M. David Johnson
http://www.bds-soft.com
info@bds-soft.com

False Disk Routines
for the
ML Foundation System

by M. David Johnson

2023/04/22

Abstract

The False Disk System is presented, as a subsystem of the ML Foundation System, to run on top of the ML Foundation Core, and to provide the convenience of using Linear Sector Numbers (0-2447) instead of the CoCo Disk System's Standard Drive, Track, and Sector numbers.

____

This paper and its associated code are available online at:

http://www.bds-soft.com/cocoPapers.php .

=====

Table of Contents

Introduction

The False Disk System is the first subsystem of the ML Foundation System to be developed, and it is designed to run on top of the ML Foundation Core.

Suppose we're designing a game which will need to use the CoCo disk system for temporary storage (for example, to store the contents of one screen of graphic data while we're working on another such screen).

How much more convenient would it be - instead of having to refer to Track 13, Sector 17; Track 13, Sector 18; Track 14, Sector 1; and Track 14, Sector 2 - to simply be able to refer to Linear Sectors 251, 252, 253, and 254.

That's what this False Disk System is all about.

Direct access to disk sectors is well documented in the CoCo manuals. In the BASIC Language, we can use DSKI$ and DSKO$ (DSOM, p.62).

And, in Assembly Language, we can use DSKCON (DSOM, p.60).

In fact, this False Disk System is built on top of the ML Foundation Core and uses that Core's DISKRW routine (MDJ01, p.200) which is itself built on top of DSKCON.

But none of this speaks to the desire to simply address disk sectors with the simplified concept of Linear Sector Numbers.

In this paper, I describe:

1. The BASIC Language programs to initialize, name, rename, copy, and test-fill False Disks of this system; and a program to display the contents of a specified Linear Sector Number.

2. The BASIC Language subroutines to translate a Linear Sector Number into a Drive Number, Track Number, and Sector Number (and vice-versa).

3. The BASIC Language subroutines to put a specified buffer to (and get a specified buffer from) a specified Linear Sector Number, using the translation subroutines of item 2 above.

4, The Assembly Language subroutines to translate a Linear Sector Number into a Drive Number, Track Number, and Sector Number (and vice-versa).

5. The Assembly Language subroutines to put a specified buffer to (and get a specified buffer from) a specified Linear Sector Number, using the translation subroutines of item 4 above.

6. The BASIC Language MAKEFALS.BAS program to collect the Assembly Language subroutines of items 4 and 5 above into a single FLSYS.BIN file.

The False Disk System, being duplicated in both BASIC and Assembly Language in this manner, also provides the additional service of facilitating the transfer of data between BASIC and Assembly Language programs and routines in 256-byte chunks; in addition to the simpler, but only single- and double-byte chunks of the ML Foundation Core's REGXFR transfer variables (MDJ01, p.23).

———

A Note on Numbers: To keep everything simple to understand, and also neatly lined-up, I frequently refer to numbers as decimal bytes with three full digits, e.g. 004, 027, 229, etc. See Appendix A for conversions between the decimal and hexadecimal representations of bytes. The leading zeroes are NOT intended to indicate octal notation. Octal notation is not used anywhere in this paper.

In works of this complexity (at least for me) typos and other errors are bound to sneak in. Please let me know about any you discover so I can note and correct them.

M.D.J. 2023/04/22
info@bds-soft.com

=====

General Methodology

Directly accessing disk sectors, whether via Linear Sector Numbers or not, carries a danger which Tandy warned all of us about (DSOM, p.62):

> DSKO$… outputs data directly to the sector you specify. Since it bypasses the disk filing system, it will output data without opening a file and listing its location in the directory. For this reason you need to be careful:
>
> 1. not to output data over the directory sectors unless you no longer plan to use the directory.
>
> 2. not to output data over other data you presently have stored on the disk.

In order to minimize these dangers, I am maintaining a modified Disk Directory and File Allocation Table (FAT) as a special identifier in the False Disk System. The Standard CoCo Floppy Disk consists of 35 Tracks of 18 Sectors each, for a total of 630 available separate sectors.

But, as with standard disks, I am reserving the entirety of Track 17 for a False Directory and FAT. The False Directory will provide a special indicator filename to help avoid the dangers noted above. Along with the False Directory, I'm also providing a False FAT which will simply indicate that the False Directory File takes up all 68 grans (i.e. all 34 remaining tracks) on the False Disk. Thus the False Disk, upon a DIR command, will typically display:

```
XXXXXXXX XXX  3 B 68
```

or some other filename.ext chosen by the user. And FREE will typically display 0.

Given this model, each floppy disk drive will provide 612 usable Linear Sector Numbers which will be allocated to the individual disk drives as:

Drive Number	Linear Sector Numbers
0	0 - 611
1	612 - 1223
2	1224 - 1835
3	1836 - 2447

With this arrangement, users designing a game might, for example, decide to specifically name their False Disks something like:

Drive Number	Filename.ext

```
0          GAMENAME.DR0
1          GAMENAME.DR1
2          GAMENAME.DR2
3          GAMENAME.DR3
```

_____

Note that the four Assembly Language Routines of the False Disk System are herein configured to LOADM at &H4416, the byte immediately following the last byte of the ML Foundation Core. However, they can be re-assembled and thus re-positioned at any addresses above &H4416 (including addresses above &H7FFF) which the user may prefer.

This can be simply accomplished by changing the “ORG” statements at the head of each routine, and changing the corresponding FLXLTL EQU addresses above the “ORG” in FLPUT.ASM and FLGET.ASM .

=====

FALSINIX.BAS: Program to Initialize a False File Allocation Table (FAT) And a False Directory On a False Blank Disk

```
1000 '*****
1010 '*
1020 '* FALSINIX.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* INITIALIZES A FALSE
1060 '* FILE ALLOCATION
1070 '* TABLE (FAT) AND A
1080 '* FALSE DIRECTORY ON
1090 '* A FALSE BLANK DISK
1100 '* INTENDED FOR USE BY
1110 '* DSKI$, DSKO$, AND
1120 '* THEIR MACHINE LANGUAGE
1130 '* EQUIVALENTS ONLY.
1140 '*
1150 '* THE FALSE XXXXXXXX.XXX
1160 '* DIRECTORY ENTRY AND
1170 '* FREE = 0 WILL SIGNAL
1180 '* TO THE USER THAT THE
1190 '* DISK IS A FALSE DISK.
1200 '*
1210 '* UPON COMPLETION, THE
1220 '* DIRECTORY WILL LOOK
1230 '* LIKE THIS:
1240 '*
1250 '* XXXXXXXX XXX  3 B 68
1260 '*
1270 '* AND WILL PROVIDE
1280 '* THIS RESULT:
1290 '*
1300 '* PRINT FREE(0) --> 0
1310 '*
1320 '*****
1330 '

1500 CLEAR &H1000
```

```
1510 '

1700 PRINT
1710 PRINT "    PUT THE DISK IN DRIVE 0"
1720 PRINT "  ***  ***  WARNING  ***  ***"
1730 PRINT "  *** DISK WILL BE ERASED ***"
1740 PRINT "    PRESS ANY KEY WHEN READY"
1750 A$ = INKEY$
1760 IF A$ = "" GOTO 1750
1770 PRINT
1780 PRINT "WORKING *";
1790 '

2000 'ERASE THE DISK
2010 X$ = "                                "
2020 Z$ = X$+X$+X$+X$
2030 FOR I = 1 TO 128
2040   MID$(Z$,I,1) = CHR$(0)
2050 NEXT I
2060 FOR T = 0 TO 34
2070   FOR S = 1 TO 18
2080     DSKO$ 0,T,S,Z$,Z$
2090   NEXT S
2100   PRINT "*";
2110 NEXT T
2120 '

2200 'GENERATE THE FALSE FAT
2210 F$ = Z$
2220 'SET GRANULES 0 TO 66
2230 FOR I = 1 TO 67
2240   MID$(F$,I,1) = CHR$(I)
2250 NEXT I
2260 'SET GRANULE 67
2270 '==> USE ALL 9 SECTORS
2280 MID$(F$,68,1) = CHR$(&HC9)
2290 'PUT TRACK 17, SECTOR 2 TO DISK
2300 DSKO$ 0,17,2,F$,Z$
2310 PRINT "*";
2320 '

2500 'GENERATE THE FALSE DIRECTORY
2510 D$ = Z$
2520 'SET FALSE FILENAME
2530 FOR I = 1 TO 8
2540   MID$(D$,I,1) = "X"
2550 NEXT I
```

```
2560 'SET FALSE EXTENSION
2570 FOR I = 9 TO 11
2580   MID$(D$,I,1) = "X"
2590 NEXT I
2600 'SET TYPE =
2610 '  TEXT EDITOR SOURCE
2620 MID$(D$,12,1) = CHR$(3)
2630 'SET FORMAT = BINARY
2640 MID$(D$,13,1) = CHR$(0)
2650 'SET NUMBER OF THE
2660 '  FIRST GRANULE
2670 MID$(D$,14,1) = CHR$(0)
2680 'NUMBER OF BYTES USED
2690 '  IN LAST SECTOR = 256
2700 MID$(D$,15,1) = CHR$(1)
2710 MID$(D$,16,1) = CHR$(0)
2720 'PUT TRACK 17, SECTOR 3 TO DISK
2730 DSKO$ 0,17,3,D$,Z$
2740 PRINT "*";
2750 '

2900 PRINT
2910 PRINT "FALSE INI X = DONE"
2920 '

32767 END
```

=====

FALSINIC.BAS: Program to Initialize a False File Allocation Table (FAT) And a False Directory On a False Blank Disk Using a User-Supplied Filename and extension

```
1000 '*****
1010 '*
1020 '* FALSINIC.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* INITIALIZES A FALSE
1060 '* FILE ALLOCATION
1070 '* TABLE (FAT) AND A
1080 '* FALSE DIRECTORY ON
1090 '* A FALSE BLANK DISK
1100 '*
1110 '* UTILIZES A FILENAME
1120 '* AND EXTENSION CHOSEN
1130 '* BY THE USER, HERE
1140 '* REPRESENTED BY:
1150 '* FILENAME.EXT
1160 '*
1170 '* INTENDED FOR USE BY
1180 '* DSKI$, DSKO$, AND
1190 '* THEIR MACHINE LANGUAGE
1200 '* EQUIVALENTS ONLY.
1210 '*
1220 '* THE FALSE FILENAME.EXT
1230 '* DIRECTORY ENTRY AND
1240 '* FREE = 0 WILL SIGNAL
1250 '* TO THE USER THAT THE
1260 '* DISK IS A FALSE DISK.
1270 '*
1280 '* UPON COMPLETION, THE
1290 '* DIRECTORY WILL LOOK
1300 '* LIKE THIS:
```

```
1310 '*
1320 '* FILENAME EXT  3 B 68
1330 '*
1340 '* AND WILL PROVIDE
1350 '* THIS RESULT:
1360 '*
1370 '* PRINT FREE(0) --> 0
1380 '*
1390 '*****
1400 '

1500 CLEAR &H1000
1510 '

1700 PRINT
1710 PRINT "    PUT THE DISK IN DRIVE 0"
1720 PRINT "  ***  ***  WARNING  ***  ***"
1730 PRINT "  *** DISK WILL BE ERASED ***"
1740 PRINT "    PRESS ANY KEY WHEN READY"
1750 A$ = INKEY$
1760 IF A$ = "" GOTO 1750
1770 PRINT
1780 '

2000 PRINT
2010 PRINT "ENTER FILENAME WITHOUT"
2020 PRINT "EXTENSION; 1-8 CHARS."
2030 INPUT FT$
2040 L1 = LEN(FT$)
2050 IF ((L1 < 1) OR (L1 > 8)) GOTO 3710  'ERROR
2060 L1 = 8 - L1
2070 FOR I = 1 TO L1
2080   FT$ = FT$ + " "
2090 NEXT I
2100 '

2200 PRINT
2210 PRINT "ENTER EXTENSION;"
2220 PRINT "1-3 CHARS."
2230 INPUT EX$
2240 L1 = LEN(EX$)
2250 IF ((L1 < 1) OR (L1 > 3)) GOTO 3710  'ERROR
2260 L1 = 3 - L
2270 FOR I = 1 TO L1
2280   EX$ = EX$ + " "
2290 NEXT I
2300 PRINT "WORKING *";
```

```
2310 '

2400 'ERASE THE DISK
2410 X$ = "                                "
2420 Z$ = X$+X$+X$+X$
2430 FOR I = 1 TO 128
2440   MID$(Z$,I,1) = CHR$(0)
2450 NEXT I
2460 FOR T = 0 TO 34
2470   FOR S = 1 TO 18
2480     DSKO$ 0,T,S,Z$,Z$
2490   NEXT S
2500   PRINT "*";
2510 NEXT T
2520 '

2700 'GENERATE THE FALSE FAT
2710 F$ = Z$
2720 'SET GRANULES 0 TO 66
2730 FOR I = 1 TO 67
2740   MID$(F$,I,1) = CHR$(I)
2750 NEXT I
2760 'SET GRANULE 67
2770 '==> USE ALL 9 SECTORS
2780 MID$(F$,68,1) = CHR$(&HC9)
2790 'PUT TRACK 17, SECTOR 2 TO DISK
2800 DSKO$ 0,17,2,F$,Z$
2810 PRINT "*";
2820 '

3100 'GENERATE THE FALSE DIRECTORY
3110 D$ = Z$
3120 'SET FALSE FILENAME
3130 FOR I = 1 TO 8
3140   MID$(D$,I,1) = MID$(FT$,I,1)
3150 NEXT I
3160 'SET FALSE EXTENSION
3170 FOR I = 9 TO 11
3180   MID$(D$,I,1) = MID$(EX$,I-8,1)
3190 NEXT I
3200 'SET TYPE =
3210 '  TEXT EDITOR SOURCE
3220 MID$(D$,12,1) = CHR$(3)
3230 'SET FORMAT = BINARY
3240 MID$(D$,13,1) = CHR$(0)
3250 'SET NUMBER OF THE
3260 '  FIRST GRANULE
```

```
3270 MID$(D$,14,1) = CHR$(0)
3280 'NUMBER OF BYTES USED
3290 '  IN LAST SECTOR = 256
3300 MID$(D$,15,1) = CHR$(1)
3310 MID$(D$,16,1) = CHR$(0)
3320 'PUT TRACK 17, SECTOR 3 TO DISK
3330 DSKO$ 0,17,3,D$,Z$
3340 PRINT "*";
3350 '

3500 PRINT
3510 PRINT "FALSE INI C = DONE"
3520 GOTO 32767
3530 '

3700 'ERROR HANDLING
3710 PRINT
3720 PRINT "*** ERROR ENCOUNTERED ***"
3730 PRINT "***  RUN  TERMINATED  ***"
3740 PRINT
3750 '

32767 END
```

=====

FALSCHGN.BAS: Program to Change the Name of a False Disk Using a User-Supplied Filename and extension

```
1000 '*****
1010 '*
1020 '* FALSCHGN.BAS
1030 '* MDJ 2023/02/24
1040 '*
1050 '* CHANGES THE NAME OF
1060 '* A FALSE DISK
1100 '*
1110 '* UTILIZES A FILENAME
1120 '* AND EXTENSION CHOSEN
1130 '* BY THE USER, HERE
1140 '* REPRESENTED BY:
1150 '* FILENAME.EXT
1160 '*
1170 '* INTENDED FOR USE BY
1180 '* DSKI$, DSKO$, AND
1190 '* THEIR MACHINE LANGUAGE
1200 '* EQUIVALENTS ONLY.
1210 '*
1220 '* THE FALSE FILENAME.EXT
1230 '* DIRECTORY ENTRY AND
1240 '* FREE = 0 WILL SIGNAL
1250 '* TO THE USER THAT THE
1260 '* DISK IS A FALSE DISK.
1270 '*
1280 '* UPON COMPLETION, THE
1290 '* DIRECTORY WILL LOOK
1300 '* LIKE THIS:
1310 '*
1320 '* FILENAME EXT  3 B 68
1330 '*
1340 '* AND WILL PROVIDE
1350 '* THIS RESULT:
1360 '*
1370 '* PRINT FREE(0) --> 0
1380 '*
1390 '*****
1400 '
```

```
1500 CLEAR &H1000
1510 X$ = "                                "
1520 Z$ = X$+X$+X$+X$
1530 FOR I = 1 TO 128
1540   MID$(Z$,I,1) = CHR$(0)
1550 NEXT I
1560 '

1700 PRINT
1710 PRINT "PUT THE FALSE DISK IN DRIVE 0"
1740 PRINT "PRESS ANY KEY WHEN READY"
1750 A$ = INKEY$
1760 IF A$ = "" GOTO 1750
1770 PRINT
1790 '

2500 PRINT
2510 PRINT "ENTER FILENAME WITHOUT"
2520 PRINT "EXTENSION; 1-8 CHARS."
2530 INPUT FT$
2540 L1 = LEN(FT$)
2550 IF ((L1 < 1) OR (L1 > 8)) GOTO 3710  'ERROR
2560 L1 = 8 - L1
2570 FOR I = 1 TO L1
2580   FT$ = FT$ + " "
2590 NEXT I
2600 '

2800 PRINT
2810 PRINT "ENTER EXTENSION;"
2820 PRINT "1-3 CHARS."
2830 INPUT EX$
2840 L1 = LEN(EX$)
2850 IF ((L1 < 1) OR (L1 > 3)) GOTO 3710  'ERROR
2860 L1 = 3 - L
2870 FOR I = 1 TO L1
2880   EX$ = EX$ + " "
2890 NEXT I
2900 '

3100 'GENERATE THE FALSE DIRECTORY
3110 D$ = Z$
3120 'SET FALSE FILENAME
3130 FOR I = 1 TO 8
3140   MID$(D$,I,1) = MID$(FT$,I,1)
3150 NEXT I
```

```
3160 'SET FALSE EXTENSION
3170 FOR I = 9 TO 11
3180   MID$(D$,I,1) = MID$(EX$,I-8,1)
3190 NEXT I
3200 'SET TYPE =
3210 '  TEXT EDITOR SOURCE
3220 MID$(D$,12,1) = CHR$(3)
3230 'SET FORMAT = BINARY
3240 MID$(D$,13,1) = CHR$(0)
3250 'SET NUMBER OF THE
3260 '  FIRST GRANULE
3270 MID$(D$,14,1) = CHR$(0)
3280 'NUMBER OF BYTES USED
3290 '  IN LAST SECTOR = 256
3300 MID$(D$,15,1) = CHR$(1)
3310 MID$(D$,16,1) = CHR$(0)
3320 'PUT TRACK 17, SECTOR 3 TO DISK
3330 DSKO$ 0,17,3,D$,Z$
3340 '

3500 'CLEAN THE DIRECTORY TRACK
3510 DSKO$ 0,17,1,Z$,Z$
3520 DSKO$ 0,17,4,Z$,Z$
3530 DSKO$ 0,17,5,Z$,Z$
3540 DSKO$ 0,17,6,Z$,Z$
3550 DSKO$ 0,17,7,Z$,Z$
3560 DSKO$ 0,17,8,Z$,Z$
3570 DSKO$ 0,17,9,Z$,Z$
3580 DSKO$ 0,17,10,Z$,Z$
3590 DSKO$ 0,17,11,Z$,Z$
3600 DSKO$ 0,17,12,Z$,Z$
3610 DSKO$ 0,17,13,Z$,Z$
3620 DSKO$ 0,17,14,Z$,Z$
3630 DSKO$ 0,17,15,Z$,Z$
3640 DSKO$ 0,17,16,Z$,Z$
3650 DSKO$ 0,17,17,Z$,Z$
3660 DSKO$ 0,17,18,Z$,Z$
3670 '

3800 PRINT
3810 PRINT "FALSE CHANGE DISK NAME = DONE"
3820 GOTO 32767
3830 '

4000 'ERROR HANDLING
4010 PRINT
4020 PRINT "*** ERROR ENCOUNTERED ***"
```

```
4030 PRINT "***  RUN  TERMINATED  ***"
4040 PRINT
4050 '

32767 END
```

=====

FALSCOPY.BAS: Program to Copy a Source Disk To a Target Disk

```
1000 '*****
1010 '*
1020 '* FALSCOPY.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* COPIES A SOURCE DISK
1060 '* TO A TARGET DISK
1070 '*
1080 '* ANY DATA ON THE
1090 '* TARGET DISK WILL
1100 '* BE LOST
1110 '*
1120 '* DOES A
1130 '* SECTOR-BY-SECTOR COPY
1140 '*
1150 '* BOTH DRIVE 0 AND
1160 '* DRIVE 1 ARE REQUIRED
1170 '* BY THIS PROGRAM - USE
1180 '* THE BASIC COPY
1190 '* COMMAND INSTEAD IF
1200 '* ONLY DRIVE 0 IS
1210 '* AVAILABLE
1220 '*
1230 '* NOTE: THIS PROGRAM
1240 '* WILL WORK WITH
1250 '* REGULAR DISKS AS
1260 '* WELL AS FALSE DISKS.
1270 '*
1280 '*****
1290 '

1300 CLEAR &H1000
1310 '

1400 PRINT
1410 PRINT "   PUT SOURCE DISK IN DRIVE 0"
1420 PRINT "   AND TARGET DISK IN DRIVE 1"
1430 PRINT "   ***  *** WARNING  ***  ***"
1440 PRINT "   ** ALL TARGET DISK DATA **"
1450 PRINT "   ***    WILL BE LOST    ***"
```

```
1460 PRINT "    PRESS ANY KEY WHEN READY"
1470 A$ = INKEY$
1480 IF A$ = "" GOTO 1470
1490 PRINT
1500 '

1600 PRINT "WORKING ";
1620 FOR T = 0 TO 34
1630   T3 = INT(T/10)
1640   T4 = T - (T3 * 10)
1650   T1$ = CHR$(T3 + &H30)
1660   T2$ = CHR$(T4 + &H30)
1670   FOR S = 1 TO 18
1680     S3 = INT(S/10)
1690     S4 = S - (S3 * 10)
1700     S1$ = CHR$(S3 + &H30)
1710     S2$ = CHR$(S4 + &H30)
1720     DSKI$ 0,T,S,X$,Y$
1730     DSKO$ 1,T,S,X$,Y$
1740   NEXT S
1750   PRINT "*";
1760 NEXT T
1770 PRINT:PRINT
1780 '

2000 PRINT "FALSE DISK COPY = DONE"
2010 '

32767 END
```

=====

FALSFILL.BAS: Program to Fill a False Disk With Test Data

```
1000 '*****
1010 '*
1020 '* FALSFILL.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* FILLS A FALSE DISK
1060 '* WITH TEST DATA
1070 '*
1080 '*****
1090 '

1400 CLEAR &H1000

1500 PRINT
1510 PRINT "PUT A FALSE DISK IN DRIVE 0"
1520 PRINT "PRESS ANY KEY WHEN READY"
1530 A$ = INKEY$
1540 IF A$ = "" GOTO 1530
1550 PRINT "WORKING ";
1570 W3$ = "TRK 00 SEC 01 - "
1580 '

2010 ' GENERATE THE TEST DATA
2020 FOR T = 0 TO 34
2030   'SKIP TRACK 17
2040   IF T = 17 GOTO 2220
2050   T3 = INT(T/10)
2060   T4 = T - (T3 * 10)
2070   T1$ = CHR$(T3 + &H30)
2080   T2$ = CHR$(T4 + &H30)
2090   FOR S = 1 TO 18
2100     S3 = INT(S/10)
2110     S4 = S - (S3 * 10)
2120     S1$ = CHR$(S3 + &H30)
2130     S2$ = CHR$(S4 + &H30)
2140     MID$(W3$,5,1) = T1$
2150     MID$(W3$,6,1) = T2$
2160     MID$(W3$,12,1) = S1$
2170     MID$(W3$,13,1) = S2$
2180     S1$ = W3$+W3$+W3$+W3$+W3$+W3$+W3$+W3$
```

```
2190     DSKO$ 0,T,S,S1$,S1$
2200   NEXT S
2210   PRINT "*";
2220 NEXT T
2230 '

2500 'CLEAN THE DIRECTORY TRACK
2510 X$ = "                                    "
2520 Z$ = X$+X$+X$+X$
2530 FOR I = 1 TO 128
2540   MID$(Z$,I,1) = CHR$(0)
2550 NEXT I
2560 DSKO$ 0,17,1,Z$,Z$
2570 DSKO$ 0,17,4,Z$,Z$
2580 DSKO$ 0,17,5,Z$,Z$
2590 DSKO$ 0,17,6,Z$,Z$
2600 DSKO$ 0,17,7,Z$,Z$
2610 DSKO$ 0,17,8,Z$,Z$
2620 DSKO$ 0,17,9,Z$,Z$
2630 DSKO$ 0,17,10,Z$,Z$
2640 DSKO$ 0,17,11,Z$,Z$
2650 DSKO$ 0,17,12,Z$,Z$
2660 DSKO$ 0,17,13,Z$,Z$
2670 DSKO$ 0,17,14,Z$,Z$
2680 DSKO$ 0,17,15,Z$,Z$
2690 DSKO$ 0,17,16,Z$,Z$
2700 DSKO$ 0,17,17,Z$,Z$
2710 DSKO$ 0,17,18,Z$,Z$
2720 '

2800 PRINT
2810 PRINT "FALSE FILL = DONE"
2820 '

32767 END
```

=====

FALSDISP.BAS: Program to Display the Data Contained in a Selected Linear Sector

Note: this program utilizes the FALSXLTL.BAS subroutine - See the third page following this one.

```
1000 '*****
1010 '*
1020 '* FALSDISP.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* DISPLAYS THE DATA
1060 '* CONTAINED IN A
1070 '* SELECTED LINEAR
1080 '* SECTOR
1090 '*
1100 '*****
1120 '

1150 CLEAR &H1000

1200 PRINT
1210 PRINT "MAKE SURE FALSE DISK IS"
1220 PRINT "IN THE DESIRED DRIVE."
1230 PRINT "PRESS ANY KEY WHEN READY"
1240 A$ = INKEY$
1250 IF A$ = "" GOTO 1240
1260 PRINT
1270 '

1400 PRINT "ENTER THE DESIRED LINEAR"
1410 PRINT "SECTOR NUMBER (0 - 2447) ";
1420 INPUT L
1430 L = INT(L)
1440 IF ((L<0) OR (L>2447)) GOTO 2410  'ERROR
1450 PRINT
1460 '

1600 'GO TRANSLATE
1610 'LINEAR SECTOR NUMBER.
1620 'RETURNS D, T, S
1630 GOSUB 20000
1640 IF (D < 0) GOTO 2410  'ERROR
```

```
1650 '

1800 PRINT "CONTENTS:"
1810 DSKI$ D,T,S,X$,Y$
1820 PRINT X$;
1830 PRINT Y$
1840 PRINT
1850 '

2000 PRINT "CHECK ANOTHER SECTOR (Y/N) ";
2010 A$ = INKEY$
2020 IF A$ = "" GOTO 2010
2030 IF A$ = "Y" GOTO 1400
2040 PRINT
2050 '

2200 PRINT "FALSE DISPLAY = DONE"
2210 GOTO 32767
2220 '

2400 'ERROR HANDLING
2410 PRINT
2420 PRINT "*** ERROR ENCOUNTERED ***"
2430 PRINT "***  RUN  TERMINATED  ***"
2440 PRINT
2450 GOTO 32767
2460 '

20000 '*****
20010 '*
20020 '* FALSXLTL.BAS
20030 '* MDJ 2023/02/21
20040 '*
20050 '* FALSE DISK
20060 '* L TO D,T,S
20070 '* TRANSLATION SUBROUTINE
20080 '*
20090 '* ENTRY CONDITIONS:
20100 '* L (0-2447)
20110 '* = LINEAR SECTOR NUMBER
20120 '*
20130 '* EXIT CONDITIONS:
20140 '* D (0-3)
20150 '* = DRIVE NUMBER
20160 '* T (0-34)
20170 '* = TRACK NUMBER
20180 '* S (1-18)
```

```
20190 '* = SECTOR NUMBER
20200 '*
20210 '* INTERNAL TEMPORARY
20220 '* VARIABLES =
20230 '*   V1, V2
20240 '*
20250 '* ON EXIT:
20260 '* D < 0 ==> ERROR
20270 '*
20280 '*****
20290 '

20400 'ERROR TRAPPING
20410 V1 = INT(L)
20420 IF V1 < 0 GOTO 20810
20430 IF V1 > 2447 GOTO 20810
20440 '

20500 'GET DRIVE NUMBER
20510 'AND REMAINDER = V2
20520 D = INT(V1 / 612)
20530 V2 = V1 - (D * 612)
20540 '

20600 'GET TRACK NUMBER
20610 'AND SECTOR NUMBER
20620 T = INT(V2 / 18)
20630 S = V2 - (T * 18) + 1
20640 'SKIP TRACK 17
20650 IF (T < 17) GOTO 20670
20660 T = T + 1
20670 RETURN
20680 '

20800 ' ERROR HANDLING
20810 D = -9
20820 RETURN
20830 '

20900 '*****
20910 '*
20920 '* END OF SUBROUTINE
20930 '*
20940 '*****
20950 '

32767 END
```

FALSXLTL.BAS: Subroutine to Translate a Linear Sector Number Into a Standard Drive Number, Track Number, and Sector Number

This is not a stand-alone program. It is a subroutine which must be called from within a stand-alone or similar program. Here, we use the FLTLTEST.BAS program to test the routine.

———

The BASIC Language Subroutine:

```
20000 '*****
20010 '*
20020 '* FALSXLTL.BAS
20030 '* MDJ 2023/02/21
20040 '*
20050 '* FALSE DISK
20060 '* L TO D,T,S
20070 '* TRANSLATION SUBROUTINE
20080 '*
20090 '* ENTRY CONDITIONS:
20100 '* L (0-2447)
20110 '* = LINEAR SECTOR NUMBER
20120 '*
20130 '* EXIT CONDITIONS:
20140 '* D (0-3)
20150 '* = DRIVE NUMBER
20160 '* T (0-34)
20170 '* = TRACK NUMBER
20180 '* S (1-18)
20190 '* = SECTOR NUMBER
20200 '*
20210 '* INTERNAL TEMPORARY
20220 '* VARIABLES =
20230 '*   V1, V2
20240 '*
20250 '* ON EXIT:
20260 '* D < 0 ==> ERROR
20270 '*
```

```
20280 '*****
20290 '

20400 'ERROR TRAPPING
20410 V1 = INT(L)
20420 IF V1 < 0 GOTO 20810
20430 IF V1 > 2447 GOTO 20810
20440 '

20500 'GET DRIVE NUMBER
20510 'AND REMAINDER = V2
20520 D = INT(V1 / 612)
20530 V2 = V1 - (D * 612)
20540 '

20600 'GET TRACK NUMBER
20610 'AND SECTOR NUMBER
20620 T = INT(V2 / 18)
20630 S = V2 - (T * 18) + 1
20640 'SKIP TRACK 17
20650 IF (T < 17) GOTO 20670
20660 T = T + 1
20670 RETURN
20680 '

20800 ' ERROR HANDLING
20810 D = -9
20820 RETURN
20830 '

20900 '*****
20910 '*
20920 '* END OF SUBROUTINE
20930 '*
20940 '*****
20950 '
```

The BASIC Language Test Program:

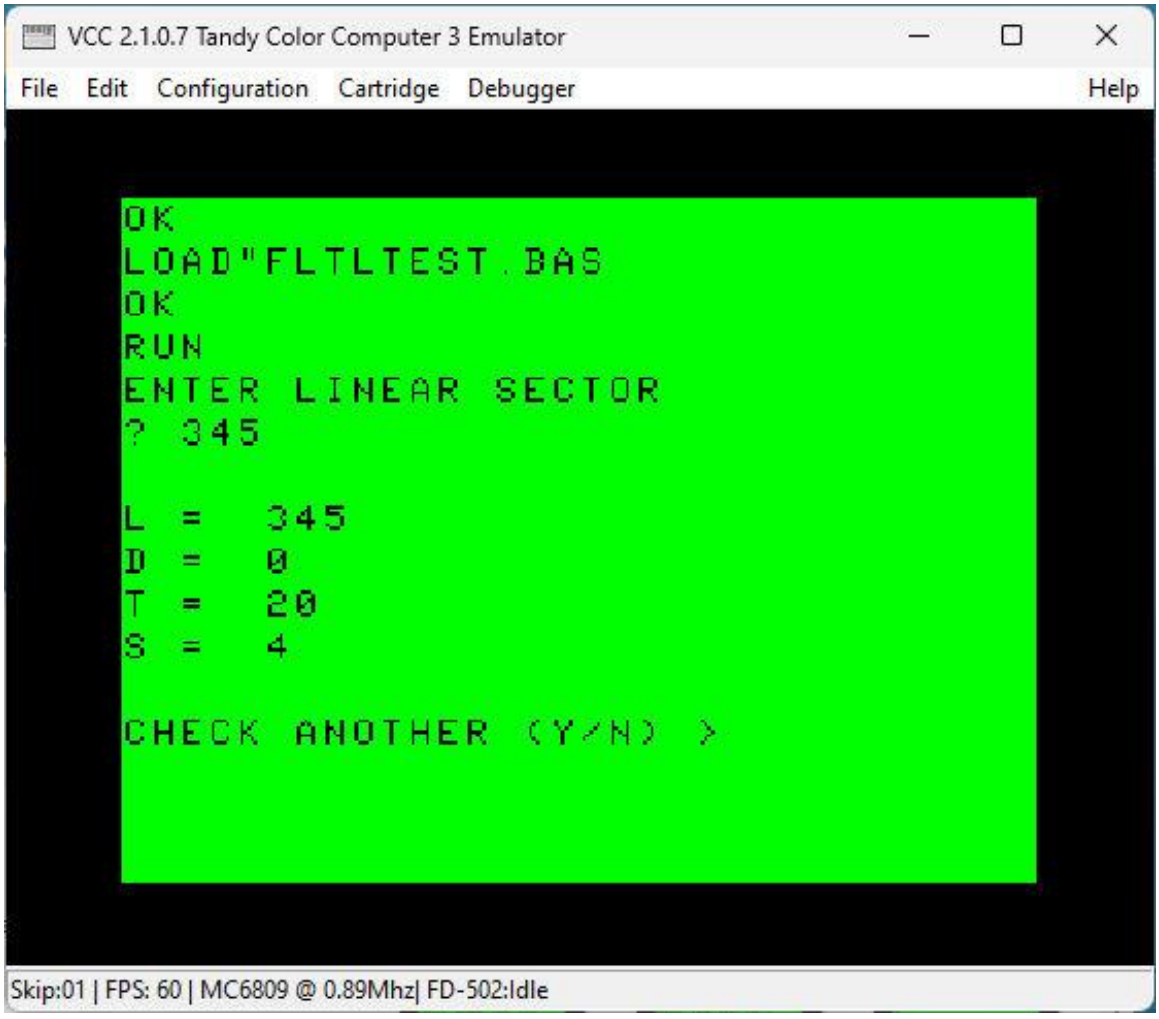
```
1000 '*****
1020 '*
1030 '* FLTLTEST.BAS
1040 '* MDJ 2023/02/22
1050 '*
```

```
1060 '* TESTS THE
1070 '* FALSXLTL
1080 '* SUBROUTINE
1090 '*
1100 '*****
1110 '

1400 CLEAR &H1000

2000 PRINT "ENTER LINEAR SECTOR"
2020 PRINT "NUMBER (0-2447) >";
2010 INPUT L
2020 L = INT(L)
2030 IF ((L >= 0) AND (L <= 2447)) GOTO 2200
2040 PRINT "OUT OF RANGE - TRY AGAIN:"
2050 GOTO 2000
2060 `

2600 ' GO TRANSLATE L TO D,T,S
2610 GOSUB 20410
2620 IF (D < 0) GOTO 3410  'ERROR
2630 PRINT
2640 PRINT "L = "; L
2650 PRINT "D = "; D
2660 PRINT "T = "; T
2670 PRINT "S = "; S
2680 PRINT
2690 '

2800 PRINT "CHECK ANOTHER (Y/N) >";
2810 A$ = INKEY$
2820 IF A$ = "" GOTO 2810
2830 IF A$ = "Y" GOTO 2000
2840 '

3200 PRINT "FLTLTEST = DONE"
3210 GOTO 32767
3220 '

3400 'ERROR HANDLING
3410 PRINT "*** ERROR ENCOUNTERED ***"
3420 PRINT "***  RUN  TERMINATED  ***"
3430 PRINT
3440 GOTO 32767
3450 '

20000 '*****
```

```
20010 '*
20020 '* FALSXLTL.BAS
20030 '* MDJ 2023/02/21
20040 '*
20050 '* FALSE DISK
20060 '* L TO D,T,S
20070 '* TRANSLATION SUBROUTINE
20080 '*
20090 '* ENTRY CONDITIONS:
20100 '* L (0-2447)
20110 '* = LINEAR SECTOR NUMBER
20120 '*
20130 '* EXIT CONDITIONS:
20140 '* D (0-3)
20150 '* = DRIVE NUMBER
20160 '* T (0-34)
20170 '* = TRACK NUMBER
20180 '* S (1-18)
20190 '* = SECTOR NUMBER
20200 '*
20210 '* INTERNAL TEMPORARY
20220 '* VARIABLES =
20230 '*   V1, V2
20240 '*
20250 '* ON EXIT:
20260 '* D < 0 ==> ERROR
20270 '*
20280 '*****
20290 '

20400 'ERROR TRAPPING
20410 V1 = INT(L)
20420 IF V1 < 0 GOTO 20810
20430 IF V1 > 2447 GOTO 20810
20440 '

20500 'GET DRIVE NUMBER
20510 'AND REMAINDER = V2
20520 D = INT(V1 / 612)
20530 V2 = V1 - (D * 612)
20540 '

20600 ' ADJUST FOR SKIPPING
20610 ' OF FAT SECTOR AND
20620 ' DIRECTORY SECTORS
20630 IF V2 < 307 GOTO 20710
20640 V2 = V2 + 18
```

```
20650 '

20700 ' GET TRACK & SECTOR
20710 T = INT(V2/18)
20720 S = V2 - (T*18) + 1
20730 RETURN
20740 '

20800 ' ERROR HANDLING
20810 D = -9
20820 RETURN
20630 '

20900 '*****
20910 '*
20920 '* END OF SUBROUTINE
20930 '*
20940 '*****
20950 '

32767 END
```

Result:



As expected.

=====

FALSXLTD.BAS: Subroutine to Translate a Standard Drive Number, Track Number, and Sector Number Into a Linear Sector Number

This is not a stand-alone program. It is a subroutine which must be called from within a stand-alone or similar program. Here, we use the FLTDTEST.BAS program to test the routine.

———

The BASIC Language Subroutine:

```
21000 '*****
21010 '*
21020 '* FALSXLTD.BAS
21030 '* MDJ 2023/02/21
21040 '*
21050 '* FALSE DISK
21060 '* D,T,S TO L
21070 '* TRANSLATION SUBROUTINE
21080 '*
21090 '* ENTRY CONDITIONS:
21100 '* D (0-3)
21110 '* = DRIVE NUMBER
21120 '* T (0-34)
21130 '* = TRACK NUMBER
21140 '* S (1-18)
21150 '* = SECTOR NUMBER
21160 '*
21170 '* EXIT CONDITIONS:
21180 '* L (0-2447)
21190 '* = LINEAR SECTOR NUMBER
21200 '*
21210 '* INTERNAL TEMPORARY
21220 '* VARIABLES =
21230 '*   V3, V4, V5
21240 '*
21250 '* ON EXIT:
21260 '* L < 0 ==> ERROR
21270 '*
```

```
21280 '*****
21290 '

21400 'ERROR TRAPPING
21410 V3 = INT(D)
21420 IF V3 < 0 GOTO 22410
21430 IF V3 > 3 GOTO 22410
21440 V4 = INT(T)
21450 IF V4 < 0 GOTO 22410
21460 IF V4 > 34 GOTO 22410
21470 IF V4 = 17 GOTO 22410
21480 V5 = INT(S)
21490 IF V5 < 1 GOTO 22410
21500 IF V5 > 18 GOTO 22410
21510 '

21800 ' GO DO SPECIAL HANDLING
21810 ' IF TRACK > 17
21820 IF V4 > 17 GOTO 22240
21830 '
21840 ' REGULAR HANDLING
21850 ' 18 SECTORS PER TRACK
21860 L = (V3 * 620) + (V4 * 18) + V5
21870 RETURN
21880 '

22200 ' SPECIAL HANDLING
22210 ' FOR TRACKS > 17
22220 ' ADJUST FOR SKIPPING
22230 ' FAT AND DIRECTORY SECTORS
22240 L = (V3 * 620) + (V4 * 18) + V5 - 18
22250 RETURN
22260 '

22400 ' ERROR HANDLING
22410 L = -9
22420 RETURN
22430 '

22600 '*****
22610 '*
22620 '* END OF SUBROUTINE
22630 '*
22640 '*****
22650 '
```

The BASIC Language Test Program:

```
1000 '*****
1020 '*
1030 '* FLTDTEST.BAS
1040 '* MDJ 2023/02/22
1050 '*
1060 '* TESTS THE
1070 '* FALSXLTD
1080 '* SUBROUTINE
1090 '*
1100 '*****
1110 '

1400 CLEAR &H1000

2000 PRINT "ENTER DRIVE NUMBER (0-3) >";
2010 INPUT D
2020 D = INT(D)
2030 IF ((D >= 0) AND (D <= 3)) GOTO 2200
2040 PRINT "OUT OF RANGE - TRY AGAIN:"
2050 GOTO 2000
2060 '

2200 PRINT "ENTER TRACK NUMBER (0-35) >";
2210 INPUT T
2220 T = INT(T)
2230 IF T = 17 GOTO 2250
2240 IF ((T >= 0) AND (T <= 34)) GOTO 2400
2250 PRINT "OUT OF RANGE - TRY AGAIN:"
2260 GOTO 2200
2270 '

2400 PRINT "ENTER SECTOR NUMBER (1-18) >";
2410 INPUT S
2420 S = INT(S)
2430 IF ((S >= 1) AND (S <= 18)) GOTO 2610
2440 PRINT "OUT OF RANGE - TRY AGAIN:"
2450 GOTO 2400
2460 '

2600 ' GO TRANSLATE D,T,S TO L
2610 GOSUB 21410
2620 IF (L < 0) GOTO 3410  'ERROR
2630 PRINT
```

```
2640 PRINT "D = "; D
2650 PRINT "T = "; T
2660 PRINT "S = "; S
2670 PRINT "L = "; L
2680 PRINT
2690 '

2800 PRINT "CHECK ANOTHER (Y/N) >";
2810 A$ = INKEY$
2820 IF A$ = "" GOTO 2810
2830 IF A$ = "Y" GOTO 2000
2840 '

3200 PRINT "FLTDTEST = DONE"
3210 GOTO 32767
3220 '

3400 'ERROR HANDLING
3410 PRINT "*** ERROR ENCOUNTERED ***"
3420 PRINT "***  RUN  TERMINATED  ***"
3430 PRINT
3440 GOTO 32767
3450 '

21000 '*****
21010 '*
21020 '* FALSXLTD.BAS
21030 '* MDJ 2023/02/21
21040 '*
21050 '* FALSE DISK
21060 '* D,T,S TO L
21070 '* TRANSLATION SUBROUTINE
21080 '*
21090 '* ENTRY CONDITIONS:
21100 '* D (0-3)
21110 '* = DRIVE NUMBER
21120 '* T (0-34)
21130 '* = TRACK NUMBER
21140 '* S (1-18)
21150 '* = SECTOR NUMBER
21160 '*
21170 '* EXIT CONDITIONS:
21180 '* L (0-2447)
21190 '* = LINEAR SECTOR NUMBER
21200 '*
21210 '* INTERNAL TEMPORARY
21220 '* VARIABLES =
```

```
21230 '*   V3, V4, V5
21240 '*
21250 '* ON EXIT:
21260 '* L < 0 ==> ERROR
21270 '*
21280 '*****
21290 '

21400 'ERROR TRAPPING
21410 V3 = INT(D)
21420 IF V3 < 0 GOTO 22410
21430 IF V3 > 3 GOTO 22410
21440 V4 = INT(T)
21450 IF V4 < 0 GOTO 22410
21460 IF V4 > 34 GOTO 22410
21470 IF V4 = 17 GOTO 22410
21480 V5 = INT(S)
21490 IF V5 < 1 GOTO 22410
21500 IF V5 > 18 GOTO 22410
21510 '

21800 ' GO DO SPECIAL HANDLING
21810 ' IF TRACK > 17
21820 IF V4 > 17 GOTO 22240
21830 '
21840 ' REGULAR HANDLING
21850 ' 18 SECTORS PER TRACK
21860 L = (V3 * 620) + (V4 * 18) + V5
21870 RETURN
21880 '

22200 ' SPECIAL HANDLING
22210 ' FOR TRACKS > 17
22220 ' ADJUST FOR SKIPPING
22230 ' FAT AND DIRECTORY SECTORS
22240 L = (V3 * 620) + (V4 * 18) + V5 - 18
22250 RETURN
22260 '

22400 ' ERROR HANDLING
22410 L = -9
22420 RETURN
22430 '

22600 '*****
22610 '*
22620 '* END OF SUBROUTINE
```

```
22630 '*
22640 '*****
22650 '

32767 END
```

Result:

As expected.

=====

FALSPUT.BAS: Subroutine to Put a Buffer To a Linear Sector

This is not a stand-alone program. It is a subroutine which must be called from within a stand-alone or similar program. Here, we use the FALSPUTT.BAS program to test the routine.

———

The BASIC Language Subroutine:

```
10000 '*****
10010 '*
10020 '* FALSPUT.BAS
10030 '* MDJ 2023/02/21
10040 '*
10050 '* FALSE DISK SYSTEM
10060 '* PUT BUFFER TO
10070 '* LINEAR SECTOR
10080 '* SUBROUTINE
10090 '*
10100 '* PUTS A LINEAR SECTOR
10110 '* (0-2447)
10120 '* TO A FALSE DISK
10130 '* FROM A SPECIFIED
10140 '* DISK BUFFER
10150 '*
10160 '* ENTRY CONDITIONS:
10170 '* B = DISK BUFFER ADDRESS
10180 '* L = LINEAR SECTOR NUMBER
10190 '*
10200 '* INTERNALLY USED
10210 '* PERMANENT VARIABLES
10220 '* D = DRIVE NUMBER
10230 '* T = TRACK NUMBER
10240 '* S = SECTOR NUMBER
10250 '*
10260 '* INTERNALLY USED
10270 '* TEMPORARY VARIABLES:
10280 '* A1, A2, C1$. C2$,
10290 '* I, E, V6, V7,
10300 '* X2$, X3$, Z$
10310 '*
```

```
10320 '* ON EXIT:
10330 '* L < 0 ==> ERROR
10340 '*
10350 '*****
10360'

10400 'THERE IS NO INTERNAL
10410 'CHECK ON THE VALIDITY
10420 'OF THE BUFFER ADDRESS
10430 'HERE - PERFORM EXTERNALLY
10440 'BEFORE CALLING THIS
10450 'SUBROUTINE.
10460 V6 = INT(B)
10470 'ADDRESSES FOR
10480 'SETTING STRINGS
10490 V6 = V6 - 1
10500 V7 = V6 + &H80
10510 '

10600 'GO TRANSLATE
10610 'LINEAR SECTOR NUMBER.
10620 'RETURNS D, T, S
10630 GOSUB 20000
10640 IF (D < 0) GOTO 11100  'ERROR
10650 '

10700 ' GET DATA FROM THE BUFFER
10710 ' AND SET STRINGS
10720 ' (MUST PRELOAD STRINGS
10730 ' WITH SPACES)
10740 Z$ = "                                "
10750 X2$ = Z$+Z$+Z$+Z$
10760 X3$ = Z$+Z$+Z$+Z$
10770 FOR I = 1 TO 128
10780   A1 = V6 + I
10790   A2 = V7 + I
10800   C1$ = CHR$(PEEK(A1))
10810   C2$ = CHR$(PEEK(A2))
10820   MID$(X2$,I,1) = C1$
10830   MID$(X3$,I,1) = C2$
10840 NEXT I
10850 '

10900 ' PUT THE DATA TO DISK
10910 DSKO$ D,T,S,X2$,X3$
10920 '
```

```
11000 ' CHECK STATUS FOR WRITE ERROR
11010 E = PEEK(&H00F0)
11020 IF (E<>0) GOTO 11100  'ERROR
11030 RETURN
11040 '

11100 ' ERROR HANDLING
11110 ' L = -9
11120 RETURN
11130 '

11200 '*****
11210 '*
11220 '* END OF SUBROUTINE
11230 '*
11240 '*****
11250 '
```

The BASIC Language Test Program:

```
1000 '*****
1010 '*
1020 '* FALSPUTT.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* TESTS THE BASIC
1060 '* FALSPUT MECHANISM
1070 '*
1080 '*****
1090 '

1100 CLEAR &H1000
1100 '

1200 PRINT
1210 PRINT "MAKE SURE FALSE DISK IS"
1220 PRINT "IN THE DESIRED DRIVE."
1230 PRINT "PRESS ANY KEY WHEN READY"
1240 A$ = INKEY$
1250 IF A$ = "" GOTO 1240
1260 PRINT
1270 '

1400 PRINT "ENTER THE DESIRED LINEAR"
```

```
1410 PRINT "SECTOR NUMBER (0 - 2447) ";
1420 INPUT L
1430 L = INT(L)
1440 IF ((L<0) OR (L>2447)) GOTO 2610  'ERROR
1450 PRINT "WORKING ";
1460 'FAKE WORKING DELAYS TO SHOW ACTIVITY
1470 FOR K = 1 TO 500:NEXT K
1480 PRINT "*";
1490 FOR K = 1 TO 500:NEXT K
1500 PRINT "*";
1510 '

1600 'THIS TEST USES THE STANDARD
1610 'DISK BASIC BUFFER AT &H0600
1620 B = &H0600
1630 PRINT "*";
1640 '

1800 'GO FILL THE BUFFER WITH THE
1810 'DESIRED TEST DATA
1820 GOSUB 5060
1830 IF (L < 0) GOTO 2610  'ERROR
1840 PRINT "*";
1850 '

2000 'GO GET THE DATA FROM THE BUFFER
2010 'AND PUT IT TO THE DISK SECTOR
2020 GOSUB 10460
2030 IF (L < 0) GOTO 2610  'ERROR
2040 PRINT "*";
2050 '

2200 PRINT:PRINT "FILL ANOTHER SECTOR (Y/N) ";
2210 A$ = INKEY$
2220 IF A$ = "" GOTO 2210
2230 IF A$ = "Y" GOTO 1400
2240 PRINT
2250 '

2400 PRINT:PRINT "FALSPUTT TEST = DONE"
2410 GOTO 32767
2420 '

2600 'ERROR HANDLING
2610 PRINT
2620 PRINT "*** ERROR ENCOUNTERED ***"
2630 PRINT "***  RUN  TERMINATED  ***"
```

```
2640 PRINT
2650 GOTO 32767
2660 '

5000 '*****
5010 '*
5020 '* FILL BUFFER SUBROUTINE
5030 '*
5040 '*****
5050 '

5060 PRINT
5070 PRINT "USE STANDARD TEST DATA (Y/N)"
5080 PRINT ">";
5090 A$ = INKEY$
5100 IF A$ = "" GOTO 5090
5110 IF A$ = "Y" GOTO 6020
5120 '

5200 PRINT "ENTER UP TO 128 CHARACTERS"
5210 PRINT "FOR THE FIRST HALF OF THE"
5220 PRINT "TEST DATA, I.E. FOUR LINES"
5230 PRINT "ON THE SCREEN >"
5240 INPUT D1$
5250 L1 = LEN(D1$)
5260 IF (L1 > 128) GOTO 5320
5270 L1 = 128 - L1
5280 FOR I = 1 TO L1
5290   D1$ = D1$ + " "
5300 NEXT I
5310 GOTO 5600
5320 D1$ = MID$(D1$, 1, 128)
5330 '

5600 PRINT "ENTER UP TO 128 CHARACTERS"
5610 PRINT "FOR THE SECOND HALF OF THE"
5620 PRINT "TEST DATA, I.E. FOUR LINES"
5630 PRINT "ON THE SCREEN >"
5640 INPUT D2$
5650 L1 = LEN(D2$)
5660 IF (L1 > 128) GOTO 5720
5670 L1 = 128 - L1
5680 FOR I = 1 TO L1
5690   D2$ = D2$ + " "
5700 NEXT I
5710 GOTO 6410
5720 D2$ = MID$(D2$, 1, 128)
```

```
5730 GOTO 6410
5740 '

6000 'GENERATE STANDARD TEST DATA
6010 'GO TRANSLATE L TO D,T,S
6020 GOSUB 20420
6030 IF (D < 0) GOTO 6610  'ERROR
6040 W3$ = "TRK 00 SEC 01 - "
6050 T3 = INT(T/10)
6060 T4 = T - (T3 * 10)
6070 T1$ = CHR$(T3 + &H30)
6080 T2$ = CHR$(T4 + &H30)
6090 S3 = INT(S/10)
6100 S4 = S - (S3 * 10)
6110 S1$ = CHR$(S3 + &H30)
6120 S2$ = CHR$(S4 + &H30)
6130 MID$(W3$,5,1) = T1$
6140 MID$(W3$,6,1) = T2$
6150 MID$(W3$,12,1) = S1$
6160 MID$(W3$,13,1) = S2$
6170 D1$ = W3$+W3$+W3$+W3$+W3$+W3$+W3$+W3$
6180 D2$ = W3$+W3$+W3$+W3$+W3$+W3$+W3$+W3$

6400 'FILL THE BUFFER WITH THE TEST DATA
6410 PRINT "WORKING ";
6420 FOR I = 1 TO 128
6430   A8 = &H05FF + I
6440   A9 = &H067F + I
6450   C8$ = MID$(D1$,I,1)
6460   C9$ = MID$(D2$,I,1)
6470   POKE A8, ASC(C8$)
6480   POKE A9, ASC(C9$)
6490   J = INT(I / 4)
6500   J = I - (J * 4)
6510   IF (J <> 0) GOTO 6530
6520   PRINT "*";
6530 NEXT I
6540 RETURN

6600 'ERROR HANDLING
6610 L = -9
6620 RETURN
6030 '

6200 '*****
6210 '*
6220 '* END OF SUBROUTINE
```

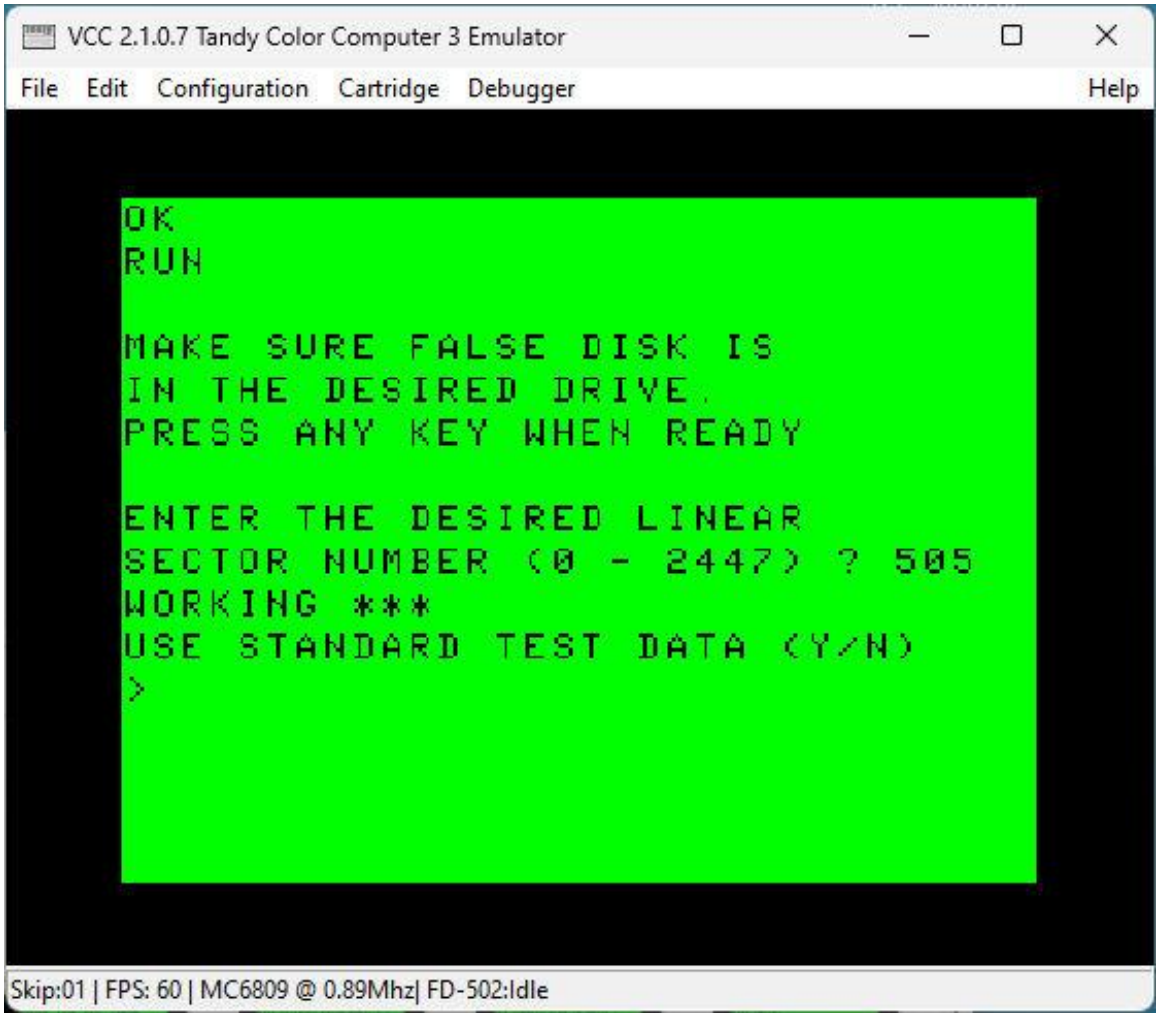
```
6230 '*
6240 '*****
6250 '

10000 '*****
10010 '*
10020 '* FALSPUT.BAS
10030 '* MDJ 2023/02/21
10040 '*
10050 '* FALSE DISK SYSTEM
10060 '* PUT BUFFER TO
10070 '* LINEAR SECTOR
10080 '* SUBROUTINE
10090 '*
10100 '* PUTS A LINEAR SECTOR
10110 '* (0-2511)
10120 '* TO A FALSE DISK
10130 '* FROM A SPECIFIED
10140 '* DISK BUFFER
10150 '*
10160 '* ENTRY CONDITIONS:
10170 '* B = DISK BUFFER ADDRESS
10180 '* L = LINEAR SECTOR NUMBER
10190 '*
10200 '* INTERNALLY USED
10210 '* PERMANENT VARIABLES
10220 '* D = DRIVE NUMBER
10230 '* T = TRACK NUMBER
10240 '* S = SECTOR NUMBER
10250 '*
10260 '* INTERNALLY USED
10270 '* TEMPORARY VARIABLES:
10280 '* A1, A2, C1$. C2$,
10290 '* I, E, V6, V7,
10300 '* X2$, X3$, Z$
10310 '*
10320 '* ON EXIT:
10330 '* L < 0 ==> ERROR
10340 '*
10350 '*****
10360'

10400 'THERE IS NO INTERNAL
10410 'CHECK ON THE VALIDITY
10420 'OF THE BUFFER ADDRESS
10430 'HERE - PERFORM EXTERNALLY
10440 'BEFORE CALLING THIS
```

```
10450 'SUBROUTINE.
10460 V6 = INT(B)
10470 'ADDRESSES FOR
10480 'SETTING STRINGS
10490 V6 = V6 - 1
10500 V7 = V6 + &H80
10510 '

10600 'GO TRANSLATE
10610 'LINEAR SECTOR NUMBER.
10620 'RETURNS D, T, S
10630 GOSUB 20000
10640 IF (D < 0) GOTO 11100  'ERROR
10650 '

10700 ' GET DATA FROM THE BUFFER
10710 ' AND SET STRINGS
10720 ' (MUST PRELOAD STRINGS
10730 ' WITH SPACES)
10740 Z$ = "                                "
10750 X2$ = Z$+Z$+Z$+Z$
10760 X3$ = Z$+Z$+Z$+Z$
10770 FOR I = 1 TO 128
10780   A1 = V6 + I
10790   A2 = V7 + I
10800   C1$ = CHR$(PEEK(A1))
10810   C2$ = CHR$(PEEK(A2))
10820   MID$(X2$,I,1) = C1$
10830   MID$(X3$,I,1) = C2$
10840 NEXT I
10850 '

10900 ' PUT THE DATA TO DISK
10910 DSKO$ D,T,S,X2$,X3$
10920 '

11000 ' CHECK STATUS FOR WRITE ERROR
11010 E = PEEK(&H00F0)
11020 IF (E<>0) GOTO 11100  'ERROR
11030 RETURN
11040 '

11100 ' ERROR HANDLING
11110 ' L = -9
11120 RETURN
11130 '
```

```
11200 '*****
11210 '*
11220 '* END OF SUBROUTINE
11230 '*
11240 '*****
11250 '

20000 '*****
20010 '*
20020 '* FALSXLTL.BAS
20030 '* MDJ 2023/02/21
20040 '*
20050 '* FALSE DISK
20060 '* L TO D,T,S
20070 '* TRANSLATION SUBROUTINE
20080 '*
20090 '* ENTRY CONDITIONS:
20100 '* L (0-2447)
20110 '* = LINEAR SECTOR NUMBER
20120 '*
20130 '* EXIT CONDITIONS:
20140 '* D (0-3)
20150 '* = DRIVE NUMBER
20160 '* T (0-34)
20170 '* = TRACK NUMBER
20180 '* S (1-18)
20190 '* = SECTOR NUMBER
20200 '*
20210 '* INTERNAL TEMPORARY
20220 '* VARIABLES =
20230 '*   V1, V2
20240 '*
20250 '* ON EXIT:
20260 '* D < 0 ==> ERROR
20270 '*
20280 '*****
20290 '

20400 'ERROR TRAPPING
20410 V1 = INT(L)
20420 IF V1 < 0 GOTO 20810
20430 IF V1 > 2447 GOTO 20810
20440 '

20500 'GET DRIVE NUMBER
20510 'AND REMAINDER = V2
20520 D = INT(V1 / 612)
```

```
20530 V2 = V1 - (D * 612)
20540 '

20600 'GET TRACK NUMBER
20610 'AND SECTOR NUMBER
20620 T = INT(V2 / 18)
20630 S = V2 - (T * 18) + 1
20640 'SKIP TRACK 17
20650 IF (T < 17) GOTO 20670
20660 T = T + 1
20670 RETURN
20680 '

20800 ' ERROR HANDLING
20810 D = -9
20820 RETURN
20830 '

20900 '*****
20910 '*
20920 '* END OF SUBROUTINE
20930 '*
20940 '*****
20950 '

32767 END
```

Result at first question:



Result at test completion:

As expected.

=====

FALSGET.BAS: Subroutine to Get a Buffer From a Linear Sector

This is not a stand-alone program. It is a subroutine which must be called from within a stand-alone or similar program. Here, we use the FALSGETT.BAS program to test the routine.

———

The BASIC Language Subroutine:

```
12000 '*****
12010 '*
12020 '* FALSGET.BAS
12030 '* MDJ 2023/02/22
12040 '*
12050 '* FALSE DISK SYSTEM
12060 '* GET BUFFER FROM
12070 '* LINEAR SECTOR
12080 '* SUBROUTINE
12090 '*
12100 '* GETS A LINEAR SECTOR
12110 '* (0-2447)
12120 '* FROM A FALSE DISK
12130 '* TO A SPECIFIED
12140 '* DISK BUFFER
12150 '*
12160 '* ENTRY CONDITIONS:
12170 '* L = LINEAR SECTOR NUMBER
12180 '* B = DISK BUFFER ADDRESS
12190 '*
12200 '* INTERNALLY USED
12210 '* PERMANENT VARIABLES
12220 '* D = DRIVE NUMBER
12230 '* T = TRACK NUMBER
12240 '* S = SECTOR NUMBER
12250 '*
12260 '* INTERNALLY USED
12270 '* TEMPORARY VARIABLES:
12280 '* A1, A2, C1$. C2$,
12290 '* I, E, V6, V7,
12300 '* X1$, X2$
```

```
12310 '*
12320 '* ON EXIT:
12330 '* L < 0 ==> ERROR
12340 '*
12350 '*****
12360'

12400 'THERE IS NO INTERNAL
12410 'CHECK ON THE VALIDITY
12420 'OF THE BUFFER ADDRESS
12430 'HERE - PERFORM EXTERNALLY
12440 'BEFORE CALLING THIS
12450 'SUBROUTINE.
12460 V6 = INT(B)
12470 'ADDRESSES FOR
12480 'SETTING STRINGS
12490 V6 = V6 - 1
12500 V7 = V6 + &H80
12510 '

12600 'GO TRANSLATE
12610 'LINEAR SECTOR NUMBER.
12620 'RETURNS D, T, S
12630 GOSUB 20000
12640 IF (D < 0) GOTO 13100  'ERROR
12650 '

12700 ' GET THE DATA FROM DISK
12710 DSKI$ D,T,S,X1$,X2$
12720 '

12800 ' CHECK STATUS FOR READ ERROR
12810 E = PEEK(&H00F0)
12820 IF (E<>0) GOTO 13100  'ERROR
12840 '

12900 ' GET DATA FROM THE STRINGS
12910 ' AND PUT TO BUFFER
12920 FOR I = 1 TO 128
12930   A1 = V6 + I
12940   A2 = V7 + I
12950   C1$ = MID$(X1$,I,1)
12960   C2$ = MID$(X2$,I,1)
12970   POKE A1, ASC(C1$)
12980   POKE A2, ASC(C2$)
12990 NEXT I
13000 RETURN
```

```
13010 '

13100 ' ERROR HANDLING
13110 ' L = -9
13120 RETURN
13130 '

13200 '*****
13210 '*
13220 '* END OF SUBROUTINE
13230 '*
13240 '*****
13250 '
```

The BASIC Language Test Program:

```
1000 '*****
1010 '*
1020 '* FALSGETT.BAS
1030 '* MDJ 2023/02/22
1040 '*
1050 '* TESTS THE BASIC
1060 '* FALSGET MECHANISM
1070 '*
1080 '*****
1090 '

1100 CLEAR &H1000
1110 '

1200 PRINT
1210 PRINT "MAKE SURE FALSE DISK IS"
1220 PRINT "IN THE DESIRED DRIVE."
1230 PRINT "PRESS ANY KEY WHEN READY"
1240 A$ = INKEY$
1250 IF A$ = "" GOTO 1240
1260 PRINT
1270 '

1400 PRINT "ENTER THE DESIRED LINEAR"
1410 PRINT "SECTOR NUMBER (0 - 2447) ";
1420 INPUT L
1430 L = INT(L)
1440 IF ((L<0) OR (L>2447)) GOTO 3010  'ERROR
```

```
1450 PRINT "WORKING ";
1460 'FAKE WORKING DELAYS TO SHOW ACTIVITY
1470 FOR K = 1 TO 500:NEXT K
1480 PRINT "*";
1490 FOR K = 1 TO 500:NEXT K
1500 PRINT "*";
1510 '

1600 'THIS TEST USES THE STANDARD
1610 'DISK BASIC BUFFER AT &H0600
1620 B = &H0600
1630 PRINT "*";
1630 '

2000 'GO GET THE DATA FROM THE DISK
2010 'AND PUT IT IN THE BUFFER
2020 GOSUB 12460
2030 IF (L < 0) GOTO 3010  'ERROR
2040 PRINT "*";
2050 '

2200 'GO DISPLAY THE BUFFER CONTENTS
2210 X$ = "                                "
2220 D1$ = X$ + X$ + X$ + X$
2230 D2$ = X$ + X$ + X$ + X$
2240 FOR I = 1 TO 128
2250   A8 = &H05FF + I
2260   A9 = &H067F + I
2270   C1$ = CHR$(PEEK(A8))
2280   C2$ = CHR$(PEEK(A9))
2290   MID$(D1$,I,1) = C1$
2300   MID$(D2$,I,1) = C2$
2310   J = INT(I / 4)
2320   J = I - (J * 4)
2330   IF (J <> 0) GOTO 2350
2340   PRINT "*";
2350 NEXT I
2360 PRINT:PRINT
2370 PRINT D1$;
2380 PRINT D2$
2390 '

2600 PRINT "CHECK ANOTHER SECTOR (Y/N) ";
2610 A$ = INKEY$
2620 IF A$ = "" GOTO 2610
2630 IF A$ = "Y" GOTO 1400
2640 PRINT
```

```
2650 '

2800 PRINT:PRINT "FALSGETT TEST = DONE"
2810 GOTO 32767
2820 '

3000 'ERROR HANDLING
3010 PRINT
3020 PRINT "*** ERROR ENCOUNTERED ***"
3030 PRINT "***  RUN  TERMINATED  ***"
3040 PRINT
3050 GOTO 32767
3060 '

12000 '*****
12010 '*
12020 '* FALSGET.BAS
12030 '* MDJ 2023/02/22
12040 '*
12050 '* FALSE DISK SYSTEM
12060 '* GET BUFFER FROM
12070 '* LINEAR SECTOR
12080 '* SUBROUTINE
12090 '*
12100 '* GETS A LINEAR SECTOR
12110 '* (0-2511)
12120 '* FROM A FALSE DISK
12130 '* TO A SPECIFIED
12140 '* DISK BUFFER
12150 '*
12160 '* ENTRY CONDITIONS:
12170 '* L = LINEAR SECTOR NUMBER
12180 '* B = DISK BUFFER ADDRESS
12190 '*
12200 '* INTERNALLY USED
12210 '* PERMANENT VARIABLES
12220 '* D = DRIVE NUMBER
12230 '* T = TRACK NUMBER
12240 '* S = SECTOR NUMBER
12250 '*
12260 '* INTERNALLY USED
12270 '* TEMPORARY VARIABLES:
12280 '* A1, A2, C1$. C2$,
12290 '* I, E, V6, V7,
12300 '* X1$, X2$
12310 '*
12320 '* ON EXIT:
```

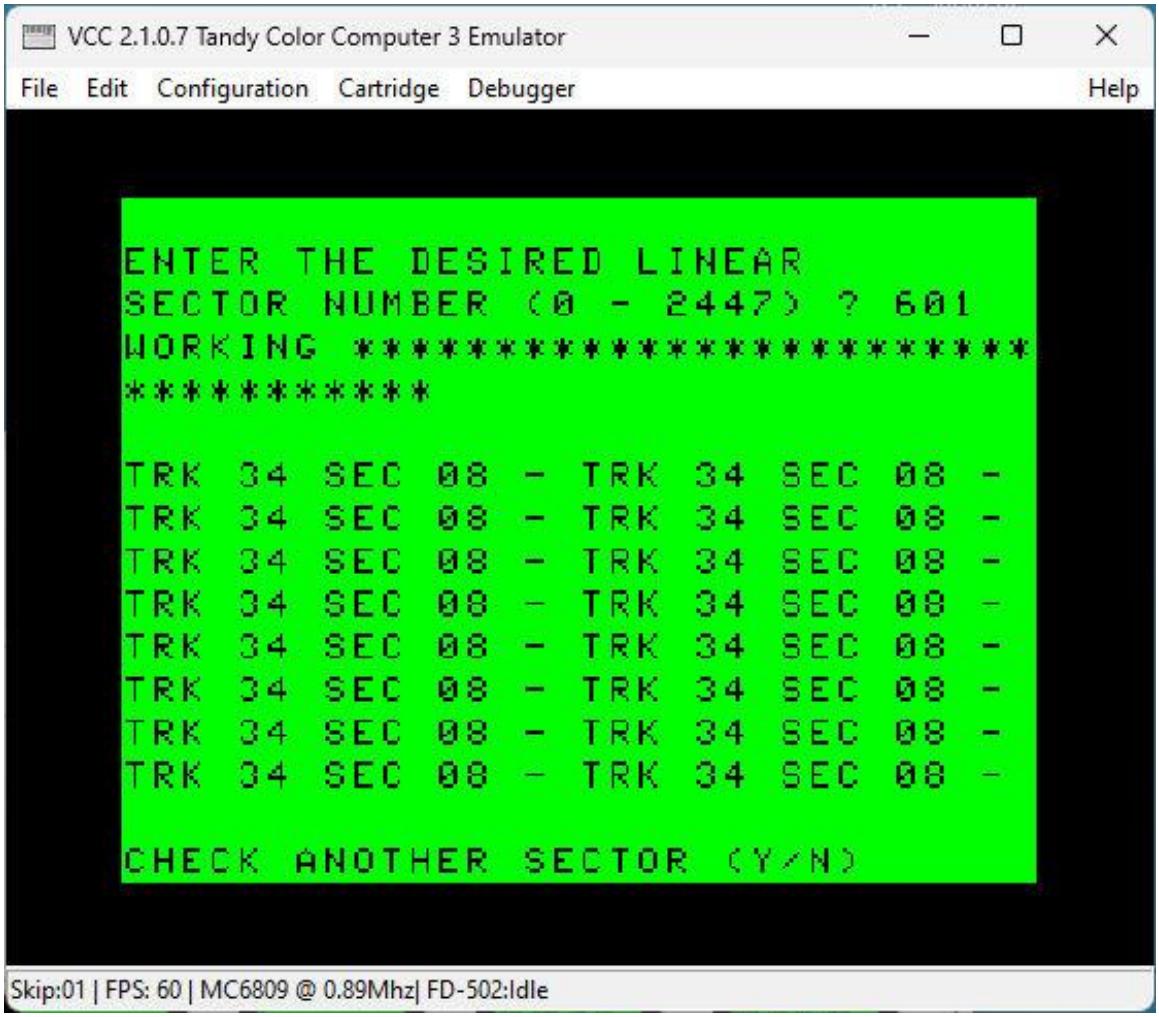
```
12330 '* L < 0 ==> ERROR
12340 '*
12350 '*****
12360'

12400 'THERE IS NO INTERNAL
12410 'CHECK ON THE VALIDITY
12420 'OF THE BUFFER ADDRESS
12430 'HERE - PERFORM EXTERNALLY
12440 'BEFORE CALLING THIS
12450 'SUBROUTINE.
12460 V6 = INT(B)
12470 'ADDRESSES FOR
12480 'SETTING STRINGS
12490 V6 = V6 - 1
12500 V7 = V6 + &H80
12510 '

12600 'GO TRANSLATE
12610 'LINEAR SECTOR NUMBER.
12620 'RETURNS D, T, S
12630 GOSUB 20000
12640 IF (D < 0) GOTO 13100  'ERROR
12650 '

12700 ' GET THE DATA FROM DISK
12710 DSKI$ D,T,S,X1$,X2$
12720 '

12800 ' CHECK STATUS FOR READ ERROR
12810 E = PEEK(&H00F0)
12820 IF (E<>0) GOTO 13100  'ERROR
12840 '

12900 ' GET DATA FROM THE STRINGS
12910 ' AND PUT TO BUFFER
12920 FOR I = 1 TO 128
12930   A1 = V6 + I
12940   A2 = V7 + I
12950   C1$ = MID$(X1$,I,1)
12960   C2$ = MID$(X2$,I,1)
12970   POKE A1, ASC(C1$)
12980   POKE A2, ASC(C2$)
12990 NEXT I
13000 RETURN
13010 '
```

```
13100 ' ERROR HANDLING
13110 ' L = -9
13120 RETURN
13130 '

13200 '*****
13210 '*
13220 '* END OF SUBROUTINE
13230 '*
13240 '*****
13250 '

20000 '*****
20010 '*
20020 '* FALSXLTL.BAS
20030 '* MDJ 2023/02/21
20040 '*
20050 '* FALSE DISK
20060 '* L TO D,T,S
20070 '* TRANSLATION SUBROUTINE
20080 '*
20090 '* ENTRY CONDITIONS:
20100 '* L (0-2447)
20110 '* = LINEAR SECTOR NUMBER
20120 '*
20130 '* EXIT CONDITIONS:
20140 '* D (0-3)
20150 '* = DRIVE NUMBER
20160 '* T (0-34)
20170 '* = TRACK NUMBER
20180 '* S (1-18)
20190 '* = SECTOR NUMBER
20200 '*
20210 '* INTERNAL TEMPORARY
20220 '* VARIABLES =
20230 '*   V1, V2
20240 '*
20250 '* ON EXIT:
20260 '* D < 0 ==> ERROR
20270 '*
20280 '*****
20290 '

20400 'ERROR TRAPPING
20410 V1 = INT(L)
20420 IF V1 < 0 GOTO 20810
20430 IF V1 > 2447 GOTO 20810
```

```
20440 '

20500 'GET DRIVE NUMBER
20510 'AND REMAINDER = V2
20520 D = INT(V1 / 612)
20530 V2 = V1 - (D * 612)
20540 '

20600 'GET TRACK NUMBER
20610 'AND SECTOR NUMBER
20620 T = INT(V2 / 18)
20630 S = V2 - (T * 18) + 1
20640 'SKIP TRACK 17
20650 IF (T < 17) GOTO 20670
20660 T = T + 1
20670 RETURN
20680 '

20800 ' ERROR HANDLING
20810 D = -9
20820 RETURN
20830 '

20900 '*****
20910 '*
20920 '* END OF SUBROUTINE
20930 '*
20940 '*****
20950 '

32767 END
```

Result at first question:

Result at test completion:



As expected.

=====

FLXLTL.ASM: Routine to Translate a Linear Sector Number Into a Standard Drive Number, Track Number, and Sector Number

```
                    00100 *****
                    00110 *
                    00120 * FLXLTL.ASM
                    00130 * MDJ 2023/02/28
                    00140 *
                    00150 * FALSE DISK
                    00160 * L TO D,T,S
                    00170 * TRANSLATION SUBROUTINE
                    00180 *
                    00190 * ENTRY CONDITIONS
                    00200 * REGISTER X = L (0-2447)
                    00210 * = LINEAR SECTOR NUMBER
                    00220 *
                    00230 * EXIT CONDITIONS
                    00240 * REGISTER X = D (0-3)
                    00250 * = DRIVE NUMBER
                    00260 * REGISTER A = T (0-34)
                    00270 * = TRACK NUMBER
                    00280 * REGISTER B = S (1-18)
                    00290 * = SECTOR NUMBER
                    00300 *
                    00310 * ON EXIT:
                    00320 * REGISTER X = $FFFF
                    00330 *           A = $FF
                    00340 *           B = $FF
                    00350 * ==> ERROR
                    00360 *
                    00370 *****
                    00380
                    00390 * GENERAL EQUATES
          0264      00400 DDIV    EQU     612     DRIVE
DIVISOR = $264
          0012      00410 TDIV    EQU     18      TRACK
DIVISOR = $12
                    00420
4416                00430         ORG     $4416
```

```
                    00440
                    00450 * ERROR TRAPPING
4416 8C   098F      00460 FLXLTL  CMPX    #2447   MAX LINEAR
SECTOR #
4419 22   68        00470         BHI     LBL008  ERROR IF
GREATER
                    00480
                    00490 * AT THIS POINT, THE STACK
CONTAINS:
                    00500 *    1,S RTS LOCATION LOW BYTE
                    00510 *     ,S RTS LOCATION HIGH BYTE
                    00520
                    00530 * MAKE SPACE ON THE STACK FOR:
                    00540 *    9,S RTS LOCATION LOW BYTE
                    00550 *    8,S RTS LOCATION HIGH BYTE
                    00560 *    7,S LITERAL SECTOR NUMBER LOW
BYTE
                    00570 *    6,S LITERAL SECTOR NUMBER
HIGH BYTE
                    00580 *    5,S REMAINDER LOW BYTE
                    00590 *    4,S REMAINDER HIGH BYTE
                    00600 *    3,S DRIVE NUMBER LOW BYTE
                    00610 *    2,S DRIVE NUMBER HIGH BYTE
(ALL ZEROES)
                    00620 *    1,S TRACK NUMBER
                    00630 *     ,S SECTOR NUMBER
441B 32   78        00640         LEAS    -8,S
                    00650
                    00660 * LOAD THE STACK
441D AF   66        00670         STX     6,S     LITERAL
SECTOR NUMBER
441F 4F             00680         CLRA
4420 A7   62        00690         STA     2,S     DRIVE #
HIGH BYTE = 0
                    00700
                    00710 * GET DRIVE NUMBER AND REMAINDER
                    00720 * BY EMULATING DIVISION VIA
                    00730 * REPEATED SUBTRACTION
                    00740 * REGD = LITERAL SECTOR NUMBER =
DIVIDEND
4422 EC   66        00750         LDD     6,S
4424 1083 0264      00760         CMPD    #DDIV   IS IT >=
THE DIVISOR?
4428 24   07        00770         BHS     LBL001  GO IF YES
442A ED   64        00780         STD     4,S     REMAINDER
442C 4F             00790         CLRA
```

```
442D A7   63       00800          STA     3,S     DRIVE
NUMBER = 0
442F 20   2B       00810          BRA     LBL004
4431 83   0264     00820 LBL001   SUBD    #DDIV   FIRST
DIVISION
4434 1083 0264     00830          CMPD    #DDIV   IS IT
>=THE DIVISOR?
4438 24   08       00840          BHS     LBL002  GO IF YES
443A ED   64       00850          STD     4,S     REMAINDER
443C 86   01       00860          LDA     #1
443E A7   63       00870          STA     3,S     DRIVE
NUMBER = QUOTIENT = 1
4440 20   1A       00880          BRA     LBL004
4442 83   0264     00890 LBL002   SUBD    #DDIV   SECOND
DIVISION
4445 1083 0264     00900          CMPD    #DDIV   IS IT >=
DIVISOR?
4449 24   08       00910          BHS     LBL003  GO IF YES
444B ED   64       00920          STD     4,S     REMAINDER
444D 86   02       00930          LDA     #2
444F A7   63       00940          STA     3,S     DRIVE
NUMBER = QUOTIENT = 2
4451 20   09       00950          BRA     LBL004
4453 83   0264     00960 LBL003   SUBD    #DDIV   THIRD
DIVISION
                   00970 *                        NOW NOT >=
DIVISOR
4456 ED   64       00980          STD     4,S     REMAINDER
4458 86   03       00990          LDA     #3
445A A7   63       01000          STA     3,S     DRIVE
NUMBER = QUOTIENT = 3
                   01010
                   01020 * GET TRACK NUMBER AND SECTOR
NUMBER
                   01030 * BY EMULATING DIVISION VIA
                   01040 * REPEATED SUBTRACTION
445C EC   64       01050 LBL004   LDD     4,S     REMAINDER
445E 8E   0000     01060          LDX     #0      COUNTER
4461 1083 0012     01070 LBL005   CMPD    #TDIV   IS
REMAINDER >= THE DIVISOR
4465 25   07       01080          BLO     LBL006  GO IF NO
4467 83   0012     01090          SUBD    #TDIV   DO A
"DIVIDE"
446A 30   01       01100          LEAX    1,X     INCREMENT
THE TRACK NUMBER
446C 20   F3       01110          BRA     LBL005  RETURN FOR
NEXT "DIVIDE"
```

```
446E 5C            01120 LBL006  INCB            ADJUST
SECTOR NUMBER
                   01130 *                         FOR 1-18
RANGE
446F E7   E4       01140         STB     ,S      SECTOR
NUMBER
4471 1F   10       01150         TFR     X,D
                   01160 * SKIP TRACK 17
4473 C1   11       01170         CMPB    #17     IS TRACK
NUMBER < 17
4475 25   01       01180         BLO     LBL007  GO IF YES
4477 5C            01190         INCB            INCREMENT
TRACK NUMBER
4478 E7   61       01200 LBL007  STB     1,S     TRACK
NUMBER
                   01210
                   01220 * GET THE EXIT CONDITIONS FROM THE
STACK
447A AE   62       01230         LDX     2,S     DRIVE
NUMBER
447C A6   61       01240         LDA     1,S     TRACK
NUMBER
447E E6   E4       01250         LDB     ,S      SECTOR
NUMBER
                   01260
                   01270 * CLEAN THE STACK AND RETURN
4480 32   68       01280         LEAS    8,S
4482 39            01290         RTS
                   01300
                   01310 * ERROR HANDLING
4483 8E   FFFF     01320 LBL008  LDX     #$FFFF  ERROR CODE
4486 CC   FFFF     01330         LDD     #$FFFF
4489 39            01340         RTS
                   01350
          0000     01360         END
```

The Assembly Language Test Routine:

```
00100 *****
00110 *
00120 * TEST0101.ASM
00130 * MDJ 2023/02/28
00140 *
00150 * FLXLTL TEST
```

```
                    00160 *
                    00170 *****
                    00180
                    00190 * LOW RAM CURSOR ADDRESS
          0088      00200 CURPOS  EQU     $0088
                    00210
                    00220 * SCREEN ADDRESSES
          0400      00230 VIDRAM  EQU     $0400
          0600      00240 VIDEND  EQU     $0600
                    00250
                    00260 * RAMROM TRIGGER ADDRESS
          FFDE      00270 RAMROM  EQU     $FFDE
                    00280
                    00290 * ALLRAM TRIGGER ADDRESS
          FFDF      00300 ALLRAM  EQU     $FFDF
                    00310
                    00320 * ML FOUNDATION
                    00330 * CORE ADDRESSES
          400E      00340 VIDCLS  EQU     $400E
          4025      00350 PUTBYT  EQU     $4025
          40D7      00360 CRLF    EQU     $40D7
          4169      00370 PUTWRD  EQU     $4169
          420A      00380 PRTS00  EQU     $420A
          43B6      00390 SNIRQS  EQU     $43B6
                    00400
                    00410 * FALSE DISK
                    00420 * SYSTEM ADDRESS
          4416      00430 FLXLTL  EQU     $4416
                    00440
7000                00450         ORG     $7000
7000 7E   7012      00460         JMP     LBL001
                    00470
7003      4C        00480 STRL    FCC     /L=/
          3D
7005      00        00490         FCB     $00     NUL
7006      44        00500 STRD    FCC     /D=/
          3D
7008      00        00510         FCB     $00     NUL
7009      54        00520 STRT    FCC     /T=/
          3D
700B      00        00530         FCB     $00     NUL
700C      53        00540 STRS    FCC     /S=/
          3D
700E      00        00550         FCB     $00     NUL
700F      2C        00560 STRSEP  FCC     /, /
          20
7011      00        00570         FCB     $00     NUL
```

```
                    00580
                    00590 * SETUP THE NEW INTERRUPT HANDLERS
                    00600 * AND ENTER ALLRAM MODE
7012 BD   43B6      00610 LBL001  JSR     SNIRQS
7015 B7   FFDF      00620         STA     ALLRAM
7018 34   07        00630         PSHS    A,B,CC
                    00640
                    00650 * AT THIS POINT, THE STACK
CONTAINS:
                    00660 *    4,S RTS LOCATION LOW BYTE
                    00670 *    3,S RTS LOCATION HIGH BYTE
                    00680 *    2,S REGISTER B
                    00690 *    1,S REGISTER A
                    00700 *     ,S REGISTER CC
                    00710
                    00720 * MAKE SPACE ON THE STACK FOR:
                    00730 *   10,S RTS LOCATION LOW BYTE
                    00740 *    9,S RTS LOCATION HIGH BYTE
                    00750 *    8,S REGISTER B
                    00760 *    7,S REGISTER A
                    00770 *    6,S REGISTER CC
                    00780 *    5,S L-VALUE LOW BYTE
                    00790 *    4,S L-VALUE HIGH BYTE
                    00800 *    3,S D-VALUE LOW BYTE
                    00810 *    2,S D-VALUE HIGH BYTE
                    00820 *    1,S T-VALUE
                    00830 *     ,S S-VALUE
701A 32   7A        00840         LEAS    -6,S
                    00850
                    00860 * SETUP THE SCREEN
701C BD   400E      00870         JSR     VIDCLS
701F 8E   0400      00880         LDX     #VIDRAM TOP OF
SCREEN
7022 9F   88        00890         STX     CURPOS  CURSOR
                    00900
                    00910 * TEST RUNS INPUT DATA
7024 8E   0000      00920         LDX     #0      = $0000
7027 BD   706B      00930         JSR     LBL002  FLXLTL
CONTROL
702A BD   7077      00940         JSR     LBL003  REPORTER
702D 8E   098F      00950         LDX     #2447   = $098F
7030 BD   706B      00960         JSR     LBL002  FLXLTL
CONTROL
7033 BD   7077      00970         JSR     LBL003  REPORTER
7036 8E   0990      00980         LDX     #2448   = $0990
7039 BD   706B      00990         JSR     LBL002  FLXLTL
CONTROL
```

```
703C BD   7077     01000          JSR     LBL003  REPORTER
703F 8E   0131     01010          LDX     #305    = $0131
7042 BD   706B     01020          JSR     LBL002  FLXLTL
CONTROL
7045 BD   7077     01030          JSR     LBL003  REPORTER
7048 8E   0132     01040          LDX     #306    = $0132
704B BD   706B     01050          JSR     LBL002  FLXLTL
CONTROL
704E BD   7077     01060          JSR     LBL003  REPORTER
7051 8E   0263     01070          LDX     #611    = $0264
7054 BD   706B     01080          JSR     LBL002  FLXLTL
CONTROL
7057 BD   7077     01090          JSR     LBL003  REPORTER
705A 8E   0264     01100          LDX     #612    = $0265
705D BD   706B     01110          JSR     LBL002  FLXLTL
CONTROL
7060 BD   7077     01120          JSR     LBL003  REPORTER
                   01130
                   01140 * RETURN TO RAMROM MODE,
                   01150 * CLEAN THE STACK, AND EXIT
7063 B7   FFDE     01160          STA     RAMROM
7066 32   66       01170          LEAS    6,S
7068 35   07       01180          PULS    A,B,CC
706A 39            01190          RTS
                   01200
                   01210 * FLXLTL CONTROL SUBROUTINE
                   01220 * AT THIS POINT, WE HAVE ADDED THE
TWO-BYTE
                   01230 * FLXLTL CONTROL SUBROUTINE RETURN
ADDRESS
                   01240 * TO THE STACK, AND THUS MUST ADD
2 TO ALL
                   01250 * L,D,T,S STACK ACCESSES
706B AF   66       01260 LBL002  STX     6,S     PUT L-
VALUE TO STACK
706D BD   4416     01270          JSR     FLXLTL  GO DO THE
TRANSLATION
7070 AF   64       01280          STX     4,S     PUT D-
VALUE TO STACK
7072 A7   63       01290          STA     3,S     PUT T-
VALUE TO STACK
7074 E7   62       01300          STB     2,S     PUT S-
VALUE TO STACK
7076 39            01310          RTS
                   01320
                   01330 * REPORTER SUBROUTINE
```

```
                    01340 * AT THIS POINT, WE HAVE ADDED THE
TWO-BYTE
                    01350 * REPORTER SUBROUTINE RETURN
ADDRESS
                    01360 * TO THE STACK, AND THUS MUST ADD
2 TO ALL
                    01370 * L,D,T,S STACK ACCESSES
                    01380
                    01390 * PRINT L-VALUE
7077 8E   7003      01400 LBL003  LDX     #STRL   /L=/
707A BD   420A      01410         JSR     PRTS00  PRINT /L=/
707D 9E   88        01420         LDX     CURPOS  PRINT L-
VALUE
707F EC   66        01430         LDD     6,S
7081 BD   4169      01440         JSR     PUTWRD
7084 9F   88        01450         STX     CURPOS
7086 8E   700F      01460         LDX     #STRSEP SEPARATOR
7089 BD   420A      01470         JSR     PRTS00  PRINT IT
                    01480
                    01490 * PRINT D-VALUE
708C 8E   7006      01500         LDX     #STRD   /D=/
708F BD   420A      01510         JSR     PRTS00  PRINT /D=/
7092 9E   88        01520         LDX     CURPOS  PRINT D-
VALUE
7094 EC   64        01530         LDD     4,S
7096 BD   4169      01540         JSR     PUTWRD
7099 9F   88        01550         STX     CURPOS
709B 8E   700F      01560         LDX     #STRSEP SEPARATOR
709E BD   420A      01570         JSR     PRTS00  PRINT IT
                    01580
                    01590 * PRINT T-VALUE
70A1 8E   7009      01600         LDX     #STRT   /T=/
70A4 BD   420A      01610         JSR     PRTS00  PRINT /T=/
70A7 9E   88        01620         LDX     CURPOS  PRINT T-
VALUE
70A9 A6   63        01630         LDA     3,S
70AB BD   4025      01640         JSR     PUTBYT
70AE 9F   88        01650         STX     CURPOS
70B0 8E   700F      01660         LDX     #STRSEP SEPARATOR
70B3 BD   420A      01670         JSR     PRTS00  PRINT IT
                    01680
                    01690 * PRINT S-VALUE
70B6 8E   700C      01700         LDX     #STRS   /S=/
70B9 BD   420A      01710         JSR     PRTS00  PRINT /S=/
70BC 9E   88        01720         LDX     CURPOS  PRINT S-
VALUE
70BE A6   62        01730         LDA     2,S
```

```
70C0 BD   4025     01740          JSR     PUTBYT
70C3 9F   88       01750          STX     CURPOS
70C5 BD   40D7     01760          JSR     CRLF
70C8 39            01770          RTS
                   01780
          0000     01790          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0101.BAS
1030 '* MDJ 2023/02/28
1040 '*
1050 '* FLXLTL TEST
1060 '*
1070 '*****
1080 '

1100 'SETUP MEMORY
1110 CLEAR 200, &H4000
1120 PCLEAR 4
1130 '

1200 'LOAD THE
1210 'ML FOUNDATION CORE
1220 LOADM "MLCORE.BIN"
1230 '

1300 'LOAD THE ML WORD
1310 'TO BE TESTED
1320 LOADM "FLXLTL.BIN"
1330 '

1400 'LOAD THE
1410 'ML TEST ROUTINE
1420 LOADM "TEST0101.BIN"
1430 '

2000 'REFERENCE THE
2010 'TRANSFER VARIABLES
2080 RA = &H400A 'REGPCH
2090 RB = &H400B 'REGPCL
2100 '
```

```
3000 'SETUP THE
3010 'RUN ADDRESS
3020 C = &H7000
3030 C1 = INT(C/256)
3040 C2 = INT(C-(C1*256))
3050 POKE RA, C1
3060 POKE RB, C2
3070 '

6000 'JUMP TO CORE
6010 'STARTUP ROUTINE
6020 EXEC &H4403
6030 '

9000 'MEMORY AND DISK
9010 'STATUS CHECK
9020 PRINT
9030 PRINT " MEM = ";MEM
9040 PRINT "FREE = ";FREE(0)

32767 END
```

Result:

As expected.

=====

FLXLTD.ASM: Routine to Translate a Standard Drive Number, Track Number, and Sector Number Into a Linear Sector Number

```
                    00100 *****
                    00110 *
                    00120 * FLXLTD.ASM
                    00130 * MDJ 2023/02/24
                    00140 *
                    00150 * FALSE DISK
                    00160 * D,T,S TO L
                    00170 * TRANSLATION SUBROUTINE
                    00180 *
                    00190 * ENTRY CONDITIONS:
                    00200 * REGISTER X = D (0-3)
                    00210 * = DRIVE NUMBER
                    00220 * REGISTER A = T (0-34)
                    00230 * = TRACK NUMBER
                    00240 * REGISTER B = S (1-18)
                    00250 * = SECTOR NUMBER
                    00260 *
                    00270 * EXIT CONDITIONS
                    00280 * REGISTER X = L (0-2447)
                    00290 * = LINEAR SECTOR NUMBER
                    00300 *
                    00310 * ON EXIT:
                    00320 * REGISTER X = $FFFF
                    00330 * ==> ERROR
                    00340 *
                    00350 *****
                    00360
                    00370 * GENERAL EQUATES
          0264      00380 DMUL    EQU     612     DRIVE
MULTIPLICAND = $264
          0012      00390 TMUL    EQU     18      TRACK
MULTIPLICAND = $12
                    00400
448A                00410         ORG     $448A
                    00420
                    00430 * ERROR TRAPPING
```

```
448A 8C   0003     00440 FLXLTD  CMPX    #3      MAX DRIVE
#
448D 22   5F       00450         BHI     LBL007  ERROR IF
GREATER
448F 81   22       00460         CMPA    #34     MAX TRACK
#
4491 22   5B       00470         BHI     LBL007  ERROR IF
GREATER
4493 81   11       00480         CMPA    #17     DIRECTORY
TRACK # (SKIP)
4495 27   57       00490         BEQ     LBL007  ERROR IF =
TRACK 17
4497 C1   12       00500         CMPB    #18     MAX SECTOR
#
4499 22   53       00510         BHI     LBL007  ERROR IF
GREATER
449B C1   00       00520         CMPB    #0      ILLEGAL
SECTOR #
449D 27   4F       00530         BEQ     LBL007  ERROR IF #
= 0
                   00540
                   00550 * AT THIS POINT, THE STACK
CONTAINS:
                   00560 *    1,S RTS LOCATION LOW BYTE
                   00570 *     ,S RTS LOCATION HIGH BYTE
                   00580
                   00590 * MAKE SPACE ON THE STACK FOR:
                   00600 *    7,S RTS LOCATION LOW BYTE
                   00610 *    6,S RTS LOCATION HIGH BYTE
                   00620 *    5,S TEMPORARY VALUE LOW BYTE
                   00630 *    4,S TEMPORARY VALUE HIGH BYTE
                   00640 *    3,S REGISTER X LOW BYTE
(DRIVE NUMBER)
                   00650 *    2,S REGISTER X HIGH BYTE (ALL
ZEROES)
                   00660 *    1,S REGISTER A (TRACK NUMBER)
                   00670 *     ,S REGISTER B (SECTOR
NUMBER)
449F 32   7A       00680         LEAS    -6,S
                   00690
                   00700 * SAVE ENTRY VALUES TO THE STACK
44A1 AF   62       00710         STX     2,S     DRIVE
NUMBER
44A3 A7   61       00720         STA     1,S     TRACK
NUMBER
44A5 E7   E4       00730         STB     ,S      SECTOR
NUMBER
```

```
                    00740
                    00750 * L = (D * 612) + (T * 18) + S
                    00760 * IF (T > 17) THEN L = L - 18
                    00770
                    00780 * CALCULATE FIRST TERM = (D * 612)
                    00790 * BY EMULATING MULTIPLICATION VIA
                    00800 * DIRECT SUBSTITUTION
44A7 A6   63       00810          LDA     3,S     DRIVE
NUMBER LOW BYTE
44A9 81   00       00820          CMPA    #0      IS IT
DRIVE #0 ?
44AB 27   17       00830          BEQ     LBL003  GO IF YES
44AD 81   01       00840          CMPA    #1      IS IT
DRIVE #1 ?
44AF 27   0E       00850          BEQ     LBL002  GO IF YES
44B1 81   02       00860          CMPA    #2      IS IT
DRIVE #2 ?
44B3 27   05       00870          BEQ     LBL001  GO IF YES
44B5 CC   072C     00880          LDD     #DMUL*3 FOR DRIVE
#3
44B8 20   0D       00890          BRA     LBL004
44BA CC   04C8     00900 LBL001   LDD     #DMUL*2 FOR DRIVE
#2
44BD 20   08       00910          BRA     LBL004
44BF CC   0264     00920 LBL002   LDD     #DMUL   FOR DRIVE
#1
44C2 20   03       00930          BRA     LBL004
44C4 CC   0000     00940 LBL003   LDD     #0      FOR DRIVE
#0
44C7 ED   62       00950 LBL004   STD     2,S     SAVE FIRST
TERM ON STACK
                    00960
                    00970 * CALCULATE THE SECOND TERM = (T *
18)
44C9 A6   61       00980          LDA     1,S     TRACK
NUMBER
44CB C6   12       00990          LDB     #TMUL   NUMBER OF
SECTORS PER TRACK
44CD 3D            01000          MUL             DO THE
MULTIPLY
44CE ED   64       01010          STD     4,S
TEMPORARILY SAVE TO STACK
                    01020
                    01030 * ADD THE TERMS
44D0 E6   E4       01040          LDB     ,S      SECTOR
NUMBER (1-18)
```

```
44D2 5A            01050         DECB            ADJUST
SECTOR NUMBER
                   01060 *                         TO ZERO-
BASE (0-17)
44D3 4F            01070         CLRA            SECTOR
NUMBER NOW IN LOW BYTE O
F D
44D4 E3   64       01080         ADDD    4,S     ADD THE
SECOND TERM
44D6 E3   62       01090         ADDD    2,S     ADD THE
FIRST TERM
44D8 ED   64       01100         STD     4,S
TEMPORARILY SAVE RESULT TO STAC
K
                   01110
                   01120 * SPECIAL HANDLING FOR TRACK
NUMBERS > 17
44DA A6   61       01130         LDA     1,S     TRACK
NUMBER
44DC 81   11       01140         CMPA    #17     IS IT
GREATER THAN 17
44DE 22   04       01150         BHI     LBL005  GO IF YES
                   01160
                   01170 * RETURN REGULAR RESULT FOR TRACKS
< 17
44E0 AE   64       01180         LDX     4,S     LINEAR
SECTOR NUMBER
44E2 20   07       01190         BRA     LBL006
                   01200
                   01210 * RETURN SPECIAL RESULT FOR TRACK
NUMBERS > 17
44E4 EC   64       01220 LBL005  LDD     4,S     TEMPORARY
VALUE
44E6 83   0012     01230         SUBD    #TMUL   SKIP TRACK
17
44E9 1F   01       01240         TFR     D,X     LINEAR
SECTOR NUMBER
                   01250
                   01260 * CLEAN THE STACK AND RETURN
44EB 32   66       01270 LBL006  LEAS    6,S
44ED 39            01280         RTS
                   01290
                   01300 * ERROR HANDLING
44EE 8E   FFFF     01310 LBL007  LDX     #$FFFF  ERROR CODE
44F1 39            01320         RTS
                   01330
          0000     01340         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0102.ASM
                    00130 * MDJ 2023/02/28
                    00140 *
                    00150 * FLXLTD TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * LOW RAM CURSOR ADDRESS
          0088      00200 CURPOS  EQU     $0088
                    00210
                    00220 * SCREEN ADDRESSES
          0400      00230 VIDRAM  EQU     $0400
          0600      00240 VIDEND  EQU     $0600
                    00250
                    00260 * RAMROM TRIGGER ADDRESS
          FFDE      00270 RAMROM  EQU     $FFDE
                    00280
                    00290 * ALLRAM TRIGGER ADDRESS
          FFDF      00300 ALLRAM  EQU     $FFDF
                    00310
                    00320 * ML FOUNDATION
                    00330 * CORE ADDRESSES
          400E      00340 VIDCLS  EQU     $400E
          4025      00350 PUTBYT  EQU     $4025
          40D7      00360 CRLF    EQU     $40D7
          4169      00370 PUTWRD  EQU     $4169
          420A      00380 PRTS00  EQU     $420A
          43B6      00390 SNIRQS  EQU     $43B6
                    00400
                    00410 * FALSE DISK
                    00420 * SYSTEM ADDRESS
          448A      00430 FLXLTD  EQU     $448A
                    00440
7000                00450         ORG     $7000
7000 7E   7012      00460         JMP     LBL001
                    00470
7003      4C        00480 STRL    FCC     /L=/
          3D
7005      00        00490         FCB     $00     NUL
7006      44        00500 STRD    FCC     /D=/
```

```
          3D
7008      00        00510          FCB     $00     NUL
7009      54        00520 STRT     FCC     /T=/
          3D
700B      00        00530          FCB     $00     NUL
700C      53        00540 STRS     FCC     /S=/
          3D
700E      00        00550          FCB     $00     NUL
700F      2C        00560 STRSEP   FCC     /, /
          20
7011      00        00570          FCB     $00     NUL
                    00580
                    00590 * SETUP THE NEW INTERRUPT HANDLERS
                    00600 * AND ENTER ALLRAM MODE
7012 BD   43B6      00610 LBL001   JSR     SNIRQS
7015 B7   FFDF      00620          STA     ALLRAM
7018 34   07        00630          PSHS    A,B,CC
                    00640
                    00650 * AT THIS POINT, THE STACK
CONTAINS:
                    00660 *    4,S RTS LOCATION LOW BYTE
                    00670 *    3,S RTS LOCATION HIGH BYTE
                    00680 *    2,S REGISTER B
                    00690 *    1,S REGISTER A
                    00700 *     ,S REGISTER CC
                    00710
                    00720 * MAKE SPACE ON THE STACK FOR:
                    00730 *   10,S RTS LOCATION LOW BYTE
                    00740 *    9,S RTS LOCATION HIGH BYTE
                    00750 *    8,S REGISTER B
                    00760 *    7,S REGISTER A
                    00770 *    6,S REGISTER CC
                    00780 *    5,S L-VALUE LOW BYTE
                    00790 *    4,S L-VALUE HIGH BYTE
                    00800 *    3,S D-VALUE LOW BYTE
                    00810 *    2,S D-VALUE HIGH BYTE
                    00820 *    1,S T-VALUE
                    00830 *     ,S S-VALUE
701A 32   7A        00840          LEAS    -6,S
                    00850
                    00860 * SETUP THE SCREEN
701C BD   400E      00870          JSR     VIDCLS
701F 8E   0400      00880          LDX     #VIDRAM TOP OF
SCREEN
7022 9F   88        00890          STX     CURPOS  CURSOR
                    00900
                    00910 * TEST RUNS INPUT DATA
```

```
7024 8E   0000     00920          LDX     #0      D = $0000
7027 86   00       00930          LDA     #0      T = $00
7029 C6   01       00940          LDB     #1      S = $01
702B BD   70BB     00950          JSR     LBL002  FLXLTD
CONTROL
702E BD   70C7     00960          JSR     LBL003  REPORTER
                   00970
7031 8E   0001     00980          LDX     #1      D = $0001
7034 86   00       00990          LDA     #0      T = $00
7036 C6   01       01000          LDB     #1      S = $01
7038 BD   70BB     01010          JSR     LBL002  FLXLTD
CONTROL
703B BD   70C7     01020          JSR     LBL003  REPORTER
                   01030
703E 8E   0002     01040          LDX     #2      D = $0002
7041 86   00       01050          LDA     #0      T = $00
7043 C6   01       01060          LDB     #1      S = $01
7045 BD   70BB     01070          JSR     LBL002  FLXLTD
CONTROL
7048 BD   70C7     01080          JSR     LBL003  REPORTER
                   01090
704B 8E   0003     01100          LDX     #3      D = $0003
704E 86   00       01110          LDA     #0      T = $00
7050 C6   01       01120          LDB     #1      S = $01
7052 BD   70BB     01130          JSR     LBL002  FLXLTD
CONTROL
7055 BD   70C7     01140          JSR     LBL003  REPORTER
                   01150
7058 8E   0003     01160          LDX     #3      D = $0003
705B 86   22       01170          LDA     #34     T = $22
705D C6   12       01180          LDB     #18     S = $12
705F BD   70BB     01190          JSR     LBL002  FLXLTD
CONTROL
7062 BD   70C7     01200          JSR     LBL003  REPORTER
                   01210
7065 8E   0004     01220          LDX     #4      D = $0004
= ERROR
7068 86   00       01230          LDA     #0      T = $00
706A C6   01       01240          LDB     #1      S = $01
706C BD   70BB     01250          JSR     LBL002  FLXLTD
CONTROL
706F BD   70C7     01260          JSR     LBL003  REPORTER
                   01270
7072 8E   0000     01280          LDX     #0      D = $0000
7075 86   10       01290          LDA     #16     T = $10
7077 C6   12       01300          LDB     #18     S = $12
```

```
7079 BD   70BB     01310          JSR     LBL002  FLXLTD
CONTROL
707C BD   70C7     01320          JSR     LBL003  REPORTER
                   01330
707F 8E   0000     01340          LDX     #0      D = $0000
7082 86   23       01350          LDA     #35     T = $23 =
ERROR
7084 C6   12       01360          LDB     #18     S = $12
7086 BD   70BB     01370          JSR     LBL002  FLXLTD
CONTROL
7089 BD   70C7     01380          JSR     LBL003  REPORTER
                   01390
708C 8E   0000     01400          LDX     #0      D = $0000
708F 86   12       01410          LDA     #18     T = $12
7091 C6   01       01420          LDB     #1      S = $01
7093 BD   70BB     01430          JSR     LBL002  FLXLTD
CONTROL
7096 BD   70C7     01440          JSR     LBL003  REPORTER
                   01450
7099 8E   0000     01460          LDX     #0      D = $0000
709C 86   00       01470          LDA     #0      T = $00
709E C6   00       01480          LDB     #0      S = $00 =
ERROR
70A0 BD   70BB     01490          JSR     LBL002  FLXLTD
CONTROL
70A3 BD   70C7     01500          JSR     LBL003  REPORTER
                   01510
70A6 8E   0000     01520          LDX     #0      D = $0000
70A9 86   11       01530          LDA     #17     T = $11 =
ERROR
70AB C6   01       01540          LDB     #1      S = $01
70AD BD   70BB     01550          JSR     LBL002  FLXLTD
CONTROL
70B0 BD   70C7     01560          JSR     LBL003  REPORTER
                   01570
                   01580 * RETURN TO RAMROM MODE,
                   01590 * CLEAN THE STACK, AND EXIT
70B3 B7   FFDE     01600          STA     RAMROM
70B6 32   66       01610          LEAS    6,S
70B8 35   07       01620          PULS    A,B,CC
70BA 39            01630          RTS
                   01640
                   01650 * FLXLTD CONTROL SUBROUTINE
                   01660 * AT THIS POINT, WE HAVE ADDED THE
TWO-BYTE
                   01670 * FLXLTD CONTROL SUBROUTINE RETURN
ADDRESS
```

```
                    01680 * TO THE STACK, AND THUS MUST ADD
2 TO ALL
                    01690 * L,D,T,S STACK ACCESSES
70BB AF   64      01700 LBL002  STX     4,S     PUT D-
VALUE TO STACK
70BD A7   63      01710         STA     3,S     PUT T-
VALUE TO STACK
70BF E7   62      01720         STB     2,S     PUT S-
VALUE TO STACK
70C1 BD   448A    01730         JSR     FLXLTD  GO DO THE
TRANSLATION
70C4 AF   66      01740         STX     6,S     PUT L-
VALUE TO STACK
70C6 39           01750         RTS
                  01760
                  01770 * REPORTER SUBROUTINE
                  01780 * AT THIS POINT, WE HAVE ADDED THE
TWO-BYTE
                  01790 * REPORTER SUBROUTINE RETURN
ADDRESS
                  01800 * TO THE STACK, AND THUS MUST ADD
2 TO ALL
                  01810 * L,D,T,S STACK ACCESSES
                  01820
                  01830 * PRINT D-VALUE
70C7 8E   7006    01840 LBL003  LDX     #STRD   /D=/
70CA BD   420A    01850         JSR     PRTS00  PRINT /D=/
70CD 9E   88      01860         LDX     CURPOS  PRINT D-
VALUE
70CF EC   64      01870         LDD     4,S
70D1 BD   4169    01880         JSR     PUTWRD
70D4 9F   88      01890         STX     CURPOS
70D6 8E   700F    01900         LDX     #STRSEP SEPARATOR
70D9 BD   420A    01910         JSR     PRTS00  PRINT IT
                  01920
                  01930 * PRINT T-VALUE
70DC 8E   7009    01940         LDX     #STRT   /T=/
70DF BD   420A    01950         JSR     PRTS00  PRINT /T=/
70E2 9E   88      01960         LDX     CURPOS  PRINT T-
VALUE
70E4 A6   63      01970         LDA     3,S
70E6 BD   4025    01980         JSR     PUTBYT
70E9 9F   88      01990         STX     CURPOS
70EB 8E   700F    02000         LDX     #STRSEP SEPARATOR
70EE BD   420A    02010         JSR     PRTS00  PRINT IT
                  02020
                  02030 * PRINT S-VALUE
```

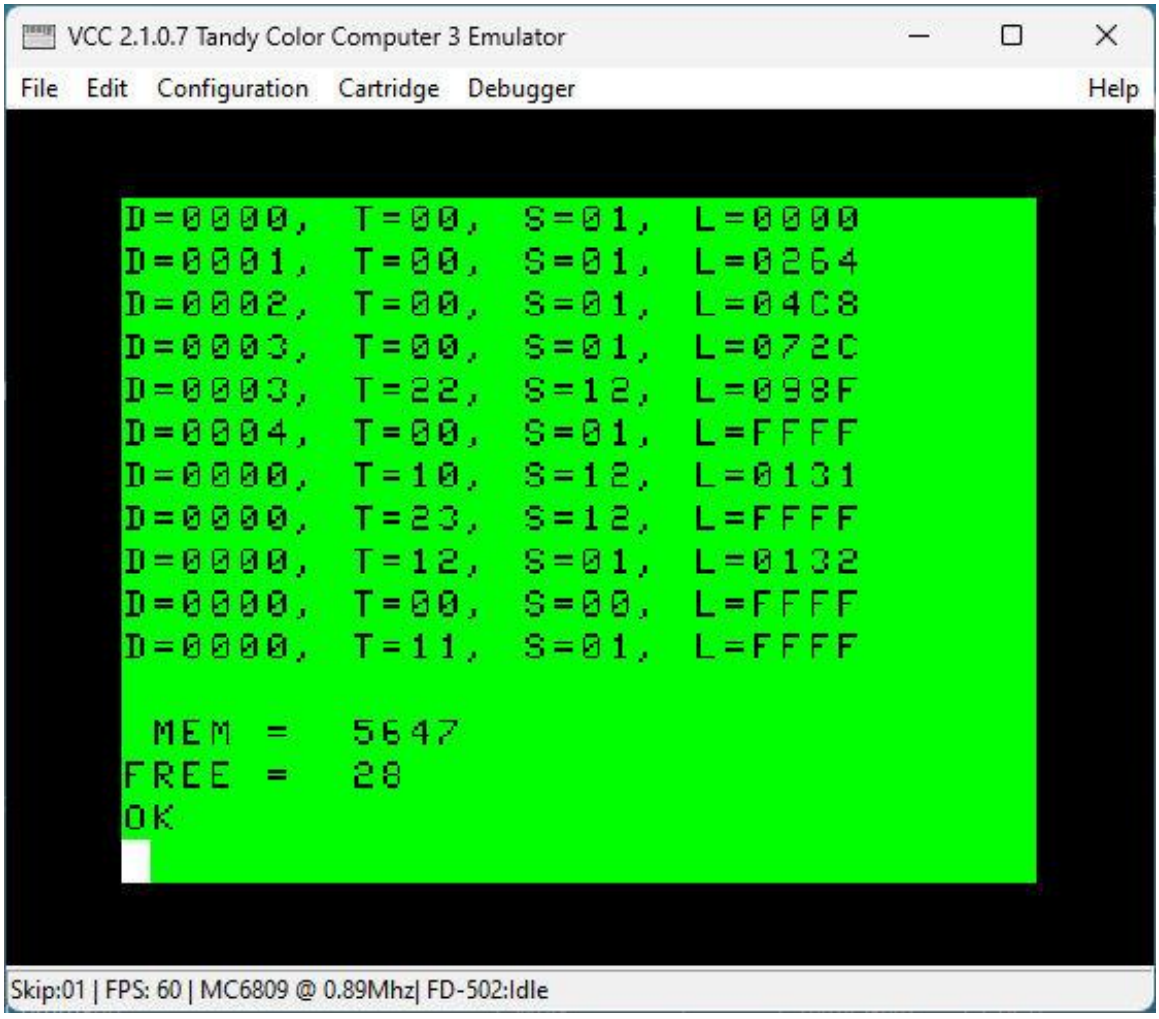
```
70F1 8E   700C     02040          LDX     #STRS   /S=/
70F4 BD   420A     02050          JSR     PRTS00  PRINT /S=/
70F7 9E   88       02060          LDX     CURPOS  PRINT S-
VALUE
70F9 A6   62       02070          LDA     2,S
70FB BD   4025     02080          JSR     PUTBYT
70FE 9F   88       02090          STX     CURPOS
7100 8E   700F     02100          LDX     #STRSEP SEPARATOR
7103 BD   420A     02110          JSR     PRTS00  PRINT IT
                   02120
                   02130 * PRINT L-VALUE
7106 8E   7003     02140          LDX     #STRL   /L=/
7109 BD   420A     02150          JSR     PRTS00  PRINT /L=/
710C 9E   88       02160          LDX     CURPOS  PRINT L-
VALUE
710E EC   66       02170          LDD     6,S
7110 BD   4169     02180          JSR     PUTWRD
7113 9F   88       02190          STX     CURPOS
7115 BD   40D7     02200          JSR     CRLF
7118 39            02210          RTS
                   02220
          0000     02230          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0102.BAS
1030 '* MDJ 2023/02/28
1040 '*
1050 '* FLXLTD TEST
1060 '*
1070 '*****
1080 '

1100 'SETUP MEMORY
1110 CLEAR 200, &H4000
1120 PCLEAR 4
1130 '

1200 'LOAD THE
1210 'ML FOUNDATION CORE
1220 LOADM "MLCORE.BIN"
1230 '
```

```
1300 'LOAD THE ML WORD
1310 'TO BE TESTED
1320 LOADM "FLXLTD.BIN"
1330 '

1400 'LOAD THE
1410 'ML TEST ROUTINE
1420 LOADM "TEST0102.BIN"
1430 '

2000 'REFERENCE THE
2010 'TRANSFER VARIABLES
2080 RA = &H400A 'REGPCH
2090 RB = &H400B 'REGPCL
2100 '

3000 'SETUP THE
3010 'RUN ADDRESS
3020 C = &H7000
3030 C1 = INT(C/256)
3040 C2 = INT(C-(C1*256))
3050 POKE RA, C1
3060 POKE RB, C2
3070 '

6000 'JUMP TO CORE
6010 'STARTUP ROUTINE
6020 EXEC &H4403
6030 '

9000 'MEMORY AND DISK
9010 'STATUS CHECK
9020 PRINT
9030 PRINT " MEM = ";MEM
9040 PRINT "FREE = ";FREE(0)

32767 END
```

Result:



As expected.

=====

FLPUT.ASM: Routine to Put a Buffer To a Linear Sector

```
                00100 *****
                00110 *
                00120 * FLPUT.ASM
                00130 * MDJ 2023/02/24
                00140 *
                00150 * FALSE DISK SYSTEM
                00160 * PUT BUFFER TO
                00170 * LINEAR SECTOR
                00180 * ON DISK
                00190 *
                00200 * PUTS A LINEAR SECTOR
                00210 * (0-2447)
                00220 * TO A FALSE DISK
                00230 * FROM A SPECIFIED
                00240 * DISK BUFFER
                00250 *
                00260 * ENTRY CONDITIONS:
                00270 * X = LINEAR SECTOR NUMBER
                00280 * Y = DISK BUFFER ADDRESS
                00290 *
                00300 * THERE IS NO INTERNAL
                00310 * CHECK ON THE VALIDITY
                00320 * OF THE BUFFER ADDRESS
                00330 * HERE - PERFORM EXTERNALLY
                00340 * BEFORE CALLING THIS
                00350 * SUBROUTINE.
                00360 *
                00370 * ON EXIT:
                00380 * REGISTER X = $FFFF
                00390 * ==> ERROR
                00400 *
                00410 *****
                00420
                00430 * LOW RAM VARIABLES
00EA            00440 DCOPC   EQU     $00EA
00EB            00450 DCDRV   EQU     $00EB
00EC            00460 DCTRK   EQU     $00EC
00ED            00470 DCSEC   EQU     $00ED
00EE            00480 DCBPT   EQU     $00EE
00F0            00490 DCSTA   EQU     $00F0
```

```
                    00500
                    00510 * ML FOUNDATION
                    00520 * CORE ADDRESS
          4253      00530 DISKRW  EQU     $4253
                    00540
                    00550 * EXTERNAL ROUTINE
                    00560 * ADDRESS
          4416      00570 FLXLTL  EQU     $4416
                    00580
44F2                00590         ORG     $44F2
                    00600
44F2 34   06        00610 FLPUT   PSHS    A,B
                    00620
                    00630 * AT THIS POINT, THE STACK
CONTAINS:
                    00640 *    3,S RTS LOCATION LOW BYTE
                    00650 *    2,S RTS LOCATION HIGH BYTE
                    00660 *    1,S REGISTER B
                    00670 *     ,S REGISTER A
                    00680
                    00690 * MAKE SPACE ON THE STACK FOR:
                    00700 *   11,S RTS LOCATION LOW BYTE
                    00710 *   10,S RTS LOCATION HIGH BYTE
                    00720 *    9,S REGISTER B
                    00730 *    8,S REGISTER A
                    00740 *    7,S BUFFER ADDRESS LOW BYTE
                    00750 *    6,S BUFFER ADDRESS HIGH BYTE
                    00760 *    5,S L-VALUE LOW BYTE
                    00770 *    4,S L-VALUE HIGH BYTE
                    00780 *    3,S D-VALUE LOW BYTE
                    00790 *    2,S D-VALUE HIGH BYTE (ALL
ZEROES)
                    00800 *    1,S T-VALUE
                    00810 *     ,S S-VALUE
44F4 32   78        00820         LEAS    -8,S
                    00830
                    00840 * SAVE L AND BUFFER ADDRESS ON THE
STACK
44F6 AF   64        00850         STX     4,S
44F8 10AF 66        00860         STY     6,S
                    00870
                    00880 * TRANSLATE L TO D,T,S
44FB AE   64        00890         LDX     4,S
44FD BD   4416      00900         JSR     FLXLTL
4500 8C   FFFF      00910         CMPX    #$FFFF
TRANSLATION ERROR?
4503 27   26        00920         BEQ    LBL001  GO IF YES
```

```
                    00930
                    00940 * SAVE D,T,S ON THE STACK
4505 AF   62        00950         STX     2,S
4507 A7   61        00960         STA     1,S
4509 E7   E4        00970         STB     ,S
                    00980
                    00990 * SET LOW RAM VARIABLES FOR DISKRW
450B EC   66        01000         LDD     6,S
450D DD   EE        01010         STD     DCBPT   BUFFER
ADDRESS
450F A6   E4        01020         LDA     ,S
4511 97   ED        01030         STA     DCSEC   SECTOR
NUMBER
4513 A6   61        01040         LDA     1,S
4515 97   EC        01050         STA     DCTRK   TRACK
NUMBER
4517 A6   63        01060         LDA     3,S
4519 97   EB        01070         STA     DCDRV   DRIVE
NUMBER
451B 86   03        01080         LDA     #3      WRITE CODE
TO:
451D 97   EA        01090         STA     DCOPC   OPERATION
CODE
                    01100
                    01110 * GO PUT BUFFER CONTENTS TO DISK
451F BD   4253      01120         JSR     DISKRW  GO DO THE
DISK WRITE
                    01130
                    01140 * CHECK FOR DISK ERROR
4522 96   F0        01150         LDA     DCSTA   GET STATUS
4524 26   05        01160         BNE     LBL001  GO IF DISK
ERROR
                    01170
                    01180 * NORMAL EXIT
                    01190 * CLEAN THE STACK, AND EXIT
4526 32   68        01200         LEAS    8,S
4528 35   06        01210         PULS    A,B
452A 39             01220         RTS
                    01230
                    01240 * ERROR HANDLING
452B 8E   FFFF      01250 LBL001  LDX     #$FFFF
                    01260 * CLEAN THE STACK, AND EXIT
452E 32   68        01270         LEAS    8,S
4530 35   06        01280         PULS    A,B
4532 39             01290         RTS
                    01300
          0000      01310         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0103.ASM
                    00130 * MDJ 2023/02/28
                    00140 *
                    00150 * FLPUT TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * RAMROM TRIGGER ADDRESS
          FFDE      00200 RAMROM  EQU     $FFDE
                    00210
                    00220 * ALLRAM TRIGGER ADDRESS
          FFDF      00230 ALLRAM  EQU     $FFDF
                    00240
                    00250 * ML FOUNDATION
                    00260 * CORE ADDRESS
          43B6      00270 SNIRQS  EQU     $43B6
                    00280
                    00290 * FALSE DISK
                    00300 * SYSTEM ADDRESS
          44F2      00310 FLPUT   EQU     $44F2
                    00320
7000                00330         ORG     $7000
7000 34   30        00340         PSHS    X,Y
                    00350
                    00360 * SETUP THE NEW INTERRUPT HANDLERS
                    00370 * AND ENTER ALLRAM MODE
7002 BD   43B6      00380         JSR     SNIRQS
7005 B7   FFDF      00390         STA     ALLRAM
                    00400
                    00410 * SETUP PARAMETERS
                    00420 * L = (D * 612) + (T * 18) + S - 1
                    00430 * D = 1, T = 1, S = 1 ==> L = 630
= $0276
7008 8E   0276      00440         LDX     #630    L
700B 108E 0600      00450         LDY     #$0600  STD DISK
BUFFER
                    00460
                    00470 * WRITE BUFFER TO DISK
700F BD   44F2      00480         JSR     FLPUT
```

```
                    00490
                    00500 * RETURN TO RAMROM MODE AND EXIT
7012 B7   FFDE      00510         STA     RAMROM
7015 35   30        00520         PULS    X,Y
7017 39             00530         RTS
                    00540
          0000      00550         END
```

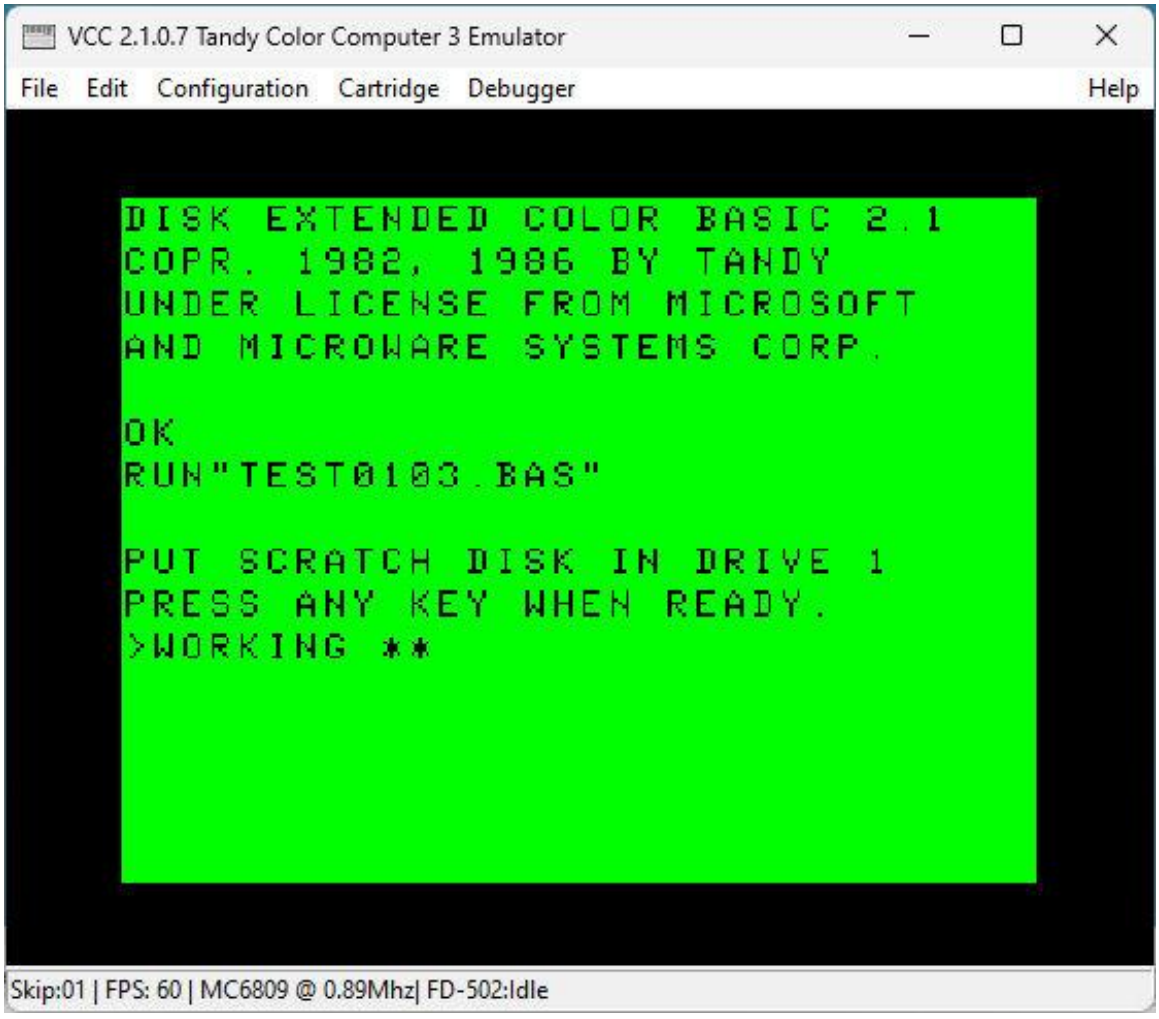
The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0103.BAS
1030 '* MDJ 2023/02/28
1040 '*
1050 '* FLPUT TEST
1060 '*
1070 '*****
1080 '

1100 'SETUP MEMORY
1110 CLEAR &H1000
1120 '

1200 'LOAD THE
1210 'ML FOUNDATION CORE
1220 LOADM "MLCORE.BIN"
1230 '

1300 'LOAD THE ML WORD
1310 'TO BE TESTED
1320 LOADM "FLPUT.BIN"
1330 'AND THAT WORD'S
1340 'DEPENDENCIES
1350 LOADM "FLXLTL.BIN"
1360 '

1400 'LOAD THE
1410 'ML TEST ROUTINE
1420 LOADM "TEST0103.BIN"
1430 '

1450 'L = (D * 612) + (T * 18) + S - 1
1460 'D = 1, T = 1, S = 1 ==> L = 630 = &H0276
```

```
1470 '

1500 PRINT:PRINT "PUT SCRATCH DISK IN DRIVE 1"
1530 PRINT "PRESS ANY KEY WHEN READY."
1540 PRINT ">";
1550 Z$ = INKEY$
1560 IF Z$ = "" GOTO 1550
1570 PRINT "WORKING *";
1580 '

1600 'LOAD THE STANDARD DISK BUFFER
1610 A$ = "WHOEVER CONFESSES THAT JESUS    "
1620 A$ = A$ + "IS THE SON OF GOD, GOD ABIDES   "
1630 A$ = A$ + "IN HIM, AND HE IN GOD. SO WE    "
1640 A$ = A$ + "HAVE COME TO KNOW AND TO        "
1650 '
1660 B$ = "BELIEVE THE LOVE THAT GOD HAS   "
1670 B$ = B$ + "FOR US. GOD IS LOVE, AND        "
1680 B$ = B$ + "WHOEVER ABIDES IN LOVE ABIDES   "
1690 B$ = B$ + "IN GOD, AND GOD ABIDES IN HIM.  "
1700 '
1710 '1 JOHN 4:15-16 (ESV)
1720 '
1730 A = &H0600
1740 B = &H0680
1750 FOR I = 0 TO 127
1760   X$ = MID$(A$,I+1,1)
1770   Y$ = MID$(B$,I+1,1)
1820   POKE (A + I), ASC(X$)
1830   POKE (B + I), ASC(Y$)
1840   IF I = 15 THEN PRINT "*";
1850   IF I = 31 THEN PRINT "*";
1860   IF I = 47 THEN PRINT "*";
1870   IF I = 63 THEN PRINT "*";
1880   IF I = 79 THEN PRINT "*";
1890   IF I = 95 THEN PRINT "*";
1900   IF I = 111 THEN PRINT "*";
1910   IF I = 127 THEN PRINT "*";
1920 NEXT I
1930 '

2000 PRINT:PRINT "BUFFER CHECK:":PRINT
2010 FOR I = &H0600 TO &H06FF
2020   A1$ = CHR$(PEEK(I))
2030   PRINT A1$;
2040 NEXT I
2050 PRINT:PRINT "PRESS ANY KEY TO WRITE TO DISK"
```

```
2060 PRINT ">";
2070 Z$ = INKEY$
2080 IF Z$ = "" GOTO 2070
2090 '

2900 'REFERENCE THE
2910 'TRANSFER VARIABLES
2920 RA = &H400A 'REGPCH
2930 RB = &H400B 'REGPCL
2940 '

3000 'SETUP THE
3010 'RUN ADDRESS
3020 C = &H7000
3030 C1 = INT(C/256)
3040 C2 = INT(C-(C1*256))
3050 POKE RA, C1
3060 POKE RB, C2
3070 '

6000 'JUMP TO CORE
6010 'STARTUP ROUTINE
6020 EXEC &H4403
6030 '

7000 PRINT:PRINT "DISK CHECK:":PRINT
7010 DSKI$ 1, 1, 1, P$, Q$
7020 PRINT P$;Q$
7030 '

9000 'MEMORY AND DISK
9010 'STATUS CHECK
9020 PRINT
9030 PRINT " MEM = ";MEM
9040 PRINT "FREE = ";FREE(0)
9050 '

32767 END
```

Result at first question:



Result at second question:

Result at test completion:

As expected.

Bible Note: This is 1 John 4:15-16 (ESV). This is a promise from God to us. If we rest our weight on this promise, God will comfort us with the inner conviction of the truth of this promise so that we can rest secure in His love even in the very midst of the worst turmoil, want, danger, and persecution that Satan and his minions can throw at us. He will strengthen us to remember that "he who is in you is greater than he who is in the world" (1 John 1:4, ESV).

=====

FLGET.ASM: Routine to Get a Buffer From a Linear Sector

```
          00100 *****
          00110 *
          00120 * FLGET.ASM
          00130 * MDJ 2023/02/24
          00140 *
          00150 * GET BUFFER FROM
          00160 * LINEAR SECTOR
          00170 * ON DISK
          00180 *
          00190 * GETS A LINEAR SECTOR
          00200 * (0-2447)
          00210 * FROM A FALSE DISK
          00220 * TO A SPECIFIED
          00230 * DISK BUFFER
          00240 *
          00250 * ENTRY CONDITIONS:
          00260 * X = LINEAR SECTOR NUMBER
          00270 * Y = DISK BUFFER ADDRESS
          00280 *
          00290 * THERE IS NO INTERNAL
          00300 * CHECK ON THE VALIDITY
          00310 * OF THE BUFFER ADDRESS
          00320 * HERE - PERFORM EXTERNALLY
          00330 * BEFORE CALLING THIS
          00340 * SUBROUTINE.
          00350 *
          00360 * ON EXIT:
          00370 * REGISTER X = $FFFF
          00380 * ==> ERROR
          00390 *
          00400 *****
          00410
          00420 * LOW RAM VARIABLES
00EA      00430 DCOPC   EQU     $00EA
00EB      00440 DCDRV   EQU     $00EB
00EC      00450 DCTRK   EQU     $00EC
00ED      00460 DCSEC   EQU     $00ED
00EE      00470 DCBPT   EQU     $00EE
00F0      00480 DCSTA   EQU     $00F0
          00490
```

```
                    00500 * ML FOUNDATION
                    00510 * CORE ADDRESS
          4253      00520 DISKRW  EQU     $4253
                    00530
                    00540 * EXTERNAL ROUTINE
                    00550 * ADDRESS
          4416      00560 FLXLTL  EQU     $4416
                    00570
4533                00580         ORG     $4533
                    00590
4533 34   06        00600 FLGET   PSHS    A,B
                    00610
                    00620 * AT THIS POINT, THE STACK
CONTAINS:
                    00630 *    3,S RTS LOCATION LOW BYTE
                    00640 *    2,S RTS LOCATION HIGH BYTE
                    00650 *    1,S REGISTER B
                    00660 *     ,S REGISTER A
                    00670
                    00680 * MAKE SPACE ON THE STACK FOR:
                    00690 *   11,S RTS LOCATION LOW BYTE
                    00700 *   10,S RTS LOCATION HIGH BYTE
                    00710 *    9,S REGISTER B
                    00720 *    8,S REGISTER A
                    00730 *    7,S BUFFER ADDRESS LOW BYTE
                    00740 *    6,S BUFFER ADDRESS HIGH BYTE
                    00750 *    5,S L-VALUE LOW BYTE
                    00760 *    4,S L-VALUE HIGH BYTE
                    00770 *    3,S D-VALUE LOW BYTE
                    00780 *    2,S D-VALUE HIGH BYTE (ALL
ZEROES)
                    00790 *    1,S T-VALUE
                    00800 *     ,S S-VALUE
4535 32   78        00810         LEAS    -8,S
                    00820
                    00830 * SAVE L AND BUFFER ADDRESS ON THE
STACK
4537 AF   64        00840         STX     4,S
4539 10AF 66        00850         STY     6,S
                    00860
                    00870 * TRANSLATE L TO D,T,S
453C AE   64        00880         LDX     4,S
453E BD   4416      00890         JSR     FLXLTL
4541 8C   FFFF      00900         CMPX    #$FFFF
TRANSLATION ERROR?
4544 27   26        00910         BEQ    LBL001  GO IF YES
                    00920
```

```
                    00930 * SAVE D,T,S ON THE STACK
4546 AF   62        00940         STX     2,S
4548 A7   61        00950         STA     1,S
454A E7   E4        00960         STB     ,S
                    00970
                    00980 * SET LOW RAM VARIABLES FOR DISKRW
454C EC   66        00990         LDD     6,S
454E DD   EE        01000         STD     DCBPT   BUFFER
ADDRESS
4550 A6   E4        01010         LDA     ,S
4552 97   ED        01020         STA     DCSEC   SECTOR
NUMBER
4554 A6   61        01030         LDA     1,S
4556 97   EC        01040         STA     DCTRK   TRACK
NUMBER
4558 A6   63        01050         LDA     3,S
455A 97   EB        01060         STA     DCDRV   DRIVE
NUMBER
455C 86   02        01070         LDA     #2      READ CODE
TO:
455E 97   EA        01080         STA     DCOPC   OPERATION
CODE
                    01090
                    01100 * GO GET BUFFER CONTENTS FROM DISK
4560 BD   4253      01110         JSR     DISKRW  GO DO THE
DISK WRITE
                    01120
                    01130 * CHECK FOR DISK ERROR
4563 96   F0        01140         LDA     DCSTA   GET STATUS
4565 26   05        01150         BNE     LBL001  GO IF DISK
ERROR
                    01160
                    01170 * NORMAL EXIT
                    01180 * CLEAN THE STACK, AND EXIT
4567 32   68        01190         LEAS    8,S
4569 35   06        01200         PULS    A,B
456B 39             01210         RTS
                    01220
                    01230 * ERROR HANDLING
456C 8E   FFFF      01240 LBL001  LDX     #$FFFF
                    01250 * CLEAN THE STACK, AND EXIT
456F 32   68        01260         LEAS    8,S
4571 35   06        01270         PULS    A,B
4573 39             01280         RTS
                    01290
          0000      01300         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0104.ASM
                    00130 * MDJ 2023/02/28
                    00140 *
                    00150 * FLGET TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * RAMROM TRIGGER ADDRESS
          FFDE      00200 RAMROM  EQU     $FFDE
                    00210
                    00220 * ALLRAM TRIGGER ADDRESS
          FFDF      00230 ALLRAM  EQU     $FFDF
                    00240
                    00250 * ML FOUNDATION
                    00260 * CORE ADDRESS
          43B6      00270 SNIRQS  EQU     $43B6
                    00280
                    00290 * FALSE DISK
                    00300 * SYSTEM ADDRESS
          4533      00310 FLGET   EQU     $4533
                    00320
7000                00330         ORG     $7000
7000 34   30        00340         PSHS    X,Y
                    00350
                    00360 * SETUP THE NEW INTERRUPT HANDLERS
                    00370 * AND ENTER ALLRAM MODE
7002 BD   43B6      00380         JSR     SNIRQS
7005 B7   FFDF      00390         STA     ALLRAM
                    00400
                    00410 * SETUP PARAMETERS
                    00420 * L = (D * 612) + (T * 18) + S - 1
                    00430 * D = 1, T = 1, S = 1 ==> L = 630
= $0276
7008 8E   0276      00440         LDX     #630    L
700B 108E 0600      00450         LDY     #$0600  STD DISK
BUFFER
                    00460
                    00470 * READ BUFFER FROM DISK
700F BD   4533      00480         JSR     FLGET
                    00490
```

```
                    00500 * RETURN TO RAMROM MODE AND EXIT
7012 B7   FFDE      00510         STA     RAMROM
7015 35   30        00520         PULS    X,Y
7017 39             00530         RTS
                    00540
          0000      00550         END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0104.BAS
1030 '* MDJ 2023/02/28
1040 '*
1050 '* FLGET TEST
1060 '*
1070 '*****
1080 '

1100 'SETUP MEMORY
1110 CLEAR &H1000
1120 '

1200 'LOAD THE
1210 'ML FOUNDATION CORE
1220 LOADM "MLCORE.BIN"
1230 '

1300 'LOAD THE ML WORD
1310 'TO BE TESTED
1320 LOADM "FLGET.BIN"
1330 'AND THAT WORD'S
1340 'DEPENDENCIES
1350 LOADM "FLXLTL.BIN"
1360 '

1400 'LOAD THE
1410 'ML TEST ROUTINE
1420 LOADM "TEST0103.BIN"
1430 '

1450 'L = (D * 612) + (T * 18) + S - 1
1460 'D = 1, T = 1, S = 1 ==> L = 630 = &H0276
1470 '
```

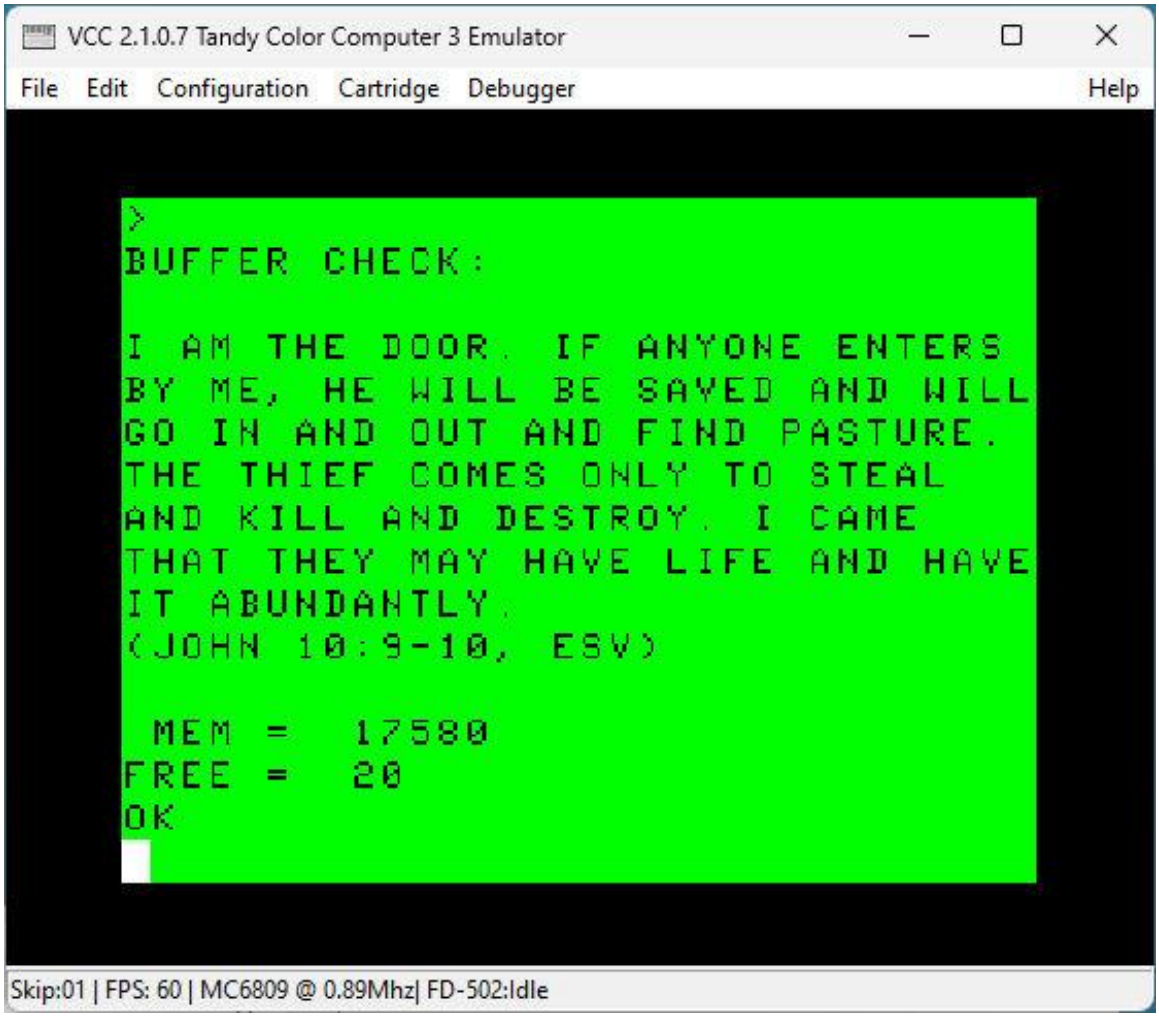
```
1500 PRINT:PRINT "PUT SCRATCH DISK IN DRIVE 1"
1530 PRINT "PRESS ANY KEY WHEN READY."
1540 PRINT ">";
1550 Z$ = INKEY$
1560 IF Z$ = "" GOTO 1550
1570 '

1600 PRINT:PRINT "DISK CHECK:":PRINT
1610 DSKI$ 1, 1, 1, P$, Q$
1620 PRINT P$;Q$
1630 '
1640 PRINT:PRINT "PRESS ANY KEY TO READ FROM
DISK"
1650 PRINT ">";
1660 Z$ = INKEY$
1670 IF Z$ = "" GOTO 1660
1680 '

2900 'REFERENCE THE
2910 'TRANSFER VARIABLES
2920 RA = &H400A 'REGPCH
2930 RB = &H400B 'REGPCL
2940 '

3000 'SETUP THE
3010 'RUN ADDRESS
3020 C = &H7000
3030 C1 = INT(C/256)
3040 C2 = INT(C-(C1*256))
3050 POKE RA, C1
3060 POKE RB, C2
3070 '

6000 'JUMP TO CORE
6010 'STARTUP ROUTINE
6020 EXEC &H4403
6030 '

7000 PRINT:PRINT "BUFFER CHECK:":PRINT
7010 FOR I = &H0600 TO &H06FF
7020   A1$ = CHR$(PEEK(I))
7030   PRINT A1$;
7040 NEXT I

9000 'MEMORY AND DISK
9010 'STATUS CHECK
```

```
9020 PRINT
9030 PRINT " MEM = ";MEM
9040 PRINT "FREE = ";FREE(0)
9050 '

32767 END
```

Result at first question:

Result at second question:

Result at test completion:



As expected.

Bible Note: This is a promise directly from Jesus to us. If we have received Jesus as our Lord and Savior, then He will touch our hearts with this reality that He is _THE_ door; not just _A_ door; that Jesus is the exclusive door: If people don't come into Heaven and Eternity through Him, then they're doomed to remain outside.

We're safe and secure within Jesus' sheepfold. There is none that can snatch us out of His hand! Even though we may face many troubles and much tribulation in this life, we know Whom we have believed and (Yes!) we are fully persuaded that He is indeed able to keep that which we've committed unto Him.

=====

MAKEFALS.BAS: Program to Make the FLSYS.BIN File

This program combines the FLXLTL,BIN , FLXLTD.BIN, FLPUT.BIN , and FLGET.BIN files into the single FLSYS.BIN file so that the entire machine language False Disk System can be loaded as a single entity.

```
1000 '*****
1010 '*
1020 '* MAKEFALS.BAS
1030 '* MDJ 2023/03/07
1040 '*
1050 '* MAKES THE
1060 '* FLSYS.BIN FILR
1070 '*
1080 '*****
1090 '

1500 CLEAR 200, &H4000
1510 '

2000 LOADM "FLXLTL.BIN"
2010 LOADM "FLXLTD.BIN"
2020 LOADM "FLPUT.BIN"
2030 LOADM "FLGET.BIN"
2040 '

3000 SAVEM "FLSYS.BIN", &H4416, &H4573, &H4416
3010 '

32767 END
```

=====

Appendix A
Decimal to Hexadecimal Conversions

DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX
000	00	032	20	064	40	096	60
001	01	033	21	065	41	097	61
002	02	034	22	066	42	098	62
003	03	035	23	067	43	099	63
004	04	036	24	068	44	100	64
005	05	037	25	069	45	101	65
006	06	038	26	070	46	102	66
007	07	039	27	071	47	103	67
008	08	040	28	072	48	104	68
009	09	041	29	073	49	105	69
010	0A	042	2A	074	4A	106	6A
011	0B	043	2B	075	4B	107	6B
012	0C	044	2C	076	4C	108	6C
013	0D	045	2D	077	4D	109	6D
014	0E	046	2E	078	4E	110	6E
015	0F	047	2F	079	4F	111	6F
016	10	048	30	080	50	112	70
017	11	049	31	081	51	113	71
018	12	050	32	082	52	114	72
019	13	051	33	083	53	115	73
020	14	052	34	084	54	116	74
021	15	053	35	085	55	117	75
022	16	054	36	086	56	118	76
023	17	055	37	087	57	119	77
024	18	056	38	088	58	120	78
025	19	057	39	089	59	121	79
026	1A	058	3A	090	5A	122	7A
027	1B	059	3B	091	5B	123	7B
028	1C	060	3C	092	5C	124	7C
029	1D	061	3D	093	5D	125	7D
030	1E	062	3E	094	5E	126	7E
031	1F	063	3F	095	5F	127	7F

DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX
128	80	160	A0	192	C0	224	E0
129	81	161	A1	193	C1	225	E1
130	82	162	A2	194	C2	226	E2
131	83	163	A3	195	C3	227	E3
132	84	164	A4	196	C4	228	E4
133	85	165	A5	197	C5	229	E5
134	86	166	A6	198	C6	230	E6
135	87	167	A7	199	C7	231	E7
136	88	168	A8	200	C8	232	E8
137	89	169	A9	201	C9	233	E9
138	8A	170	AA	202	CA	234	EA
139	8B	171	AB	203	CB	235	EB
140	8C	172	AC	204	CC	236	EC
141	8D	173	AD	205	CD	237	ED
142	8E	174	AE	206	CE	238	EE
143	8F	175	AF	207	CF	239	EF
144	90	176	B0	208	D0	240	F0
145	91	177	B1	209	D1	241	F1
146	92	178	B2	210	D2	242	F2
147	93	179	B3	211	D3	243	F3
148	94	180	B4	212	D4	244	F4
149	95	181	B5	213	D5	245	F5
150	96	182	B6	214	D6	246	F6
151	97	183	B7	215	D7	247	F7
152	98	184	B8	216	D8	248	F8
153	99	185	B9	217	D9	249	F9
154	9A	186	BA	218	DA	250	FA
155	9B	187	BB	219	DB	251	FB
156	9C	188	BC	220	DC	252	FC
157	9D	189	BD	221	DD	253	FD
158	9E	190	BE	222	DE	254	FE
159	9F	191	BF	223	DF	255	FF

=====

Appendix B: My CoCo Philosophy

The CoCo community enjoys a great diversity of interests.

Some choose to concentrate on hardware innovations and modifications such as interfacing with VGA and HDMI monitors, SD Card data storage, and 104-key keyboards. This interest is at least partly born of necessity, since composite monitors, floppy diskettes, and CoCo spare parts are no longer manufactured and are in increasingly short supply.

Others concentrate on expanding the software horizons of the CoCo 3, using NitrOS-9 and other operating systems to make the multitasking CoCo behave ever closer to modern Windows, Mac, and Linux machines.

Still others are devoted to emulating the CoCo on other platforms by developing emulators such as VCC, OVCC, MAME, and XRoar.

And some just love retro gaming.

My personal interest is twofold:

1. To see VCC increasingly used as a learning tool for budding software developers.

2. To see just how much I can cram into a 64K CoCo 2.

First, VCC: Today's Grade School, Junior High, and High School students have a wealth of available learning tools. Micro-bits, Arduinos, and Raspberry Pi supermicro devices provide highly affordable entry-level introductions to computer programming and interfacing. Maker-Spaces and Innovation Centers in our schools and libraries help foster growth and experience.

But these devices do have limitations. Even these simple(?) computers can have rather steep learning curves, and their low initial cost can quickly expand as new peripherals and experimental equipment and supplies are added.

VCC is free, and can be used on any Windows computer: just download it, install it, and it runs. If you don't own a Windows computer, your school, library, or a friend probably does. The included BASIC language is easy to learn and can readily serve as a stepping-stone towards more complex programming languages. (And, no, learning structured programming does not require a language that enforces structure. In fact, I think learning to structure your programs is actually more effective when you do so on your own.)

I prefer VCC to the other emulators for these purposes because its setup is trivial: Again, just download it, install it, and it runs. OVCC, MAME, and XRoar have their advantages, but ease of setup is not one of them. Even with their available Windows binary packages, they require pre-installation of other bits and pieces of software before they can be downloaded, installed, and run. This may not be a major problem for a reasonably adept aficionado, but it forms a significant barrier for the newbie. And, it's the newbie whom we're trying to reach, interest, and encourage here; the newbie who may not yet recognize even the tiniest awakening of interest in things computational.

But, for these purposes, VCC has one glaring weakness: its instruction manual is woefully terse. I would like to see VCC bundled with a selection of tutorials, manuals, and examples suited to guiding even the most newbie of newbies into the wonders of computing.

Second, The Stuffed CoCo: I'm simply fascinated by the challenge of seeing how much functional capability I can sandwich into the nooks and crannies of the 64K space. Whether it's working in the available RAM left by the 32K ROM and the dedicated RAM that supports that ROM, or whether it's jumping right into ALLRAM mode and just filling the entire 64K to near-overflowing; it's an investigative gauntlet which goes right to the heart of my enchantment with puzzles in general.

It's great fun!

M.D.J. 2021/08/29

=====

Appendix C: New BDS Software License

This New Software License applies to all software found on the BDS Software site, and supersedes all previous copyright notices and licensing provisions which may appear in the software itself or in any documentation therefor.

All software which has previously been placed in the public domain remains in the public domain.

All other software, programs, experiments and reports, documentation, and any other material on this site (other than that attributed to outside sources) is hereby copyright © 2018 (or later if so marked) by M. David Johnson.

All software, documentation, and other information on the BDS Software site is available for you to freely download without cost.

Whether you downloaded such items directly from this site, or you obtained them by any other means, you are hereby licensed to copy them, to sell or give away such copies, to use them, and to excerpt from them, in any way whatsoever, so long as nothing you do with them would denigrate the name of our Lord and Savior, Jesus Christ.

I make absolutely no warranty whatsoever for any of these items. You use them entirely at your own risk.

If they don't work for you, I commiserate.

If they crash your system, I sympathize.

But I accept no responsibility whatsoever for any such consequences. Under no circumstances will BDS Software or M. David Johnson be liable for any negative results of any kind which you may experience from downloading or using these items.

BDS Software's former mail address at P.O. Box 485 in Glenview, IL is no longer valid. Any mail sent to that address will be rejected by the U.S. Postal Service. See my Contact page.

M.D.J. 2018/06/08

=====

Works Cited

[DSOM] Tandy. *Color Computer Disk System Owner's Manual & Programming Guide*. Fort Worth, TX: Tandy, 1981. Print.

[MDJ01] Johnson, M. David. *Back To (Almost) Bare-Metal Programming*, Version 0.0.2. https://www.bds-soft.com/cocoPapers.php , 2023. Online.

=====