DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

DI-SECTOR V3.0

STARPOINT SOFTWARE

```
0101001101010000001000001010101000001101001001
0100000010100110000100000000101010001001000
0100000010100111001001010110101001100100000
0101010001001111100100000000100110101001001
0100101101000101001000000000100101000101110
0010000000100100001000001010100111001010010
0101100100100000001000001100100111101010010
0010000000101010001001010000100010100100000
0100111001001001010000000100100000100100 1100
0100010100100000001000000010100111001000100
0010000000100000101010100100101010001010011
001000000010000010010000000101000001101001011
01010101010100000001000010010000100100001
```

DI-SECTOR V3.0

STARPOINT SOFTWARE

122 S. Broadway Yreka, CA 96097 (916) 842-6183

TABLE OF CONTENTS

INTRODUCTION

Welcome to the world of disk utilities. This DI-SECTOR diskette contains many disk utilities which represents over two years of hard work inside of the 1541 drive. This version of DI-SECTOR itself is credited with one full year of work. With all of this research & development we feel that we have created the best disk utility for the Commodore 1541 disk drive to date.

DI-SECTOR V3.0 is fully menu driven, and contains many disk utilities for the beginning and advanced Commodore user. It contains a protected and unprotected disk copier for both 1 and 2 disk drives, an easy to use file copier to transfer individual files between diskettes, a sector editor which allows individual sectors to be repaired or modified in HEX, ASCII, and assembler a full featured machine language monitor for both the computer and the disk drive, which supports undefined opcodes, indirect searching, etc, a format editor which has many advanced utilities such as creating/repairing disk errors, a block identifier utility which allows viewing of the latest protection schemes and a new Art's backup which copies the latest Electronic Arts (tm) games.

MAKING BACKUPS & PIRACY

The DI-SECTOR diskette is not protected for many reasons. One of which is that we (the producers) feel that you (the customer) have a right to a backup of your diskette in the case that it is damaged. We recommend that you make a backup of the DI-SECTOR diskette before you start using it, and put it in a safe place in the case of an emergency. You can backup the DI-SECTOR diskette with the unprotected disk copier (option B).

We also feel that the protection of diskettes has gotten out of hand. Ever since the introduction of DI-SECTOR V2.0 there has been mass confusion on both sides of the protection wars. The vendors make protection that the current copiers can't copy, and the copy program vendors all race to copy the latest protection schemes. Copy protection is very expensive and time consuming; not only does it raise the price of the programs, but it delays the release as well.

The only way that lack of disk protection will work is if people only use the backup copies for themselves, and not give them to other people.

SYSTEM REQUIREMENTS

DI-SECTOR is compatible with the C-64 & C-128 computers, with 1 or 2 disk drives.

Assorted modules of the DI-SECTOR utility achieve their speed through the re-writing of the disk operating system (DOS). The modules that do this can only be used with Commodore 1541 disk drives and COMPATIBLES. The follow-

ing is a partial list of drives that are considered "compatible drives":

TRUE COMPATIBLE WITH 1541	NOT TRUE COMPAT.
Commodore 1540, 1541, & 1571	MSD SD-1 & SD-2
Comtel Enhancer 2000	Comm. SFD-1001
Indus GT disk drive	Comm. 4040
Most "copy cat" drives	Hard disks

Some of the modules will work with "non-true-compatible" drives. The following is a list of the modules, and their specific requirements:

DI-SECTOR MODULE	REQUIREMENTS
Option A: Nibble Backup	True compat.
Option B: Unprotected Backup	True compat.
Option C: File Backup	All disk drives
Option D: Sector Editor	All disk drives
Option E: Format Editor	True compat.
Option F: Art's Backup	True compat.
Option G: STARMON Monitor	All disk drives
Option H: Renumber Drive	True compat./MSD

Note: When using a C-128, you must be in C-64 mode.

DI-SECTOR MAIN MENU

The DI-SECTOR main menu allows you to load any of the many "sub-programs" in the DI-SECTOR system. To load the main menu simply type:

```
LOAD ":*",8,1  [RETURN]
```

After a few moments, you will be presented with a menu of the available options and, at the bottom of the menu, the current source & destination drive settings.

If the source & destination drive settings are acceptable, simply type the letter (A-I) that corresponds to the module you wish to load.

If a single disk drive is present, it will be used as both the source and destination drive. If a second disk drive is available, then it will be chosen as the default destination drive.

If you do not wish to use the default source & destination settings, you may change them with the following keys:

[F1]	increment source device number
[F3]	decrement source device number
[F5]	increment dest. device number
[F7]	decrement dest. device number

Devices not present on the serial cable are displayed in light red. Pressing the [RETURN] key will restore the source and destination default settings.

The SECTOR EDITOR, FORMAT EDITOR, STARMON, & ART'S BACKUP modules will default to the source drive for all I/O operations.

Note: Pressing the left-arrow key allows you to look at the DI-SECTOR cover screen.

OPTION A: PROTECTED BACKUP

This module allows you to backup protected diskettes, and should be used for duplicating commercially protected software.

* COMMAND SUMMARY *

[C]	Begin copy process
[D]	Directory of diskette
[B]	Boot DI-SECTOR diskette (return to menu)
[Q]	Quit- Return to BASIC (warm start)

If desired you may change, the beginning & ending track number and the track increment by pressing [RETURN] to move the cursor to the respective prompt and typing the appropriate numbers. The numbers accepted are 0-40; half tracks are allowed.

Press [C] to begin the copy process, and you will be prompted to insert the appropriate disk(s). When using a single drive, five read/write passes will be required with a 6th required for disks needing parameters. You will be prompted to insert the required disks when necessary. If using 2 drives, no disk swapping will be required.

Note: The parameters for a disk will be activated when using 1 drive only. If you wish to copy disks that need parameters (see enclosure), copy the disk using a single drive.

Another Note: When copying with 2 drives, the nibbler will "bump" the head of the secondary drive for calibration purposes the first time it is used.

OPTION B: UNPROTECTED BACKUP

This module allows you to backup unprotected diskettes. This program should be used for copying your own programs, public domain disks, or disks that aren't protected; it will copy an entire disk in the fastest time possible.

When using a single drive, 3 read/write passes will be required to duplicate the disk, and you will be prompted to swap disks when appropriate. Duplication with two drives requires no disk swapping and is considerably faster.

Note: When copying with 2 drives, the copier will "bump" the head of the secondary drive for calibration purposes the first time it is used.

OPTION C: FILE BACKUP

This option is used for copying individual files from diskette to diskette.

* COMMAND SUMMARY *

[R]	Read directory of diskette
[CRSR KEYS]	Move red "X" cursor around

[SPACE BAR]	select/de-select file to be copied
[A]	Reverse file select status of all files
[C]	Copy selected files
[S]	Scratch selected files
[F]	Format a diskette
[Q]	Quit- Return to main menu

Note: This file copier will copy a file of any length, even files larger that 64k long. In order to achieve this, the program uses the "append" command which may allocate an extra sector of the disk. Because of this, you may wish to validate the destination disk (DOS "V" command) after duplicating an extr-emely large file, (over 200 blocks.)

OPTION D: DISK SECTOR EDITOR

This module allows the user to display infor-mation about an individual disk. From this point you are presented with a sub-menu of 4 items:

COMMAND 1: EDIT SECTORS

This major portion of the Sector Editor will allow you to read sectors, modify them in Hex or ASCII and write them back to the disk again. This section includes a help screen, which can be invoked by typing a question mark. The full command set is as follows:

H	Hex Entry. This allows you to enter Hex data.
J	Jump. When executed with the cursor on the first byte of the sector, will jump to the next sector of the current file.
M	Monitor. Sends you off into Starmon. Return by typing 'X'.
R	Read Sector, this will move the cursor to the T window, and allow you to type in a track and sector. When that is completed, the selected track and sector will be read.
W	Write Sector. This will write the selected track and sector back to the disk after being modified.
T	Text Entry. This allows you to enter ASCII text into the buffer directly.
F1	Will increment the track pointer and read the new track/sector.
F3	Will decrement track pointer and read the new track/sector.
F5	Will increment the sector number and read new track/sector. If the sector number is greater than the number of sectors on the track then the track will be incremented and the sector number will be zero.
F7	Will decrement sector number and read new track/sector. If sector is less than zero then the track number will be decremented and the sector will be set to the last sector of that track.

The screen layout during this program includes the ASCII display, the disk status line, followed by the HEX display. To the right hand

side of the screen is the Track pointer, Sector poitner, Position pointer, and Decimal, which is the decimal value of the byte under the cursor.

COMMAND 2: DISPLAY BAM

This feature will display the Block Allocation Map of the disk in the drive. A dash mark is un-allocated, while the dot symbol means the sector is used.

COMMAND 3: SEND DOS COMMAND

Here you can send a valid DOS command or take a directory (see apendix C.)

COMMAND 4: RETURN TO MAIN MENU

Return to the Di-Sector menu after a brief interlude with the cover mess.

OPTION E: FORMAT EDITOR

This module quickly formats disks and allows you to experiment with disk protection schemes, both primitive and advanced. To take FULL advantage of some of these options a fairly detailed background knowledge of the 1541 is desired. We recommend reading Inside Commodore DOS (see back of manual).

* COMMAND SUMMARY *

[C]	Create disk errors
[T]	Check track for errors
[W]	Check whole disk for errors
[R]	Repair a track of read errors
[F]	Format a diskette
[B]	Block identifier utility
[P]	Software write protect a disk
[U]	Software un-write protect a disk
[Q]	Quit- return to main menu

COMMAND C: CREATE DISK ERRORS

This option allows you to create disk read errors 20, 21, 22, 23, 27 & 29. Disk errors were widely used for disk protection around the summer of 1983. Errors are a good way to start experimenting with disk protection.

In order to use the create error utility, you simply have to enter the error number you wish to create, the track number, and the beginning and ending sector numbers. The following is a list of errors, the description of the errors, along with any comments and/or limitations of that error:

NOTE: Error 21 is a very flaky error to write. Any slight deviation of your disk rotations speed will cause one sector too many, or one sector too few to be written to the disk. Do not try to adjust your speed, however, just repair the track and try again. In most instances it does not matter if you write too many error 21s to the track as most software protection schemes will not check.

Also, if you are writing error 21, you MUST write at least FOUR consecutive sectors with that error to any particular track.

```
ERROR 20: BLOCK HEADER NOT FOUND
ERROR 21: NO SYNC CHARACTER
ERROR 22: DATA BLOCK NOT PRESENT
          (must enter data block identifier)
ERROR 23: CHECKSUM ERROR IN DATA BLOCK
          (must enter data block checksum)
ERROR 27: CHECKSUM ERROR IN HEADER BLOCK
          (WHOLE TRACK ONLY- must  enter header
           block checksum for sector 0)
ERROR 29: DISK ID MISMATCH
          (WHOLE TRACK ONLY- must enter ID)
```

COMMAND T: CHECK TRACK FOR ERRORS

This option allows you to check any track (1-35) for disk read errors. These are the same read errors that appear in the previous section (COMMAND C: CREATE DISK ERRORS). The information that is given is as follows:

TRACK NUMBER, SECTOR NUMBER, ID HI BYTE, ID LO BYTE, HEADER BLOCK CHECKSUM, DATA BLOCK CHECKSUM, and ERROR NUMBER if applicable.

COMMAND W: CHECK WHOLE DISK FOR ERRORS

This option simply places the previous option (OPTION T: CHECK TRACK FOR ERRORS), in a loop checking tracks 1 through 35. If an error is encountered on a track, the computer waits for a key press before continuing the check of the disk.

COMMAND R: REPAIR TRACK OF READ ERRORS

This option repairs read errors on a disk. To use, simply enter number of track to be repaired (1-35). This utility reads all of the data off of the track, re-formats the track, and then writes the data back to the disk.

COMMAND F: FORMAT A DISKETTE

This is just a "simple" 8 second format utility, which allows you to format a diskette like the DOS "NEW0:" command. This utility also allows you to enter a 5 byte ID. The first 2 bytes are the standard ID bytes, and the other three bytes are just for show. On a "normally" formatted diskette these bytes are usually ' 2A'. If you only type 2 bytes for the ID, ' 2A' will be inserted automatically.

Note: The first time you format a diskette, it will "bump" the head for positioning. Multiple formats with this utility, however, will only "bump" the head once.

COMMAND B: BLOCK IDENTIFIER UTILITY

This is the most "advanced" utility in the Format Editor. It was conceived for the sake of looking at the newer protection schemes that don't use the standard DOS format. This means that you won't be shown if their are any DOS errors on the disk. This utility is

ideal for viewing "re-defined sector layout" protection schemes. Like the Epyx Breakdance protection having two sector 3's on track 16. Or looking at how track 34 extends to 34.5, 35, and 35.5 on Electronic Art's protection. This utility is also useful in viewing "extra sectors", "re-defined block ID bytes" and "half-tracking" like DI-SECTOR V2.0's track 8.5.

This utility shows the following:

BYTE RIGHT AFTER SYNC MARK (BLOCK ID)- If it's a $07 it prints "DATA", if it's a $08 then it prints "HEADER" otherwise it prints NS(x), where 'x' is the byte it found.

THE NEXT 2 BYTES ARE THE 3rd & 4th bytes found after the sync mark. These bytes usually indicate the track and sector numbers on a header block. This is a common thing to change on the newer generic protection schemes.

CURRENT DENSITY- The density indicator can be found on the upper left corner of the screen. It shows the current density. It shows up in light-red if you indicated a non-standard density for that track. You can indicate a non-standard density with the [D] command.

NUMBER OF SECTORS- This shows the number of sectors found on that track. If it is not the standard number of sectors, this number shows in light-red.

The following is a command summary of the Block ID utility.

[R]	Re-read current track
[D]	Change density of track, then re-read
[?]	Keyboard help screen (shows you this information)
[Q]	Quit- return to track prompt

USE THE ARROW KEYS FOR THE FOLLOWING COMMANDS

[UP]	Step head out a full track
[DN]	Step head in a full track
[LF]	Step head out a half track
[RT]	Step head in a half track

Note: The drive keeps running until you quit at the track prompt, and return to the format editor. The drive is initialized at this time.

HOW IT WORKS/LIMITATIONS:
This utility basically waits for a sync mark, records the next 5 GCR bytes and records the timer values in between sync marks. It does this 46 times. After this it converts all of the GCR bytes to data. It uses the timer values in between sync marks to determine where the "start" of the track is (the sector after the largest timer value). After this it tries to find similarities between the data to find the end of the track. With this information it assumes how many sectors are on the track.

There are a lot of limitations in this utility. One is if there are sectors with the exact same header information for every header and data block on the entire track. This means

that the significant bytes in each sector lie after the first 5 GCR bytes. A very good example of this is Datamost's Conan. Another limitation is not being able to look at multiple densities in a single sector, or picking apart a single sector at all.

COMMAND P: SOFTWARE WRITE PROTECT A DISK

This utility basically is an extension of the CREATE DISK ERRORS module. It creates a unique error- ERROR 73: CBM DOS V2.6 1541. This error is useful for simulating a write protect tab on a disk. Even though reading of a disk works properly, this error is incurred whenever a write to the disk is attempted.

All this option is doing, is changing the DOS ID byte (track 18, sector 0, byte 2) from a $41 to a $40.

COMMAND U: SOFTWARE UN-WRITE PROTECT A DISK

This option "repairs" an ERROR 73 created with the previous option, SOFTWARE WRITE PROTECT A DISK. This returns both reading and writing of a disk back to "normal".

This option changes the DOS ID byte back to $41.

OPTION E: ELECTRONIC ARTS (TM) BACKUP

This module allows you to backup Electronic Arts (tm) programs. To use this module you first copy the original disk with the Nibble Backup (option A), and then select this option. You then press [F] to fix the backup of the disk. You will then be prompted to insert disk to be fixed (press [RETURN] to continue). At the end of this process (about 5 seconds later) you will be prompted to remove the diskette, and place a write-protect tab on it.

Other options in this module are:
[D] Directory of diskette
[B] Boot DI-SECTOR diskette (return to menu)
[Q] Quit- Return to BASIC (warm start)

Note: This module works on the principle of re-writing the boot so that checks to tracks 34 & 35 are trapped and ignored. Electronic Arts' (tm) protection is written with a double wide head on tracks 34 & 35. You can view this with the Block Identifier Utility, and note how reliable the reading of sectors are on tracks 34-35.5 (most disk drives like the 1541, screw up half tracks when writing standard tracks because of the trim heads).

The machine the creates these errors is very expensive, and a very good example of how disk protection has gotten out of hand.

OPTION G: MACHINE LANGUAGE MONITOR

This option contains a very advanced machine language monitor which will allow you to write and debug programs of your own creation, or those of others. It is well suited to the beginner, as well as the advanced user. You will be prompted with a menu asking you where the monitor should load. You have a choice of 800 hex, 8000, or C000. Or you may at this time abort to the main menu. The monitor is also available from BASIC. The BASIC loading versions are loadable at three address for your convienience and are called the following:

MON RUN
MON 32768
MON 49152

To load the loaction $800 version of the monitor type LOAD "MON RUN",8 and the RUN. to load the other versions type LOAD"filename",8,1 then SYS addr (i.e. SYS 32768)

Notation Conventions in Monitor Documentation

<Numeric> is any of the following:

Hexadecimal: consisting of an optional '$' followed by hexadecimal digits.
Examples: $2020, 3F, $1ABD

Decimal: '!' followed by decimal digits.
Examples: !21, !32767, !100

Octal: '&' followed by octal digits.

Examples: &37, &10

Binary: '%' followed by binary digits.
 Examples: %11001010, %1100

Operators: Any numeric can include the math operators + and -. For instance $100-!25 is a valid numeric, with the value of 231 decimal.

<start-address> is any valid <numeric>, between $0000 and $FFFF.

<end-mark> is a <numeric> is the address of the end of a memory range. It can be indicated with either an address or a comma followed by a <numeric> length.
 Example: M 8000 8100 or M 8000,100
 will display both display $100 byte of memory beginning at location $8000.

<byte> is a <numeric> between 0 and $FF or text that is inside of quotes, which will generate one <byte> per ASCII charactor.

Examples: $34, 2D, !100, or &37.
Example of text: "ABCD" which generates the bytes $41 $42 $43 $44.

<device> a valid <numeric> device number between 0 and 15 decimal usually 8 for the first disk drive.

Note that operands in braces { and } are optional, and will default to various values depending upon the command.

Starmon has 6 command classes: Memory, Assembler, Execution, Register, Redirection, File manipulation, and Miscellaneous.

Summary of StarMon Commands:

Memory Commands
- : Set Memory
- F Fill Memory
- T Transfer Memory
- I Interpret as ASCII
- H Hunt memory
- C Compare memory

Assembler Commands
- A Assemble bytes
- D Disassemble bytes
- U Undefined opcode disassembly

Execution Commands
- G Go
- J JSR

Register Commands
- R Register Display
- * Register Modify

Redirection Command
- O Operate

File Commands
- L Load
- S Save

Misc Commands
- X Exit
- = Base conversion/Math
- @ Dos Wedge

Commands that operate on memory

The following commands allow you to display, and modify memory:

- : (set memory)
- Fill memory
- Transfer
- Memory display
- Interpret memory as ASCII
- Hunt for pattern.
- Compare memory.

[SET MEMORY]

```
 : <start address> <byte> <byte> <byte> ...
```

The set memory command will begin modifying memory at start address for as many bytes as are given on the command line.

Examples:

```
:4000 10 &21 %10101 'HELLO'
```

Will set memory from $4000 with the hexadecimal values $10, $11, $15, $48, $45, $4C, $4C and $4F.

[FILL MEMORY]

```
 F <start address> <end mark> <pattern>
```

where <pattern> is a string of up to 32 bytes.

Examples:

```
F C000 C100 'PATTERN ' 10 %1010
```

Will fill the memory from C000 to C100 inclusive, with the repeating hexadecimal bytes $50 $41 $54 $54 $45 $52 $4E $20 $10 $0A.

```
F !2048,8 &37
```

Will fill the memory from 32767 decimal, for 8 bytes with the value 37 octal.

[TRANSFER MEMORY]

```
 T <source start address> <end mark> <destination start address>
```

Transfers bytes from source area to destination area. Examples:

```
T 8000 8100 4000
```

Will move the bytes residing at $8000 through $8100 and duplicate them in the block $4000 through $4100.

```
T 1000,!100 $9000
```

Will move the bytes residing at 1000 hex, for 100 decimal bytes, to location 9000 hexadecimal. Note that the '$' in the '$9000' is optional.

Note that this command is especially useful in conjunction with the 'O' command explained later, for copying RAM to and from the disk drive.

[MEMORY DISPLAY]

```
 M { <start address> { <end mark> } }
```

This command will display bytes from memory starting at start address and continuing until end mark. If no end mark is given, it assumes 1 screenful of data. If no start address is given, it commences from the last address displayed. This command will always display at least eight bytes, and rounds the number of bytes displayed up to the next multiple of eight.

Examples:
M 8000
Will display one screenful of data in hexadecimal, and ASCII starting at 8000 hex.
M D020,!64
Will display 64 decimal bytes beginning at hex location D020, in hex and ASCII.
M
This if executed directly after the M D020,!64 will display the next screenful of memory, which will begin at D060.

[INTERPRET ASCII]

```
 I { <start address> { <end mark> } }
```

This command will display the memory selected in ASCII. If no end mark is specified, one screen full will be displayed. This command, similar to 'M' will display at least 32 bytes, and will round up to the nearest multiple of 32 bytes. Examples:

I 8000
Will display from 8000 hex, one screenful of memory in ASCII.

I !4096,!96
Will display 96 decimal bytes of data starting at 4096 decimal, in ASCII format.

[HUNT MEMORY]

```
 H <start address> <end mark> <pattern>
```

This command will search through memory searching for <pattern>, which may include question marks for any position, which is a wildcard match. Examples:

Assume that memory from $4000 contains the following data:

```
:4000 34 22 11 45 21 56 3F 1A
:4008 2B 33 4F C2 19 24 2E 5F
```

H 4000,F 1?
Will return all of the addresses between 4000 and 400F where the top nibble equals 1, in this example, 4002, 4007, and 400C.

```
H 4000 7000 "F?SH" %1???????
```
Will Find such things as "fish", "fosh", "fesh" etc. as long as they are followed by a byte with the hi bit set.

```
H 4000 7000 %01??????
```
Will return all of the addresses in the range which contain bytes which have bit 7 reset, and bit 6 set.

This is probably the most useful location for the binary numerics, as it allows you to search for specific bit patterns or set of instructions.

[COMPARE MEMORY]

```
C <start address> <end mark> <compare address>
```

This command will compare two areas of memory for differences. If a byte between the <start address> and <end mark> does not match a corresponding byte in the comparison area, then it's address will be printed.

NOTE: This can be quite a list if you attempt to compare two unrelated areas of memory; press stop to abort.

Commands that manipulate Assembly Language

There are 3 commands that allow you to program in assembly language.
- Assemble
- Disassemble

- Undefined opcode disassemble
- ; Assemble a single line of code

[ASSEMBLE]

```
A <address> <op-code> {<operand>}
```

This will begin assembling code at the address you specify.

Examples:
A 8000 LDA #$41
Will begin assembly mode, inserting the instruction LDA #$41 at location 8000. The monitor will then print on the following line, the next address, whereupon you have the option of hitting return again, and exiting assembly mode, or typing in another opcode such as:

```
A 8002 LDX #$04
Continuing...
A 8004 SEC
A 8005 DEX
A 8006 LDA #"A"
A 8008 <ret>
.
```

Note that the computer's output is underlined. If an illegal opcode, or the wrong operand for an opcode is attempted, the monitor will print a question mark at the point where the syntax error ocurred, and assembly will be aborted. Undefined opcodes may be used in the assembler. This interactivity is well suited to the beginning programming student. Also note that in the operand field, the default base for numerics is decimal (!), and not hex ($).

For more information on assembly language, consult a book specifically written for that purpose.

[DISASSEMBLE]

```
 D { <start address> { <end mark> } }
```

This will begin disassembling code in memory, displaying it in hex, 6502 instructions, and at the right-hand edge of the screen, ASCII. Illegal instructions are marked with ??? in the opcode field. Note that at the beginning of each line is the character ;, which will allow you to modify your code with the screen editor, and automatically re-assemble it. If an instruction passes over the end mark boundry, it will complete the display of that instruction. If the end mark is omitted, the disassembly will continue for 1 screen. If the start address is omitted, the disassembly will commence from the current address.

[UNDEFINED OPCODE DISASSEMBLY]

```
 U { <start address> { <end mark> } }
```

Identical to Disassemble, excepting that it will disassemble certain undefined opcodes. See Appendix A for a list of valid undefined opcodes.

[ASSEMBLE A SINGLE LINE]

```
; opcode {operand}
```

This command is for conviniance of modification with the full screen editor. When doing a disassembly a ";" is printed a the begining of each line to facilitate modification of M.L. statements. To change an instruction, simply overtype it and press return.
WARNING! typing an instruction longer in bytes than the current instruction will cause the next memory locations to be destroyed.

Commands which execute code
There are two commands that allow you to execute portions of your code. They are:

Go, and
JSR or Jump SubRoutine.

[GO]

```
 G { <address> }
```

Where address is any valid numeric where execution is to begin. When this command is executed the program counter is loaded with <address>, if given, and execution is passed to that location. If no address is given, then execution will continue with the current value of the program counter. Program execution will continue until it locates a BRK instruction ($00) or until the operator taps the <RESTORE> key. If the program encounters the BRK instruction, the monitor will print out the registers, and the program counter will point to the BRK

instruction. If the <RESTORE> key was pressed, the monitor will return with the program counter pointing to the instruction that was to be executed before interrupted.
Examples:
G !32768
Will begin execution at 32768 decimal.

G
Will begin execution at current value of program counter.

[JSR]

J { <address> }

This command is identical to GO, except that control will be returned to the monitor at the next RTS instruction. This is useful for testing subroutines.

<u>Commands that involve the registers</u>

There are two commands that allow the manipul-ation of registers.

<u>R</u>egister display, and
<u>*</u> Set Registers

[REGISTER DISPLAY]

R

This command will print out the contents of the program counter, the accumulator, the X

register, the Y register, stack pointer and the flags.
Examples:

```
R
      PC   AC XR YR SP  NV-BDIZC
     *2020 04 34 21 A9 %01000010
```

[SET REGISTERS]

*

This command is designed to be used with the register display command. Note that when the registers are displayed, the first character on the line is a '*'. This facilitates using the screen editor to modify registers, yet this command can be used stand-alone, but with great caution.
Example:

```
R
      PC   AC XR YR SP  NV-BDIZC
     *8020 10 23 34 B3 %01001001
```

The registers and flags can be changed by typing the new contents in the respective fields, and pressing the return key.

<u>Command for memory redirection</u>

[OPERATE]

The 'O' command, which stands for Operate, allows you to write code for your disk drive. The command takes one of three forms:
O <return> which displays the current Input and Output memory,

O <i/o device> which sets both input and output to the same device memory, and
O <input device> <output device> which allows any combination.

By example:

O 8 (Sets all input and output to go to device 8)
thus - M 1000 will display one screen of the drive memory.
O 8 0 (Sets to input from drive, and output to computer.)
thus - T 1000,100 4000 moves $100 bytes from drive memory location $1000 to $4000 in the computer memory.)

This feature is for advanced programmers wishing to write custom code for their disk drive, or other intelligent peripherals. All memory commands will work with redirection, as will execution and assembly commands. Register commands, however will not work.

Commands for loading and saving files

The two commands Load and Save are for loading machine language programs from disk, and saving them to disk. Load and save only operate with computer memory. Memory redirection has no effect.

[LOAD]

L "<file name>" <device> {<start address>}

This command will go to the disk specified by <device> and attempt to load <file name>. If no start address is given, then the start address the file was saved with is used. If a start address is given, it will load the program at that location.
Example:

L "FISH" 8
Will go to the disk, and attempt to locate the file FISH and load it into memory at it's default location. If the file is not found it will leave memory unmodified and return with a prompt. If the file is found, it will be loaded into memory and the prompt returned.

L "FROG" 8 2020
Will go to the disk, attempt to locate the file FROG and load it into memory starting at 2020 hex.

[SAVE]

S "<file name>" <device> <start address> <end mark>

This command will save the bytes between start address and end mark, minus one byte, under the <file name> on <device>. If the file already exists, it will not be overwritten.

Example:
```
 A 4000 INC $D020
 A 4001 JMP $4000
 A 4004 <ret>
 .S "FISH" 8 4000 4004
 SAVING "FISH"
 .
```

Miscellaneous Commands

There are, finally, three miscellaneous commands, @ which allows you to execute DOS commands, = which does base conversion and math, and X for exit.

[DOS WEDGE]

```
 @ <dos command>
```

This command allows the user to execute one of several DOS commands. For a complete list of DOS commands, and how to use them, refer to the manual for your disk drive.
Example:

```
@$
```

Will return the directory information for the current drive.

```
@NO: FISH,SA
```
Will format the selected disk, naming it FISH and giving it the ID code, SA.

[BASE CONVERSION/MATH]

```
 = <numeric>
```

This routine will convert one word from any of the valid bases to all of the other bases.

Example:
```
 = !32767
 * $7FFF !32767 %01111111 11111111
```
Note that instead of an asterisk, there will be a graphics character.

```
 = "S"+3
 V  $0056 !86 %00000000 01010110

 = &40
    $0020 !32 %00000000 00100000
```

[EXIT]

```
 X
```

This command will return you to BASIC via the BASIC "warm start". When using monitor in sector editor, you will return to the edit sector screen.

OPTION H: RENUMBER DRIVE

This option contains a utility that allows you to renumber a device if you have two drives that are the same device number. This option is usually for 2 drive use when you don't own both drives. If you own both drives, the most practical way is to permanently renumber the disk drive. This is explained in the Commodore 1541 users manual.

After selecting this option, you will first be prompted to select the device number to be changed to. The default device number is 09, but you can change this (08-11) with the [F1] & [F3] keys. You can return to the main menu at this time with the [Q] key.

To continue, turn off all other drives except the drive to be renumbered, and press the [RETURN] key.

You will then be prompted to turn on your other drives, and press [RETURN]. The drive is now renumbered, and you will return to the main menu.

OPTION I: QUIT- RETURN TO BASIC

This option simply returns you to BASIC via the old BASIC warm start.

* DI-SECTOR V3.0 CREDITS *

These are the people that wish to be remembered for their efforts in bringing you, DI-SECTOR V3.0.

Scott M. Blum.....	Unprotected Backup
	File Backup
	Format Editor
	Menu, Cover Screen & Layout
	Manual, label & logo artwork
Bruce Q. Hammond..	Sector Editor
	STARMON Mach. Lang. Monitor
	Original DI-SECTOR concept
	Fast Loader, and Manual

Special thanks to Bryce Nesbitt for GCR convert routine in Unprotected Backup, 3.0 raster routine on main menu, and for assorted technical help, and suggestions.

Thanks to Stan Krute for teaching us assembly language, and encouraging interest in microcomputers (whatever they are).

Thanks to Chip Gracey for technical help, and non-technical suggestions.

Thanks to Scott Statton, who prefers edlin to Word Perfect because of its many bugs, for work on manual, assorted criticism, and borrowing money.

Thanks to the entire Starpoint staff for sending out update notices, and surviving a half-dozen hackers wandering around writing disk utilities (VAH!).

Thanks to KILROY for assuming responsibility for ALL bugs in this manual, DI-SECTOR, and ISEPIC.

Three cheers for Bob Nolan, and his help on the DI-SECTOR logo, label, ad, and manual cover artwork.

Thanks to Mr. Murphy for inventing laws that can be used for blaiming when ANYTHING goes wrong.

And a very special thanks goes to the Pepsi Corporation for inventing Diet-Pepsi, without which this project would not have been possible.

And now, the part we have all been waiting for, your favorite and mine, the tm notices. (yea!)

RECOMMENDED READING

INSIDE COMMODORE DOS, by Richard Immers & Gerald Neufeld, Datamost- This book is an excellent introduction for all Commodore 1541 disk drive programmers. It gives a rundown on what all the subroutines are, in a readable source form. It also explains GCR encoding, etc. The main problem with this book is that the authors didn't receive any royalties because Datamost is in "chapter 11".

COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE, Commodore Business Machines- This book is a must for all serious Commodore 64 programmers It gives all of the KERNAL locations, and a rundown of what they all do, among other things. It's poorly laid out, but the information's all there.

THE ANATOMY OF THE 1541 DISK DRIVE, Abacus Software. This is an OK version of the 1541 disk drive's source code. The introduction is not worth buying the book for, and there are many errors in the source code itself. I would recommend listing the source code yourself, if you're serious about drive programming.

THE HITCHHIKER'S GUIDE TO THE GALAXY, by Douglas Adams. A great book! This book is a comical science fiction adventure of Arthur Dent. This book holds the answer to the universe, and gives the reader incredible insight to life the universe & everything.

IDEAS AND OPINIONS, by Albert Einstein. On the serious side. This book contains many essays which reveals the simple genius that Einstein is famous for. Physics are explained as well as many humanitarian essays.

RECOMMENDED WARES

MERLIN MACRO ASSEMBLER, by Glen Bredon- THE BEST macro assembler for any 6502 machine. Available for the Commodore 64, and Apple II. Fully featured line editor, searching, assembling from memory, monitor, Sourceror disassembler all included. Very well documented. Used to develop DI-SECTOR V3.0.

II/64 SYSTEM, by Chip Gracey- A development system which connects an Apple II to a Commodore 64. Assembly language development is done on the Apple version of MERLIN, and down loads the code to the Commodore after ASM. You only have to save your source once a day. A must for serious Commodore programmers. Used to develop DI-SECTOR V3.0.

KWIK WRITE WORD PROCESSOR, Datamost- The most under rated computer program to date. A very easy to use word processor for the Commodore 64, with very clean editing. Fully menu driven, and on-line help. KWIK WRITE is under rated because it's only $19.95!

MAC WRITE, Encore Systems- A fine example of a decent word processor, that best of all comes free with your Macintosh. Used in early drafts of this documentation.

AMIGA PAINT, Electronic Arts- The best graphics development package on any microcomputer. Similar to Macpaint but with very advanced features and the ability to take advantage of the Amiga's stunning graphics. Worth buying an Amiga for.

RECOMMENDED LISTENING

THE BEATLES- SGT. PEPPERS LONELY HEARTS CLUB BAND, THE WHITE ALBUM, & ABBEY ROAD. 1960's rock at it's best. Listen to the phenomenal "fab-four" from Liverpool, England at their best. John Lennon & Paul McCartney together form the world's most famous song writing team. George Harrison & Ringo Starr also perform some of their own songs.

SIMON & GARFUNKEL- BOOKENDS, SOUNDS OF SILENCE, & BRIDGE OVER TROUBLED WATER. Paul Simon (voice accompanied by Art Garfunkel) tactfully combines beautiful poetry, witty criticism, folk and rock music to achieve an easy listening music experience containing magic which cannot be denied.

MANNHEIM STEAMROLLER- FRESH AIRE I-V. A wonderful combination of classical and rock music. Music written by Chip Davis. Instruments used are piano, harpsichord, synthesizer, bass, lute, classical guitar, drums, recorder, oboe. Must be heard to appreciate.

JEAN MICHAEL JARRE- EQUINOX. Some of the most creative and inspired synthesized music of the twentieth century. This, as well as other albums by this composer, combine a unique blend of digital effects and musical virtuosity.

MOST GLAMOUROUS MEMORY LOCATIONS

The following list of memory locations are for our benefit only. We figure that if we put these locations in the DI-SECTOR manual, there will always be a copy of it laying around the Starpoint office. So much for honesty.

* COMPUTER MEMORY LOCATIONS *

STATUS.........	$90	MSGFLG.........	$9D
DEVICE.........	$BA	CURSBLINK......	$CC
CURSCOL........	$D3	CHARCOLOR......	$0286
REPEATFLAG.....	$028A	TBUFFER........	$0334
PRINTDECIMAL...	$BDCD	BASICEND.......	$9FFF
VICCONTROL.....	$D011	DATAPORTA......	$DD00
ACPTR..........	$FFA5	CHKIN..........	$FFC6
CHKOUT.........	$FFC9	CHRIN..........	$FFCF
CHROUT.........	$FFD2	CIOUT..........	$FFA8
CLALL..........	$FFE7	CLOSE..........	$FFC3
CLRCHN.........	$FFCC	GETIN..........	$FFE4
IOINIT.........	$FF84	LISTEN.........	$FFB1
LOAD...........	$FFD5	OPEN...........	$FFC0
PLOT...........	$FFF0	RESTORE........	$FF8A
SAVE...........	$FFD8	SECOND.........	$FF93
SETLFS.........	$FFBA	SETMSG.........	$FF90
SETNAM.........	$FFBD	TALK...........	$FFB4
TKSA...........	$FF96	UNLSN..........	$FFAE
UNTLK..........	$FFAB	VECTOR.........	$FF8D
BASICWARMST....	$FCE2		

* 1541 JOB QUEUE COMMANDS RUNDOWN *

READ SECTOR....	$80	WRITE SECTOR...	$90
VERIFY SECTOR..	$A0	SEEK SECTOR....	$B0
BUMP HEAD......	$C0	JUMP TO CODE...	$D0
EXECUTE CODE...	$E0		

* 1541 JOB QUEUE ERRORS RUNDOWN *

OK, EVERYTHING COOL (ERROR 0)	$01
HEADER BLOCK NOT FOUND (ERROR 20)	$02
SYNC TIMEOUT- NO SYNC FOUND (ERROR 21)	$03
DATA BLOCK NOT FOUND (ERROR 22)	$04
DATA BLOCK CHECKSUM ERROR (ERROR 23)	$05
WRITE VERIFY ERROR (ERROR 25)	$07
WRITE PROTECT TAB ON (ERROR 26)	$08
HEADER BLOCK CHECKSUM ERROR (ERROR 27)	$09
ID MISMATCH ERROR (ERROR 29)	$0B

* 1541 JOB QUEUE LOCATIONS RUNDOWN *

BUFF #0:	CNTRL:$00	TK:$06	SC:$07	LOC:$0300
BUFF #1:	CNTRL:$01	TK:$08	SC:$09	LOC:$0400
BUFF #2:	CNTRL:$02	TK:$0A	SC:$0B	LOC:$0500
BUFF #3:	CNTRL:$03	TK:$0C	SC:$0D	LOC:$0600
BUFF #4:	CNTRL:$04	TK:$0E	SC:$0F	LOC:$0700
BUFF #5:	CNTRL:$05	TK:$10	SC:$11	LOC:NORAM

* 1541 MEMORY LOCATIONS *

IDHI	$16	IDLO	$17
TRACK	$18	SECTOR	$19
HEADERCKSUM	$1A	DRIVESTATUS	$20
DVPNTR	$30	DVPNTRIND	$34
DATABLKID	$38	HDRBLKID	$39
CHECKSUM	$3A	SERIALPORT	$1800
DISKCNTRL	$1C00	DATAPORTA	$1C01
DATADIR4PORTA	$1C03	PCR	$1C0C
DISKINIT	$EAA0	WARMINIT	$EB22
MAXSEC	$F24B	PUT4GB	$F6D0
GET4GB	$F7E6	PUT4GBTABLE	$F8A0
GET4GBTABLE	$F8C0	GCR2BIN (BLK)	$F8E0
BIN2GCR (BLK)	$F78F		

APPENDIX A: 6502 UNDEFINED OPCODES

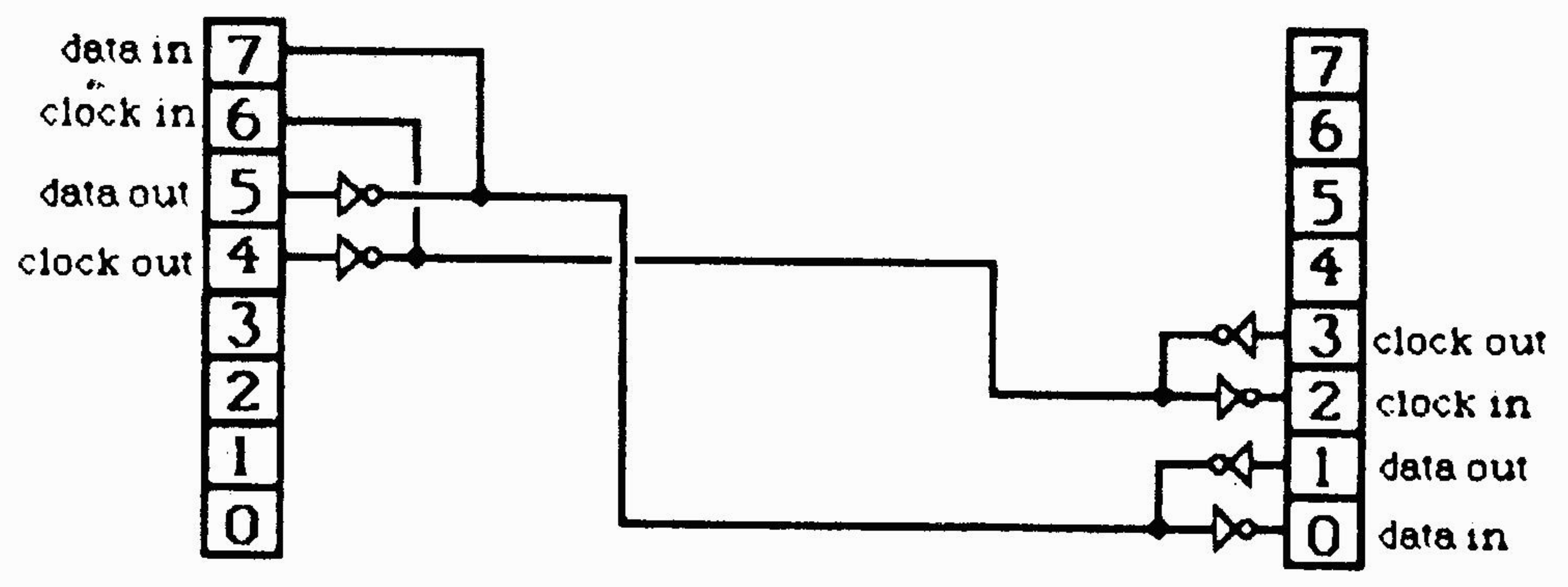
ASO:	ASL then ORA the result with the accum.
RLA:	ROL then AND the result with the accum.
LSE:	LSR then EOR the result with the accum.
RRA:	ROR then ADC the result with the accum.
AXS:	Store the result of A AND X
LAX:	LDA and LDX with the same data
DCM:	DEC memory and CMP the result with accum.
INS:	INC memory then SBC the result from accum.
ALR:	AND the accum. with data and LSR result
ARR:	AND the accum. with data and ROR result
XAA:	Store X AND data in the accumulator
OAL:	ORA with #$EE, AND result with data, then TAX
SAX:	SBC data from A AND result with X and store result in X
NOP:	No operation
SKB:	Skip byte (i.e. branch of +1)
SKW:	Skip word (i.e. branch of +2)

6502 UNDEFINED OPCODES

Intruction	Abs	Abs,X	Abs,Y	Zer	Zer,X	Zer,Y	(Ind,X)	(Ind),Y	Imm
ASO (ASL,ORA)	0F	1F	1B	07	17		03	13	0B
RLA (ROL,AND)	2F	3F	3B	27	37		23	33	2B
LSE (LSR,EOR)	4F	5F	5B	47	57		43	53	
RRA (ROR,ADC)	6F	7F	7B	67	77		63	73	
AXS (STX,STA)	8F			87		97	83		
LAX (LDX,LDA)	AF		BF	A7	B7		A3	B3	
DCM (DEC,CMP)	CF	DF	DB	C7	D7		C3	D3	
INS (INC,SBC)	EF	FF	FB	E7	F7		E3	F3	
ALR (AND,LSR)									4B
ARR (AND,ROR)									6B
XAA (TXA,)									8B
OAL (TAX,LDA)									AB
SAX (DEX,CMP)									CB

NOP	1A,3A,5A,7A,DA,FA
SKB	80,82,C2,E2,04,14,34,44,54,64,74,D4,F4
SKW	0C,1C,3C,5C,7C,DC,FC

SERIAL BUS FROM COMPUTER TO DRIVE



Computer side ($DD00)

Drive side ($1800)

DI-SECTOR V3.0

STARPOINT SOFTWARE

SOF0000212 $ 49.950
STA DISECTOR VER 3

DI-SECTOR V3.0

Starpoint Software revolutionalized the Commodore 64 computer software market with the release of the very first disk utility package DI-SECTOR V1.0. Then in the summer of 1984, Starpoint Software did it again with the release of the first 3 minute backup, Drive MON and Format Editor with the infamous DI-SECTOR V2.0. Now, Starpoint Software attempts to dumbfound all with the release of the fabled DI-SECTOR V3.0. Many people don't even believe their own eyes when they see this remarkable product in action. The following are some of the impressive features of this product:

* Unprotected disk backup, archives disks in only 48 seconds with verify!
* Protected disk backup, archives disks in only 1 minute!
* Copies ALL of the latest protection schemes including density switching, half tracks, sync tracks, long blocks, duplicate blocks, data/header block errors and more!
* Format a diskette in only 8 seconds!
* Ultra fast file copier allows the copying of files larger than 64k long! File Copier will copy between 1541, MSD, SFD 1001, hard disk drives, etc. easily and automatically!
* All copiers have options for 1 or 2 1541 disk drives!
* Full featured sector editor allows modifying of individual sectors in ASCII, HEX, or Assembler Language.
* Powerful machine language monitor allows modifications and debugging of programs in the computer and disk drive memory! Many features of monitor include assemble, disassembly of undefined opcodes, indirect searching, searching with wildcards, etc.
* Block Identifier Utility allows viewing of the latest protection schemes, including "Density Switching", "Half Tracks", "Redefined Sector Layout", etc.
* DI-SECTOR diskette NOT PROTECTED for unlimited archival backups!!!
* Create errors 20-21-22-23-27-29 easily and reliably!
* Repair damaged diskette by <u>removing</u> read errors!
* All these utilities included on one diskette.

STARPOINT SOFTWARE
122 S. Broadway Yreka, CA 96097 (916) 842-6183