directory is required, do not do this:

```
OPEN 4,4
CMD 4
LOAD "$Ø",8 or >$Ø
```

...instead, perform two seperate operations...

```
LOAD "$Ø",8
OPEN 4,4
CMD 4
LIST
PRINT#4
CLOSE 4
```

Note that this works because only one bus device is involved at a time.

Appendix B - Programming considerations

Because of the efforts taken to ensure maximum invisibility, very few precautions need be taken when programming with the BusCard installed.

1) - All input/output operations should be done through the KERNAL ROM routines. They may be called directly but the preferred method uses the Commodore jump table. None of the Serial routines should be entered in the middle; the BusCard only traps the normal entry points.

2) - Avoid using the time of day or alarm register in the 6526 at $DD00; use the other 6526 at $DC00.

3) - Use a soft-loaded monitor program if it is necessary to program in the RAM behind the ROMs.