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

Back To

(Almost)

Bare-Metal Programming

Version 0.0.2

by M. David Johnson

2023/04/22

Abstract

Going back to (almost) bare metal, The ML Foundation Core Version 0.0.2 is presented as a base-level system upon which to build machine-language games and other software.

This system is intended for simplifying and organizing the development of 6809 Assembly Language Programs for the 64K Radio Shack Color Computer 2; most specifically in the areas of text processing and unsigned integer numerical computation.

This Version 0.0.2 iteration corrects bugs involving Disk Basic variables and routines in low ram. It also incorporates space for 8 pages of PMODE graphics.

____

This paper and its associated code are available online at:

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

=====

Table of Contents

=====

Introduction

From Techopedia:

Bare-metal programming is a term for programming that operates without various layers of abstraction or, as some experts describe it, "without an operating system supporting it." Bare-metal programming interacts with a system at the hardware level, taking into account the specific build of the hardware.

____

In this paper, I present Version 0.0.2 of the Core of The ML Foundation System, a system which will provide the bedrock upon which you can build more extensive Assembly Language games and other systems of your own.

This ML Foundation Core contains routines which could be expected to be utilized in almost any game or other software system. It specifically includes eight mechanisms which would be expected to be necessary for almost any game:

1. A minimal interface between BASIC and Machine Language.
2. Access to Keyboard Input.
3. Control of Output to the 32 x 16 VIDRAM Text Screen.
4. Access to Disk Storage.
5. 8-bit Sound.
6. Unsigned 8-bit and 16-bit Integer Math.
7. Various Text Processing Routines.
8. An Unsigned 16-bit Integer Pseudo-Random Number Generator.

This ML Foundation Core is intended to be used in three different ways.

First, the Core itself - The Core, all by itself, might be used as the base on which to build simple games and software. Unsigned Integers, for instance, may be quite sufficient for such endeavors. Signed Integers, Floating Point Numbers, etc. might be overkill. Users could then build their simple systems directly on top of the core.

Second, the primary ML Foundation. Certain more complicated routines will be added onto the Core in the future to produce a more complex ML Foundation. Users could then build their more complicated systems on top of that ML Foundation.

Third, alternate ML Foundations. Various combinations of more advanced software will be combined to form ML Foundations specific to certain types of systems. For, example one ML Foundation might be developed for games having high graphics content. Another ML Foundation might be designed to smooth the development of scientific computations. Still another might be designed to support the teaching of computational subject matter from within my proposed VCC Bundle. (See

[MDJ02]). Users would then be able to choose which ML Foundation was most suitable for their application.

——

When I first started learning 6809 Assembly Language back in the late 1980's, I found it to be quite a struggle. Books by experts such as Bill Barden, Lance Leventhal, and others were significant helps, but getting past the first hurdle of being able to put information into the computer and get stuff back out was daunting, to say the least.

With the completion of this paper, you have a minimally complete system; capable of receiving input from the Keyboard, generating output to the Video RAM (**`VIDRAM`**) Screen, and storing and retrieving data and code to and from the disk drives.

At this point, Version 0.0.2 of the ML Foundation Core is deemed to be functionally complete. It's should be sufficient for experimenting with, and as a foundation upon which to build additional software layers.

——

Back in 1985, James and Victor Perotti wrote:

> There is no better way to learn about computers than by learning to program in Assembly. With it you are directly manipulating the CPU ; you are writing in the language of the machine; you are learning how the computer works. With other languages, you program in the environment of that particular software. Assembly is the lowest level language, the one closest to the raw binary code that the CPU really processes. (Perotti 68)

And more recently, Ed Snider writes, "In programming the Color Computer, my language of choice is assembly language. It's the only way to get full performance out of the machine, and to control the hardware directly."

——

In order to build anything, including software, you need a strong and solid foundation on which to build (cf. what Jesus said about the house that was built on sand: Matthew 7:24-27). What I hope to provide here is a solid but simple foundation; one which is easy to understand and won't distract you by sending you down a rabbit hole, chasing after the true meaning of a tricky bit of code.

Efficiency is good, but clarity is better.

Consider, for a moment, the following BASIC program.

```
1DATA182,28,0,31,137,61,253,28,0,57:
PCLEAR1:CLEAR200,&H1C00:FORI=0TO9:RE
ADN1:POKE&H6000+I,N1:NEXTI
2INPUTN:N=INT(ABS(N)):IFN>255THENN=2
55
3POKE&H1C00,N:EXEC&H6000:N2=(((PEEK(
&H1C00))*256)+PEEK(&H1C01)):?N;"**2=
";N2:?"MORE(Y/N)";:INPUTZ$:IFZ$="Y"G
OTO2:END
```

DON’T TURN THE PAGE YET !

This three-line program is named SB.BAS —

Can you tell what this program is supposed to do?

And, if you can tell what it’s supposed to do, can you tell how it goes about doing it?

THREE… HOURS… LATER…

Okay, now you can turn the page.

——

This is the same program, only not quite so crunched up.

```
1'SQRBYTSV.BAS
2'E.G. SQRBYT SEMI-VERBOSE
3'MDJ 2021/09/13
4PCLEAR1
5CLEAR200,&H1C00
6DATA182,28,0,31,137,61,253,28,0,57
7FORI=0TO9
8READN1
9POKE&H6000+I,N1
10NEXTI
11INPUTN
12N=INT(ABS(N))
13IFN>255THENN=255
14POKE&H1C00,N
15EXEC&H6000
16N2=(((PEEK(&H1C00))*256)+PEEK(&H1C
01))
17PRINTN;"**2=";N2
18PRINT"MORE(Y/N)";
19INPUTZ$
20IFZ$="Y"GOTO11
21END
```

Is that any better?

“Not much,” you say.

Okay, go on to the next page.

——

This is essentially the same program again, but now fully commented and with some better direction for the I/O tasks:

```
1000 '*****
1010 '*
1020 '* SQRBYTV.BAS
1030 '* I.E. SQRBYT VERBOSE
1040 '* MDJ 2021/09/13
1050 '*
1060 '* SQUARES AN
1070 '* UNSIGNED BYTE
1080 '* IN MACHINE CODE
1090 '*
1100 '*****
1110 '
1120 'SETUP MEMORY
1130 PCLEAR 1
1140 CLEAR 200, &H1C00
1150 '
1160 'THE MACHINE LANGUAGE
1170 'ROUTINE FROM SQRBYT.ASM
1180 DATA 182,28,0,31,137
1190 DATA 61,253,28,0,57
1200 '
1210 'PUT THE MACHINE LANGUAGE
1220 'PROGRAM INTO MEMORY
1230 FOR I = 0 TO 9
1240 READ N1
1250 POKE &H6000+I,N1
1260 NEXT I
1270 '
1280 'UNCOMMENT FOR DEBUGGING
1290 'PRINT
1300 'FOR I = 0 TO 9
1310 'N = PEEK(&H6000+I)
1320 'PRINT N;",";
1330 'NEXT I
1340 'PRINT
1350 '
1360 'ENTER A NUMBER
1370 A$="ENTER AN UNSIGNED "
1380 B$="BYTE: "
1390 PRINT A$;B$;
1400 INPUT N
1410 '
1420 'NO NEGATIVE NUMBERS
1430 N=ABS(N)
```

```
1440 '
1450 'ONLY UNSIGNED INTEGERS
1460 N=INT(N)
1470 '
1480 'MAXIMUM SIZE = 8-BITS
1490 IF N>255 THEN N=255
1500 '
1510 'PUT N TO TRANSFER REGA
1520 POKE &H1C00,N
1530 '
1540 'GO DO THE SQUARE
1550 EXEC &H6000
1560 '
1570 'GET THE SQUARE FROM
1580 'TRANSFER REGD
1590 NA=PEEK(&H1C00)
1600 '
1610 'UNCOMMENT FOR DEBUGGING
1620 'PRINT"NA = ";NA
1630 '
1640 NB=PEEK(&H1C01)
1650 '
1660 'UNCOMMENT FOR DEBUGGING
1670 'PRINT"NB = ";NB
1680 '
1690 N2=((NA * 256) + NB)
1700 '
1710 'REPORT THE RESULTS
1720 A$="THE SQUARE OF "
1730 B$=" IS "
1740 PRINT A$;N;B$;N2
1750 '
1760 'DO IT AGAIN?
1770 PRINT
1780 PRINT "DO ANOTHER (Y/N)?"
1790 INPUT Z$
1800 PRINT
1810 IF Z$="Y" GOTO 1370
32767 END
```

Now, you should pretty much be able to tell what the program does at a glance.

But, it’s still difficult to tell HOW it does it, until we add the associated Assembly Language Routine which is provided on the following page.

BTW, the debugging statements are not just included as eye-candy. I originally had a typo in Line 1190, having typed in a “38” instead of a “28”.

At this point, everything should be clear.

```
                         *****
                         *
                         * SQRBYT.ASM
                         * MDJ 2021/09/13
                         *
                         * SQUARES AN
                         * UNSIGNED BYTE
                         *
                         * ENTRY CONDITIONS:
                         *   A = THE BYTE
                         *
                         * EXIT CONDITIONS:
                         *   D = THE SQUARE
                         *
                         *****

                         * 8-BIT TRANSFER
                         * REGISTER A
          1C00           REGA    EQU     $1C00

                         * 16-BIT TRANSFER
                         * REGISTER D
          1C00           REGD    EQU     $1C00

                         * A = HIGH BYTE OF D
                         * THUS THEY HAVE THE
                         * SAME ADDRESS

6000                             ORG     $6000

                         * GET THE BYTE
6000 B6   1C00                   LDA     REGA

                         * COPY IT TO REGISTER B
6003 1F   89                     TFR     A,B

                         * DO THE SQUARING
6005 3D                          MUL

                         * PUT THE 16-BIT RESULT
6006 FD   1C00                   STD     REGD

                         * EXIT
6009 39                          RTS
          0000                   END
```

Note that this code consists of ten bytes:

```
$B6, $1C, $00, $1F, $89, $3D, $FD, $1C, $00, $39
```

whose decimal equivalents are:

```
182, 028, 000, 031, 137, 061, 253, 028, 000, 057
```

or more compactly:

```
182,28,0,31,137,61,253,28,0,57
```

which represents the actual machine language code equivalent of the above assembly language routine.

——

There are times when super-crunched code like the original 3-Liner above is appropriate. Like when you're entering a CoCo-Stuffing contest. But, even then, you'd be wise to have a well-commented copy on hand somewhere.

In such cases, I like to keep a simple Microsoft Access database table on my Big Iron which includes:

1. A 12-character text field for the 8.3 crunched filename,
2. A 36-character text field for a brief description,
3. A 12-character text field for the 8.3 fully commented BASIC filename,
4. A 12-character text field for the 8.3 Assembly Language filename,
5. A 255-character text field for the first (up to) 255 characters of the text of the crunched file (MS Access makes it easy to re-order the table in alphanumeric order on any field), and
6. A memo field of effectively unlimited length.

This Version 0.0.2 is intended to present a strong, solid, and simple Color Computer machine language foundation core ("The ML Foundation Core") upon which you can build systems of your own. This ML Foundation is well-commented throughout — not only for you, but also for me when I come back to it and try to understand what I was doing five months (five minutes?) ago.

Others can, and have, written fancier and faster code than I have here. But my goal is to provide you with an easy-to-use place to begin. Fast and fancy can come later.

Now, even a foundation needs a foundation. When you build a house, the first thing you build is its foundation. But even before you start laying that foundation, you make sure you've dug down to solid rock.

This paper presents the foundation of the ML Foundation, i.e. the ML Foundation Core. The ML Foundation is not completed in this paper - there remains much yet to be done.

As Winston Churchill remarked at the Lord Mayor's dinner at Mansion House in London on November 10, 1942, just after the victory at El Alamein, "Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning." (Manchester and Reid 591).

____

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. 2022/04/22
info@bds-soft.com

=====

General Methodology

I'm developing The ML Foundation in two parts:

> 1. The ML Foundation Core, which includes the lowest level code, and which is established in specific locations in Low RAM. This part is treated in this paper.
>
> 2. The rest of The ML Foundation, which will be modular and which will be "relocatable" in that it will be built for simplicity of re-assembly. These parts of the ML Foundation will be addressed in future papers.
>
> I had originally intended to make this code truly relocatable, but I've determined that (at least for me) such relocatable code is not worth the cost in bytes, cycles, nor complexity of code. See Appendix C.

I am developing The ML Foundation, even within the Core, to be modular, rather than monolithic. This will allow each routine to be developed, written, and tested independently, greatly (hopefully) reducing both subsequent development and debugging time.

For each of the following Core Routines, I generally present the Assembly Language Routine itself, a Test Routine also written in Assembly Language, a BASIC Language Control Program to initiate and drive the Test Routine, and the results of the testing.

=====

MC6809 Register Set

The Register Set of the MC6809 is presented graphically on the Programmer's Card on the following page; from (Warren 154).

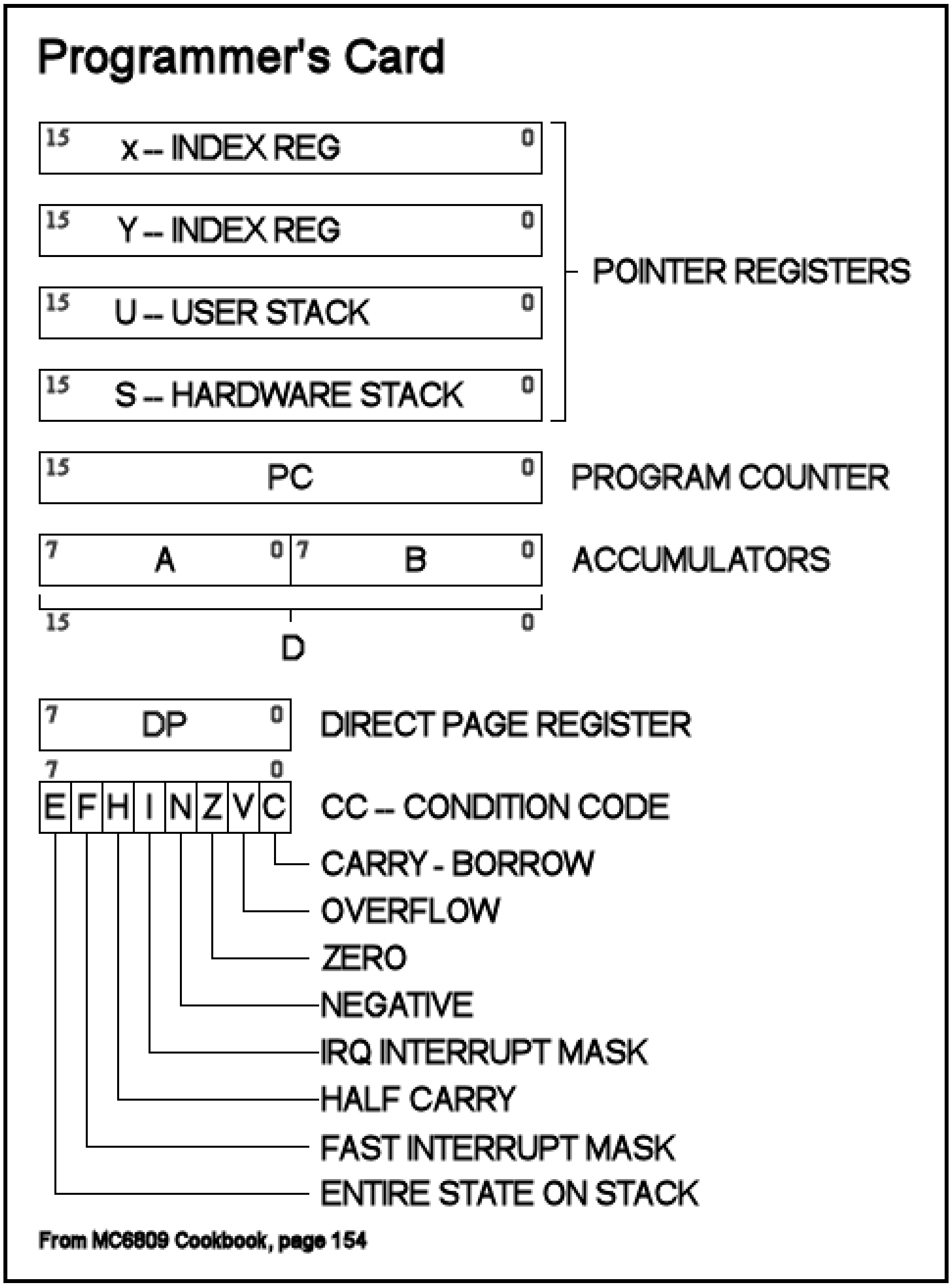
The X, Y, U, S, and PC Registers are each 16-bits wide.

The A, B, DP, and CC Registers are each 8-bits wide.

The A and B Registers also combine to form the 16-bit D Register, with the A Register serving as the D Register's High Byte, and the B Register serving as its Low Byte.

In the 16-bit Registers, as shown on the Programmer's Card, the bits are numbered from 0 to 15 from right-to-left.

The bits in the 8-bit Registers are numbered from 0 to 7 from right-to-left.


Programmer's Card
15
X -- INDEX REG
0
15
Y -- INDEX REG
0
15
U -- USER STACK
0
15
S -- HARDWARE STACK
0
POINTER REGISTERS
15
PC
0
PROGRAM COUNTER
7
A
0
7
B
0
ACCUMULATORS
15
D
0
7
DP
0
DIRECT PAGE REGISTER
7
0
E F H I N Z V C
CC -- CONDITION CODE
CARRY - BORROW
OVERFLOW
ZERO
NEGATIVE
IRQ INTERRUPT MASK
HALF CARRY
FAST INTERRUPT MASK
ENTIRE STATE ON STACK
From MC6809 Cookbook, page 154

=====

64K CoCo 2 Memory Map
ML Foundation Core

In 2021, [MDJ03] was presented as the beginning of this system.

In that paper, I reported:

> Also, during the earliest part of this development project, I discovered that stuff I put into Graphic Page 1 memory, even after "PCLEAR 0", often became corrupted for no reason I could easily discern.

In 2022, [MDJ04] discovered and reported that the problem was a heel-of-hand-smacking-forehead type of really embarrassing error: I had failed to note that the CoCo Disk System used RAM assigned to the Disk System from 0x0600 through 0x0DFF. Therefore, in Disk BASIC, the Graphics Memory starts at 0x0E00 instead of at 0x0600.

Here's the 64K CoCo 2 Disk System ML Foundation Core Memory Map, as modified for Version 0.0.2 (cf. the TRS-80 CoCo Wiki):

```
64K CoCo 2 ML Foundation Core Memory Map
========================================

Decimal         Address Contents    Hex Address
-------         ----------------    -----------

                    SYSTEM LOW RAM:

0-1023          System Use          0000-03FF

1024-1535       Text Screen Memory  0400-05FF

1536-3583       Disk System RAM     0600-0DFF

                Graphic
                Screen Memory
3584-5119       Page 1              0E00-13FF
5120-6655       Page 2              1400-19FF
6656-8191       Page 3              1A00-1FFF
8192-9727       Page 4              2000-25FF
9728-11263      Page 5              2600-2BFF
11264-12799     Page 6              2C00-31FF
12800-14335     Page 7              3200-37FF
14336-15871     Page 8              3800-3DFF
```

15872-16383	BASIC Language Initialization* Program Storage	3E00-3FFF
16384-17429	ML Foundation Core Assembly Language** Program Storage	4000-4415
17430-32767	General Assembly Language*** Program Storage	4416-7FFF
	SYSTEM UPPER RAM BANK:	
32768-65279	Assembly Language*** Program Storage	8000-FEFF
	SYSTEM UPPER ROM BANK:	
32768-40959	Extended Color BASIC	8000-9FFF
40960-49151	Color BASIC	A000-BFFF
49152-57343	Disk Basic or Cartridge Memory	C000-DFFF
57344-65279	Super-Extended (Enhanced) Basic	E000-FEFF
	Hardware Registers, I/O, and Interrupt Vectors:	
65280-65535	Registers & Vectors	FF00-FFFF

This arrangement will allow a total of 48,896 (0xBF00) bytes of RAM for Assembly Language Program Storage.

* The BASIC Language Initialization Program Storage is intended to be used for a (very short) BASIC Program which will simply setup a Run Location variable and then jump into the ML Foundation Core. The BASIC Language Control Programs, presented herein during Core Routine testing, are all examples of such Initialization Programs.

** The general initialization process is for the BASIC Program to POKE a Run Address into REGPC at $400A and then EXEC the Core Startup Code at STRTUP at $4403. That Startup Code then sets up the new Low RAM interrupt routines, sets ALLRAM Mode, puts the user stack at the top of High RAM, and jumps to the Run Location Address provided by the BASIC Program.

*** This memory is for additional ML Foundation code and for user code.

Note that the "ALLRAM" Mode discussed herein is different from the similarly-named mode in other CoCo-related documents. In those documents, the ROM is copied into High RAM. In this system, the entirety of the High RAM remains available for ML Foundation and user routines. Any necessary access to ROM routines is accomplished via bank-switching.

=====

REGXFR: Transfer Variables

So. To begin.

In developing and testing the ML Foundation System, the first thing we need to be able to do is to transfer information between the Assembly Language Routines to be developed and tested, and the BASIC Language Control Programs being used to test those routines.

To provide that capability, I devised a set of Register Variables in memory where BASIC Programs can POKE bytes to be used by the Assembly Language routines, and from which BASIC Programs can PEEK bytes returned from those Routines.

From the Assembly Language side, these memory locations will simply be accessed via suitable LD and ST instructions.

We establish these Transfer Variables as follows:

```
00100 *****
00110 *
00120 * REGXFR.ASM
00130 * MDJ 2023/01/17
00140 *
00150 * REGISTER TRANSFER
00160 * VARIABLES
00170 *
00180 * THIS ROUTINE IS USED
00190 * FOR TRANSFERRING
00200 * REGISTER VALUES
00210 * BETWEEN BASIC AND
00220 * ASSEMBLY LANGUAGE
00230 * PROGRAMS
00240 *
00250 * REGA = REGD HIGH BYTE
00260 * REGB = REGD LOW BYTE
00270 *
00280 * THUS REGD = REGA:REGB
00290 * AND REGDH = REGA
00300 *     REGDL = REGB
00310 *
00320 * AS THE 6809 IS A
00330 * BIG-ENDIAN MACHINE
00340 * THE LOWER ADDRESS
00350 * OF A TWO-BYTE
00360 * REGISTER IS THE MOST
00370 * SIGNIFICANT BYTE
```

```
                    00380 *
                    00390 *****
                    00400
                    00410 * ALTERNATE LABELS
          4000      00420 REGD     EQU     $4000
          4000      00430 REGDH    EQU     $4000
          4001      00440 REGDL    EQU     $4001
          4002      00450 REGX     EQU     $4002
          4004      00460 REGY     EQU     $4004
          4006      00470 REGS     EQU     $4006
          4008      00480 REGU     EQU     $4008
          400A      00490 REGPC    EQU     $400A
                    00500
4000                00510          ORG     $4000
                    00520
4000                00530 REGA     RMB     1
4001                00540 REGB     RMB     1
4002                00550 REGXH    RMB     1
4003                00560 REGXL    RMB     1
4004                00570 REGYH    RMB     1
4005                00580 REGYL    RMB     1
4006                00590 REGSH    RMB     1
4007                00600 REGSL    RMB     1
4008                00610 REGUH    RMB     1
4009                00620 REGUL    RMB     1
400A                00630 REGPCH   RMB     1
400B                00640 REGPCL   RMB     1
400C                00650 REGDP    RMB     1
400D                00660 REGCC    RMB     1
                    00670
          0000      00680          END
```

———

For example, to transfer a 16-bit value **N1** from a BASIC program to the D Register for use by an Assembly Language routine, and to then get the Assembly Language Routine's result from the X Register, the BASIC program would do something like:

```
100 'N1 = 16-BIT NUMBER TO BE
110 'TRANSFERRED TO REGISTER D
120 '
130 'N2 = HIGH BYTE
140 N2 = INT(N1 / 256)
150 '
160 'N3 = LOW BYTE
170 N3 = INT(N1 - (N2 * 256))
```

```
180 `
190 `REGISTER D TRANSFER
200 `VARIABLE ADDRESS =
210 `&H4000:&H4001
220 POKE &H4000, N2
230 POKE &H40O1, N3
240 `
250 ` GO DO ASSEMBLY LANGUAGE
260 ` ROUTINE
270 EXEC &H7000
280 `
290 `REGISTER X TRANSFER
300 `VARIABLE ADDRESS =
310 `&H4002:&H40O3
320 N2 = PEEK(&H40O2) `HIGH BYTE
330 N3 = PEEK(&H4003) `LOW BYTE
340 `
350 ` N1 = 16-BIT RESULT
360 N1 = (N2 * 256) + N3
370 PRINT N1
32767 END
```

Meanwhile, the Assembly Language routine would do something like:

```
00100 REGD   EQU       $4000
00110 REGX   EQU       $4002
00120
00130        ORG       $7000
00140        PSHS      A,B,X
00150
00160 *GET VALUE INTO D
00170 *FROM TRANSFER VARIABLE
00180        LDD       REGD
00190
00200 *DO SOME PROCESSING
00210
00220 *PUT VALUE FROM X
00230 *INTO TRANSFER VARIABLE
00240        STX       REGX
00250
00260 *EXIT
00270        PULS      A,B,X
00280        RTS
00290        END
```

As an actual test of the Transfer Variables, we’ll use the following Assembly Language Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0001.ASM
                   00130 * MDJ 2023/02/10
                   00140 *
                   00150 * REGXFR TEST
                   00160 *
                   00170 *****
                   00180
                   00190 * TRANSFER VARIABLES
          4000     00200 REGA    EQU     $4000
          4001     00210 REGB    EQU     $4001
          4002     00220 REGX    EQU     $4002
          4004     00230 REGY    EQU     $4004
          400A     00240 REGPC   EQU     $400A
                   00250
7000               00260         ORG     $7000
                   00270
7000 34   36       00280         PSHS    A,B,X,Y
7002 B6   4000     00290         LDA     REGA
7005 4C            00300         INCA
7006 4C            00310         INCA
7007 B7   4000     00320         STA     REGA
700A F6   4001     00330         LDB     REGB
700D 5C            00340         INCB
700E F7   4001     00350         STB     REGB
7011 BE   4002     00360         LDX     REGX
7014 30   05       00370         LEAX    5,X
7016 BF   4002     00380         STX     REGX
7019 10BE 4004     00390         LDY     REGY
701D 31   28       00400         LEAY    8,Y
701F 10BF 4004     00410         STY     REGY
                   00420
                   00430 * EXIT
7023 35   36       00440         PULS    A,B,X,Y
7025 39            00450         RTS
          0000     00460         END
```

along with the following BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0001.BAS
```

```
1030 '* MDJ 2023/02/10
1040 '*
1050 '* REGXFR 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0001.BIN"
1330 '

2000 'REFERENCE THE
2010 'TRANSFER VARIABLES
2020 R0 = &H4000 'REGA
2030 R1 = &H4001 'REGB
2040 R2 = &H4002 'REGXH
2050 R3 = &H4003 'REGXL
2060 R4 = &H4004 'REGYH
2070 R5 = &H4005 'REGYL
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 '

4000 'BASIC PREAMBLE
4010 A = 82
4020 B = 27
4030 X = &H1023 ' 4131 DECIMAL
```

```
4040 Y = &HAC37 '44087 DECIMAL
4050 PRINT "ON ENTRY:"
4060 PRINT "  A = "; A
4070 PRINT "  B = "; B
4080 PRINT "  X = "; X
4090 PRINT "  Y = "; Y
4100 '

5000 'TRANSFER DATA TO
5010 'REGXFR REGISTERS
5020 POKE R0, A
5030 POKE R1, B
5040 X1 = INT(X/256)
5050 X2 = INT(X-(X1*256))
5060 POKE R2, X1
5070 POKE R3, X2
5080 Y1 = INT(Y/256)
5090 Y2 = INT(Y-(Y1*256))
5100 POKE R4, Y1
5110 POKE R5, Y2
5120 '

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

7000 'TRANSFER DATA FROM
7010 'REGXFR REGISTERS
7020 A = PEEK(R0)
7030 B = PEEK(R1)
7040 X1 = PEEK(R2)
7050 X2 = PEEK(R3)
7060 X = INT((X1*256)+X2)
7070 Y1 = PEEK(R4)
7080 Y2 = PEEK(R5)
7090 Y = INT((Y1*256)+Y2)
7100 '

8000 'BASIC POSTAMBLE
8010 PRINT "ON EXIT:"
8020 PRINT "  A = "; A
8030 PRINT "  B = "; B
8040 PRINT "  X = "; X
8050 PRINT "  Y = "; Y
8060 '
```

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

32767 END
```

—

Results:

```
ON ENTRY:
  A =  82
  B =  27
  X =  4131
  Y =  44087
ON EXIT:
  A =  84
  B =  28
  X =  4136
  Y =  44095
```

____

The results are as expected.

=====

VIDCLS: Clear the VIDRAM Screen

Once we're able to transfer information back-and-forth between BASIC Programs and Assembly Language Routines, the next thing to develop is the ability to display information on the screen directly from Machine Language.

In order to establish that ability, the first step is to be able to clear the screen to make way for the display of new data. The following Assembly Language routine accomplishes that task. Recall that [MDJ01] noted that POKE Code 96 does the same thing as ASCII Print Code 32; it produces a plain green blank character.

Because I'm writing this system as a collection of small modules, within and distinct to each module I'm using labels like LBL001, LBL002, … , LBLnnn as local labels at locations where no external reference is anticipated.

.

```
                00100 *****
                00110 *
                00120 * VIDCLS.ASM
                00130 * MDJ 2023/01/17
                00140 *
                00150 * CLEARS THE
                00160 * VIDEO RAM
                00170 * AT &H0400 TO &H05FF
                00180 * TO ALL BYTES = $60
                00190 * I.E. ALL GREEN BLANKS
                00200 * I.E. 96 DECIMAL
                00210 *
                00220 * ENTRY CONDITIONS:
                00230 * NONE
                00240 *
                00250 * EXIT CONDITIONS
                00260 * NONE
                00270 *
                00280 *****
                00290
                00300 * FIRST BYTE OF VIDRAM
0400            00310 VIDRAM  EQU     $0400
                00320
                00330 * ONE BYTE PAST THE
                00340 * LAST BYTE OF VIDRAM
0600            00350 VIDEND  EQU     $0600
                00360
                00370 * GREEN BLANK CHAR CODE
0060            00380 CHR60   EQU     $60
```

```
                    00390
400E                00400          ORG     $400E
                    00410
400E 34   12        00420 VIDCLS   PSHS    A,X
4010 86   60        00430          LDA     #CHR60
4012 8E   0400      00440          LDX     #VIDRAM
4015 A7   80        00450 LBL001   STA     ,X+
4017 8C   0600      00460          CMPX    #VIDEND
401A 26   F9        00470          BNE     LBL001
                    00480
                    00490 * EXIT
401C 35   12        00500          PULS    A,X
401E 39             00510          RTS
                    00520
          0000      00530          END
```

——-

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0002.ASM
                    00130 * MDJ 2023/02/10
                    00140 *
                    00150 * VIDCLS TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * ML FOUNDATION
                    00200 * CORE ADDRESS
          400E      00210 VIDCLS   EQU     $400E
                    00220
7000                00230          ORG     $7000
                    00240
                    00250 * VIDCLS TEST
                    00260
7000 34   02        00270          PSHS    A
                    00280
7002 BD   400E      00290          JSR     VIDCLS
                    00300
                    00310 * HOLD THE SCREEN
7005 20   FE        00320 LBL001   BRA     LBL001
                    00330
7007 35   02        00340          PULS    A
7009 39             00350          RTS
```

```
              00360
     0000     00370            END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0002.BAS
1030 '* MDJ 2023/02/10
1040 '*
1050 '* VIDCLS 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0002.BIN"
1330 '

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
9010 PRINT " MEM = ";MEM
9020 PRINT "FREE = ";FREE(0)

32767 END
```

—

The Result is as expected: a completely blank green screen.

=====

PUTCHR: Put a Character To the VIDRAM Screen At a Specific Position

To continue with the development of our ability to display information on the screen, we have here a little routine that simply puts a character to a specified location in **`VIDRAM`**. This is perhaps the simplest routine we'll ever have to write during this coding adventure.

`PUTCHR` does not advance the cursor, and it provides no mechanism for scrolling the screen if necessary. To incorporate those provisions, use **`PUTCHA`** instead.

Also note that PUTCHR and PUTCHA may not produce exactly the results you might expect, because they use Print Codes instead of Poke Codes, cf. [MDJ01]. For example, instead of a blank, all-green character for ASCII Code 32 ($20 = Space), PUTCHR produces a blank, all-black character. For normal output, use PRTCHR and PRTCHA instead of PUTCHR and PUTCHA.

```
                    00100 *****
                    00110 *
                    00120 * PUTCHR.ASM
                    00130 * MDJ 2023/01/17
                    00140 *
                    00150 * PUT A CHARACTER CODE
                    00160 * TO THE VIDEO RAM
                    00170 *
                    00180 * ENTRY CONDITIONS:
                    00190 * A = CHARACTER CODE
                    00200 * X = SCREEN LOCATION
                    00210 * ($0400 - $05FF)
                    00220 *
                    00230 * EXIT CONDITIONS:
                    00240 * NONE
                    00250 *
                    00260 *****
                    00270
401F                00280         ORG     $401F
                    00290
401F A7   84        00300 PUTCHR  STA     ,X
                    00310
                    00320 * EXIT
4021 39             00330         RTS
                    00340
          0000      00350         END
```

I'll delay testing this routine until the next section.

=====

GETCHR: Get a Character From a Specific Position on the VIDRAM Screen

Although a form of input rather than output, the **GETCHR.ASM** routine is a mirror of the simple **PUTCHR.ASM** routine. It simply retrieves a character code from a specified location in **VIDRAM**..

```
                    00100 *****
                    00110 *
                    00120 * GETCHR.ASM
                    00130 * MDJ 2023/01/17
                    00140 *
                    00150 * GET THE CHARACTER CODE
                    00160 * FROM A SPECIFIED
                    00170 * LOCATION
                    00180 * IN VIDEO RAM
                    00190 *
                    00200 * ENTRY CONDITIONS:
                    00210 * X = SCREEN LOCATION
                    00220 * ($0400 - $05FF)
                    00230 *
                    00240 * EXIT CONDITIONS:
                    00250 * A = CHARACTER CODE
                    00260 *
                    00270 *****
                    00280
4022                00290         ORG     $4022
                    00300
4022 A6   84        00310 GETCHR  LDA     ,X
                    00320
                    00330 * EXIT
4024 39             00340         RTS
          0000      00350         END
```

——-

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0003.ASM
                    00130 * MDJ 2023/02/11
                    00140 *
```

```
                    00150 * PUTCHR/GETCHR
                    00160 * + HOLD TEST
                    00170 *
                    00180 * CLEARS THE SCREEN
                    00190 * THEN PUTS CHAR CODES
                    00200 * $00 THROUGH $FF
                    00210 * (000-255 DECIMAL)
                    00220 * TO VIDRAM, THEN GETS
                    00230 * THE CHARACTER AT $044D
                    00240 * (=M) AND REPORTS IT
                    00250 * TO THE SCREEN AND THEN
                    00260 * HOLDS THE SCREEN
                    00270 *
                    00280 *****
                    00290
                    00300 * SCREEN ADDRESS
          0400      00310 VIDRAM  EQU     $0400
                    00320
                    00330 * ML FOUNDATION
                    00340 * CORE ADDRESSES
          400E      00350 VIDCLS  EQU     $400E
          401F      00360 PUTCHR  EQU     $401F
          4022      00370 GETCHR  EQU     $4022
                    00380
7000                00390         ORG     $7000
                    00400
                    00410 * PUTCHR/GETCHR TEST
                    00420
7000 34   12        00430         PSHS    A,X
                    00440
                    00450 * CLEAR THE SCREEN
7002 BD   400E      00460         JSR     VIDCLS
                    00470
                    00480 * LOAD THE FIRST
                    00490 * CHAR CODE
7005 86   00        00500         LDA     #$00
                    00510
                    00520 * LOAD THE SCREEN
                    00530 * ADDRESS
7007 8E   0400      00540         LDX     #VIDRAM
                    00550
                    00560 * FILL THE FIRST PART OF
                    00570 * THE SCREEN WITH THE
                    00580 * CHARACTER SET
700A BD   401F      00590 LBL001  JSR     PUTCHR
700D 4C             00600         INCA
700E 30   01        00610         LEAX    1,X
```

```
7010 81   FF       00620          CMPA    #$FF
7012 26   F6       00630          BNE     LBL001
                   00640
                   00650 * POINT TO CHARACTER M'S
                   00660 * ADDRESS IN VIDRAM
7014 8E   044D     00670          LDX     #$044D
                   00680
                   00690 * GET THE CHARACTER M
                   00700 * FROM THE SCREEN
7017 BD   4022     00710          JSR     GETCHR
                   00720
                   00730 * POINT FURTHER DOWN
                   00740 * ON THE SCREEN
                   00750 * I.E. IN VIDRAM
701A 8E   0580     00760          LDX     #$0580
                   00770
                   00780 * PUT THE M TO THE
                   00790 * SCREEN
701D BD   401F     00800         JSR     PUTCHR
                   00810
                   00820 * HOLD THE SCREEN
7020 20   FE       00830 LBL002  BRA     LBL002
                   00840
                   00850 * EXIT
7022 35   12       00860          PULS    A,X
7024 39            00870          RTS
                   00880
          0000     00890          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0003.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* PUTCHR/GETCHR 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0003.BIN"
1330 '

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
9010 PRINT " MEM = ";MEM
9020 PRINT "FREE = ";FREE(0)

32767 END
```

Result:

As expected.

Also note that, except for the reported M at the lower left side of the screen, this is the same result you would get if you ran the following in BASIC:

```
1000 CLS
1010 CD = &H00
1010 FOR AD = &H0400 TO &H04FF
1020 POKE AD, CD
1030 CD = CD + 1
1040 NEXT AD
1050 GOTO 1050
32767 END
```

=====

PUTBYT: Put an 8-bit Number To the VIDRAM Screen As Two Hexadecimal Digits At a Specific Position

Now that we're able to put a character into the **VIDRAM**, its pretty much just a mechanical task to put any text anywhere on the screen. But, it's also important to be able to put numbers to the screen as well.

For the moment, I'm going to concentrate on working with unsigned integers only. And, I'm going to restrict myself to only displaying those integers in hexadecimal format. The following **PUTBYT.ASM** Routine places an 8-bit number (byte) on the screen in the form of two consecutive hexadecimal digits, i.e. from "**00**" to "**FF**".

After that, displaying 16-bit, 32-bit, 64-bit integers, etc. is simply a matter of processing one byte after another with this same routine.

PUTBYT does not advance the cursor, and it provides no mechanism for scrolling the screen if necessary. To incorporate those provisions, use **PUTBYA** instead. .

```
00100 *****
00110 *
00120 * PUTBYT.ASM
00130 * MDJ 2023/01/24
00140 *
00150 * PUTS AN 8-BIT NUMBER
00160 * TO VIDRAM AS TWO
00170 * HEXADECIMAL DIGITS
00180 *
00190 * ENTRY CONDITIONS:
00200 * A = THE 8-BIT NUMBER
00210 * X = SCREEN LOCATION
00220 * ($0400 - $05FE)
00230 * CANNOT BE $05FF
00240 * BECAUSE NEED
00250 * ROOM TO PUT
00260 * 2 CHARACTERS
00270 *
00280 * EXIT CONDITIONS:
00290 * X = NEW SCREEN LOC
00300 * ($0402 - $0600)
```

```
                    00310 * $0600 INDICATES
                    00320 * END OF VIDRAM
                    00330 * HAS BEEN PASSED
                    00340 *
                    00350 *****
                    00360
                    00370 * SCRATCHPAD VARIABLES
                    00380 * THE 8-BIT NUMBER
          0076      00390 L0076   EQU     $0076
                    00400
                    00410 * THE HIGH NIBBLE
          0077      00420 L0077   EQU     $0077
                    00430
                    00440 * THE LOW NIBBLE
          00F3      00450 L00F3   EQU     $00F3
                    00460
                    00470 * EXTERNAL ROUTINE
                    00480 * ADDRESS
          401F      00490 PUTCHR  EQU     $401F
                    00500
4025                00510         ORG     $4025
                    00520
                    00530 * SAVE THE NUMBER
4025 97   76        00540 PUTBYT  STA     L0076
                    00550
                    00560 * DIVIDE BY 16
4027 44             00570         LSRA
4028 44             00580         LSRA
4029 44             00590         LSRA
402A 44             00600         LSRA
                    00610
                    00620 * SAVE THE HIGH NIBBLE
402B 97   77        00630         STA L0077
                    00640
                    00650 * MULTIPLY BY 16
402D 48             00660         LSLA
402E 48             00670         LSLA
402F 48             00680         LSLA
4030 48             00690         LSLA
                    00700
                    00710 * SAVE TEMP RESULT
4031 97   F3        00720         STA     L00F3
                    00730
                    00740 * GET THE NUMBER AGAIN
4033 96   76        00750         LDA     L0076
                    00760
                    00770 * SUBTRACT TEMP RESULT
```

```
4035 90   F3        00780         SUBA    L00F3
                    00790
                    00800 * SAVE LOW NIBBLE
4037 97   F3        00810         STA     L00F3
                    00820
                    00830 * IS LOW NIBBLE <= 9
4039 81   09        00840         CMPA    #9
                    00850
                    00860 * GO IF NO
403B 22   04        00870         BHI     LBL001
                    00880
                    00890 * ADD ZERO OFFSET
403D 8B   70        00900         ADDA    #112
403F 20   02        00910         BRA     LBL002
                    00920
                    00930 * ADD "A" OFFSET
4041 8B   37        00940 LBL001  ADDA    #55
                    00950
                    00960 * SAVE LOW NIBBLE CHAR
4043 97   F3        00970 LBL002  STA     L00F3
                    00980
                    00990 * GET HIGH NIBBLE
4045 96   77        01000         LDA     L0077
                    01010
                    01020 * IS HIGH NIBBLE <= 9
4047 81   09        01030         CMPA    #9
                    01040
                    01050 * GO IF NO
4049 22   04        01060         BHI     LBL003
                    01070
                    01080 * ADD ZERO OFFSET
404B 8B   70        01090         ADDA    #112
404D 20   02        01100         BRA     LBL004
                    01110
                    01120 * ADD "A" OFFSET
404F 8B   37        01130 LBL003  ADDA    #55
                    01140
                    01150 * PUT HIGH NIBBLE CHAR
                    01160 * TO VIDRAM
4051 BD   401F      01170 LBL004  JSR     PUTCHR
                    01180
                    01190 * INCREMENT VIDRAM PTR
4054 30   01        01200         LEAX    1,X
                    01210
                    01220 * GET LOW NIBBLE CHAR
4056 96   F3        01230         LDA     L00F3
                    01240
```

```
                    01250 * PUT LOW NIBBLE CHAR
                    01260 * TO VIDRAM
4058 BD   401F      01270        JSR    PUTCHR
                    01280
                    01290 * INCREMENT VIDRAM PTR
405B 30   01        01300        LEAX    1,X
                    01310
                    01320 * EXIT
405D 39             01330        RTS
                    01340
          0000      01350        END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0004.ASM
                    00130 * MDJ 2023/02/11
                    00140 *
                    00150 * PUTBYT TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * SCREEN ADDRESSES
          0400      00200 VIDRAM  EQU     $0400
          0600      00210 VIDEND  EQU     $0600
                    00220
                    00230 * ML FOUNDATION
                    00240 * CORE ADDRESSES
          400E      00250 VIDCLS  EQU     $400E
          4025      00260 PUTBYT  EQU     $4025
                    00270
7000                00280         ORG     $7000
                    00290
                    00300 * PUTBYT TEST
                    00310
7000 34   16        00320         PSHS    A,B,X
                    00330
                    00340 * CLEAR THE SCREEN
7002 BD   400E      00350         JSR     VIDCLS
                    00360
                    00370 * LOAD THE FIRST
                    00380 * BYTE VALUE
7005 86   00        00390         LDA     #$00
                    00400
```

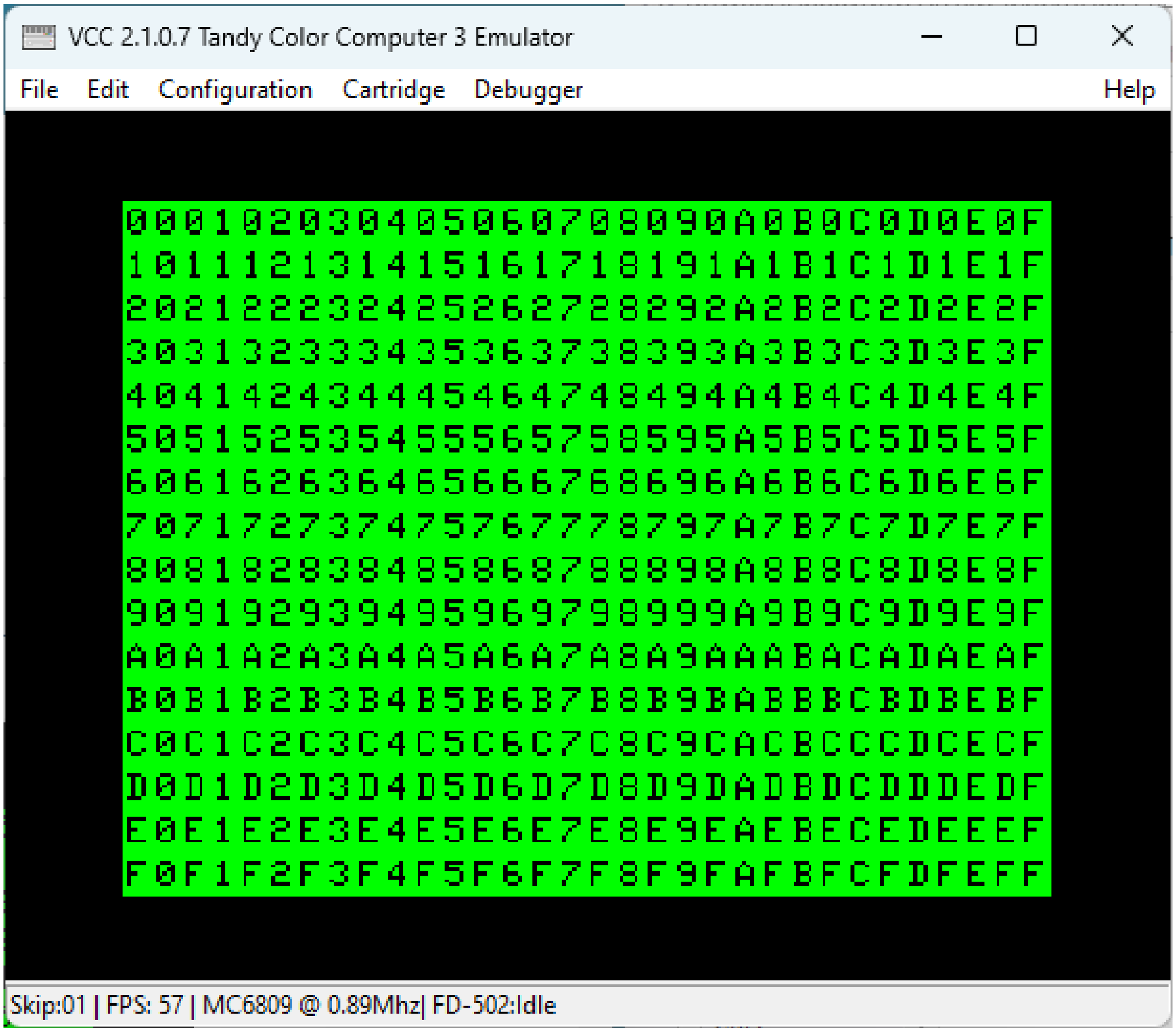
```
                    00410 * LOAD THE SCREEN
                    00420 * ADDRESS
7007 8E   0400      00430         LDX     #VIDRAM
                    00440
                    00450 * SAVE THE BYTE VALUE
700A 1F   89        00460         TFR     A,B
                    00470
                    00480 * GO PUT BYTE TO SCREEN
700C BD   4025      00490 LBL001  JSR     PUTBYT
                    00500
                    00510 * ARE WE DONE?
700F 8C   0600      00520         CMPX    #VIDEND
                    00530
                    00540 * GO IF YES
7012 24   05        00550         BHS     LBL002
                    00560
                    00570  * GET NEXT BYTE VALUE
7014 5C             00580         INCB
7015 1F   98        00590         TFR     B,A
7017 20   F3        00600         BRA     LBL001
                    00610
                    00620 * HOLD THE SCREEN
7019 20   FE        00630 LBL002  BRA     LBL002
                    00640
                    00650 * EXIT
701B 35   16        00660         PULS    A,B,X
701D 39             00670         RTS
                    00680
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0004.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* PUTBYT 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0004.BIN"
1330 '

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
9010 PRINT " MEM = ";MEM
9020 PRINT "FREE = ";FREE(0)

32767 END
```

Result:



As expected.

=====

SCROLL: Scroll the VIDRAM Screen

I expect that much of what I'm planning to do with the ML Foundation will involve putting information to specific locations in **VIDRAM**, rather than a continuous scrolling output.

Nonetheless, there may be times and applications where the ability to scroll the screen will be useful. And this seems like a convenient spot to cover scrolling. So….

```
                    00100 *****
                    00110 *
                    00120 * SCROLL.ASM
                    00130 * MDJ 2023/01/17
                    00140 *
                    00150 * SCROLLS THE SCREEN
                    00160 *
                    00170 * ENTRY CONDITIONS
                    00180 * NONE
                    00190 *
                    00200 * EXIT CONDITIONS
                    00210 * NONE
                    00220 *
                    00230 *****
                    00240
                    00250 * SCREEN ADDRESSES
          0400      00260 VIDRAM EQU $0400
          0600      00270 VIDEND EQU $0600
          0420      00280 VIDL01 EQU $0420
                    00290
405E                00300         ORG     $405E
                    00310
405E 34   32        00320 SCROLL  PSHS    A,X,Y
                    00330
                    00340 * Y = SOURCE POINTER
                    00350 * X = TARGET POINTER
                    00360
                    00370 * POINT X TO FIRST LINE
4060 8E   0400      00380         LDX     #VIDRAM
                    00390
                    00400 * POINT Y TO SECOND LINE
4063 108E 0420      00410         LDY     #VIDL01
                    00420
                    00430 * SCROLL THE SCREEN
4067 A6   A0        00440 LBL001  LDA     ,Y+
4069 A7   80        00450         STA     ,X+
                    00460
```

```
                    00470 * ARE WE DONE?
406B 108C 0600      00480         CMPY    #VIDEND
                    00490
                    00500 * GO IF NO
406F 25   F6        00510         BLO     LBL001
                    00520
                    00530 * CLEAR THE LAST LINE
                    00540 * LOAD BLANK GREEN CHAR
4071 86   60        00550         LDA     #96
4073 A7   80        00560 LBL002  STA     ,X+
                    00570
                    00580 * ARE WE DONE?
4075 8C   0600      00590         CMPX    #VIDEND
                    00600
                    00610 * GO IF NO
4078 25   F9        00620         BLO     LBL002
                    00630
                    00640 * EXIT
407A 35   32        00650         PULS    A,X,Y
407C 39             00660         RTS
                    00670
          0000      00680         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0005.ASM
                    00130 * MDJ 2023/02/11
                    00140 *
                    00150 * SCROLL TEST
                    00160 *
                    00170 * PUTS ALL BYTES
                    00180 * ($00 - $FF) TO THE
                    00190 * SCREEN, WITHOUT ANY
                    00200 * SPACES, THEN SCROLLS
                    00210 * THE SCREEN ONE LINE,
                    00220 * AND THEN HOLDS THE
                    00230 * SCREEN
                    00240 *
                    00250 *****
                    00260
                    00270 * SCREEN ADDRESSES
          0400      00280 VIDRAM  EQU     $0400
          0600      00290 VIDEND  EQU     $0600
```

```
                  00300
                  00310 * ML FOUNDATION
                  00320 * CORE ADDRESSES
          400E    00330 VIDCLS  EQU     $400E
          4025    00340 PUTBYT  EQU     $4025
          405E    00350 SCROLL  EQU     $405E
                  00360
7000              00370         ORG     $7000
                  00380
                  00390 * SCROLL TEST
                  00400
7000 34   16      00410         PSHS    A,B,X
                  00420
                  00430 * CLEAR THE SCREEN
7002 BD   400E    00440         JSR     VIDCLS
                  00450
                  00460 * LOAD THE FIRST
                  00470 * BYTE VALUE
7005 86   00      00480         LDA     #$00
                  00490
                  00500 * LOAD THE SCREEN
                  00510 * ADDRESS
7007 8E   0400    00520         LDX     #VIDRAM
                  00530
                  00540 * SAVE THE BYTE VALUE
700A 1F   89      00550         TFR     A,B
                  00560
                  00570 * GO PUT BYTE TO SCREEN
700C BD   4025    00580 LBL001  JSR     PUTBYT
                  00590
                  00600 * ARE WE DONE?
700F 8C   0600    00610         CMPX    #VIDEND
                  00620
                  00630 * GO IF YES
7012 24   05      00640         BHS     LBL002
                  00650
                  00660 * GET NEXT BYTE VALUE
7014 5C           00670         INCB
7015 1F   98      00680         TFR     B,A
7017 20   F3      00690         BRA     LBL001
                  00700
                  00710 * SCROLL THE SCREEN
                  00720 * ONE LINE
7019 BD   405E    00730 LBL002  JSR     SCROLL
                  00740
                  00750 * HOLD THE SCREEN
701C 20   FE      00760 LBL003  BRA     LBL003
```

```
701E 35   16                PULS    A,B,X
7020 39                     RTS

          0000              END
```

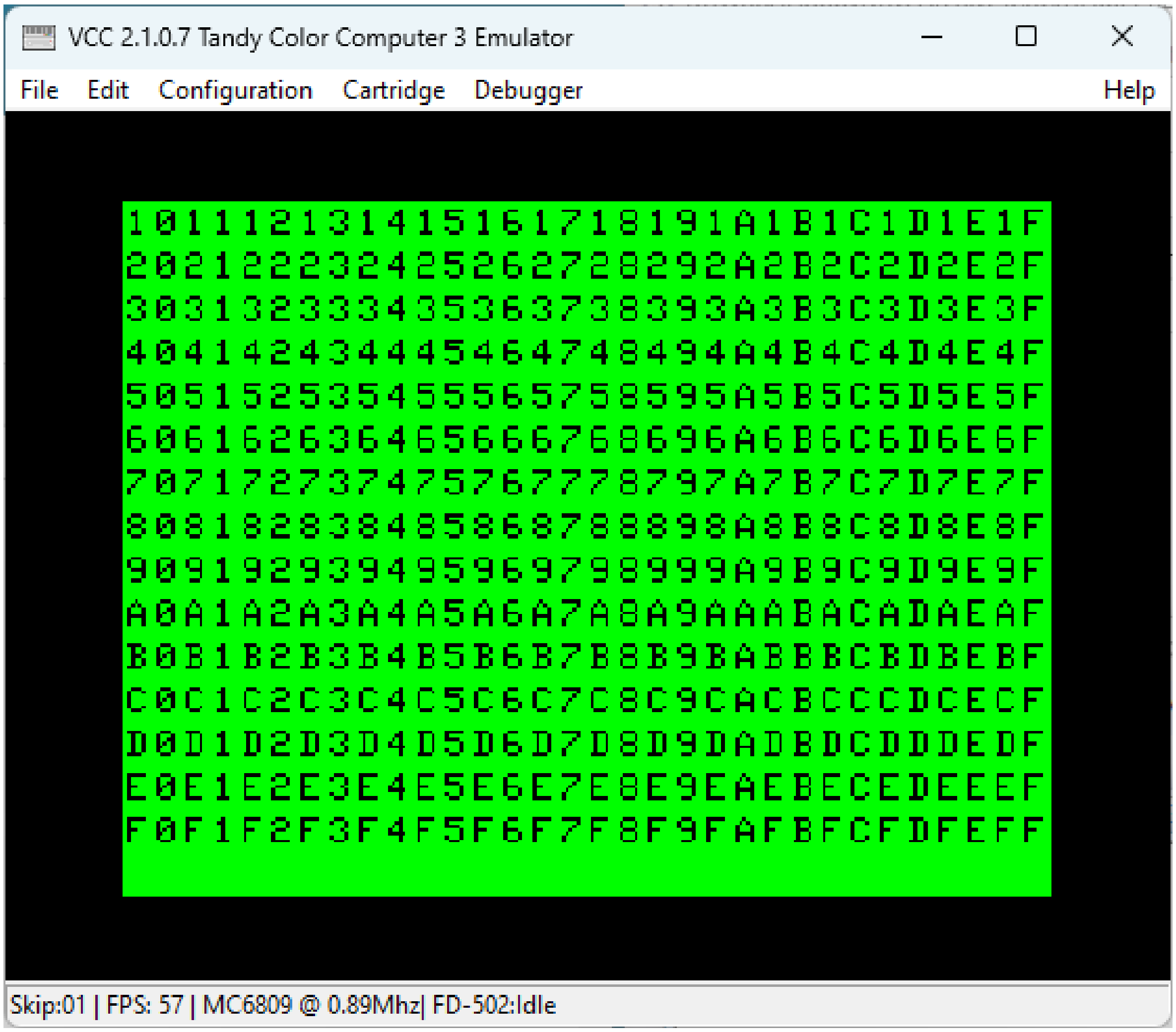
———

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0005.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* SCROLL 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0005.BIN"
1330 '

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
9010 PRINT " MEM = ";MEM
9020 PRINT "FREE = ";FREE(0)

32767 END
```

Result:



As expected.

=====

PUTCHA: Put a Character To the VIDRAM Screen at the Cursor Position and Advance the Cursor

This routine is designed to function with continuous scrolling output.

PUTCHA advances the cursor, and it also scrolls the screen when required. If those provisions are not necessary for a particular application, use **PUTCHR** instead: it uses less resources..

```
            *****
            *
            * PUTCHA.ASM
            * MDJ 2023/01/24
            *
            * PUT A CHARACTER CODE
            * TO THE VIDEO RAM
            * AT THE CURSOR POSITION
            * AND ADVANCE THE CURSOR
            *
            * SCROLLS THE SCREEN
            * IF REQUIRED
            *
            * ENTRY CONDITIONS:
            * A = CHARACTER CODE
            *
            * EXIT CONDITIONS:
            * NONE
            *
            *****

            * LOW RAM CURSOR ADDRESS
0088        CURPOS  EQU     $0088

            * START OF THE LAST
            * LINE OF VIDRAM
05E0        VIDL15  EQU     $05E0

            * ONE BYTE PAST THE
            * END OF VIDRAM
0600        VIDEND  EQU     $0600

```

```
                    00420 * EXTERNAL ROUTINE
                    00430 * ADDRESS
          405E      00440 SCROLL  EQU     $405E
                    00450
407D                00460         ORG     $407D
                    00470
407D 34   12        00480 PUTCHA  PSHS    A,X
                    00490
                    00500 * GET THE CURSOR
407F 9E   88        00510         LDX     CURPOS
                    00520
                    00530 * IS PRE-SCROLL REQUIRED?
4081 8C   0600      00540         CMPX    #VIDEND
                    00550
                    00560 * GO IF NO
4084 25   06        00570         BLO     LBL001
                    00580
                    00590 * SCROLL ONE LINE
4086 BD   405E      00600         JSR     SCROLL
4089 8E   05E0      00610         LDX     #VIDL15
                    00620
                    00630 * PUT THE CHARACTER CODE
408C A7   80        00640 LBL001  STA     ,X+
                    00650
                    00660 * END OF VIDRAM?
408E 8C   0600      00670         CMPX    #VIDEND
                    00680
                    00690 * GO IF NO
4091 25   06        00700         BLO     LBL002
                    00710
                    00720 * SCROLL ONE LINE
4093 BD   405E      00730         JSR     SCROLL
4096 8E   05E0      00740         LDX     #VIDL15
                    00750
                    00760 * STORE NEW CURSOR
4099 9F   88        00770 LBL002  STX     CURPOS
                    00780
                    00790 * EXIT
409B 35   12        00800         PULS    A,X
409D 39             00810         RTS
                    00820
          0000      00830         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0006.ASM
                    00130 * MDJ 2023/02/11
                    00140 *
                    00150 * PUTCHA TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * ML FOUNDATION
                    00200 * CORE ADDRESS
          407D      00210 PUTCHA  EQU     $407D
          409E      00220 PUTBYA  EQU     $409E
7000                00230         ORG     $7000
                    00240
                    00250 * PUTCHA TEST
                    00260
7000 34   20        00270         PSHS    Y
                    00280
7002 20   0D        00290         BRA     LBL001
                    00300
                    00310 * SEE STRING-LINE NOTE
                    00320 * IN DISCUSSION BELOW
7004      4A        00330 STRING  FCC     /JESUS/
          45
          53
          55
          53
7009      60        00340         FCB     $60
700A      4C        00350         FCC     /LIVES/
          49
          56
          45
          53
700F      61        00360         FCB     $61
7010      00        00370         FCB     $00
                    00380
                    00390 * POINT TO THE STRING
7011 108E 7004      00400 LBL001  LDY     #STRING
                    00410
                    00420 * GET A CHARACTER
7015 A6   A0        00430 LBL002  LDA     ,Y+
                    00440
                    00450 * GO IF NULL TERMINATOR
7017 27   05        00460         BEQ     LBL003
```

```
                    00470
                    00480 * PUT IT TO VIDRAM
7019 BD   407D      00490         JSR     PUTCHA
701C 20   F7        00500         BRA     LBL002
                    00510
                    00520 * EXIT
701E 35   20        00530 LBL003  PULS    Y
7020 39             00540         RTS
                    00550
          0000      00560         END
```

String Line Note: The Lines 330-370 were originally:

```
7004      4A        00330 STRING  FCC     /JESUS LIVES!/
          45
          53
          55
          53
          20
          4C
          49
          56
          45
          53
          21
7010      00        00370         FCB     $00
```

The additions and rearrangements were necessary because, in Machine Language, we're doing essentially the same thing as the **BASIC POKE** mechanism here. We're inserting the character codes directly into **VIDRAM** memory rather than using the **PRINT** mechanism to display them.

The **PRINT** mechanism codes for a space ($20 = 032 decimal) and an exclamation point ($21 = 033 decimal) display as Reversed (i.e. green-on-black) instead of the Standard black-on-green when placed on the screen by the **POKE** mechanism instead (cf. MDJ01).

Thus, we have to substitute the **POKE** mechanism codes for the space ($60 = 096 decimal) and the exclamation point ($61 = 097 decimal) instead.

Later, we'll handle this sort of thing through the **PK2PRT** and **PRT2PK** Translation Routines.

____

The BASIC Language Control Program:

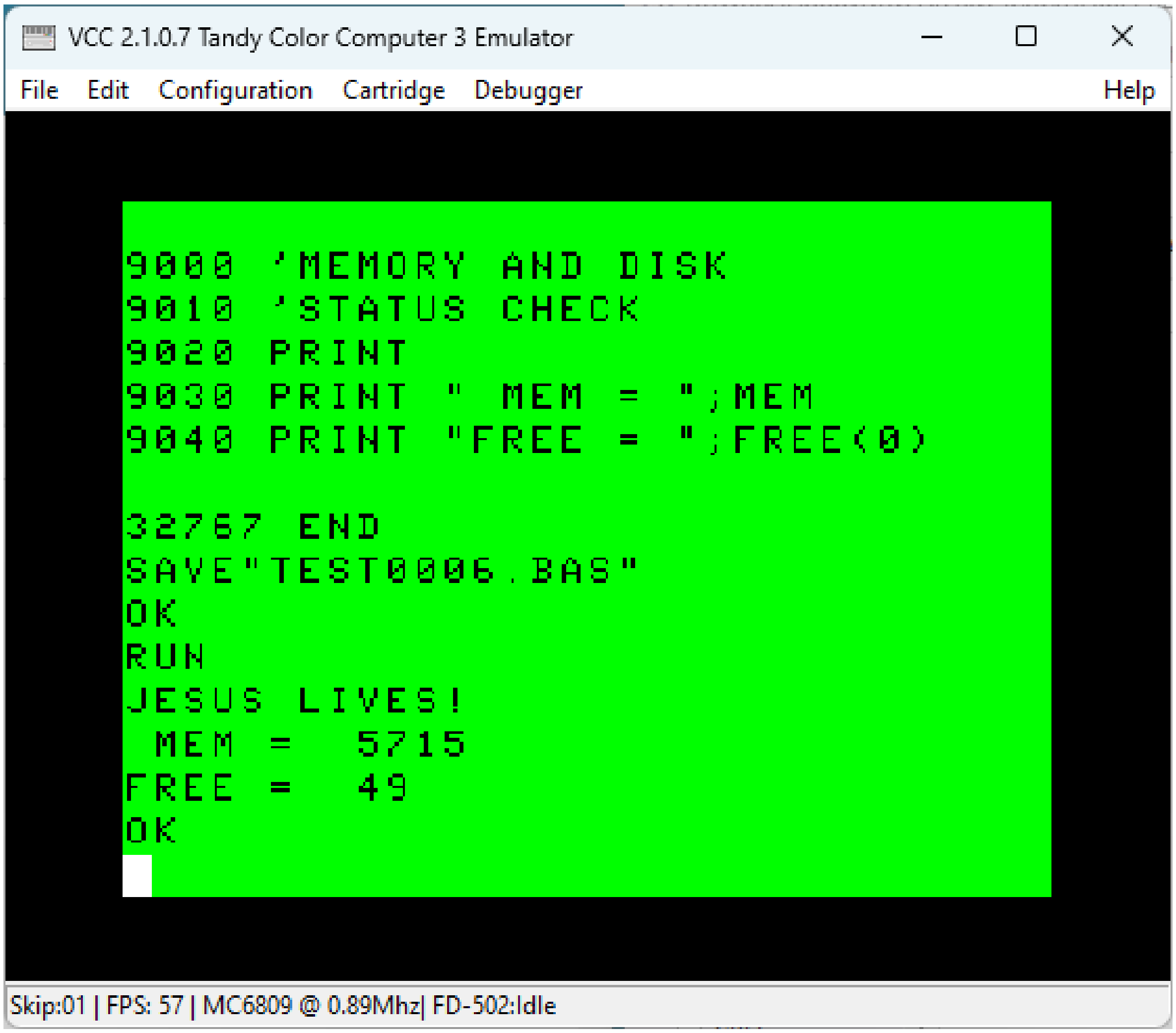
```
1000 '*****
1010 '*
```

```
1020 '* TEST0006.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* PUTCHA 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0006.BIN"
1330 '

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.

=====

PUTBYA: Put an 8-bit Number To the VIDRAM Screen As Two Hexadecimal Digits At the Cursor Position and Advance the Cursor

This routine is designed to function with continuous scrolling output.

PUTBYA advances the cursor, and it also scrolls the screen when required. If those provisions are not necessary for a particular application, use **PUTBYT** instead: it uses less resources.

```
      *****
      *
      * PUTBYA.ASM
      * MDJ 2023/01/24
      *
      * PUTS AN 8-BIT NUMBER
      * TO VIDRAM AS TWO
      * HEXADECIMAL DIGITS
      *
      * SCROLLS THE SCREEN
      * IF REQUIRED
      *
      * ENTRY CONDITIONS:
      * A = THE 8-BIT NUMBER
      *
      * EXIT CONDITIONS:
      * NONE
      *
      *****

      * SCRATCHPAD VARIABLES
      * THE 8-BIT NUMBER
0076  L0076   EQU     $0076

      * THE HIGH NIBBLE
0077  L0077   EQU     $0077

      * THE LOW NIBBLE
00F3  L00F3   EQU     $00F3

```

```
                   00400 * EXTERNAL ROUTINE
                   00410 * ADDRESS
          407D     00420 PUTCHA  EQU     $407D
                   00430
409E               00440         ORG     $409E
                   00450
409E 34   02       00460 PUTBYA  PSHS    A
                   00470
                   00480 * SAVE THE NUMBER
40A0 97   76       00490         STA     L0076
                   00500
                   00510 * DIVIDE BY 16
40A2 44            00520         LSRA
40A3 44            00530         LSRA
40A4 44            00540         LSRA
40A5 44            00550         LSRA
                   00560
                   00570 * SAVE THE HIGH NIBBLE
40A6 97   77       00580         STA     L0077
                   00590
                   00600 * MULTIPLY BY 16
40A8 48            00610         LSLA
40A9 48            00620         LSLA
40AA 48            00630         LSLA
40AB 48            00640         LSLA
                   00650
                   00660 * SAVE TEMP RESULT
40AC 97   F3       00670         STA     L00F3
                   00680
                   00690 * GET THE NUMBER AGAIN
40AE 96   76       00700         LDA     L0076
                   00710
                   00720 * SUBTRACT TEMP RESULT
40B0 90   F3       00730         SUBA    L00F3
                   00740
                   00750 * SAVE LOW NIBBLE
40B2 97   F3       00760         STA     L00F3
                   00770
                   00780 * IS LOW NIBBLE <= 9
40B4 81   09       00790         CMPA    #9
                   00800
                   00810 * GO IF NO
40B6 22   04       00820         BHI     LBL001
                   00830
                   00840 * ADD ZERO OFFSET
40B8 8B   70       00850         ADDA    #112
40BA 20   02       00860         BRA     LBL002
```

```
                        00870
                        00880 * ADD "A" OFFSET
40BC 8B    37           00890 LBL001  ADDA    #55
                        00900
                        00910 * SAVE LOW NIBBLE CHAR
40BE 97    F3           00920 LBL002  STA     L00F3
                        00930
                        00940 * GET HIGH NIBBLE
40C0 96    77           00950         LDA     L0077
                        00960
                        00970 * IS HIGH NIBBLE <= 9
40C2 81    09           00980         CMPA    #9
                        00990
                        01000 * GO IF NO
40C4 22    04           01010         BHI     LBL003
                        01020
                        01030 * ADD ZERO OFFSET
40C6 8B    70           01040         ADDA    #112
40C8 20    02           01050         BRA     LBL004
                        01060
                        01070 * ADD "A" OFFSET
40CA 8B    37           01080 LBL003  ADDA    #55
                        01090
                        01100 * PUT HIGH NIBBLE CHAR
                        01110 * TO VIDRAM
40CC BD    407D         01120 LBL004  JSR     PUTCHA
                        01130
                        01140 * GET LOW NIBBLE CHAR
40CF 96    F3           01150         LDA     L00F3
                        01160
                        01170 * PUT LOW NIBBLE CHAR
                        01180 * TO VIDRAM
40D1 BD    407D         01190         JSR     PUTCHA
                        01200
                        01210 * EXIT
40D4 35    02           01220         PULS    A
40D6 39                 01230         RTS
                        01240
           0000         01250         END
```

The Assembly Language Test Routine:

```
                        00100 *****
                        00110 *
                        00120 * TEST0007.ASM
```

```
                  00130 * MDJ 2023/02/11
                  00140 *
                  00150 * PUTBYA TEST
                  00160 *
                  00170 * PUTS 128 BYTES
                  00180 * ($00 - $7F) TO THE
                  00190 * SCREEN, BEGINNING AT
                  00200 * THE CURRENT CURSOR
                  00210 * POSITION
                  00220 *
                  00230 *****
                  00240
                  00250 * ML FOUNDATION
                  00260 * CORE ADDRESS
          409E    00270 PUTBYA  EQU     $409E
                  00280
7000              00290         ORG     $7000
                  00300
                  00310 * PUTBYA TEST
                  00320
7000 34   06      00330         PSHS    A,B
                  00340
                  00350 * LOAD THE FIRST
                  00360 * BYTE VALUE
7002 86   00      00370         LDA     #$00
                  00380
                  00390 * SAVE THE BYTE VALUE
7004 1F   89      00400        TFR     A,B
                  00410
                  00420 * GO PUT BYTE TO SCREEN
7006 BD   409E    00430 LBL001  JSR     PUTBYA
                  00440
                  00450 * ARE WE DONE?
7009 C1   7F      00460         CMPB    #$7F
                  00470
                  00480 * GO IF YES
700B 24   05      00490         BHS     LBL002
                  00500
                  00510 * GET NEXT BYTE VALUE
700D 5C           00520         INCB
700E 1F   98      00530         TFR     B,A
7010 20   F4      00540         BRA     LBL001
                  00550
                  00560 * EXIT
7012 35   06      00570 LBL002  PULS    A,B
7014 39           00580         RTS
                  00590
```

```
0000      00600           END
```

Note Line 460. We make the comparison against Register B, instead of against Register A, because **PUTBYA** alters Register A; i.e. it is not internally preserved.

——

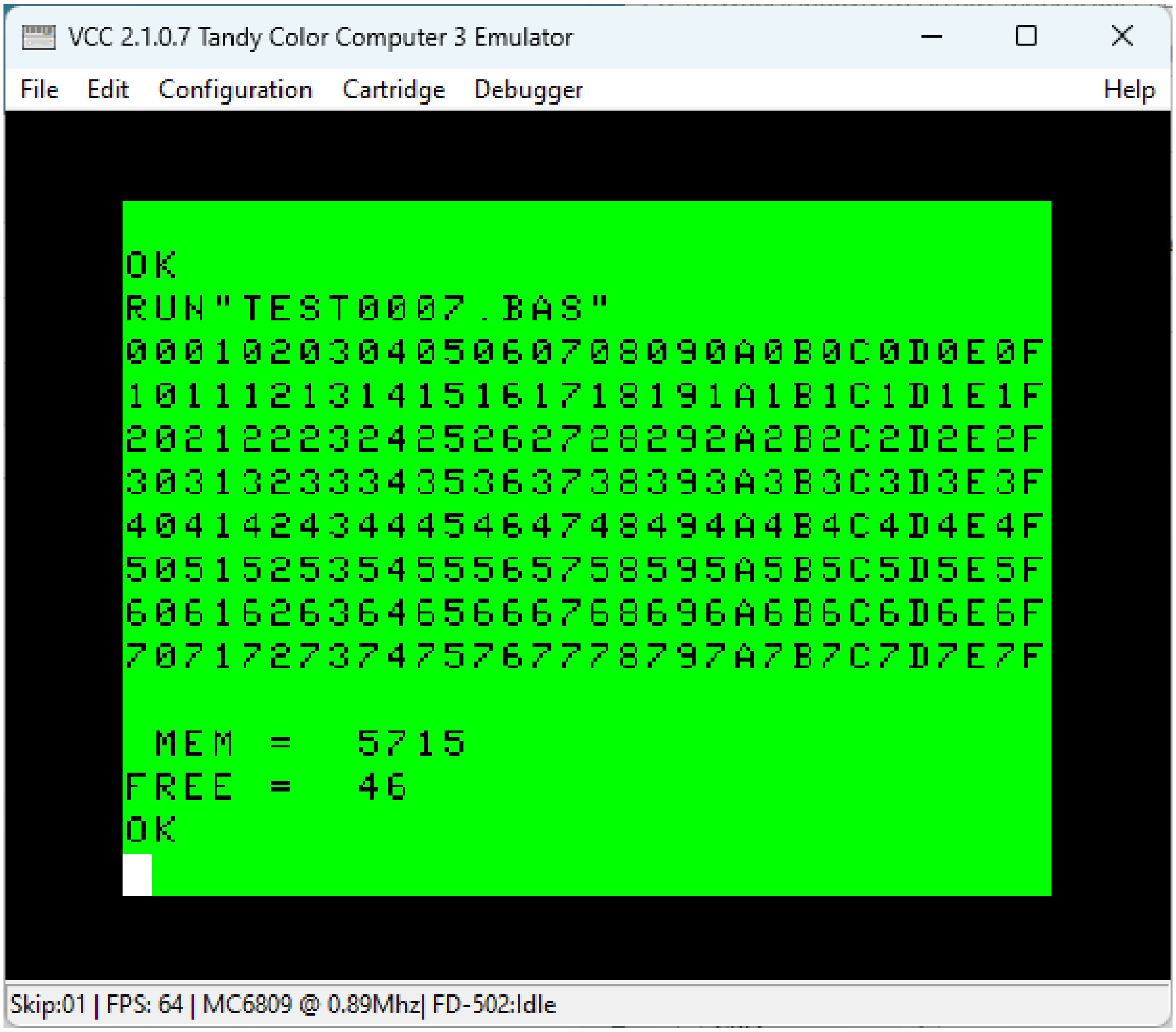
The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0007.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* PUTBYA 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0007.BIN"
1330 '

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
```

First, we perform the test with a simple **RUN** command.

Result:



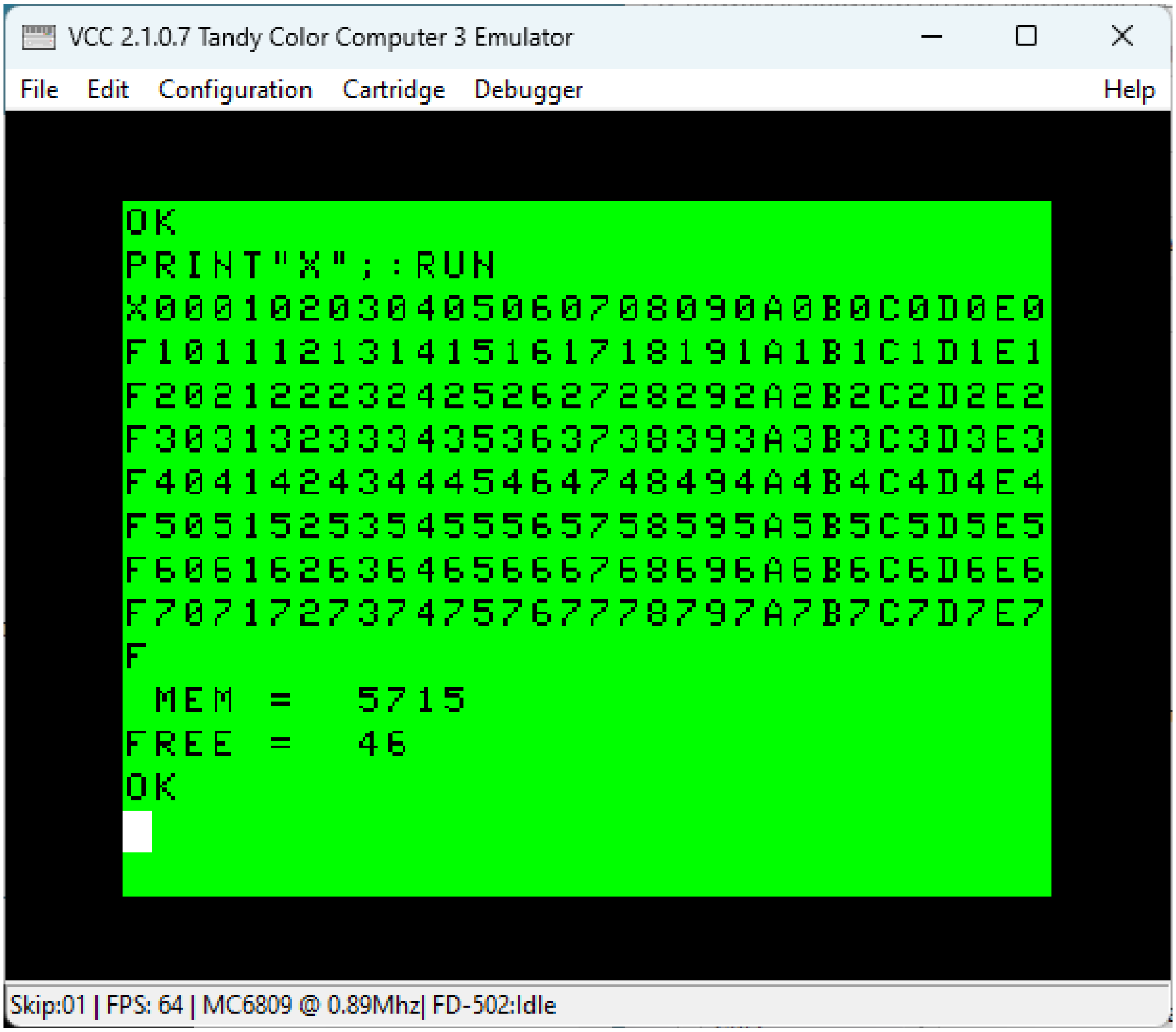
As expected.

But, note that when the scrolls occur, they all occur at the end of the byte.

To test **PUTBYA**'s ability to scroll in the middle of the byte if necessary, we use the command:

PRINT"X";:RUN

Result:

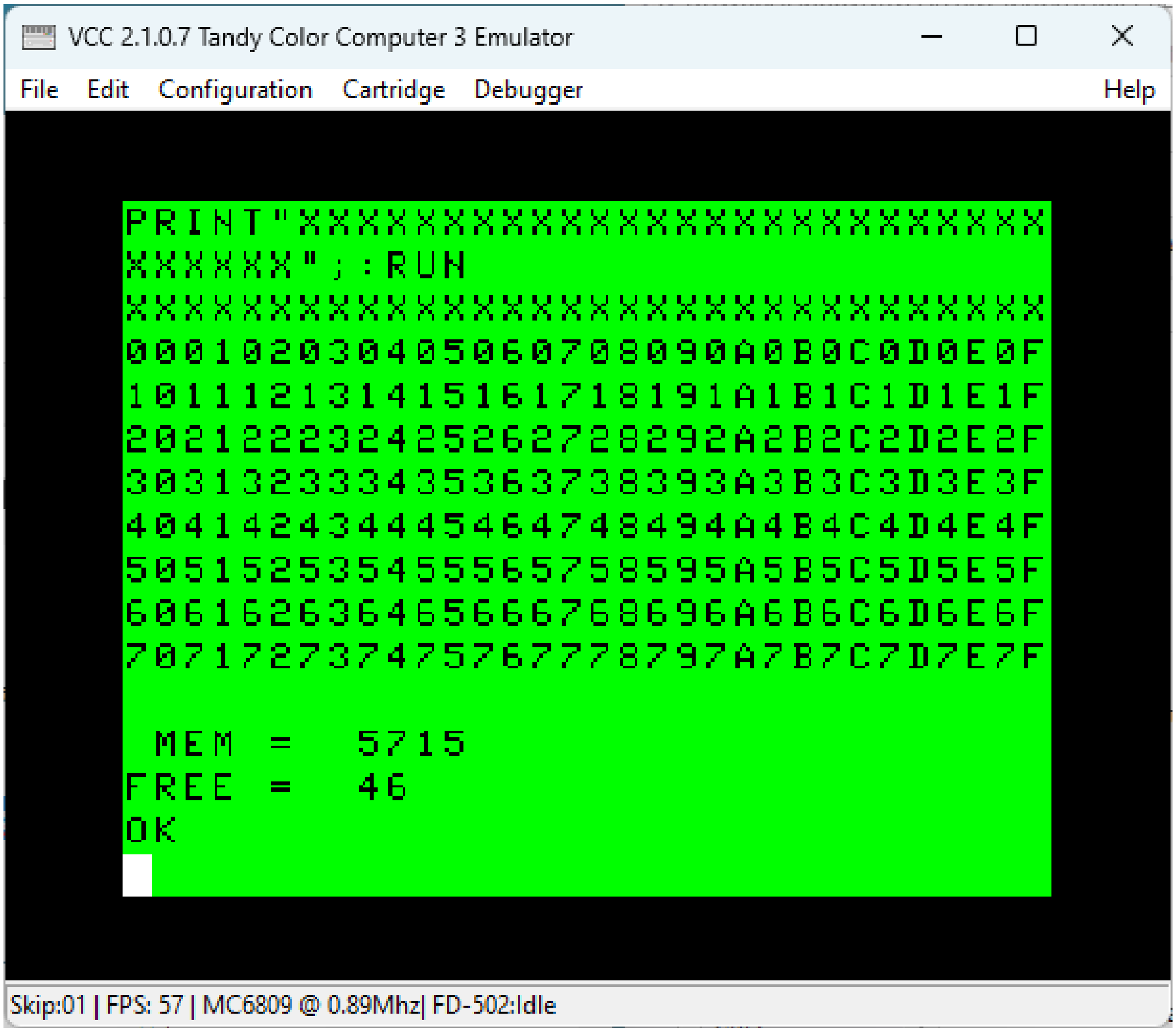


As expected.

And, finally, to test the situation where **PUTBYA** needs to do a scroll before starting to print the byte, we use the command:

PRINT”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”;:RUN

Result:



As expected.

=====

CRLF: Do a Carriage Return and Line Feed on the VIDRAM Screen

In addition to putting characters and bytes onto the screen, the ability to do a carriage return and line feed will also be useful.

```
                    00100 *****
                    00110 *
                    00120 * CRLF.ASM
                    00130 * MDJ 2023/01/24
                    00140 *
                    00150 * DO A CARRIAGE RETURN
                    00160 * AND LINE FEED
                    00170 *
                    00180 * ENTRY CONDITIONS
                    00190 * NONE
                    00200 *
                    00210 * EXIT CONDITIONS
                    00220 * NONE
                    00230 *
                    00240 *****
                    00250
                    00260 * LOW RAM CURSOR ADDRESS
          0088      00270 CURPOS  EQU     $0088
                    00280
                    00290 * SCREEN ADDRESSES
                    00300 * START OF VIDRAM
          0400      00310 VIDRAM  EQU     $0400
                    00320
                    00330 * ONE BYTE PAST THE
                    00340 * END OF VIDRAM
          0600      00350 VIDEND  EQU     $0600
                    00360
                    00370 * START OF THE LAST
                    00380 * LINE OF VIDRAM
          05E0      00390 VIDL15  EQU     $05E0
                    00400
                    00410 * EXTERNAL ROUTINE
                    00420 * ADDRESSES
          400E      00430 VIDCLS  EQU     $400E
          405E      00440 SCROLL  EQU     $405E
                    00450
40D7                00460         ORG     $40D7
                    00470
```

```
40D7 34   16        00480 CRLF    PSHS    A,B,X
                    00490
                    00500 * GET THE CURSOR
40D9 9E   88        00510         LDX     CURPOS
                    00520
                    00530 * IS IT BELOW RANGE?
40DB 8C   0400      00540         CMPX    #VIDRAM
                    00550
                    00560 * GO IF YES (ERROR)
40DE 25   1D        00570         BLO     LBL002
                    00580
                    00590 * IS IT ABOVE RANGE?
40E0 8C   0600      00600         CMPX    #VIDEND
                    00610
                    00620 * GO IF YES (ERROR)
40E3 24   18        00630         BHS     LBL002
                    00640
                    00650 * IS IT ON THE LAST LINE
                    00660 * OF THE VIDRAM SCREEN?
40E5 8C   05E0      00670         CMPX    #VIDL15
                    00680
                    00690 * GO IF YES
40E8 24   0B        00700         BHS     LBL001
                    00710
                    00720 * DO THE CRLF
40EA 1F   10        00730         TFR     X,D
40EC C4   E0        00740         ANDB    #$E0
40EE C3   0020      00750         ADDD    #$0020
40F1 1F   01        00760         TFR     D,X
40F3 20   0E        00770         BRA     LBL003
                    00780
                    00790 * LAST LINE
40F5 BD   405E      00800 LBL001  JSR     SCROLL
40F8 8E   05E0      00810         LDX     #VIDL15
40FB 20   06        00820         BRA     LBL003
                    00830
                    00840 * ERROR
40FD BD   400E      00850 LBL002  JSR     VIDCLS
4100 8E   0400      00860         LDX     #VIDRAM
                    00870
                    00880 * PUT THE CURSOR
4103 9F   88        00890 LBL003  STX     CURPOS
                    00900
                    00910 * EXIT
4105 35   16        00920         PULS    A,B,X
4107 39             00930         RTS
                    00940
```

```
        0000      00950           END
```

Note that if the cursor position has somehow been corrupted and is outside of its proper $0400 - $05FF range, **CRLF** simply clears the screen and puts the cursor at its start location, i.e. $0400.

The actual **CRLF** part of this code, i.e. lines 740-770, might benefit from a little additional explanation. The code transfers the cursor position value from Register **X** to Register **D**, massages it a bit, and then transfers it back to Register **X**.

Each of the sixteen lines on the **VIRRAM** screen begins at a multiple of $20 (= 032 decimal), i.e. the first character of each line is at the address indicated as follows:

	First Character Address		
Line	**Hexadecimal**	**Decimal**	**Binary**
00	$0400	1024	0000 0100 0000 0000
01	$0420	1056	0000 0100 0010 0000
02	$0440	1088	0000 0100 0100 0000
03	$0460	1120	0000 0100 0110 0000
04	$0480	1152	0000 0100 1000 0000
05	$04A0	1184	0000 0100 1010 0000
06	$04C0	1216	0000 0100 1100 0000
07	$04E0	1248	0000 0100 1110 0000
08	$0500	1280	0000 0101 0000 0000
09	$0520	1312	0000 0101 0010 0000
10	$0540	1344	0000 0101 0100 0000
11	$0560	1376	0000 0101 0110 0000
12	$0580	1408	0000 0101 1000 0000
13	$05A0	1440	0000 0101 1010 0000
14	$05C0	1472	0000 0101 1100 0000
15	$05E0	1504	0000 0101 1110 0000

And, for reference:

VIDRAM	$0400	1024	0000 0100 0000 0000
VIDL01	$0420	1056	0000 0100 0010 0000
VIDL15	$05E0	1472	0000 0101 1110 0000
VIDEND	$0600	1536	0110 0000 0000 0000

Notice that the last five bits in the binary format are all zeroes. **2^5 = 32 = $20.**

$10000 - $20 = $FFE0 = 1111 1111 1110 0000 binary.

Therefore, if we **AND** any screen address (**$0400 - $05FF**) with **$FFE0**, we will obtain the address of the first character of the line in which that screen address appears.

If we then add **32 = $20** to that value, we will obtain the address of the first character of the following line.

So, our logical approach would then simply seem to be:

```
ANDD    $FFE0
```

Unfortunately, there is no **ANDD** in MC6809 machine language; only **ANDA**, **ANDB**, and **ANDCC**. However, since Register A is the high byte of Register D and Register B is the low byte of Register D, i.e.:

```
D = A:B
```

We can accomplish the same task by doing:

```
ANDA    $FF
ANDB    $E0
```

and, because **ANDA $FF** will always leave Register A unchanged, we can simply leave that line of code out. Thus **ANDB $E0** will give us the address of the first character of the current line in Register D, and adding $0020 to that will give us the address of the first character in the following line. And so we have:

```
00720 * DO THE CRLF
00730         TFR     X,D
00740         ANDB    #$E0
00750         ADDD    #$0020
00760         TFR     D,X
```

———

The Assembly Language Test Routine:

```
         00100 *****
         00110 *
         00120 * TEST0008.ASM
         00130 * MDJ 2023/02/11
         00140 *
         00150 * CRLF TEST
         00160 *
         00170 *****
         00180
         00190 * ML FOUNDATION
         00200 * CORE ADDRESSES
407D     00210 PUTCHA  EQU     $407D
40D7     00220 CRLF    EQU     $40D7
         00230
```

```
7000                00240          ORG      $7000
                    00250
                    00260 * CRLF TEST
                    00270
7000 34    20       00280          PSHS     Y
7002 20    73       00290          BRA      LBL001
                    00300
7004       41       00310 STR01    FCC      /ALTHOUGH/
           4C
           54
           48
           4F
           55
           47
           48
700C       60       00320          FCB      $60      SPACE
700D       47       00330          FCC      /GOD/
           4F
           44
7010       60       00340          FCB      $60      SPACE
7011       4C       00350          FCC      /LOVES/
           4F
           56
           45
           53
7016       60       00360          FCB      $60      SPACE
7017       55       00370          FCC      /US/
           53
7019       6C       00380          FCB      $6C      COMMA
701A       00       00390          FCB      $00      NULL
                    00400
701B       48       00410 STR02    FCC      /HE/
           45
701D       67       00420          FCB      $67      APOSTROPHE
701E       53       00430          FCC      /S/
701F       60       00440          FCB      $60      SPACE
7020       41       00450          FCC      /ALSO/
           4C
           53
           4F
7024       60       00460          FCB      $60      SPACE
7025       47       00470          FCC      /GOING/
           4F
           49
           4E
           47
702A       60       00480          FCB      $60      SPACE
```

```
702B        54      00490         FCC     /TO/
            4F
702D        60      00500         FCB     $60      SPACE
702E        4A      00510         FCC     /JUDGE/
            55
            44
            47
            45
7033        60      00520         FCB     $60      SPACE
7034        55      00530         FCC     /US/
            53
7036        00      00540         FCB     $00      NULL
                    00550
7037        41      00560 STR03   FCC     /ACCORDING/
            43
            43
            4F
            52
            44
            49
            4E
            47
7040        60      00570         FCB     $60      SPACE
7041        54      00580         FCC     /TO/
            4F
7043        60      00590         FCB     $60      SPACE
7044        48      00600         FCC     /HIS/
            49
            53
7047        60      00610         FCB     $60      SPACE
7048        50      00620         FCC     /PERFECT/
            45
            52
            46
            45
            43
            54
704F        00      00630         FCB     $00      NULL
                    00640
7050        48      00650 STR04   FCC     /HOLINESS/
            4F
            4C
            49
            4E
            45
            53
            53
```

```
7058        6E      00660               FCB     $6E       PERIOD
7059        60      00670               FCB     $60       SPACE
705A        54      00680               FCC     /THAT/
            48
            41
            54
705E        67      00690               FCB     $67       APOSTROPHE
705F        53      00700               FCC     /S/
7060        60      00710               FCB     $60       SPACE
7061        53      00720               FCC     /SCARY/
            43
            41
            52
            59
7066        61      00730               FCB     $61       EXCLAMATION MARK
7067        00      00740               FCB     $00       NULL
                    00750
7068        41      00760 STR05         FCC     /ARE/
            52
            45
706B        60      00770               FCB     $60       SPACE
706C        59      00780               FCC     /YOU/
            4F
            55
706F        60      00790               FCB     $60       SPACE
7070        52      00800               FCC     /READY/
            45
            41
            44
            59
7075        7F      00810               FCB     $7F       QUESTION MARK
7076        00      00820               FCB     $00       NULL
                    00830
                    00840 * OPEN WITH A CRLF
7077 BD     40D7    00850 LBL001        JSR     CRLF
                    00860
                    00870 * THE FIRST STRING
707A 108E 7004      00880               LDY     #STR01
707E BD     70A2    00890               JSR     LBL002
                    00900
                    00910 * THE SECOND STRING
7081 108E 701B      00920               LDY     #STR02
7085 BD     70A2    00930               JSR     LBL002
                    00940
                    00950 * THE THIRD STRING
7088 108E 7037      00960               LDY     #STR03
708C BD     70A2    00970               JSR     LBL002
```

```
                   00980
                   00990 * THE FOURTH STRING
708F 108E 7050     01000         LDY     #STR04
7093 BD   70A2     01010         JSR     LBL002
                   01020
                   01030 * THE FIFTH STRING
7096 108E 7068     01040         LDY     #STR05
709A BD   70A2     01050         JSR     LBL002
                   01060
709D BD   40D7     01080         JSR     CRLF
70A0 20   0D       01090         BRA     LBL004
                   01100
                   01110 * SUBROUTINE TO PUT
                   01120 * STRING TO SCREEN
                   01130 * GET A CHARACTER
70A2 A6   A0       01140 LBL002  LDA     ,Y+
                   01150
                   01160 * GO IF NULL TERMINATOR
70A4 27   05       01170         BEQ     LBL003
                   01180
                   01190 * PUT IT TO VIDRAM
70A6 BD   407D     01200         JSR     PUTCHA
70A9 20   F7       01210         BRA     LBL002
                   01220
                   01230 * LINE-ENDING CRLF
70AB BD   40D7     01240 LBL003  JSR     CRLF
70AE 39            01250         RTS
                   01260
                   01270 * EXIT
70AF 35   20       01280 LBL004  PULS   Y
70B1 39            01290         RTS
                   01300
          0000     01310         END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0008.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* CRLF 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0008.BIN"
1330 '

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.

____

Bible Note: God created us in Adam and Eve. He gave those two only one rule: "Don't eat from the Tree of the Knowledge of Good and Evil." But they disobeyed and were cast out. We all inherited that evil, rebellious, prideful nature from Adam and Eve. And we have also added our own evil thoughts and deeds on top of it.

In addition to being perfect Love, God is also perfect Justice. By His very nature, He MUST punish every trace of evil. God has decreed that there shall be one punishment: banishment from His presence for all eternity in Hell and the Lake of Fire.

There are only two ways that penalty can be paid. Either we pay it ourselves, or we accept the fact that Jesus paid it for us and we turn away from our evil and trust Jesus to save us.

Romans 10:9-10 says, “If you confess with your mouth that Jesus is Lord and believe in your heart that God raised him from the dead, you will be saved.” (ESV).

=====

PK2PRT: Converting POKE Codes to PRINT Codes

The Video Screen VIDRAM displays text characters and Semi-Graphics characters depending upon which bytes are put to memory in the VIDRAM space ($0400-$05FF).

In the BASIC Language, as discussed in (MDJ01), for each byte ($00-$FF), the character displayed depends upon the method (PRINT or POKE) by which the byte is put to the VIDRAM.

In Assembly Language (and thus also in The ML Foundation), the method used (STA) is analogous to BASIC's POKE mechanism. But the Key Codes returned from the keyboard are PRINT codes. Thus, we need a means for converting between the two types of Key Codes.

In performing such conversions, you may expect that PRT2PK (See next Section) will be required significantly more often than PK2PRT which is discussed in this Section.

If you have a POKE Code, and you want to use it in a PRINT function, convert it to a PRINT Code. If:

000 <= POKE Code <= 031 ($00 <= POKE Code <= $1F)
Then add 096 ($60) to the POKE Code.

032 <= POKE Code <= 063 ($20 <= POKE Code <= $3F)
Then, there is no PRINT Code that Corresponds to that POKE Code;
Use PRINT Code = 32 ($20) = a blank green space.

064 <= POKE Code <= 095 ($40 <= POKE Code <= $5F)
Then the PRINT Code is the same as the POKE Code.

096 <= POKE Code <= 127 ($60 <= POKE Code <= $7F)
Then subtract 064 ($40) from the POKE Code.

128 <= POKE Code <= 255) ($80 <= POKE Code <= $FF)
Then the PRINT Code is the same as the POKE Code.

```
00100 *****
00110 *
00120 * PK2PRT.ASM
00130 * MDJ 2023/01/17
00140 *
00150 * CONVERTS POKE CODES
00160 * TO PRINT CODES
00170 *
```

```
                   00180 * ENTRY CONDITIONS:
                   00190 * A = POKE CODE
                   00200 * ($00 - $FF)
                   00210 * (000 - 255)
                   00220 *
                   00230 * EXIT CONDITIONS:
                   00240 * A = PRINT CODE
                   00250 * ($00 - $FF)
                   00260 * (000 - 255)
                   00270 *
                   00280 *****
                   00290
4108               00300         ORG     $4108
                   00310
                   00320 * 000 <= CODE <= 031 ?
4108 81   20       00330 PK2PRT  CMPA    #32
                   00340
                   00350 * GO IF YES
410A 25   16       00360         BLO     LBL003
                   00370
                   00380 * 032 <= CODE <= 063 ?
410C 81   40       00390         CMPA    #64
                   00400
                   00410 * GO IF YES
410E 25   0E       00420         BLO     LBL002
                   00430
                   00440 * 064 <= CODE <= 095 ?
4110 81   60       00450         CMPA    #96
                   00460
                   00470 * GO IF YES ==> NO CHANGE
4112 25   10       00480         BLO     LBL004
                   00490
                   00500 * 096 <= CODE <= 127 ?
4114 81   80       00510         CMPA    #128
                   00520
                   00530 * GO IF YES
4116 25   02       00540         BLO     LBL001
                   00550
                   00560 * CODE >= 128
                   00570 * NO CHANGE
4118 20   0A       00580         BRA     LBL004
                   00590
                   00600 * 096 <= CODE <= 127
411A 80   40       00610 LBL001  SUBA    #64
411C 20   06       00620         BRA     LBL004
                   00630
                   00640 * 064 <= CODE <= 095
```

```
                  00650 * NO CHANGE
                  00660
                  00670 * 032 <= CODE <= 063
                  00680 * ALL = A GREEN SPACE
411E 86   20      00690 LBL002  LDA     #32
4120 20   02      00700         BRA     LBL004
                  00710
                  00720 * 000 <= CODE <= 031
4122 8B   60      00730 LBL003  ADDA    #96
                  00740
                  00750 * EXIT
4124 39           00760 LBL004  RTS
                  00770
          0000    00780         END
```

The Assembly Language Test Routine:

```
                  00100 *****
                  00110 *
                  00120 * TEST0009.ASM
                  00130 * MDJ 2023/02/11
                  00140 *
                  00150 * PK2PRT TEST
                  00160 *
                  00170 *****
                  00180
                  00190 * ML FOUNDATION
                  00200 * CORE ADDRESSES
          407D    00210 PUTCHA  EQU     $407D
          409E    00220 PUTBYA  EQU     $409E
          40D7    00230 CRLF    EQU     $40D7
          4108    00240 PK2PRT  EQU     $4108
                  00250
7000              00260         ORG     $7000
                  00270
                  00280 * PK2PRT TEST
                  00290
7000 34   20      00300         PSHS    Y
7002 20   17      00310         BRA     LBL001
                  00320
                  00330 * OUTPUT STRINGS LIST
7004      60      00340 STR01   FCB     $60
7005      50      00350         FCC     /POKE/
          4F
          4B
```

```
             45
7009         60       00360          FCB      $60
700A         00       00370          FCB      $00
700B         50       00380 STR02    FCC      /PRINT/
             52
             49
             4E
             54
7010         60       00390          FCB      $60
7011         00       00400          FCB      $00
7012         43       00410 STR03    FCC      /CODE/
             4F
             44
             45
7016         60       00420          FCB      $60
7017         7D       00430          FCB      $7D
7018         60       00440          FCB      $60
7019         64       00450          FCB      $64
701A         00       00460          FCB      $00
                      00470
                      00480 * MAIN ROUTINE
701B 86      12       00490 LBL001   LDA      #18
701D 8D      0E       00500          BSR      LBL002
701F 86      28       00510          LDA      #40
7021 8D      0A       00520          BSR      LBL002
7023 86      57       00530          LDA      #87
7025 8D      06       00540          BSR      LBL002
7027 86      77       00550          LDA      #119
7029 8D      02       00560          BSR      LBL002
702B 20      3D       00570          BRA      LBL008
                      00580
                      00590 * PRINT ORDER SUBROUTINE
                      00600 * SAVE POKE CODE
702D 34      02       00610 LBL002   PSHS     A
                      00620
                      00630 * PUT POKE MESSAGE
702F 8D      1F       00640          BSR      LBL003
7031 8D      29       00650          BSR      LBL005
                      00660
                      00670 * RESTORE AND RE-SAVE
                      00680 * POKE CODE
7033 35      02       00690          PULS     A
7035 34      02       00700          PSHS     A
                      00710
                      00720 * PUT POKE CODE
7037 BD      409E     00730          JSR      PUTBYA
                      00740
```

```
                  00750 * PUT PRINT MESSAGE
703A BD   40D7    00760         JSR     CRLF
703D 8D   17      00770         BSR     LBL004
703F 8D   1B      00780         BSR     LBL005
                  00790
                  00800 * RESTORE POKE CODE
7041 35   02      00810         PULS    A
                  00820
                  00830 * CONVERT CODE
7043 BD   4108    00840         JSR     PK2PRT
                  00850
                  00860 * PUT PRINT CODE
7046 BD   409E    00870         JSR     PUTBYA
7049 BD   40D7    00880         JSR     CRLF
704C BD   40D7    00890         JSR     CRLF
704F 39           00900         RTS
                  00910
                  00920 * PRINT MESSAGE SUBRT
7050 108E 7004    00930 LBL003  LDY     #STR01
7054 20   0A      00940         BRA     LBL006
7056 108E 700B    00950 LBL004  LDY     #STR02
705A 20   04      00960         BRA     LBL006
705C 108E 7012    00970 LBL005  LDY     #STR03
7060 A6   A0      00980 LBL006  LDA     ,Y+
7062 27   05      00990         BEQ     LBL007
7064 BD   407D    01000         JSR     PUTCHA
7067 20   F7      01010         BRA     LBL006
7069 39           01020 LBL007  RTS
                  01030
                  01040 * EXIT
706A 35   20      01050 LBL008  PULS    Y
706C 39           01060         RTS
                  01070
          0000    01080         END
```

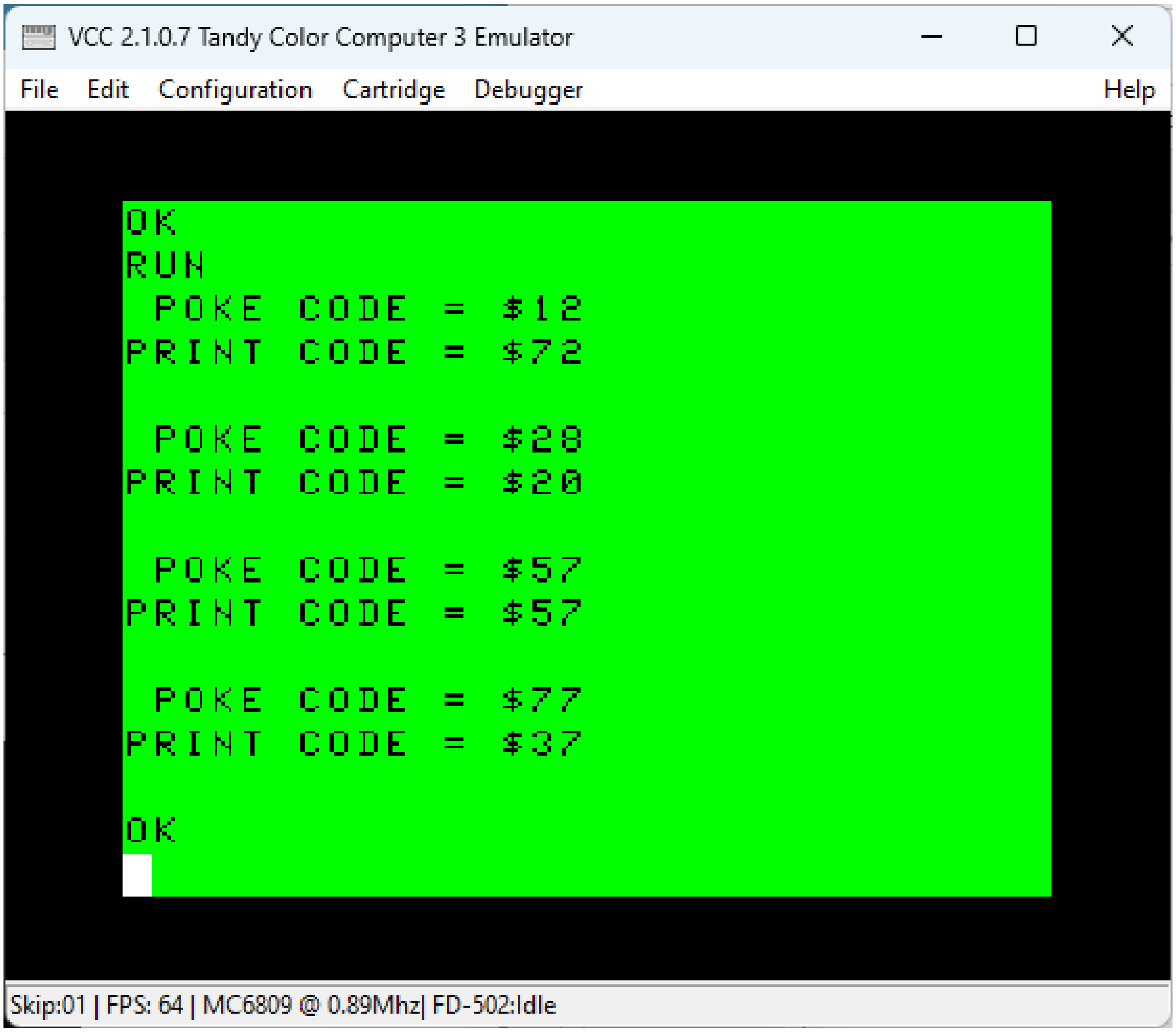
———

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0009.BAS
1030 '* MDJ 2023/02/11
1040 '*
1050 '* PK2PRT 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0009.BIN"
1330 '

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 '

32767 END
```

Result:



As expected.

=====

PRT2PK: Converting PRINT Codes to POKE Codes

The Video Screen VIDRAM displays text characters and Semi-Graphics characters depending upon which bytes are put to memory in the VIDRAM space ($0400-$05FF).

In the BASIC Language, as discussed in (MDJ01), for each byte ($00-$FF), the character displayed depends upon the method (PRINT or POKE) by which the byte is put to the VIDRAM.

In Assembly Language (and thus also in The ML Foundation), the method used (STA) is analogous to BASIC's POKE mechanism. But the Key Codes returned from the keyboard are PRINT codes. Thus, we need a means for converting between the two types of Key Codes.

In performing such conversions, you may expect that PRT2PK will be required significantly more often than PK2PRT (See previous Section) .

If you have a PRINT Code, and you want to use it in a POKE function, convert it to a POKE Code. If:

000 <= PRINT Code <= 031 ($00 <= PRINT Code <= $1F)
Then it is a blank background color space.
Use POKE Code = 096 ($60) = a blank green space.

032 <= PRINT Code <= 063 ($20 <= PRINT Code <= $3F)
Then add 064 ($40) to the PRINT Code.

064 <= PRINT Code <= 095 ($40 <= PRINT Code <= $5F)
Then the POKE Code is the same as the PRINT Code.

096 <= PRINT Code <= 127 ($60 <= PRINT Code <= $7F)
Then subtract 096 ($60) from the PRINT Code.

128 <= PRINT Code <= 255) ($80 <= PRINT Code <= $FF)
Then the POKE Code is the same as the PRINT Code.

```
00100 *****
00110 *
00120 * PRT2PK.ASM
00130 * MDJ 2023/01/17
00140 *
00150 * CONVERTS PRINT CODES
00160 * TO POKE CODES
00170 *
00180 * ENTRY CONDITIONS
```

```
                   * A = PRINT CODE
                   * ($00 - $FF)
                   * (000 - 255)
                   *
                   * EXIT CONDITIONS
                   * A = POKE CODE
                   * ($00 - $FF)
                   * (000 - 255)
                   *
                   *****

4125                       ORG     $4125

                   * 000 <= CODE <= 031 ?
4125 81   20       PRT2PK  CMPA    #32

                   * GO IF YES
4127 25   16               BLO     LBL003

                   * 032 <= CODE <= 063 ?
4129 81   40               CMPA    #64

                   * GO IF YES
412B 25   0E               BLO     LBL002

                   * 064 <= CODE <= 095 ?
412D 81   60               CMPA    #96

                   * GO IF YES ==> NO CHANGE
412F 25   10               BLO     LBL004

                   * 096 <= CODE <= 127 ?
4131 81   80               CMPA    #128

                   * GO IF YES
4133 25   02               BLO     LBL001

                   * CODE >= 128
                   * NO CHANGE
4135 20   0A               BRA     LBL004

                   * 096 <= CODE <= 127
4137 80   60       LBL001  SUBA    #96
4139 20   06               BRA     LBL004

                   * 064 <= CODE <= 095
                   * NO CHANGE
```

```
                    00660
                    00670 * 032 <= CODE <= 063
413B 8B   40        00680 LBL002  ADDA    #64
413D 20   02        00690         BRA     LBL004
                    00700
                    00710 * 000 <= CODE <= 031
                    00720 * ALL = A GREEN SPACE
413F 86   60        00730 LBL003  LDA     #96
                    00740
                    00750 * EXIT
4141 39             00760 LBL004  RTS
                    00770
          0000      00780         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0010.ASM
                    00130 * MDJ 2023/02/12
                    00140 *
                    00150 * PRT2PK TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * ML FOUNDATION
                    00200 * CORE ADDRESSES
          407D      00210 PUTCHA  EQU     $407D
          409E      00220 PUTBYA  EQU     $409E
          40D7      00230 CRLF    EQU     $40D7
          4108      00240 PK2PRT  EQU     $4108
          4125      00250 PRT2PK  EQU     $4125
                    00260
7000                00270         ORG     $7000
                    00280
                    00290 * PRT2PK TEST
                    00300
7000 34   20        00310         PSHS    Y
7002 20   17        00320         BRA     LBL001
                    00330
                    00340 * OUTPUT STRINGS LIST
7004      50        00350 STR01   FCC     /PRINT/
          52
          49
          4E
```

```
            54
7009        60      00360           FCB     $60
700A        00      00370           FCB     $00
700B        60      00380 STR02     FCB     $60
700C        50      00390           FCC     /POKE/
            4F
            4B
            45
7010        60      00400           FCB     $60
7011        00      00410           FCB     $00
7012        43      00420 STR03     FCC     /CODE/
            4F
            44
            45
7016        60      00430           FCB     $60
7017        7D      00440           FCB     $7D
7018        60      00450           FCB     $60
7019        64      00460           FCB     $64
701A        00      00470           FCB     $00
                    00480
                    00490 * MAIN ROUTINE
701B 86     12      00500 LBL001    LDA     #18
701D 8D     0E      00510           BSR     LBL002
701F 86     28      00520           LDA     #40
7021 8D     0A      00530           BSR     LBL002
7023 86     57      00540           LDA     #87
7025 8D     06      00550           BSR     LBL002
7027 86     77      00560           LDA     #119
7029 8D     02      00570           BSR     LBL002
702B 20     3D      00580           BRA     LBL008
                    00590
                    00600 * PRINT ORDER SUBROUTINE
                    00610 * SAVE PRINT CODE
702D 34     02      00620 LBL002    PSHS    A
                    00630
                    00640 * PUT PRINT MESSAGE
702F 8D     1F      00650           BSR     LBL003
7031 8D     29      00660           BSR     LBL005
                    00670
                    00680 * RESTORE AND RE-SAVE
                    00690 * PRINT CODE
7033 35     02      00700           PULS    A
7035 34     02      00710           PSHS    A
                    00720
                    00730 * PUT PRINT CODE
7037 BD     409E    00740           JSR     PUTBYA
                    00750
```

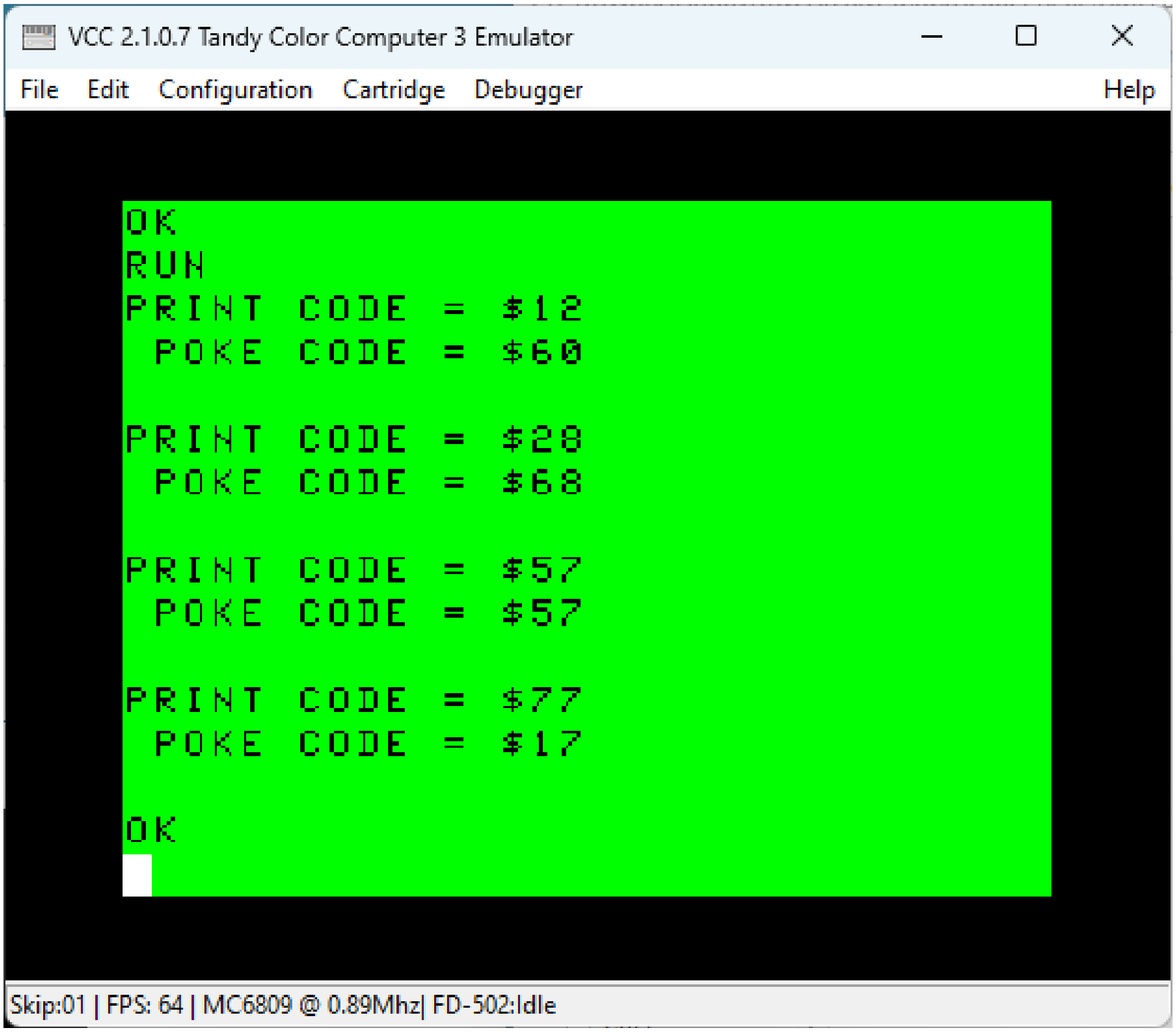
```
                     00760 * PUT POKE MESSAGE
703A BD   40D7       00770        JSR     CRLF
703D 8D   17         00780        BSR     LBL004
703F 8D   1B         00790        BSR     LBL005
                     00800
                     00810 * RESTORE PRINT CODE
7041 35   02         00820        PULS    A
                     00830
                     00840 * CONVERT CODE
7043 BD   4125       00850        JSR     PRT2PK
                     00860
                     00870 * PUT POKE CODE
7046 BD   409E       00880        JSR     PUTBYA
7049 BD   40D7       00890        JSR     CRLF
704C BD   40D7       00900        JSR     CRLF
704F 39              00910        RTS
                     00920
                     00930 * PRINT MESSAGE SUBRT
7050 108E 7004       00940 LBL003 LDY     #STR01
7054 20   0A         00950        BRA     LBL006
7056 108E 700B       00960 LBL004 LDY     #STR02
705A 20   04         00970        BRA     LBL006
705C 108E 7012       00980 LBL005 LDY     #STR03
7060 A6   A0         00990 LBL006 LDA     ,Y+
7062 27   05         01000        BEQ     LBL007
7064 17   D016       01010        LBSR    PUTCHA
7067 20   F7         01020        BRA     LBL006
7069 39              01030 LBL007 RTS
                     01040
                     01050 * EXIT
706A 35   20         01060 LBL008 PULS    Y
706C 39              01070        RTS
                     01080
          0000       01090        END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0010.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* PRT2PK 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0010.BIN"
1330 '

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 '

32767 END
```

Result:



As expected.

=====

POLCAT: Get a Key Press Character Code From the Keyboard

With the completion of this **POLCAT** Routine, we have a minimally complete system; capable of receiving input from the Keyboard and generating output to the **VIDRAM** Screen.

But that is just the bare minimum. In order to provide a more useful collection of routines in the ML Foundation Core, much remains to be developed, all of which will be done and presented in the remainder of this paper.

Beyond the core, I hope to provide additional portions of a complete ML Foundation in future papers covering other ML Foundation Modules which will build upon this core.

In conformance with my CoCo Philosophy of trying to cram as much stuff as I can into the 96K of the 64K CoCo 2, **POLCAT** simply jumps into ROM and uses its POLCAT Routine. (cf. Appendix A below for the details of my CoCo Philosophy).

I developed many of the other Routines presented in this Paper entirely in Assembly Language in order to maximize speed while also minimizing memory space used.

But, with the Keyboard, the limitations of the BASIC Language (Very Slow) are enormously overshadowed by the User Limitations (SUPER HUMONGOUS SLOW). There is thus no point in trying to write super-fast machine code for Keyboard access. It is much more efficient to just use the ROM code: it immensely saves on RAM space, and won’t materially effect overall system speed in normal use.

In the future, I will use the same approach regarding access to comparatively slow peripherals such as the Cassette and Disk Drives, and the RS-232 Port.

```
00100 *****
00110 *
00120 * POLCAT.ASM
00130 * MDJ 2021/01/17
00140 *
00150 * POLLS THE KEYBOARD
00160 * FOR A KEYSTROKE
00170 *
00180 * USES THE BUILT-IN
00190 * ROM POLCAT ROUTINE
00200 *
00210 * ENTRY CONDITIONS:
00220 * NONE
00230 *
```

```
                 00240 * EXIT CONDITIONS:
                 00250 * IF NO KEY WAS PRESSED:
                 00260 * CC Z BIT = 1
                 00270 * REG A = 0
                 00280 * IF A KEY WAS PRESSED
                 00290 * CC Z BIT = 0
                 00300 * REG A = CHAR CODE
                 00310 *
                 00320 * ANY ROUTINE CALLING
                 00330 * THIS SHOULD PSHS A,CC
                 00340 * BEFORE CALLING AND
                 00350 * PULS A,CC ON RETURN
                 00360 * AFTER CHECKING AND
                 00370 * TRANSFERRING THE DATA
                 00380 * RETURNED AS REQUIRED
                 00390 *
                 00400 *****
                 00410
                 00420 * RAMROM TRIGGER ADDRESS
          FFDE   00430 RAMROM  EQU     $FFDE
                 00440
                 00450 * ALLRAM TRIGGER ADDRESS
          FFDF   00460 ALLRAM  EQU     $FFDF
                 00470
                 00480 * ROM POLCAT JUMP ADDRESS
          A000   00490 XPOLCT  EQU     $A000
                 00500
4142             00510         ORG     $4142
                 00520
4142 34   68     00530 POLCAT  PSHS    Y,U,DP
                 00540
                 00550 * SET RAMROM MODE
4144 B7   FFDE   00560         STA     RAMROM
                 00570
                 00580 * GO DO ROM POLCAT
4147 AD   9F A000 00590        JSR     [XPOLCT]
                 00600
                 00610 * SET ALLRAM MODE
414B B7   FFDF   00620         STA     ALLRAM
                 00630
                 00640 * EXIT
414E 35   68     00650         PULS    Y,U,DP
4150 39          00660         RTS
                 00670
          0000   00680         END
```

The general concept for the Test Routine is simply to receive Key Press Codes from the Keyboard and echo them to the **VIDRAM** Screen.

The CoCo 2 Keyboard is not capable of directly generating Codes 000 - 002, 004 - 007, 011, 014 - 020, 022 - 031, 096, or 123 - 255 ($00 - $02, $04 - $07, $0B, $0E - $14, $16 - $1F, $60, or $7B - $FF).

And, for the purposes of this Test, we will ignore Codes 009, 010, 012, and 021 ($09, $0A, $0C, and $15).

When the Test receives Code 003 ($03), it will do a BREAK (i.e. exit the program) and return to the Command Prompt.

When the Test receives Code 008 ($08), it will perform the Backspace and Overwrite Function.

When the Test receives Code 013 ($0D), it will perform the Carriage Return/Linefeed Function.

For all other Codes, the Test Routine will echo the appropriate character to the **VIDRAM** Screen. The Key Press Character Codes returned from the Keyboard are PRINT Codes rather than POKE Codes. Therefore, any such codes which are to be output to the **VIDRAM** Screen must be processed through **PRT2PK** first.

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0011.ASM
                   00130 * MDJ 2023/02/12
                   00140 *
                   00150 * POLCAT TEST
                   00160 *
                   00170 *****
                   00180
                   00190 * LOW RAM CURSOR ADDRESS
          0088     00200 CURPOS  EQU     $0088
                   00210
                   00220 * ML FOUNDATION
                   00230 * CORE ADDRESSES
          401F     00240 PUTCHR  EQU     $401F
          407D     00250 PUTCHA  EQU     $407D
          40D7     00260 CRLF    EQU     $40D7
          4125     00270 PRT2PK  EQU     $4125
          4142     00280 POLCAT  EQU     $4142
                   00290
7000               00300         ORG     $7000
```

```
                     00310
                     00320 * POLCAT TEST
                     00330
7000 34   03         00340         PSHS    A,CC
                     00350
                     00360  * DISPLAY PROMPT
7002 86   7E         00370         LDA     #126
7004 BD   407D       00380         JSR     PUTCHA
7007 86   60         00390         LDA     #96
7009 BD   407D       00400         JSR     PUTCHA
                     00410
                     00420 * GO CHECK ROM
                     00430 * FOR KEY PRESS
700C BD   4142       00440 LBL001  JSR     POLCAT
                     00450
                     00460 * GO IF NO KEY PRESS
700F 27   FB         00470         BEQ     LBL001
                     00480
                     00490 * WAS IT THE BREAK KEY?
7011 81   03         00500         CMPA    #3
                     00510
                     00520 * GO IF YES
7013 27   32         00530         BEQ     LBL004
                     00540
                     00550 * WAS IT A BACKSPACE?
7015 81   08         00560         CMPA    #8
                     00570
                     00580 * GO IF YES
7017 27   1D         00590         BEQ     LBL003
                     00600
                     00610 * WAS IT A CARRIAGE
                     00620 * RETURN?
7019 81   0D         00630         CMPA    #13
                     00640
                     00650 * GO IF YES
701B 27   14         00660         BEQ     LBL002
                     00670
                     00680 * WAS IT <= 031 ?
701D 81   20         00690         CMPA    #32
                     00700
                     00710 * GO IF YES (IGNORE)
701F 25   EB         00720         BLO     LBL001
                     00730
                     00740 * WAS IT = 96 ?
7021 81   60         00750         CMPA    #96
                     00760
                     00770 * GO IF YES (IGNORE)
```

```
7023 27   E7       00780         BEQ     LBL001
                   00790
                   00800 * WAS IT >= 123 ?
7025 81   7B       00810         CMPA    #123
                   00820
                   00830 * GO IF YES (IGNORE)
7027 24   E3       00840         BHS     LBL001
                   00850
                   00860 * PUT IT TO VIDRAM
7029 BD   4125     00870         JSR     PRT2PK
702C BD   407D     00880         JSR     PUTCHA
702F 20   DB       00890         BRA     LBL001
                   00900
                   00910 * DO CRLF
7031 BD   40D7     00920 LBL002  JSR     CRLF
7034 20   D6       00930         BRA     LBL001
                   00940
                   00950 * DO BACKSPACE
7036 34   02       00960 LBL003  PSHS    A
7038 9E   88       00970         LDX     CURPOS
703A 30   1F       00980         LEAX    -1,X
703C 9F   88       00990         STX     CURPOS
703E 86   60       01000         LDA     #96
7040 BD   401F     01010         JSR     PUTCHR
7043 35   02       01020         PULS    A
7045 20   C5       01030         BRA     LBL001
                   01040
                   01050 * EXIT
7047 35   03       01060 LBL004  PULS    A,CC
7049 39            01070         RTS
                   01080
          0000     01090         END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0011.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* POLCAT 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0011.BIN"
1330 '

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.

Bible Note: The familiar John 3:16 reads, “For God so loved the world, that he gave his only begotten Son, that whosoever believeth in him should not perish, but have everlasting life.” (KJV).

But the following verses 17-21 may not be so familiar,

> [17] For God did not send his Son into the world to condemn the world,
> but to save the world through him. [18] Whoever believes in him is not
> condemned, but whoever does not believe stands condemned already
> because he has not believed in the name of God’s one and only Son. [19]
> This is the verdict: Light has come into the world, but men loved
> darkness instead of light because their deeds were evil. [20] Everyone

> who does evil hates the light, and will not come into the light for fear that his deeds will be exposed. [21] But whoever lives by the truth comes into the light, so that it may be seen plainly that what he has done has been done through God." (NIV).

=====

BKSPCE: Do a Backspace on the VIDRAM Screen

As noted in this Routine's opening comments, it backs the cursor up one position and overwrites that position with a blank space.

```
                    *****
                    *
                    * BKSPCE.ASM
                    * MDJ 2023/01/24
                    *
                    * BACKS THE CURSOR UP ONE
                    * POSITION IF IT IS NOT
                    * ALREADY AT LOWEST
                    * POSITION ON THE VIDRAM
                    * SCREEN AND OVERWRITES
                    * THAT POSITION WITH A
                    * SPACE (POKE MECHANISM
                    * CODE POINT 096)
                    *
                    * ENTRY CONDITIONS:
                    * NONE
                    *
                    * EXIT CONDITIONS:
                    * NONE
                    *
                    *****

                    * LOW RAM CURSOR ADDRESS
          0088      CURPOS EQU $0088

                    * SCREEN ADDRESSES
                    * START OF VIDRAM
          0400      VIDRAM EQU $0400

                    * EXTERNAL ROUTINE
                    * ADDRESS
          401F      PUTCHR  EQU     $401F

4151                        ORG     $4151

4151 34   12        BKSPCE  PSHS    A,X

                    * ADJUST THE CURSOR
```

```
4153 9E   88        00480          LDX     CURPOS
                    00490
                    00500 * IS IT ALREADY AT START
                    00510 * OF VIDRAM SCREEN
4155 8C   0400      00520          CMPX    #VIDRAM
                    00530
                    00540 * GO IF NO
4158 22   03        00550          BHI     LBL001
                    00560
415A 8E   0401      00570          LDX     #VIDRAM+1
415D 30   1F        00580 LBL001   LEAX    -1,X
415F 9F   88        00590          STX     CURPOS
                    00600
                    00610 * CLEAR SCREEN POSITION
4161 86   60        00620          LDA     #96
4163 BD   401F      00630          JSR     PUTCHR
                    00640
                    00650 * EXIT
4166 35   12        00660          PULS    A,X
4168 39             00670          RTS
                    00680
          0000      00690          END
```

For **BKSPCE.ASM**, both the Test Routine and the Control Program are just very simple modifications of those used for testing **POLCAT.ASM**.

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0012.ASM
                    00130 * MDJ 2023/02/12
                    00140 *
                    00150 * BKSPCE TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * LOW RAM CURSOR ADDRESS
          0088      00200 CURPOS   EQU     $0088
                    00210
                    00220 * ML FOUNDATION
                    00230 * CORE ADDRESSES
          401F      00240 PUTCHR   EQU     $401F
          407D      00250 PUTCHA   EQU     $407D
          40D7      00260 CRLF     EQU     $40D7
```

```
          4125   PRT2PK  EQU     $4125
          4142   POLCAT  EQU     $4142
          4151   BKSPCE  EQU     $4151
7000                     ORG     $7000
                 * POLCAT TEST
7000 34   03             PSHS    A,CC
                  * DISPLAY PROMPT
7002 86   7E             LDA     #126
7004 BD   407D           JSR     PUTCHA
7007 86   60             LDA     #96
7009 BD   407D           JSR     PUTCHA
                 * GO CHECK ROM
                 * FOR KEY PRESS
700C BD   4142   LBL001  JSR     POLCAT
                 * GO IF NO KEY PRESS
700F 27   FB             BEQ     LBL001
                 * WAS IT THE BREAK KEY?
7011 81   03             CMPA    #3
                 * GO IF YES
7013 27   26             BEQ     LBL004
                 * WAS IT A BACKSPACE?
7015 81   08             CMPA    #8
                 * GO IF YES
7017 27   1D             BEQ     LBL003
                 * WAS IT A CARRIAGE
                 * RETURN?
7019 81   0D             CMPA    #13
                 * GO IF YES
701B 27   14             BEQ     LBL002
                 * WAS IT <= 031 ?
701D 81   20             CMPA    #32
                 * GO IF YES (IGNORE)
701F 25   EB             BLO     LBL001
```

```
                    00740
                    00750 * WAS IT = 96 ?
7021 81   60        00760        CMPA    #96
                    00770
                    00780 * GO IF YES (IGNORE)
7023 27   E7        00790        BEQ     LBL001
                    00800
                    00810 * WAS IT >= 123 ?
7025 81   7B        00820        CMPA    #123
                    00830
                    00840 * GO IF YES (IGNORE)
7027 24   E3        00850        BHS     LBL001
                    00860
                    00870 * PUT IT TO VIDRAM
7029 BD   4125      00880        JSR     PRT2PK
702C BD   407D      00890        JSR     PUTCHA
702F 20   DB        00900        BRA     LBL001
                    00910
                    00920 * DO CRLF
7031 BD   40D7      00930 LBL002 JSR     CRLF
7034 20   D6        00940        BRA     LBL001
                    00950
                    00960 * DO BACKSPACE
7036 BD   4151      00970 LBL003 JSR     BKSPCE
7039 20   D1        00980        BRA     LBL001
                    00990
                    01000 * EXIT
703B 35   03        01010 LBL004 PULS    A,CC
703D 39             01020        RTS
                    01030
          0000      01040        END
```

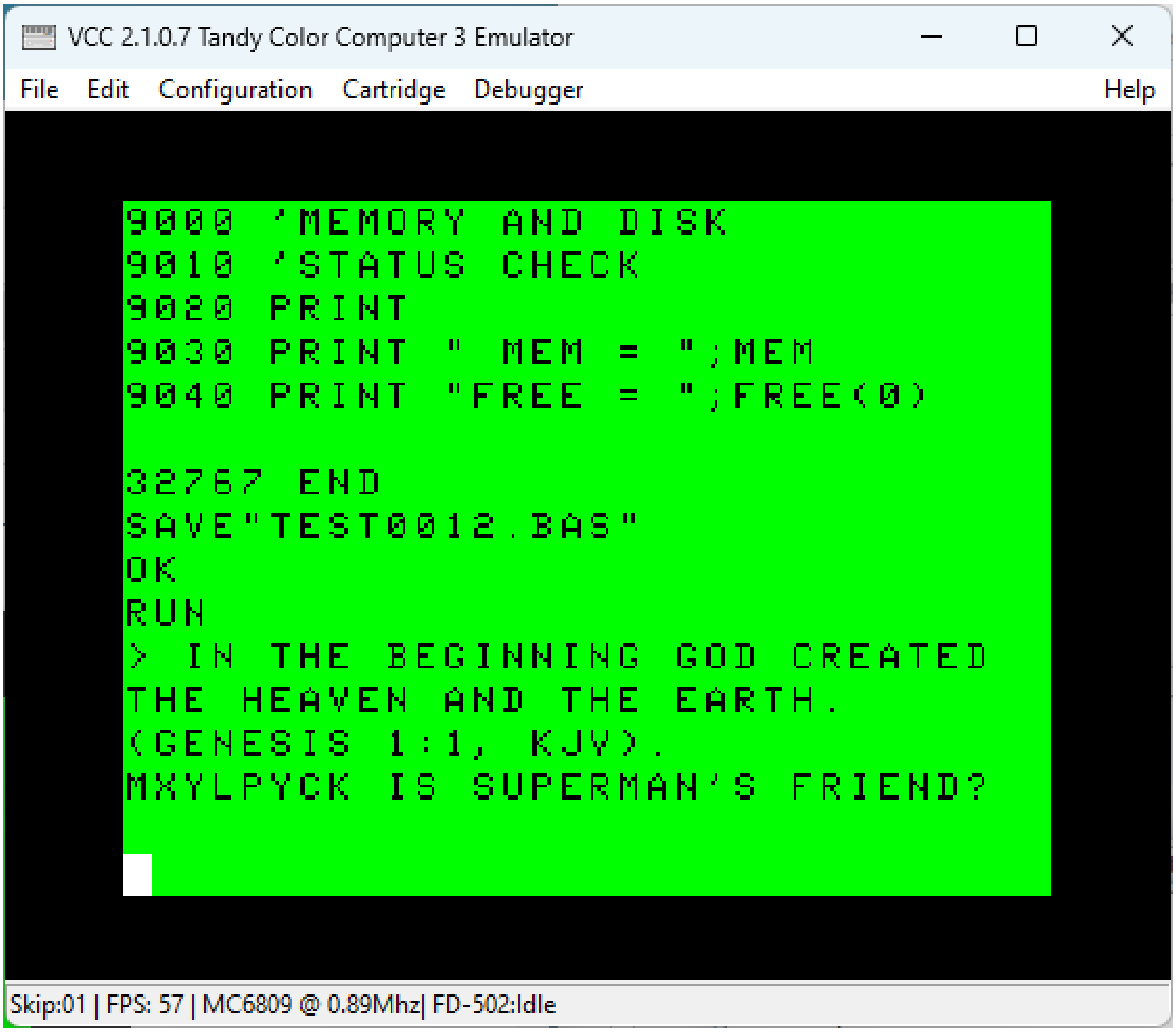
The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0012.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* BKSPCE TEST
1060 '*
1070 '*****
1080 '

1100 'SETUP MEMORY
```

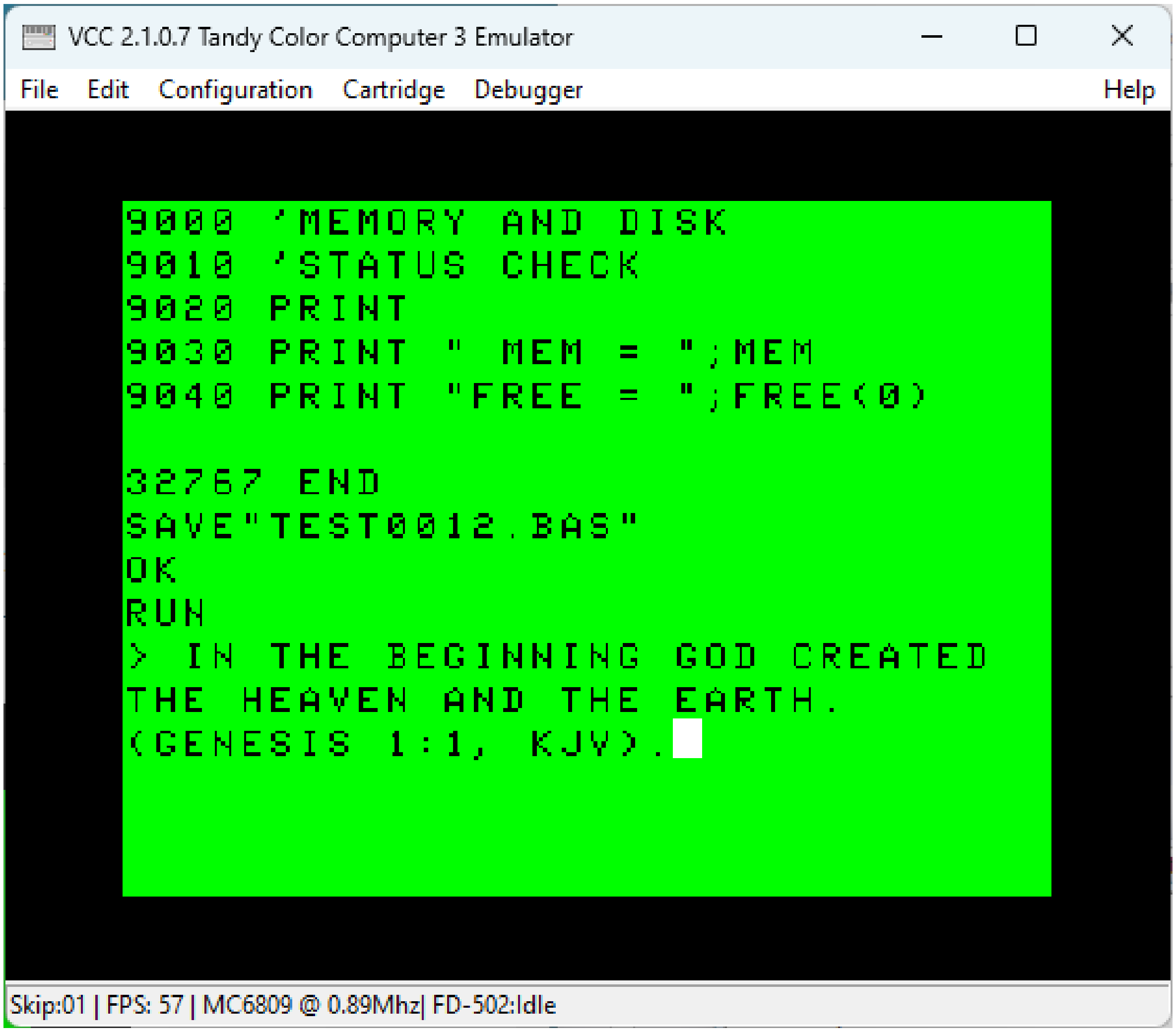
```
1110 CLEAR 200, &H4000
1120 PCLEAR 4
1130 '

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

1300 'LOAD THE
1310 'ML TEST ROUTINE
1320 LOADM "TEST0012.BIN"
1330 '

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
```

Results -

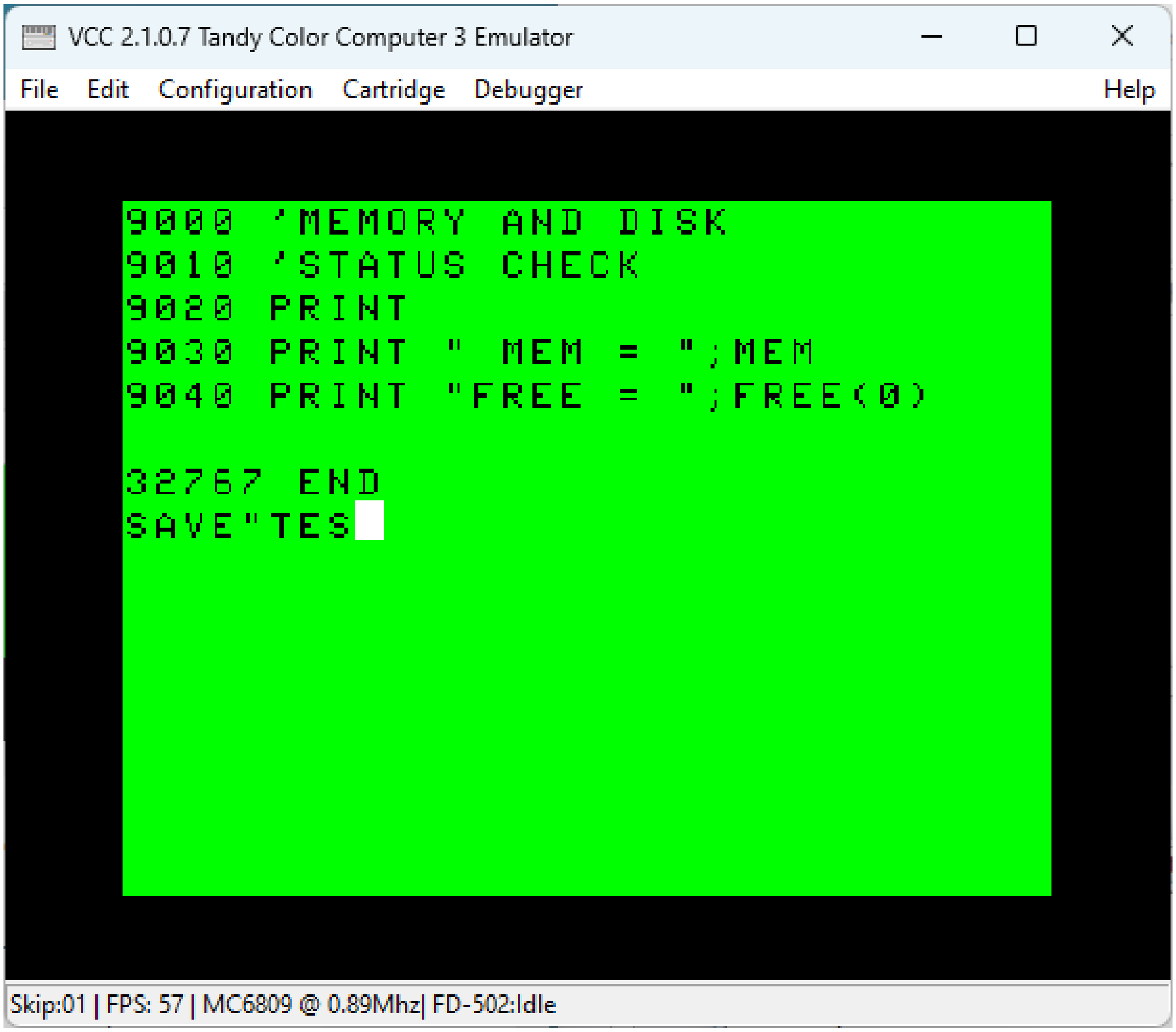
First, we start the test and type some characters:



Then, we backspace to remove some of the characters which we typed:

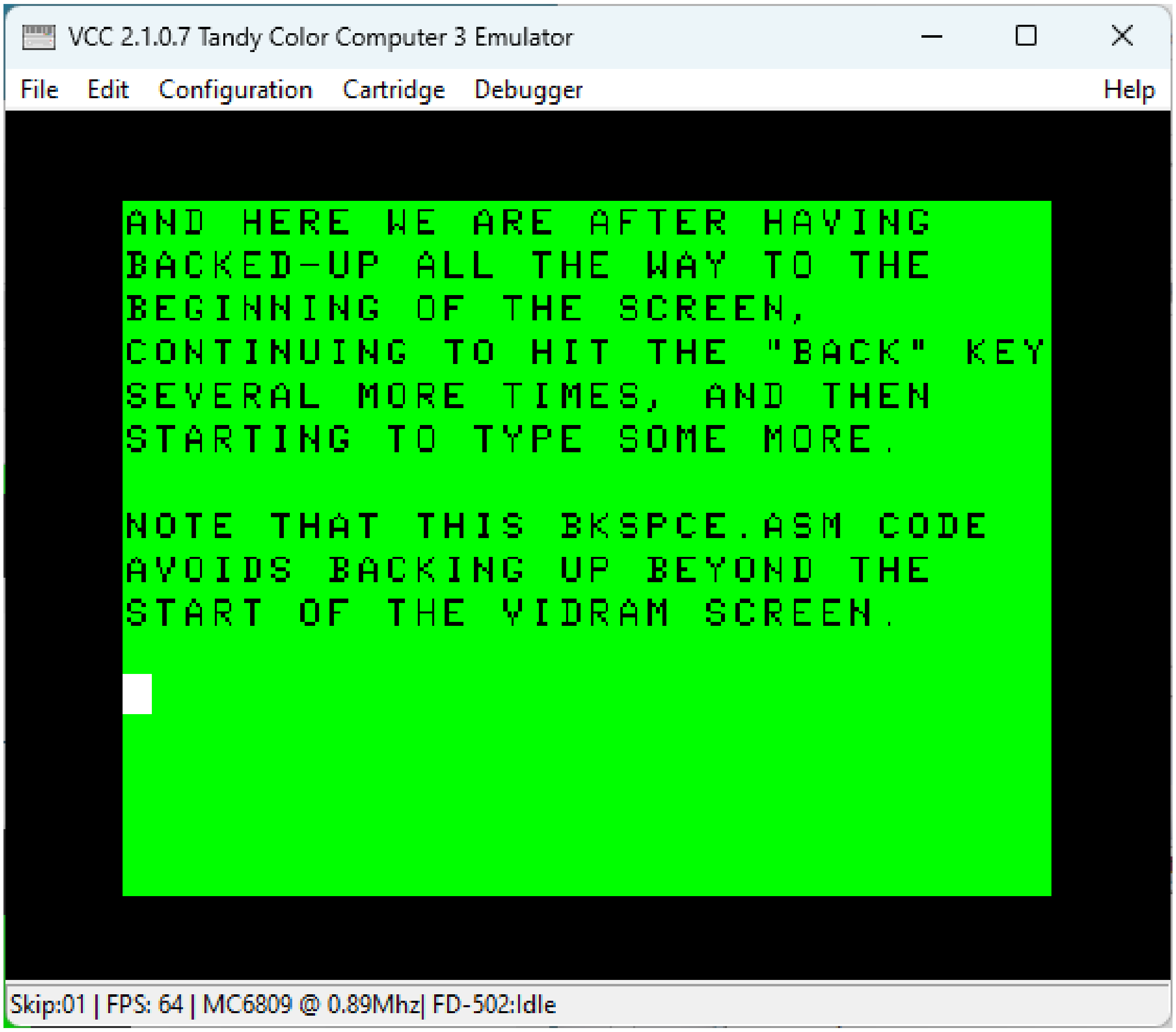


Then, we continue backing up — right past the opening prompt, the RUN command, and more:



And, we can keep on backing up, right on back to the very beginning of the VIDRAM Screen:

And then we can start all over typing again:



All this is fully as expected.

=====

PUTWRD: Put a 16-bit Number To the VIDRAM Screen As Four Hexadecimal Digits At a Specific Position

In the same way that an 8-bit number is called a byte, a 16-bit number is called a word. In this section, the PUTWRD Routine is presented. The mnemonic PUTWRD simply stands for “Put Word”. In the next section, PUTWRA will stand for “Put Word with Advance”.

```
                         *****
                         *
                         * PUTWRD.ASM
                         * MDJ 2023/01/24
                         *
                         * PUTS A 16-BIT NUMBER
                         * TO VIDRAM AS FOUR
                         * HEXADECIMAL DIGITS.
                         *
                         * ENTRY CONDITIONS:
                         * D = THE 16-BIT NUMBER
                         * X = SCREEN LOCATION
                         * ($0400 - $05FC)
                         * CANNOT BE MORE
                         * THAN $05FC
                         * BECAUSE NEED
                         * ROOM TO PUT
                         * 4 CHARACTERS
                         *
                         * EXIT CONDITIONS:
                         * X = NEW SCREEN LOC
                         * ($0404 - $0600)
                         * $0600 INDICATES
                         * END OF VIDRAM
                         * HAS BEEN PASSED
                         *
                         *****

                         * EXTERNAL ROUTINE
                         * ADDRESS
         4025            PUTBYT  EQU     $4025

4169                             ORG     $4169
```

```
                   00430
4169 34    06      00440 PUTWRD  PSHS    D
                   00450
                   00460 * SAVE THE LOW BYTE
416B 34    04      00470         PSHS    B
                   00480
                   00490 * PRINT THE HIGH BYTE
416D BD    4025    00500         JSR     PUTBYT
                   00510
                   00520 * RESTORE THE LOW BYTE
                   00530 * BUT TO REGISTER A
4170 35    02      00540         PULS    A
                   00550
                   00560 * PRINT THE LOW BYTR
4172 BD    4025    00570         JSR     PUTBYT
                   00580
                   00590 * EXIT
4175 35    06      00600         PULS    D
4177 39            00610         RTS
                   00620
           0000    00630         END
```

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0013.ASM
                   00130 * MDJ 2023/02/12
                   00140 *
                   00150 * PUTWRD TEST
                   00160 *
                   00170 *****
                   00180
                   00190 * LOW RAM CURSOR ADDRESS
           0088    00200 CURPOS  EQU     $0088
                   00210
                   00220 * ML FOUNDATION
                   00230 * CORE ADDRESSES
           40D7    00240 CRLF    EQU     $40D7
           4169    00250 PUTWRD  EQU     $4169
                   00260
7000               00270         ORG     $7000
                   00280
                   00290 * PUTWRD TEST
                   00300
```

```
7000 34   16       00310          PSHS    A,B,X
                   00320
7002 CC   7A3D     00330          LDD     #$7A3D
7005 8D   2D       00340          BSR     LBL001
7007 BD   40D7     00350          JSR     CRLF
700A CC   0002     00360          LDD     #$02
700D 8D   25       00370          BSR     LBL001
700F BD   40D7     00380          JSR     CRLF
7012 CC   0000     00390          LDD     #$0
7015 8D   1D       00400          BSR     LBL001
7017 BD   40D7     00410          JSR     CRLF
701A CC   FFFF     00420          LDD     #$FFFF
701D 8D   15       00430          BSR     LBL001
701F BD   40D7     00440          JSR     CRLF
7022 CC   03CE     00450          LDD     #$3CE
7025 8D   0D       00460          BSR     LBL001
7027 BD   40D7     00470          JSR     CRLF
702A CC   00AB     00480          LDD     #$00AB
702D 8D   05       00490          BSR     LBL001
702F BD   40D7     00500          JSR     CRLF
7032 20   08       00510          BRA     LBL002
                   00520
7034 9E   88       00530 LBL001   LDX     CURPOS
7036 BD   4169     00540          JSR     PUTWRD
7039 9F   88       00550          STX     CURPOS
703B 39            00560          RTS
                   00570
                   00580 * EXIT
703C 35   16       00590 LBL002   PULS    A,B,X
703E 39            00600          RTS
                   00610
          0000     00620          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0013.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* PUTWRD 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0013.BIN"
1330 '

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.

=====

PUTWRA: Put a 16-bit Number To the VIDRAM Screen As Four Hexadecimal Digits At the Cursor Position and Advance the Cursor

In the same way that an 8-bit number is called a byte, a 16-bit number is called a word. In the previous section, the PUTWRD Routine was presented. The mnemonic PUTWRD simply stands for “Put Word”. In this section, PUTWRA stands for “Put Word with Advance

```
                    00100 *****
                    00110 *
                    00120 * PUTWRA.ASM
                    00130 * MDJ 2023/01/24
                    00140 *
                    00150 * PUTS A 16-BIT NUMBER
                    00160 * TO VIDRAM AS FOUR
                    00170 * HEXADECIMAL DIGITS
                    00180 *
                    00190 * ADVANCES THE CURSOR;
                    00200 * SCROLLS THE SCREEN
                    00210 * IF REQUIRED
                    00220 *
                    00230 * ENTRY CONDITIONS:
                    00240 * D = THE 16-BIT NUMBER
                    00250 *
                    00260 * EXIT CONDITIONS:
                    00270 * NONE
                    00280 *
                    00290 *****
                    00300
                    00310 * EXTERNAL ROUTINE
                    00320 * ADDRESS
          409E      00330 PUTBYA  EQU     $409E
                    00340
4178                00350         ORG     $4178
                    00360
4178 34   06        00370 PUTWRA  PSHS    D
                    00380
                    00390 * SAVE THE LOW BYTE
417A 34   04        00400         PSHS    B
```

```
                    00410
                    00420 * PRINT THE HIGH BYTE
417C BD   409E      00430        JSR     PUTBYA
                    00440
                    00450 * RESTORE THE LOW BYTE
                    00460 * BUT INTO REGISTER A
417F 35   02        00470        PULS    A
                    00480
                    00490 * PRINT THE LOW BYTE
4181 BD   409E      00500        JSR    PUTBYA
                    00510
                    00520 * EXIT
4184 35   06        00530        PULS    D
4186 39             00540        RTS
                    00550
          0000      00560        END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0014.ASM
                    00130 * MDJ 2023/02/12
                    00140 *
                    00150 * PUTWRA TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * ML FOUNDATION
                    00200 * CORE ADDRESSES
          40D7      00210 CRLF   EQU     $40D7
          4178      00220 PUTWRA EQU     $4178
                    00230
7000                00240        ORG     $7000
                    00250
                    00260 * PUTWRA TEST
                    00270
7000 34   06        00280        PSHS    A,B
                    00290
7002 CC   7A3D      00300        LDD     #$7A3D
7005 BD   4178      00310        JSR     PUTWRA
7008 BD   40D7      00320        JSR     CRLF
700B CC   0002      00330        LDD     #$02
700E BD   4178      00340        JSR     PUTWRA
7011 BD   40D7      00350        JSR     CRLF
```

```
7014 CC   0000     00360          LDD     #$0
7017 BD   4178     00370          JSR     PUTWRA
701A BD   40D7     00380          JSR     CRLF
701D CC   FFFF     00390          LDD     #$FFFF
7020 BD   4178     00400          JSR     PUTWRA
7023 BD   40D7     00410          JSR     CRLF
7026 CC   03CE     00420          LDD     #$3CE
7029 BD   4178     00430          JSR     PUTWRA
702C BD   40D7     00440          JSR     CRLF
702F CC   00AB     00450          LDD     #$00AB
7032 BD   4178     00460          JSR     PUTWRA
7035 BD   40D7     00470          JSR     CRLF
                   00480
                   00490 * EXIT
7038 35   06       00500          PULS   A,B
703A 39            00510          RTS
                   00520
          0000     00530          END
```

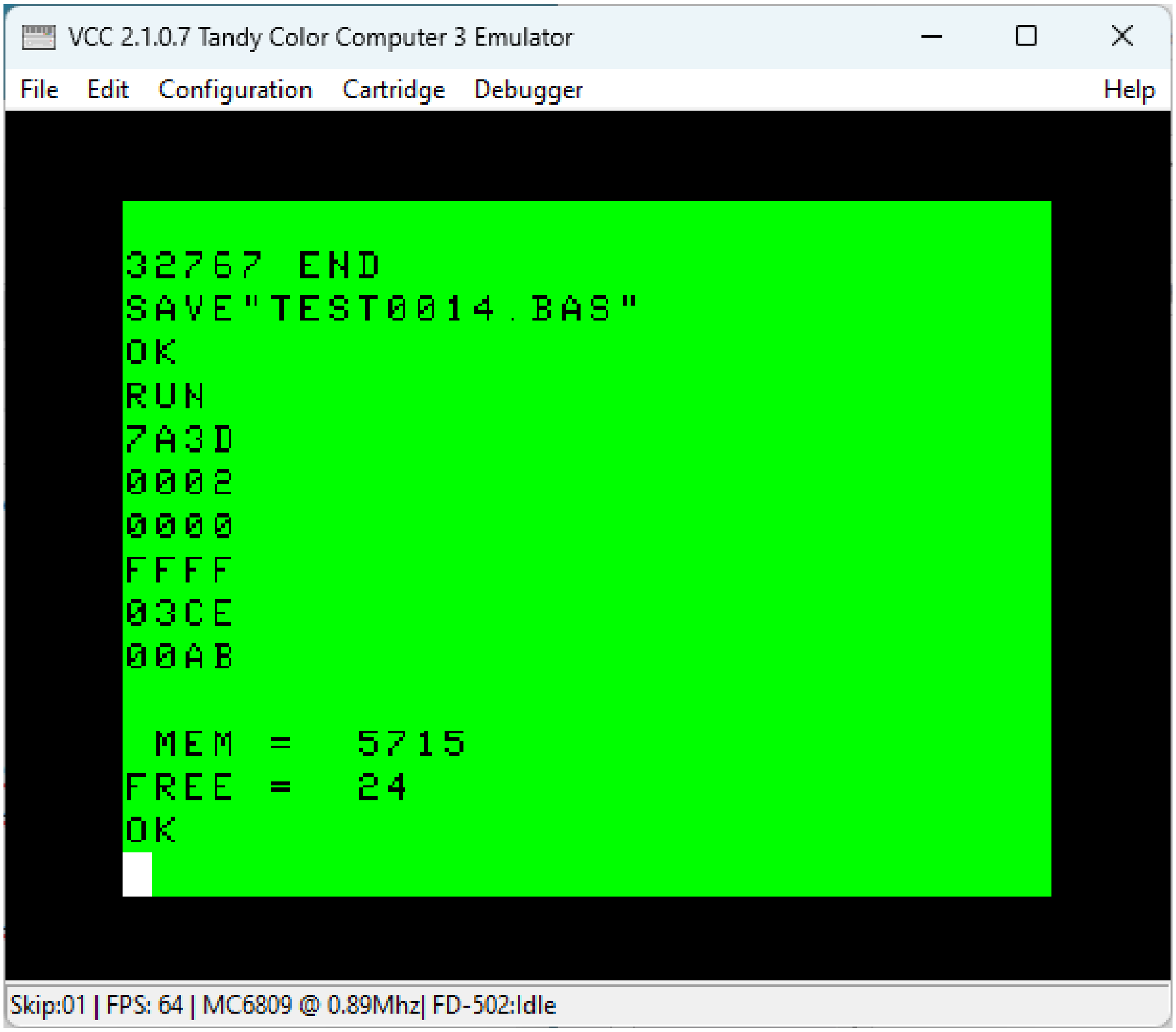
———

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0014.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* PUTWRA 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0014.BIN"
1330 '
```

```
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.

=====

SOUND: Sound a Tone of a Specified Frequency for a Specified Duration

This is essentially the BASIC ROM's SOUND command, but just called from the Machine Language of the ML Foundation Core.

```
               00100 *****
               00110 *
               00120 * SOUND.ASM
               00130 * MDJ 2023/01/24
               00140 *
               00150 * SOUNDS A TONE
               00160 * FOR A SPECIFIED
               00170 * DURATION
               00180 *
               00190 * USES THE BUILT-IN
               00200 * ROM SOUND ROUTINE
               00210 *
               00220 * ENTRY CONDITIONS:
               00230 * A = TONE (0-255)
               00240 * X = DURATION (0-65535)
               00250 *
               00260 * EXIT CONDITIONS:
               00270 * NONE
               00280 *
               00290 * ANY PROGRAM CALLING THIS
               00300 * NEEDS TO PSHS/PULS A,X
               00310 *
               00320 *****
               00330
               00340 * RAMROM TRIGGER ADDRESS
FFDE           00350 RAMROM  EQU     $FFDE
               00360
               00370 * ALLRAM TRIGGER ADDRESS
FFDF           00380 ALLRAM  EQU     $FFDF
               00390
               00400 * LOW RAM TONE
008C           00410 SNDTON  EQU     $8C
               00420
               00430 * LOW RAM DURATION
```

```
          008D     00440 SNDDUR  EQU     $8D
                   00450
                   00460 * ROM SOUND ADDRESS
          A956     00470 XSOUND  EQU     $A956
                   00480
4187               00490         ORG     $4187
                   00500
4187 97   8C       00510 SOUND   STA     SNDTON  TONE
4189 9F   8D       00520         STX     SNDDUR  DURATION
                   00530
                   00540 * SET RAMROM MODE
418B B7   FFDE     00550         STA     RAMROM
                   00560
                   00570 * GO DO ROM SOUND
418E BD   A956     00580         JSR     XSOUND
                   00590
                   00600 * SET ALLRAM MODE
4191 B7   FFDF     00610         STA     ALLRAM
                   00620
                   00630 * EXIT
4194 39            00640         RTS
                   00650
          0000     00660         END
```

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0015.ASM
                   00130 * MDJ 2023/02/12
                   00140 *
                   00150 * SOUND TEST
                   00160 *
                   00170 *****
                   00180
                   00190 * ML FOUNDATION
                   00200 * CORE ADDRESS
          4187     00210 SOUND   EQU     $4187
                   00220
7000               00230         ORG     $7000
                   00240
                   00250 * SOUND TEST
                   00260
7000 34   12       00270         PSHS    A,X
7002 86   40       00280         LDA     #64
```

```
7004 8E   0040     00290          LDX     #64
7007 BD   4187     00300          JSR     SOUND
                   00310
                   00320 * EXIT
700A 35   12       00330          PULS    A,X
700C 39            00340          RTS
                   00350
          0000     00360          END
```

____

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0015.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* SOUND 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0015.BIN"
1330 '

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:

The tone sounds as expected. But you’ll have to run the test for yourself in order to verify the result.

=====

BEEP: The Response To the ASCII $07 BEL Character

This is to be the response whenever the ASCII $07 BEL Character is encountered.

```
                    00100 *****
                    00110 *
                    00120 * BEEP.ASM
                    00130 * MDJ 2023/01/24
                    00140 *
                    00150 * SOUNDS A SHORT TONE
                    00160 * INTENDED AS THE RESPONSE
                    00170 * TO THE BEL = 07 CHARACTER
                    00180 *
                    00190 * ENTRY CONDITIONS:
                    00200 * NONE
                    00210 *
                    00220 * EXIT CONDITIONS:
                    00230 * NONE
                    00240 *
                    00250 *****
                    00260
                    00270 * EXTERNAL ROUTINE
                    00280 * ADDRESS
          4187      00290 SOUND   EQU     $4187
                    00300
4195                00310         ORG     $4195
                    00320
4195 34   12        00330 BEEP    PSHS    A,X
4197 86   C4        00340         LDA     #196    TONE
4199 8E   0001      00350         LDX     #2      DURATION
                    00360
                    00370 * GO DO THE SOUND
419C BD   4187      00380         JSR     SOUND
                    00390
                    00400 * EXIT
419F 35   12        00410         PULS    A,X
41A1 39             00420         RTS
                    00430
          0000      00440         END
```

The Assembly Language Test Routine:

```
                  00100 *****
                  00110 *
                  00120 * TEST0016.ASM
                  00130 * MDJ 2023/02/12
                  00140 *
                  00150 * BEEP TEST
                  00160 *
                  00170 *****
                  00180
                  00190 * ML FOUNDATION
                  00200 * CORE ADDRESSES
          4195    00210 BEEP    EQU     $4195
                  00220
7000              00230         ORG     $7000
                  00240
                  00250 * BEEP TEST
                  00260
7000 BD   4195    00270         JSR     BEEP
                  00280
7003 39           00290         RTS
                  00300
          0000    00310         END
```

____

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0016.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* BEEP 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0016.BIN"
1330 '

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:

The tone sounds as expected. But you'll have to run the test for yourself in order to verify the result.

=====

COLD: Performs a Cold Start to Disk Basic

Normally, whichever routine you design in assembly language (and call from BASIC) should simply end in an RTS which will return you to just after the calling point in that BASIC program.

But, you may occasionally have to escape from a misbehaving assembly language routine, and COLD.ASM will do that for you. It simply discards whatever your routine was doing and dumps you directly to a Cold Start.

You could, I suppose, perform a Warm Start instead by doing a LDA #$55 instead of CLRA, but my experience with Warm Starts has not been very good. It seems that, no matter what I try to do after the Warm Start, the result is an immediate "OS Error", or something else equally noxious. Proceed at your own risk.

COLD.ASM does a Cold Start to Disk BASIC if the Disk BASIC ROM Cartridge is installed. This is confirmed via TEST0017.ASM and TEST0017.BAS below.

Theoretically, it would perform a Cold Start to Extended Color BASIC if the Disk BASIC ROM Cartridge is not installed. However, there would be no way to load this if the Disk BASIC ROM Cartridge is not installed. For completeness, TEST0018.BAS below provides the equivalent Cold Start mechanism via poking the bytes of COLD.ASM directly into memory.

```
          *****
          *
          * COLD.ASM
          * MDJ 2023/01/26
          *
          * FORCE A COLD START
          *
          * DOES A COLD START TO
          * DISK BASIC IF THE
          * FD-502 DISK BASIC ROM
          * CARTRIDGE IS INSTALLED.
          *
          * OTHERWISE, DOES A
          * COLD START TO EXTENDED
          * COLOR BASIC.
          *
          *****

          * RAMROM TRIGGER ADDRESS
FFDE      RAMROM  EQU     $FFDE

```

```
41A2                00310          ORG     $41A2
                    00320
41A2 4F             00330 COLD     CLRA
                    00340
                    00350 * SET RAMROM MODE
41A3 B7   FFDE      00360          STA     RAMROM
                    00370
                    00380 * CLEAR WARM START FLAG
41A6 97   71        00390          STA     $71
                    00400
                    00410 * HIGH MEMORY RESET VECTOR
41A8 6E   9F FFFE   00420          JMP     [$FFFE]
                    00430
                    00440 * FOR ASSEMBLY END CHECK ONLY
                    00450 * CAN BE OVERWRITTEN AS DESIRED
41AC 12             00460 XNDCHK   NOP
                    00470
          0000      00480          END          0000     00410
END
```

——-

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0017.ASM
                    00130 * MDJ 2023/02/12
                    00140 *
                    00150 * COLD TEST
                    00160 *
                    00170 *****
                    00180
          41A2      00190 COLD     EQU     $41A2
                    00200
7000                00210          ORG     $7000
                    00220
                    00230 * COLD TEST
                    00240
7000 BD   41A2      00250          JSR     COLD
                    00260
7003 39             00270          RTS
                    00280
          0000      00290          END
```

——

The BASIC Language Control Program for when the FD-502 Disk BASIC ROM Cartridge is installed:

```
1000 '*****
1010 '*
1020 '* TEST0017.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* COLD 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0017.BIN"
1330 '

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 '
```

```
32767 END
```

The BASIC Language Control Program for when the FD-502 Disk BASIC ROM Cartridge is not installed:

```
1000 '*****
1010 '*
1020 '* TEST0018.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* COLD TEST
1060 '*
1070 '* WITHOUT THE DISK
1080 '* SYSTEM AND WITHOUT
1090 '* THE DISK BASIC ROM
1100 '*
1110 '*****
1120 '

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

2000 'POKE THE EQUIVALENT
2010 'OF THE COLD START
2020 'ROUTINE INTO MEMORY,

2050 'FOR THE NO-DISK TEST,
2060 'REMOVE THE FD-502
2070 'CARTRIDGE BEFORE
2080 'LOADING THIS PROGRAM
2090 'INTO MEMORY.

2500 POKE &H41A2, &H4F  'CLRA
2510 POKE &H41A3, &HB7  'STA   RAMROM
2520 POKE &H41A4, &HFF
2530 POKE &H41A5, &HDE
2540 POKE &H41A6, &H97  'STA   $71
2550 POKE &H41A7, &H71
2560 POKE &H41A8, &H6E  'JMP   [$FFFE]
2570 POKE &H41A9, &H9F
2580 POKE &H41AA, &HFF
2590 POKE &H41AB, &HFE

3000 'GO DO THE COLD START
```

```
3010 EXEC &H41A2

32767 END
```

—

Result:

As expected, both of these result in a Cold Start to BASIC. Try them for yourself.

=====

PRTCHR: Put a Character To the VIDRAM Screen At a Specified Position While Adjusting the Print Code To its Corresponding Poke Code

This routine does the same thing as PUTCHR.ASM, except that it runs the Poke Code through PRT2PK before putting it to the VIDRAM Screen.

```
                00100 *****
                00110 *
                00120 * PRTCHR.ASM
                00130 * MDJ 2023/01/26
                00140 *
                00150 * PUT A CHARACTER CODE
                00160 * TO THE VIDEO RAM
                00170 * WHILE HANDLING
                00180 * CONTROL CODES (0-31)
                00190 *
                00200 * ENTRY CONDITIONS:
                00210 * A = CHARACTER CODE
                00220 * X = SCREEN LOCATION
                00230 * ($0400 - $05FF)
                00240 *
                00250 * EXIT CONDITIONS
                00260 * NONE
                00270 *
                00280 *****
                00290
                00300 * LOW RAM CURSOR ADDRESS
0088            00310 CURPOS  EQU     $0088
                00320
                00330 * EXTERNAL ROUTINE
                00340 * ADDRESSES
401F            00350 PUTCHR  EQU     $401F
40D7            00360 CRLF    EQU     $40D7
4125            00370 PRT2PK  EQU     $4125
4151            00380 BKSPCE  EQU     $4151
4195            00390 BEEP    EQU     $4195
41A2            00400 COLD    EQU     $41A2
                00410
```

```
41AC                00420          ORG     $41AC
                    00430
41AC 81   20        00440 PRTCHR   CMPA    #32      IS IT A CONTROL
CODE?
41AE 24   24        00450          BHS     LBL005   GO IF NO
41B0 81   07        00460          CMPA    #7       IS IT A BEL
CHARACTER?
41B2 27   0E        00470          BEQ     LBL001   GO IF YES
41B4 81   08        00480          CMPA    #8       IS IT A BACKSPACE?
41B6 27   0F        00490          BEQ     LBL002   GO IF YES
41B8 81   0D        00500          CMPA    #13      IS IT A CR?
41BA 27   10        00510          BEQ     LBL003   GO IF YES
41BC 81   03        00520          CMPA    #$03     IS IT THE BREAK KEY
                    00530 *                          = ESC KEY ON PC
KEYBOARD
                    00540 *                          = ETX = $03 KEYCODE
= 3
41BE 27   11        00550          BEQ     LBL004   GO IF YES
41C0 20   18        00560          BRA     LBL006   OTHERWISE, IGNORE
41C2 BD   4195      00570 LBL001   JSR     BEEP     BEL CHARACTER
41C5 20   13        00580          BRA     LBL006
41C7 BD   4151      00590 LBL002   JSR     BKSPCE   BACKSPACE
41CA 20   0E        00600          BRA     LBL006
41CC BD   40D7      00610 LBL003   JSR     CRLF     CARRIAGE RETURN
41CF 20   09        00620          BRA     LBL006
41D1 7E   41A2      00630 LBL004   JMP     COLD     GO DO A COLD START
41D4 BD   4125      00640 LBL005   JSR     PRT2PK   CONVERT TO POKE CODE

41D7 BD   401F      00650          JSR     PUTCHR   PUT IT TO THE SCREEN
41DA 39             00660 LBL006   RTS
                    00670
          0000      00680          END
```

——-

The Assembly Language Test Routine:

```
00100 *****
00110 *
00120 * TEST0019.ASM
00130 * MDJ 2023/02/12
00140 *
00150 * PRTCHR 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 * ML FOUNDATION
                    00270 * CORE ADDRESSES
          4000      00280 REGXFR  EQU     $4000
          400E      00290 VIDCLS  EQU     $400E
          4142      00300 POLCAT  EQU     $4142
          41AC      00310 PRTCHR  EQU     $41AC
                    00320
7000                00330         ORG     $7000
                    00340
                    00350 * PRTCHR TEST
                    00360
7000 34   12        00370         PSHS    A,X
7002 BD   400E      00380         JSR     VIDCLS
7005 8E   0500      00390         LDX     #$0500  MIDDLE LINE OF
SCREEN
7008 9F   88        00400         STX     CURPOS  CURSOR
700A 34   03        00410         PSHS    A,CC    DO DOUBLE PSHS JUST
FOR SAFETY
DURING TESTING
700C BD   4142      00420 LBL001  JSR     POLCAT  GET KEYPRESS
700F 27   FB        00430         BEQ     LBL001  GO IF Z-BIT OF CC
SET
7011 81   39        00440         CMPA    #$39    ASCII "9" USED TO
SIGNAL NORMAL
 EXIT IN THIS TEST
7013 27   0B        00450         BEQ     LBL003
7015 81   37        00460         CMPA    #$37    ASCII "7" USED TO
SIGNAL A "BEE
P"
7017 26   02        00470         BNE     LBL002
7019 86   07        00480         LDA     #$07    CHANGE IT TO ASCII
BEL = $07 =
7
701B BD   41AC      00490 LBL002  JSR     PRTCHR  PUT CHARACTER TO
SCREEN
701E 20   EC        00500         BRA     LBL001  RETURN FOR NEXT
KEYPRESS
7020 35   03        00510 LBL003  PULS    A,CC
                    00520
                    00530 * EXIT
7022 35   12        00540         PULS   A,X
```

```
7024 39         00550       RTS
                00560
          0000  00570       END
```

____

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0019.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* PRTCHR 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0019.BIN"
1330 '

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:

This actually IS as expected. I typed in “THIS IS A BEEP TEST”. Each character overwrote its predecessor because PRTCHR does not advance the cursor. Then I pressed the “7” key and it produced the expected “BEEP”. Then I pressed the “9” key and it exited back to BASIC as expected. Try it for yourself.

=====

PRTCHA: Put a Character To the VIDRAM Screen at the Cursor Position and Advance the Cursor While Adjusting the Print Code To its Corresponding Poke Code

This routine does the same thing as PUTCHA.ASM, except that it runs the Poke Code through PRT2PK before putting it to the VIDRAM Screen.

```
               00100 *****
               00110 *
               00120 * PRTCHA.ASM
               00130 * MDJ 2023/01/26
               00140 *
               00150 * PUT A CHARACTER CODE
               00160 * TO THE VIDEO RAM
               00170 * WHILE HANDLING
               00180 * CONTROL CODES (0-31)
               00190 *
               00200 * ADVANCES THE CURSON
               00210 * AND SCROLLS THE SCREEN
               00220 * IF REQUIRED
               00230 *
               00240 * ENTRY CONDITIONS:
               00250 * A = CHARACTER CODE
               00260 *
               00270 * EXIT CONDITIONS
               00280 * NONE
               00290 *
               00300 *****
               00310
               00320 * LOW RAM CURSOR ADDRESS
0088           00330 CURPOS  EQU     $0088
               00340
               00350 * EXTERNAL ROUTINE
               00360 * ADDRESSES
407D           00370 PUTCHA  EQU     $407D
40D7           00380 CRLF    EQU     $40D7
4125           00390 PRT2PK  EQU     $4125
```

```
          4151     00400 BKSPCE  EQU     $4151
          4195     00410 BEEP    EQU     $4195
          41A2     00420 COLD    EQU     $41A2
                   00430
41DB               00440         ORG     $41DB
                   00450
41DB 81   20       00460 PRTCHA  CMPA    #32     IS IT A CONTROL
CODE?
41DD 24   24       00470         BHS     LBL005  GO IF NO
41DF 81   07       00480         CMPA    #7      IS IT A BEL
CHARACTER?
41E1 27   0E       00490         BEQ     LBL001  GO IF YES
41E3 81   08       00500         CMPA    #8      IS IT A BACKSPACE?
41E5 27   0F       00510         BEQ     LBL002  GO IF YES
41E7 81   0D       00520         CMPA    #13     IS IT A CR?
41E9 27   10       00530         BEQ     LBL003  GO IF YES
41EB 81   03       00540         CMPA    #$03    IS IT THE BREAK KEY
                   00550 *                        = ESC KEY ON PC
KEYBOARD
                   00560 *                        = ETX = $03 KEYCODE
= 3
41ED 27   11       00570         BEQ     LBL004  GO IF YES
41EF 20   18       00580         BRA     LBL006  OTHERWISE, IGNORE
41F1 BD   4195     00590 LBL001  JSR     BEEP    BEL CHARACTER
41F4 20   13       00600         BRA     LBL006
41F6 BD   4151     00610 LBL002  JSR     BKSPCE  BACKSPACE
41F9 20   0E       00620         BRA     LBL006
41FB BD   40D7     00630 LBL003  JSR     CRLF    CARRIAGE RETURN
41FE 20   09       00640         BRA     LBL006
4200 7E   41A2     00650 LBL004  JMP     COLD    GO DO A COLD START
4203 BD   4125     00660 LBL005  JSR     PRT2PK  CONVERT TO POKE CODE
4206 BD   407D     00670         JSR     PUTCHA  PUT IT TO THE SCREEN
4209 39            00680 LBL006  RTS
                   00690
          0000     00700         END
```

——-

The Assembly Language Test Routine:

```
00100 *****
00110 *
00120 * TEST0020.ASM
00130 * MDJ 2023/02/12
00140 *
00150 * PRTCHA 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 * ML FOUNDATION
                    00270 * CORE ADDRESSES
          400E      00280 VIDCLS  EQU     $400E
          4142      00290 POLCAT  EQU     $4142
          41DB      00300 PRTCHA  EQU     $41DB
                    00310
7000                00320         ORG     $7000
                    00330
                    00340 * PRTCHA TEST
                    00350
7000 34   12        00360         PSHS    A,X
7002 BD   400E      00370         JSR     VIDCLS
7005 8E   0420      00380         LDX     #$0420  SECOND LINE OF
SCREEN
7008 9F   88        00390         STX     CURPOS  CURSOR
700A 34   03        00400         PSHS    A,CC    DO DOUBLE PSHS JUST
FOR SAFETY
DURING TESTING
700C BD   4142      00410 LBL001  JSR     POLCAT  GET KEYPRESS
700F 27   FB        00420         BEQ     LBL001  GO IF Z-BIT OF CC
SET
7011 81   39        00430         CMPA    #$39    ASCII "9" USED TO
SIGNAL NORMAL
 EXIT IN THIS TEST
7013 27   0B        00440         BEQ     LBL003
7015 81   37        00450         CMPA    #$37    ASCII "7" USED TO
SIGNAL A "BEE
P"
7017 26   02        00460         BNE     LBL002
7019 86   07        00470         LDA     #$07    CHANGE IT TO ASCII
BEL = $07 =
7
701B BD   41DB      00480 LBL002  JSR     PRTCHA  PUT CHARACTER TO
SCREEN
701E 20   EC        00490         BRA     LBL001  RETURN FOR NEXT
KEYPRESS
7020 35   03        00500 LBL003  PULS    A,CC
                    00510
```

```
                    00520 * EXIT
7022 35   12        00530          PULS    A,X
7024 39             00540          RTS
                    00550
          0000      00560          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0020.BAS
1030 '* MDJ 2023/02/12
1040 '*
1050 '* PRTCHA 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0020.BIN"
1330 '

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.

Bible Note: The quote of the full section (Revelation 20:11-15, ESV) is:

> [11] Then I saw a great white throne and him who was seated on it. From his
> presence earth and sky fled away, and no place was found for them. [12] And I saw
> the dead, great and small, standing before the throne, and books were opened.
> Then another book was opened, which is the book of life. And the dead were
> judged by what was written in the books, according to what they had done. [13] And
> the sea gave up the dead who were in it, Death and Hades gave up the dead who
> were in them, and they were judged, each one of them, according to what they
> had done. [14] Then Death and Hades were thrown into the lake of fire. This is the
> second death, the lake of fire. [15] And if anyone's name was not found written in
> the book of life, he was thrown into the lake of fire.

So, how do you get your name into that book of life? Hebrews 9:27-28 (NIV) says:

> 27 Just as man is destined to die once, and after that to face judgment, 28 so Christ was sacrificed once to take away the sins of many people; and he will appear a second time, not to bear sin, but to bring salvation to those who are waiting for him.

And Acts 16:31 (ESV) says: "… Believe in the Lord Jesus, and you will be saved …"

=====

PRTS00: Prints a Null-Terminated String To the VIDRAM Screen at the Current Cursor Position

The terminating NUL is not “printed”. ETX forces a Cold Start. BEL, BS, and CR are “printed”. All other control codes are ignored.

```
00100 *****
00110 *
00120 * PRTS00.ASM
00130 * MDJ 2023/01/27
00140 *
00150 * PRINTS A NULL-TERMINATED
00160 * STRING ($00) BEGINNING
00170 * AT THE CURRENT CURSOR
00180 * LOCATION
00190 *
00200 * THE NUL IS NOT COUNTED
00210 * AS PART OF THE STRING
00220 * AND IS NOT PRINTED
00230 *
00240 * ANY INTERNAL
00250 *   ETX ($03) ESC/BREAK KEY
00260 * FORCES A COLD START
00270 *
00280 * ANY INTERNAL
00290 *   BEL ($07)
00300 *   BS  ($08) BACKSPACE KEY
00310 *   CR  ($0D) ENTER KEY
00320 * CHARACTERS ARE "PRINTED"
00330 *
00340 * ALL OTHER CONTROL CODES
00350 * ($01-$31) ARE IGNORED
00360 *
00370 * ENTRY CONDITIONS:
00380 * X = START ADDRESS
00390 *     OF THE STRING
00400 *
00410 * EXIT CONDITIONS:
```

```
                    00420 * NONE
                    00430 *
                    00440 *****
                    00450
                    00460 * EXTERNAL ROUTINE
                    00470 * ADDRESS
          41DB      00480 PRTCHA  EQU     $41DB
                    00490
420A                00500         ORG     $420A
                    00510
420A 34   02        00520 PRTS00  PSHS    A
420C A6   80        00530 LBL001  LDA     ,X+     GET A CHARACTER
420E 27   05        00540         BEQ     LBL002  GO IF NUL ($00)
4210 BD   41DB      00550         JSR     PRTCHA  GO PRINT CHARACTER
4213 20   F7        00560         BRA     LBL001  RETURN FOR NEXT
CHARACTER
4215 35   02        00570 LBL002  PULS    A
                    00580
                    00590 *EXIT
4217 39             00600         RTS
                    00610
          0000      00620         END
```

——-

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0021.ASM
                    00130 * MDJ 2023/02/15
                    00140 *
                    00150 * PRTS00 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 * ML FOUNDATION
                    00270 * CORE ADDRESSES
          400E      00280 VIDCLS  EQU     $400E
          40D7      00290 CRLF    EQU     $40D7
```

```
          420A      00300 PRTS00  EQU     $420A
                    00310
7000                00320         ORG     $7000
                    00330
                    00340 * PRTSOO TEST
                    00350
7000 34   10        00360         PSHS    X
7002 7E   7120      00370         JMP      LBL001
7005      54        00380 STR000  FCC     /THIS IS A NUL-TERMINATED
STRING/
          48
          49
          53
          20
          49
          53
          20
          41
          20
          4E
          55
          4C
          2D
          54
          45
          52
          4D
          49
          4E
          41
          54
          45
          44
          20
          53
          54
          52
          49
          4E
          47
7024      00        00390 NUL0    FCB     $00     NUL
7025      54        00400 STR001  FCC     /THIS IS A NUL-TERMINATED
STRING WITH A
N EMBED/
          48
          49
          53
```

```
            20
            49
            53
            20
            41
            20
            4E
            55
            4C
            2D
            54
            45
            52
            4D
            49
            4E
            41
            54
            45
            44
            20
            53
            54
            52
            49
            4E
            47
            20
            57
            49
            54
            48
            20
            41
            4E
            20
            45
            4D
            42
            45
            44
7052        07        00410        FCB      $07
7053        44        00420        FCC      /DED BEL/
            45
            44
            20
            42
```

```
            45
            4C
705A        00          00430 NUL1    FCB     $00     NUL
705B        54          00440 STR002  FCC     /THIS IS A NUL-TERMINATED
STR/
            48
            49
            53
            20
            49
            53
            20
            41
            20
            4E
            55
            4C
            2D
            54
            45
            52
            4D
            49
            4E
            41
            54
            45
            44
            20
            53
            54
            52
7077        08          00450         FCB     $08
7078        49          00460         FCC     /ING WITH AN EMBEDDED
BACKSPACE/
            4E
            47
            20
            57
            49
            54
            48
            20
            41
            4E
            20
            45
```

```
            4D
            42
            45
            44
            44
            45
            44
            20
            42
            41
            43
            4B
            53
            50
            41
            43
            45
7096        00        00470 NUL2    FCB     $00     NUL
7097        54        00480 STR003  FCC     /THIS IS A NUL-TERMINATED
STRING /
            48
            49
            53
            20
            49
            53
            20
            41
            20
            4E
            55
            4C
            2D
            54
            45
            52
            4D
            49
            4E
            41
            54
            45
            44
            20
            53
            54
            52
```

```
          49
          4E
          47
          20
70B7      0D       00490           FCB     $0D
70B8      57       00500           FCC     /WITH AN EMBEDDED CR/
          49
          54
          48
          20
          41
          4E
          20
          45
          4D
          42
          45
          44
          44
          45
          44
          20
          43
          52
70CB      00       00510 NUL3      FCB     $00      NUL
70CC      52       00520 STR004    FCC     /ROMANS 3:23 SAYS, "FOR ALL
HAVE/
          4F
          4D
          41
          4E
          53
          20
          33
          3A
          32
          33
          20
          53
          41
          59
          53
          2C
          20
          22
          46
          4F
```

```
            52
            20
            41
            4C
            4C
            20
            48
            41
            56
            45
70EB        00          00530 NUL4    FCB     $00     NUL
70EC        53          00540 STR005  FCC     /SINNED, AND COME SHORT OF
THE/
            49
            4E
            4E
            45
            44
            2C
            20
            41
            4E
            44
            20
            43
            4F
            4D
            45
            20
            53
            48
            4F
            52
            54
            20
            4F
            46
            20
            54
            48
            45
7109        00          00550 NUL5    FCB     $00     NUL
710A        47          00560 STR006  FCC     /GLORY OF GOD." (KJV)./
            4C
            4F
            52
            59
```

```
             20
             4F
             46
             20
             47
             4F
             44
             2E
             22
             20
             28
             4B
             4A
             56
             29
             2E
711F         00        00570 NUL6    FCB     $00     NUL
                       00580
7120 BD      400E      00590 LBL001  JSR     VIDCLS
7123 8E      0400      00600         LDX     #VIDRAM TOP OF SCREEN
7126 9F      88        00610         STX     CURPOS  CURSOR
7128 8E      7005      00620         LDX     #STR000 ADDRESS OF STRING 0
712B BD      420A      00630         JSR     PRTS00  GO PRINT STRING
712E BD      40D7      00640         JSR     CRLF
7131 8E      7025      00650         LDX     #STR001 ADDRESS OF STRING 1
7134 BD      420A      00660         JSR     PRTS00  GO PRINT STRING
7137 BD      40D7      00670         JSR     CRLF
713A 8E      705B      00680         LDX     #STR002 ADDRESS OF STRING 2
713D BD      420A      00690         JSR     PRTS00  GO PRINT STRING
7140 BD      40D7      00700         JSR     CRLF
7143 8E      7097      00710         LDX     #STR003 ADDRESS OF STRING 3
7146 BD      420A      00720         JSR     PRTS00  GO PRINT STRING
7149 BD      40D7      00730         JSR     CRLF
714C BD      40D7      00740         JSR     CRLF
714F 8E      70CC      00750         LDX     #STR004 ADDRESS OF STRING 3
7152 BD      420A      00760         JSR     PRTS00  GO PRINT STRING
7155 BD      40D7      00770         JSR     CRLF
7158 8E      70EC      00780         LDX     #STR005 ADDRESS OF STRING 3
715B BD      420A      00790         JSR     PRTS00  GO PRINT STRING
715E BD      40D7      00800         JSR     CRLF
7161 8E      710A      00810         LDX     #STR006 ADDRESS OF STRING 3
7164 BD      420A      00820         JSR     PRTS00  GO PRINT STRING
7167 BD      40D7      00830         JSR     CRLF
                       00840
                       00850  * EXIT
716A 35      10        00860         PULS    X
716C 39                00870         RTS
```

```
            00880
0000        00890           END
```

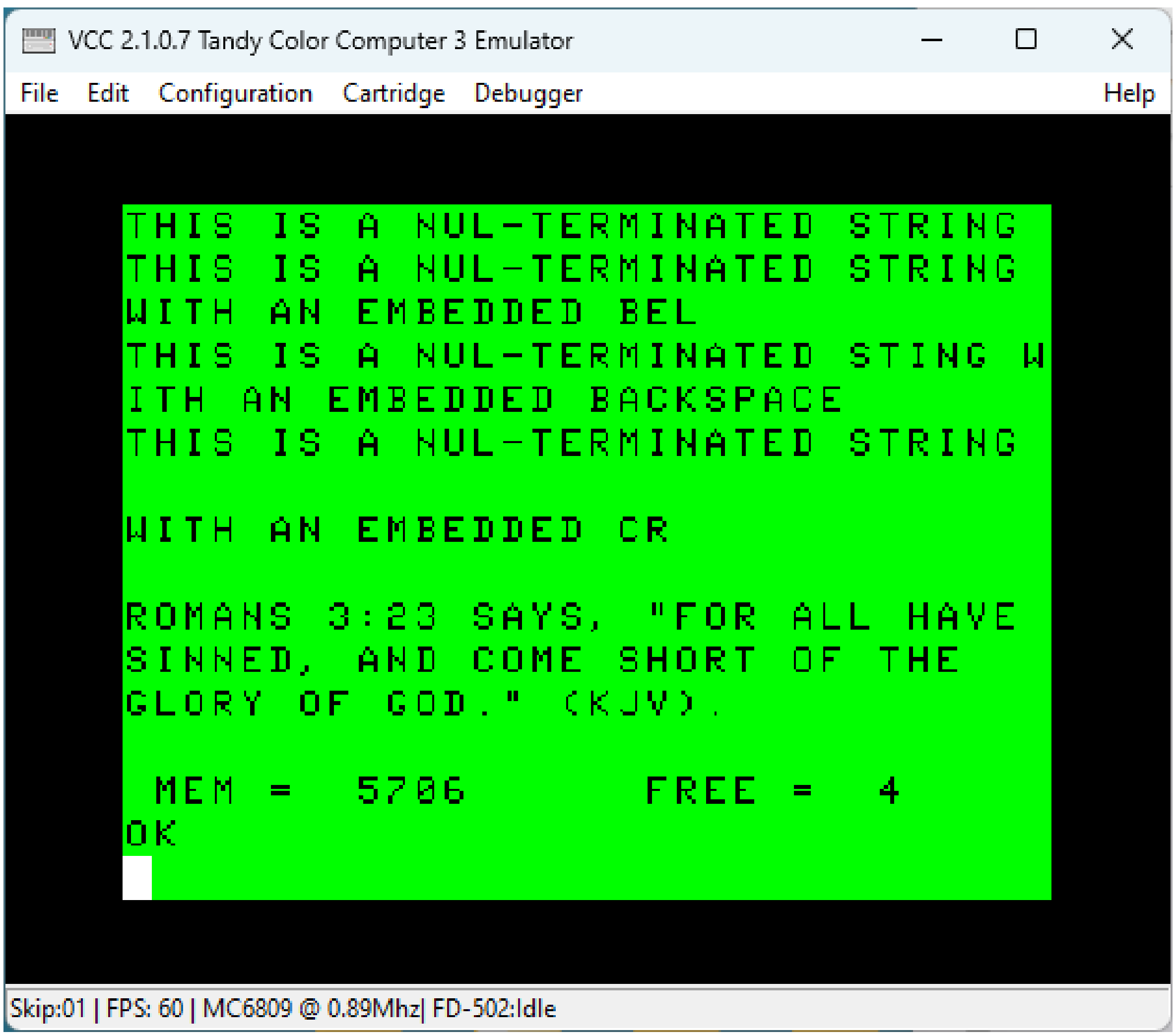
The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0021.BAS
1030 '* MDJ 2023/02/15
1040 '*
1050 '* PRTS00 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0021.BIN"
1330 '

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.

Bible Note: But there's Good News for us, because Romans 3:23 is immediately followed by Romans 3:24, "Being justified freely by his grace through the redemption that is in Christ Jesus". (KJV).

Yes, we are all flawed and have done bad things. (God calls such bad things sin. Even bad thoughts are sin.) And, because God is perfectly Holy and perfectly Just, He MUST punish every single bad thing, no matter how small. And He has decreed that the one and only punishment is eternal separation from Him in Hell and the Lake of Fire.

This causes a problem though. If there's no escape from this rule, then Heaven will be empty! God's plan for our Love and Enjoyment of Him for all eternity will never come to pass!

Praise God! Jesus is His answer! Jesus took our place! Jesus died and suffered eternal death and separation from God so that whoever believes and trusts in Him will enjoy life in His presence forever!

=====

PRTS0D: Prints a String which is Terminated By a Carriage Return To the VIDRAM Screen at the Current Cursor Position

The Carriage return is “printed”. ETX forces a Cold Start. BEL and BS are “printed”. All other control codes are ignored.

```
00100 *****
00110 *
00120 * PRTS0D.ASM
00130 * MDJ 2023/01/27
00140 *
00150 * PRINTS A CR-TERMINATED
00160 * STRING ($0D) BEGINNING
00170 * AT THE CURRENT CURSOR
00180 * LOCATION
00190 *
00200 * THE CR IS COUNTED
00210 * AS PART OF THE STRING
00220 * AND IS "PRINTED"
00230 *
00240 * ANY INTERNAL
00250 *   ETX ($03) ESC/BREAK KEY
00260 * FORCES A COLD START
00270 *
00280 * ANY INTERNAL
00290 *   BEL ($07)
00300 *   BS  ($08) BACKSPACE KEY
00310 * CHARACTERS ARE "PRINTED"
00320 *
00330 * ALL OTHER CONTROL CODES
00340 * ($01-$31) ARE IGNORED
00350 *
00360 * ENTRY CONDITIONS:
00370 * X = START ADDRESS
00380 *     OF THE STRING
00390 *
```

```
                  00400 * EXIT CONDITIONS:
                  00410 * NONE
                  00420 *
                  00430 *****
                  00440
                  00450 * EXTERNAL ROUTINE
                  00460 * ADDRESSES
         40D7     00470 CRLF    EQU     $40D7
         41DB     00480 PRTCHA  EQU     $41DB
                  00490
4218              00500         ORG     $4218
                  00510
4218 34  02       00520 PRTS0D  PSHS    A
421A A6  80       00530 LBL001  LDA     ,X+     GET A CHARACTER
421C 81  0D       00540         CMPA    #$0D    IS IT A CR ($0D)
421E 27  05       00550         BEQ     LBL002  GO IF YES
4220 BD  41DB     00560         JSR     PRTCHA  GO PRINT CHARACTER
4223 20  F5       00570         BRA     LBL001  RETURN FOR NEXT
CHARACTER
4225 BD  40D7     00580 LBL002  JSR     CRLF    PRINT THE CR
4228 35  02       00590         PULS    A
                  00600
                  00610 *EXIT
422A 39           00620         RTS
                  00630
         0000     00640         END
```

The Assembly Language Test Routine:

```
                  00100 *****
                  00110 *
                  00120 * TEST0022.ASM
                  00130 * MDJ 2023/02/15
                  00140 *
                  00150 * PRTS0D 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 * ML FOUNDATION
                    00270 * CORE ADDRESSES
          400E      00280 VIDCLS  EQU     $400E
          4218      00290 PRTS0D  EQU     $4218
                    00300
7000                00310         ORG     $7000
                    00320
                    00330 * PRTS0D TEST
                    00340
7000 34   10        00350         PSHS    X
7002 7E   7094      00360         JMP      LBL001
7005      54        00370 STR000  FCC     "THIS IS A CR-TERMINATED
STRING"
          48
          49
          53
          20
          49
          53
          20
          41
          20
          43
          52
          2D
          54
          45
          52
          4D
          49
          4E
          41
          54
          45
          44
          20
          53
          54
          52
          49
          4E
          47
7023      0D        00380 CR0     FCB     $0D     CR
7024      54        00390 STR001  FCC     "THIS IS A CR-TERMINATED
STRING WITH AN
 EMBED"
          48
```

```
          49
          53
          20
          49
          53
          20
          41
          20
          43
          52
          2D
          54
          45
          52
          4D
          49
          4E
          41
          54
          45
          44
          20
          53
          54
          52
          49
          4E
          47
          20
          57
          49
          54
          48
          20
          41
          4E
          20
          45
          4D
          42
          45
          44
7050      07        00400          FCB      $07
7051      44        00410          FCC      "DED BEL"
          45
          44
          20
```

```
            42
            45
            4C
7058        0D          00420 CR1     FCB     $0D     CR
7059        54          00430 STR002  FCC     "THIS IS A CR-TERMINATED
STR"
            48
            49
            53
            20
            49
            53
            20
            41
            20
            43
            52
            2D
            54
            45
            52
            4D
            49
            4E
            41
            54
            45
            44
            20
            53
            54
            52
7074        08          00440         FCB     $08
7075        49          00450         FCC     "ING WITH AN EMBEDDED
BACKSPACE"
            4E
            47
            20
            57
            49
            54
            48
            20
            41
            4E
            20
            45
```

```
          4D
          42
          45
          44
          44
          45
          44
          20
          42
          41
          43
          4B
          53
          50
          41
          43
          45
7093      0D        00460 CR2     FCB     $0D     CR
                    00470
7094 BD   400E      00480 LBL001  JSR     VIDCLS
7097 8E   0400      00490         LDX     #VIDRAM TOP OF SCREEN
709A 9F   88        00500         STX     CURPOS  CURSOR
709C 8E   7005      00510         LDX     #STR000 ADDRESS OF STRING 0
709F BD   4218      00520         JSR     PRTS0D  GO PRINT STRING
70A2 8E   7024      00530         LDX     #STR001 ADDRESS OF STRING 1
70A5 BD   4218      00540         JSR     PRTS0D  GO PRINT STRING
70A8 8E   7059      00550         LDX     #STR002 ADDRESS OF STRING 2
70AB BD   4218      00560         JSR     PRTS0D  GO PRINT STRING
                    00570
                    00580 * EXIT
70AE 35   10        00590         PULS    X
70B0 39             00600         RTS
                    00610
          0000      00620         END
```

———

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0022.BAS
1030 '* MDJ 2023/02/15
1040 '*
1050 '* PRTS0D 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0022.BIN"
1330 '

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.

Note the ending report, “FREE = 1”. We’ve been working with a disk which I temporarily labeled “MLFSYS15.DSK”. The “FREE = 1” report means it’s time to switch to a new disk which I’m temporarily labeling “MLFSYS16.DSK” and which currently contains only the file “MLCORE.BIN” so that “FREE = 67”. cf. *Color Computer Disk System*, pp. 57-62.

=====

PRTSLS: Prints a Length-Specified String To the VIDRAM Screen at the Current Cursor Position

ETX forces a Cold Start. BEL, BS, and CR are “printed”. All other control codes are ignored. Note that the maximum length of $FFFF is really only theoretical because such a string would fill all 64K of memory and leave no room for the code intended to be used to print the string.

The practical maximum length depends strictly on the amount of code required by a given program (and thus the amount of memory available for string storage). Experimentation will be required any time you try to push these two interdependent limits.

```
*****
*
* PRTSLS.ASM
* MDJ 2023/01/27
*
* PRINTS A LENGTH-SPECIFIED
* STRING ($0001-$FFFF)
* BEGINNING AT THE
* CURRENT CURSOR LOCATION
*
* ANY INTERNAL
*   ETX ($03) ESC/BREAK KEY
* FORCES A COLD START
*
* ANY INTERNAL
*   BEL ($07)
*   BS  ($08) BACKSPACE KEY
*   CR  ($0D)
* CHARACTERS ARE "PRINTED"
*
* ALL OTHER CONTROL CODES
* ($01-$31) ARE IGNORED,
* EXCEPT THAT THEY ARE
* INCLUDED IN THE "LENGTH"
*
* ENTRY CONDITIONS:
* X = START ADDRESS
```

```
                    00370 *     OF THE STRING
                    00380 * Y = STRING LENGTH
                    00390 *     IN CHARACTERS
                    00400 *     ($0001-$FFFF)
                    00410 *     (    1-65535)
                    00420 *
                    00430 * EXIT CONDITIONS:
                    00440 * NONE
                    00450 *
                    00460 *****
                    00470
                    00480 * EXTERNAL ROUTINE
                    00490 * ADDRESS
          41DB      00500 PRTCHA  EQU     $41DB
                    00510
422B                00520         ORG     $422B
                    00530
422B 34   02        00540 PRTSLS  PSHS    A
422D A6   80        00550 LBL001  LDA     ,X+     GET A CHARACTER
422F BD   41DB      00560         JSR     PRTCHA  GO PRINT CHARACTER
4232 31   3F        00570         LEAY    -1,Y    DECREMENT COUNT
4234 27   02        00580         BEQ     LBL002  GO IF END OF STRING
4236 20   F5        00590         BRA     LBL001  RETURN FOR NEXT
CHARACTER
4238 35   02        00600 LBL002  PULS    A
                    00610
                    00620 *EXIT
423A 39             00630         RTS
                    00640
          0000      00650         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0023.ASM
                    00130 * MDJ 2023/02/15
                    00140 *
                    00150 * PRTSLS 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 * ML FOUNDATION
                00270 * CORE ADDRESSES
     400E       00280 VIDCLS  EQU     $400E
     40D7       00290 CRLF    EQU     $40D7
     422B       00300 PRTSLS  EQU     $422B
                00310
7000            00320         ORG     $7000
                00330
                00340 * PRTSLS TEST
                00350
7000 34   30    00360         PSHS    X,Y
7002 7E   7101  00370         JMP     LBL001
7005      0049  00380 LSPEC0  FDB     $0049
7007      54    00390 STR000  FCC     /THIS IS A LENGTH SPECIFIED
STRING /
          48
          49
          53
          20
          49
          53
          20
          41
          20
          4C
          45
          4E
          47
          54
          48
          20
          53
          50
          45
          43
          49
          46
          49
          45
          44
          20
          53
          54
```

```
          52
          49
          4E
          47
          20
7029      57        00400          FCC     /WITH A LENGTH OF 73 = $0049
/
          49
          54
          48
          20
          41
          20
          4C
          45
          4E
          47
          54
          48
          20
          4F
          46
          20
          37
          33
          20
          3D
          20
          24
          30
          30
          34
          39
          20
7045      43        00410          FCC     /CHARACTERS./
          48
          41
          52
          41
          43
          54
          45
          52
          53
          2E
7050      0047      00420 LSPEC1   FDB     $0047
```

```
7052      54      00430 STR001  FCC     /THE FIRST PHRASE OF ROMANS
6:23 /
          48
          45
          20
          46
          49
          52
          53
          54
          20
          50
          48
          52
          41
          53
          45
          20
          4F
          46
          20
          52
          4F
          4D
          41
          4E
          53
          20
          36
          3A
          32
          33
          20
7072      53      00440         FCC     /SAYS, "THE WAGES OF SIN IS
/
          41
          59
          53
          2C
          20
          22
          54
          48
          45
          20
          57
          41
```

```
          47
          45
          53
          20
          4F
          46
          20
          53
          49
          4E
          20
          49
          53
          20
          20
          20
          20
          20
          20
7092      44        00450          FCC     /DEATH"./
          45
          41
          54
          48
          22
          2E
7099      0066      00460 LSPEC2   FDB     $0066
709B      42        00470 STR002   FCC     /BUT ROMANS 6:23 CONTINUES
WITH  /
          55
          54
          20
          52
          4F
          4D
          41
          4E
          53
          20
          36
          3A
          32
          33
          20
          43
          4F
          4E
```

```
           54
           49
           4E
           55
           45
           53
           20
           57
           49
           54
           48
           20
           20
70BB       22        00480          FCC      /"BUT THE GIFT OF GOD IS
ETERNAL /
           42
           55
           54
           20
           54
           48
           45
           20
           47
           49
           46
           54
           20
           4F
           46
           20
           47
           4F
           44
           20
           49
           53
           20
           45
           54
           45
           52
           4E
           41
           4C
           20
```

```
70DB       4C       00490           FCC     /LIFE THROUGH JESUS CHRIST
OUR   /
           49
           46
           45
           20
           54
           48
           52
           4F
           55
           47
           48
           20
           4A
           45
           53
           55
           53
           20
           43
           48
           52
           49
           53
           54
           20
           4F
           55
           52
           20
           20
           20
70FB       4C       00500           FCC     /LORD."/
           4F
           52
           44
           2E
           22
                    00510
7101 BD    400E     00520 LBL001    JSR     VIDCLS
7104 8E    0400     00530           LDX     #VIDRAM TOP OF SCREEN
7107 9F    88       00540           STX     CURPOS  CURSOR
7109 8E    7007     00550           LDX     #STR000 ADDRESS OF STRING 0
710C 10BE 7005      00560           LDY     LSPEC0  GET SPECIFIED LENGTH
7110 BD    422B     00570           JSR     PRTSLS  GO PRINT STRING
7113 BD    40D7     00580           JSR     CRLF
```

```
7116 BD   40D7     00590          JSR     CRLF
7119 8E   7052     00600          LDX     #STR001 ADDRESS OF STRING 1
711C 10BE 7050     00610          LDY     LSPEC1  GET SPECIFIED LENGTH
7120 BD   422B     00620          JSR     PRTSLS  GO PRINT STRING
7123 BD   40D7     00630          JSR     CRLF
7126 BD   40D7     00640          JSR     CRLF
7129 8E   709B     00650          LDX     #STR002 ADDRESS OF STRING 2
712C 10BE 7099     00660          LDY     LSPEC2  GET SPECIFIED LENGTH
7130 BD   422B     00670          JSR     PRTSLS  GO PRINT STRING
7133 BD   40D7     00680          JSR     CRLF
7136 BD   40D7     00690          JSR     CRLF
                   00700
                   00710 * EXIT
7139 35   30       00720          PULS    X,Y
713B 39            00730          RTS
                   00740
          0000     00750          END
```

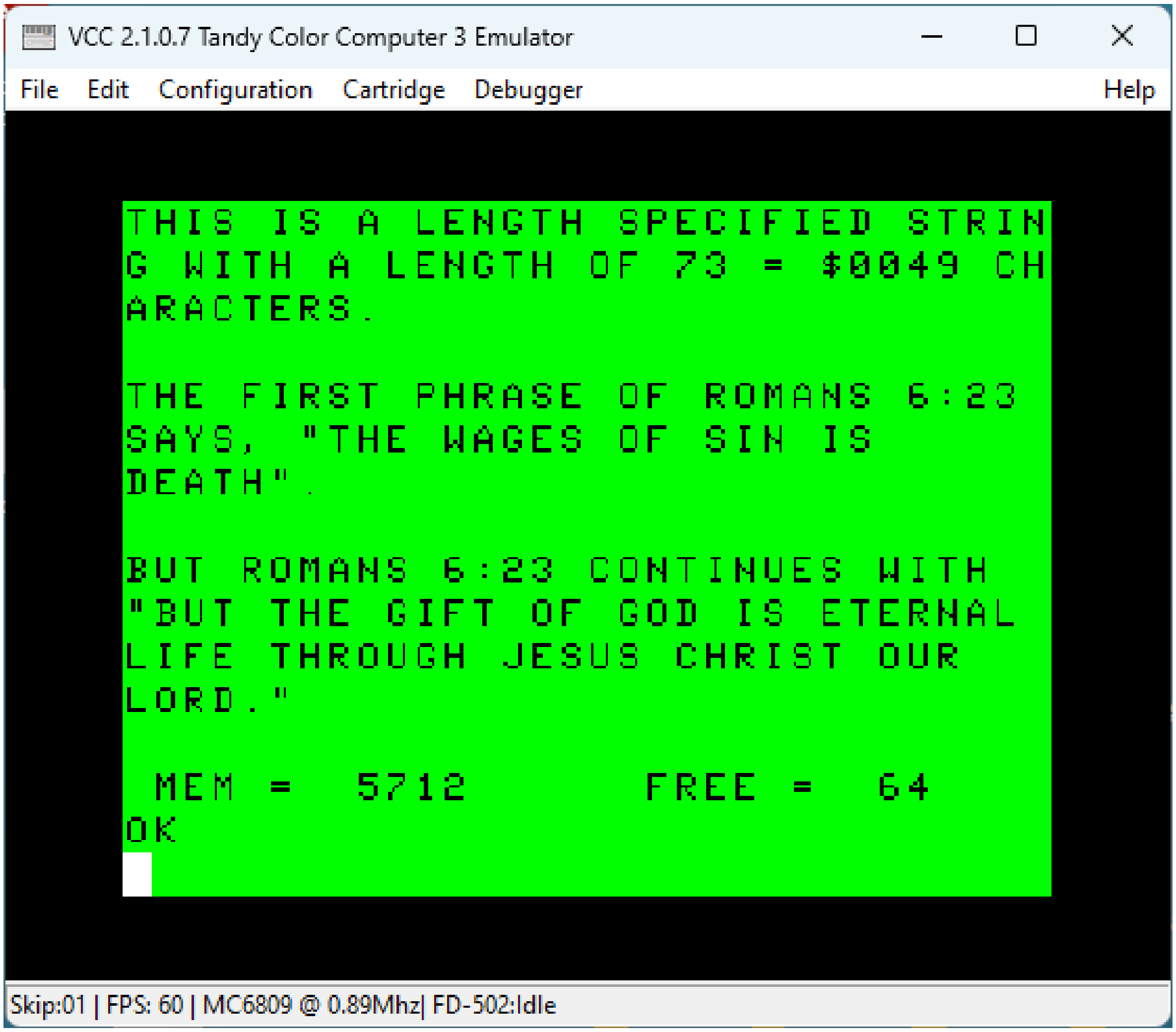
———

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0023.BAS
1030 '* MDJ 2023/02/15
1040 '*
1050 '* PRTSLS 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0023.BIN"
1330 '
```

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

32767 END
```

Result:



As expected.

=====

PRTSCL: Prints a Counted Long String To the VIDRAM Screen at the Current Cursor Position

This is similar to PRTSLS (Length-Specified String) except that the Length is specified in the first two bytes of the Counted Long String. Again, the maximum length is only theoretical.

ETX forces a Cold Start. BEL, BS, and CR are “printed”. All other control codes are ignored

```
00100 *****
00110 *
00120 * PRTSCL.ASM
00130 * MDJ 2023/01/27
00140 *
00150 * PRINTS A COUNTED LONG
00160 * STRING ($0001-$FFFF)
00170 * BEGINNING AT THE
00180 * CURRENT CURSOR LOCATION
00190 *
00200 * THE FIRST TWO BYTES AT
00210 * THE STRING ADDRESS ARE
00220 * THE CHARACTER COUNT
00230 * AKA LENGTH
00240 *
00250 * ANY INTERNAL
00260 *   ETX ($03) ESC/BREAK KEY
00270 * FORCES A COLD START
00280 *
00290 * ANY INTERNAL
00300 *   BEL ($07)
00310 *   BS  ($08) BACKSPACE KEY
00320 *   CR  ($0D)
00330 * CHARACTERS ARE "PRINTED"
00340 *
00350 * ALL OTHER CONTROL CODES
00360 * ($01-$31) ARE IGNORED,
00370 * EXCEPT THAT THEY ARE
00380 * INCLUDED IN THE "LENGTH"
00390 *
```

```
                    00400 * ENTRY CONDITIONS:
                    00410 * X = START ADDRESS
                    00420 *     OF THE COUNTED
                    00430 *     LONG STRING
                    00440 *
                    00450 * EXIT CONDITIONS:
                    00460 * NONE
                    00470 *
                    00480 *****
                    00490
                    00500 * EXTERNAL ROUTINE
                    00510 * ADDRESS
          422B      00520 PRTSLS  EQU     $422B
                    00530
423B                00540         ORG     $423B
                    00550
423B 34   20        00560 PRTSCL  PSHS    Y
423D 10AE 81        00570         LDY     ,X++    GET THE CHARACTER
COUNT
4240 BD   422B      00580         JSR     PRTSLS  GO PRINT THE STRING
4243 35   20        00590         PULS    Y
                    00600
                    00610 *EXIT
4245 39             00620         RTS
                    00630
          0000      00640         END
```

——-

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0024.ASM
                    00130 * MDJ 2023/02/15
                    00140 *
                    00150 * PRTSCL 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 * ML FOUNDATION
                   00270 * CORE ADDRESSES
          400E     00280 VIDCLS  EQU     $400E
          40D7     00290 CRLF    EQU     $40D7
          423B     00300 PRTSCL  EQU     $423B
                   00310
7000               00320         ORG     $7000
                   00330
                   00340 * PRTSCL TEST
                   00350
7000 34   10       00360         PSHS    X
7002 7E   711F     00370         JMP     LBL001
7005      0118     00380 COUNT0  FDB     $0118
7007      54       00390 STR000  FCC     "THIS IS A LONG COUNTED
STRING "
          48
          49
          53
          20
          49
          53
          20
          41
          20
          4C
          4F
          4E
          47
          20
          43
          4F
          55
          4E
          54
          45
          44
          20
          53
          54
          52
          49
          4E
          47
          20
7025      57       00400         FCC     "WITH A LENGTH OF 280 =
$0118 "
          49
```

```
           54
           48
           20
           41
           20
           4C
           45
           4E
           47
           54
           48
           20
           4F
           46
           20
           32
           38
           30
           20
           3D
           20
           24
           30
           31
           31
           38
           20
7042       43        00410          FCC      "CHARACTERS.
QWERTYUIOPASDFGHJKLZ"
           48
           41
           52
           41
           43
           54
           45
           52
           53
           2E
           20
           51
           57
           45
           52
           54
           59
           55
```

```
            49
            4F
            50
            41
            53
            44
            46
            47
            48
            4A
            4B
            4C
            5A
7062        58        00420          FCC
"XCVBNMQWERTYUIOPASDFGHJKLZXCVBNM"
            43
            56
            42
            4E
            4D
            51
            57
            45
            52
            54
            59
            55
            49
            4F
            50
            41
            53
            44
            46
            47
            48
            4A
            4B
            4C
            5A
            58
            43
            56
            42
            4E
            4D
7082        51        00430          FCC       "QWERTYUIOPASDFGHJKLZXCVBNM"
```

```
               57
               45
               52
               54
               59
               55
               49
               4F
               50
               41
               53
               44
               46
               47
               48
               4A
               4B
               4C
               5A
               58
               43
               56
               42
               4E
               4D
     709C      51         00440         FCC      "QWERTYUIOPASDFGHJKLZXCVBNM"
               57
               45
               52
               54
               59
               55
               49
               4F
               50
               41
               53
               44
               46
               47
               48
               4A
               4B
               4C
               5A
               58
               43
```

```
        56
        42
        4E
        4D
70B6    51      00450       FCC     "QWERTYUIOPASDFGHJKLZXCVBNM"
        57
        45
        52
        54
        59
        55
        49
        4F
        50
        41
        53
        44
        46
        47
        48
        4A
        4B
        4C
        5A
        58
        43
        56
        42
        4E
        4D
70D0    51      00460       FCC     "QWERTYUIOPASDFGHJKLZXCVBNM"
        57
        45
        52
        54
        59
        55
        49
        4F
        50
        41
        53
        44
        46
        47
        48
        4A
```

```
         4B
         4C
         5A
         58
         43
         56
         42
         4E
         4D
70EA     51        00470          FCC       "QWERTYUIOPASDFGHJKLZXCVBNM"
         57
         45
         52
         54
         59
         55
         49
         4F
         50
         41
         53
         44
         46
         47
         48
         4A
         4B
         4C
         5A
         58
         43
         56
         42
         4E
         4D
7104     51        00480          FCC
"QWERTYUIOPASDFGHJKLZXCVBNM."
         57
         45
         52
         54
         59
         55
         49
         4F
         50
         41
```

```
        53
        44
        46
        47
        48
        4A
        4B
        4C
        5A
        58
        43
        56
        42
        4E
        4D
        2E
                  00490
711F BD 400E      00500 LBL001  JSR     VIDCLS
7122 8E 0400      00510         LDX     #VIDRAM TOP OF SCREEN
7125 9F 88        00520         STX     CURPOS  CURSOR
7127 8E 7005      00530         LDX     #COUNT0 ADDRESS OF COUNTED
STRING
712A BD 423B      00540         JSR     PRTSCL  GO PRINT STRING
712D BD 40D7      00550         JSR     CRLF
                  00560
                  00570 * EXIT
7130 35 10        00580         PULS    X
7132 39           00590         RTS
                  00600
        0000      00610         END
```

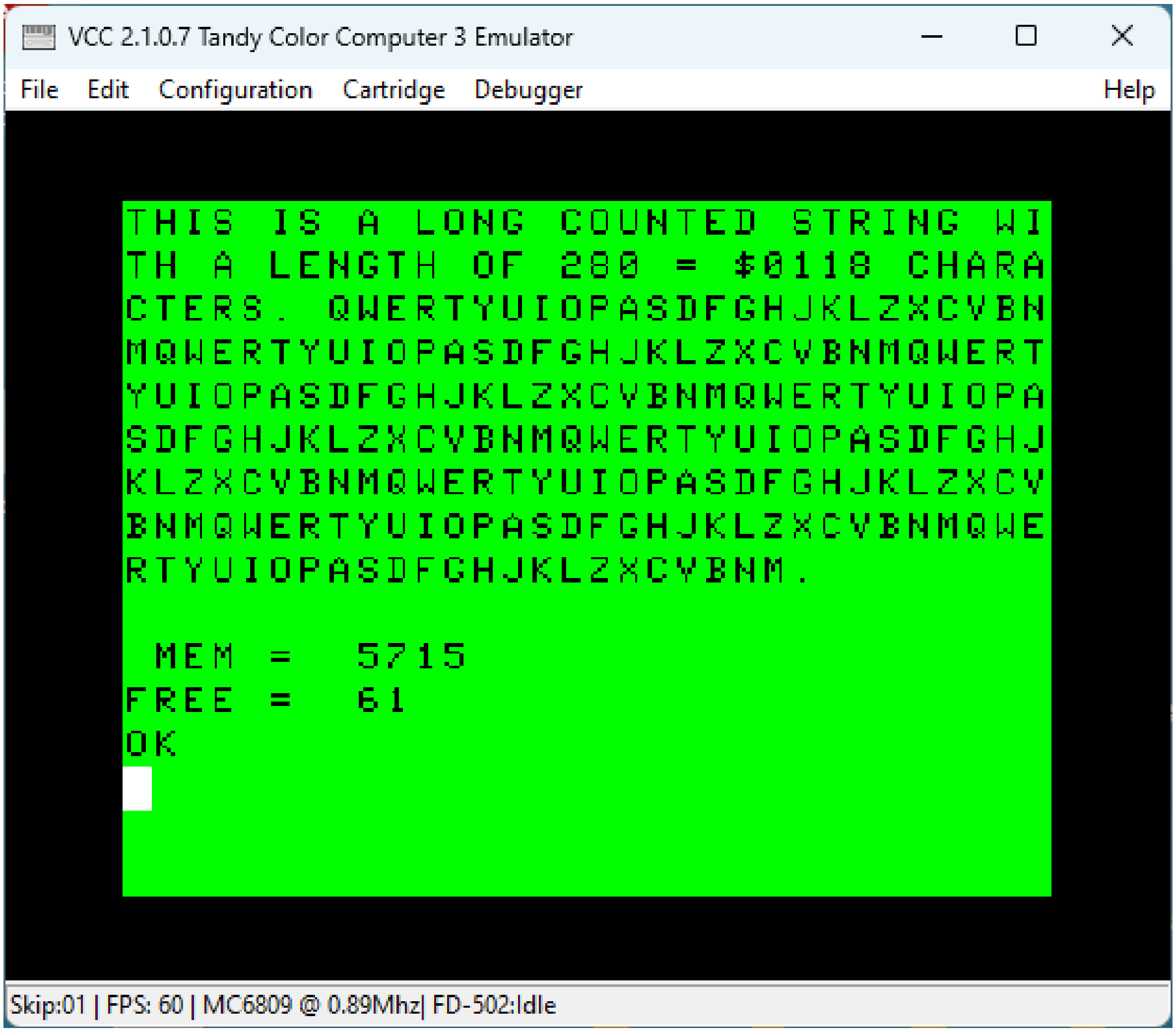
____

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0024.BAS
1030 '* MDJ 2023/02/15
1040 '*
1050 '* PRTSCL 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0024.BIN"
1330 '

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.

=====

PRTSCS: Prints a Counted Short String To the VIDRAM Screen at the Current Cursor Position

This is similar to PRTSCL (Counted Long String) except that the Length is specified in just the first byte of the Counted Short String, whose length is thus limited to 255 = $FF Characters.

Like the BASIC String, the Counted Short String cannot contain a full 256-byte Disk Sector (256 = $0100). However, a full 256-byte Disk Sector can be easily contained in either a Counted Long String or a Length-Specified String.

ETX forces a Cold Start. BEL, BS, and CR are “printed”. All other control codes are ignored

```
00100 *****
00110 *
00120 * PRTSCS.ASM
00130 * MDJ 2023/01/27
00140 *
00150 * PRINTS A COUNTED SHORT
00160 * STRING ($01-$FF)
00170 * BEGINNING AT THE
00180 * CURRENT CURSOR LOCATION
00190 *
00200 * THE FIRST BYTE AT
00210 * THE STRING ADDRESS IS
00220 * THE CHARACTER COUNT
00230 * AKA LENGTH
00240 *
00250 * ANY INTERNAL
00260 *   ETX ($03) ESC/BREAK KEY
00270 * FORCES A COLD START
00280 *
00290 * ANY INTERNAL
00300 *   BEL ($07)
00310 *   BS  ($08) BACKSPACE KEY
00320 *   CR  ($0D)
00330 * CHARACTERS ARE "PRINTED"
00340 *
00350 * ALL OTHER CONTROL CODES
```

```
                   00360 * ($01-$31) ARE IGNORED,
                   00370 * EXCEPT THAT THEY ARE
                   00380 * INCLUDED IN THE "LENGTH"
                   00390 *
                   00400 * ENTRY CONDITIONS:
                   00410 * X = START ADDRESS
                   00420 *     OF THE COUNTED
                   00430 *     SHORT STRING
                   00440 *
                   00450 * EXIT CONDITIONS:
                   00460 * NONE
                   00470 *
                   00480 *****
                   00490
                   00500 * EXTERNAL ROUTINE
                   00510 * ADDRESS
          422B     00520 PRTSLS  EQU     $422B
                   00530
4246               00540         ORG     $4246
                   00550
4246 34   26       00560 PRTSCS  PSHS    A,B,Y
4248 4F            00570         CLRA            CLEAR REGISTER D
HIGH BYTE
4249 E6   80       00580         LDB     ,X+     GET THE CHARACTER
COUNT
424B 1F   02       00590         TFR     D,Y     XFER THE COUNT TO Y
424D BD   422B     00600         JSR     PRTSLS  GO PRINT THE STRING
4250 35   26       00610         PULS    A,B,Y
                   00620
                   00630 *EXIT
4252 39            00640         RTS
                   00650
          0000     00660         END
```

——-

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0025.ASM
                   00130 * MDJ 2023/02/15
                   00140 *
                   00150 * PRTSCS 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 * ML FOUNDATION
                    00270 * CORE ADDRESSES
          400E      00280 VIDCLS  EQU     $400E
          40D7      00290 CRLF    EQU     $40D7
          4246      00300 PRTSCS  EQU     $4246
                    00310
7000                00320         ORG     $7000
                    00330
                    00340 * PRTSCS TEST
                    00350
7000 34   10        00360         PSHS    X
7002 7E   7119      00370         JMP     LBL001
7005      44        00380 COUNT0  FCB     $44
7006      54        00390 STR000  FCC     /THIS IS A SHORT COUNTED
STRING /
          48
          49
          53
          20
          49
          53
          20
          41
          20
          53
          48
          4F
          52
          54
          20
          43
          4F
          55
          4E
          54
          45
          44
          20
          53
          54
```

```
            52
            49
            4E
            47
            20
7025        57          00400          FCC      /WITH A LENGTH OF 68 = $44 /
            49
            54
            48
            20
            41
            20
            4C
            45
            4E
            47
            54
            48
            20
            4F
            46
            20
            36
            38
            20
            3D
            20
            24
            34
            34
            20
703F        43          00410          FCC      /CHARACTERS./
            48
            41
            52
            41
            43
            54
            45
            52
            53
            2E
704A        CE          00420 COUNT1   FCB      $CE
704B        41          00430 STR001   FCC      /ACTS 4:12 CONFIRMS THAT
JESUS IS/
            43
            54
```

```
            53
            20
            34
            3A
            31
            32
            20
            43
            4F
            4E
            46
            49
            52
            4D
            53
            20
            54
            48
            41
            54
            20
            4A
            45
            53
            55
            53
            20
            49
            53
706B        54          00440           FCC      /THE ONLY WAY TO GOD WHEN IT
SAYS/
            48
            45
            20
            4F
            4E
            4C
            59
            20
            57
            41
            59
            20
            54
            4F
            20
            47
```

```
          4F
          44
          20
          57
          48
          45
          4E
          20
          49
          54
          20
          53
          41
          59
          53
708B      54        00450          FCC     /THAT "NEITHER IS THERE
SALVATION/
          48
          41
          54
          20
          22
          4E
          45
          49
          54
          48
          45
          52
          20
          49
          53
          20
          54
          48
          45
          52
          45
          20
          53
          41
          4C
          56
          41
          54
          49
          4F
```

```
            4E
70AB        49          00460           FCC      /IN ANY OTHER: FOR THERE IS
NONE /
            4E
            20
            41
            4E
            59
            20
            4F
            54
            48
            45
            52
            3A
            20
            46
            4F
            52
            20
            54
            48
            45
            52
            45
            20
            49
            53
            20
            4E
            4F
            4E
            45
            20
70CB        4F          00470           FCC      /OTHER NAME UNDER HEAVEN
GIVEN   /
            54
            48
            45
            52
            20
            4E
            41
            4D
            45
            20
            55
```

```
          4E
          44
          45
          52
          20
          48
          45
          41
          56
          45
          4E
          20
          47
          49
          56
          45
          4E
          20
          20
          20
70EB      41        00480        FCC      /AMONG MEN, WHEREBY WE MUST
BE   /
          4D
          4F
          4E
          47
          20
          4D
          45
          4E
          2C
          20
          57
          48
          45
          52
          45
          42
          59
          20
          57
          45
          20
          4D
          55
          53
          54
```

```
         20
         42
         45
         20
         20
         20
710B     53      00490          FCC     /SAVED." (KJV)./
         41
         56
         45
         44
         2E
         22
         20
         28
         4B
         4A
         56
         29
         2E
                 00500
7119 BD  400E    00510 LBL001   JSR     VIDCLS
711C 8E  0400    00520          LDX     #VIDRAM TOP OF SCREEN
711F 9F  88      00530          STX     CURPOS  CURSOR
7121 8E  7005    00540          LDX     #COUNT0 ADDRESS OF COUNTED
STRING #0
7124 BD  4246    00550          JSR     PRTSCS  GO PRINT STRING
7127 BD  40D7    00560          JSR     CRLF
712A BD  40D7    00570          JSR     CRLF
712D 8E  704A    00580          LDX     #COUNT1 ADDRESS OF COUNTED
STRING #1
7130 BD  4246    00590          JSR     PRTSCS  GO PRINT STRING
7133 BD  40D7    00600          JSR     CRLF
                 00610
                 00620 * EXIT
7136 35  10      00630          PULS    X
7138 39          00640          RTS
                 00650
         0000    00660          END
```

———

The BASIC Language Control Program:

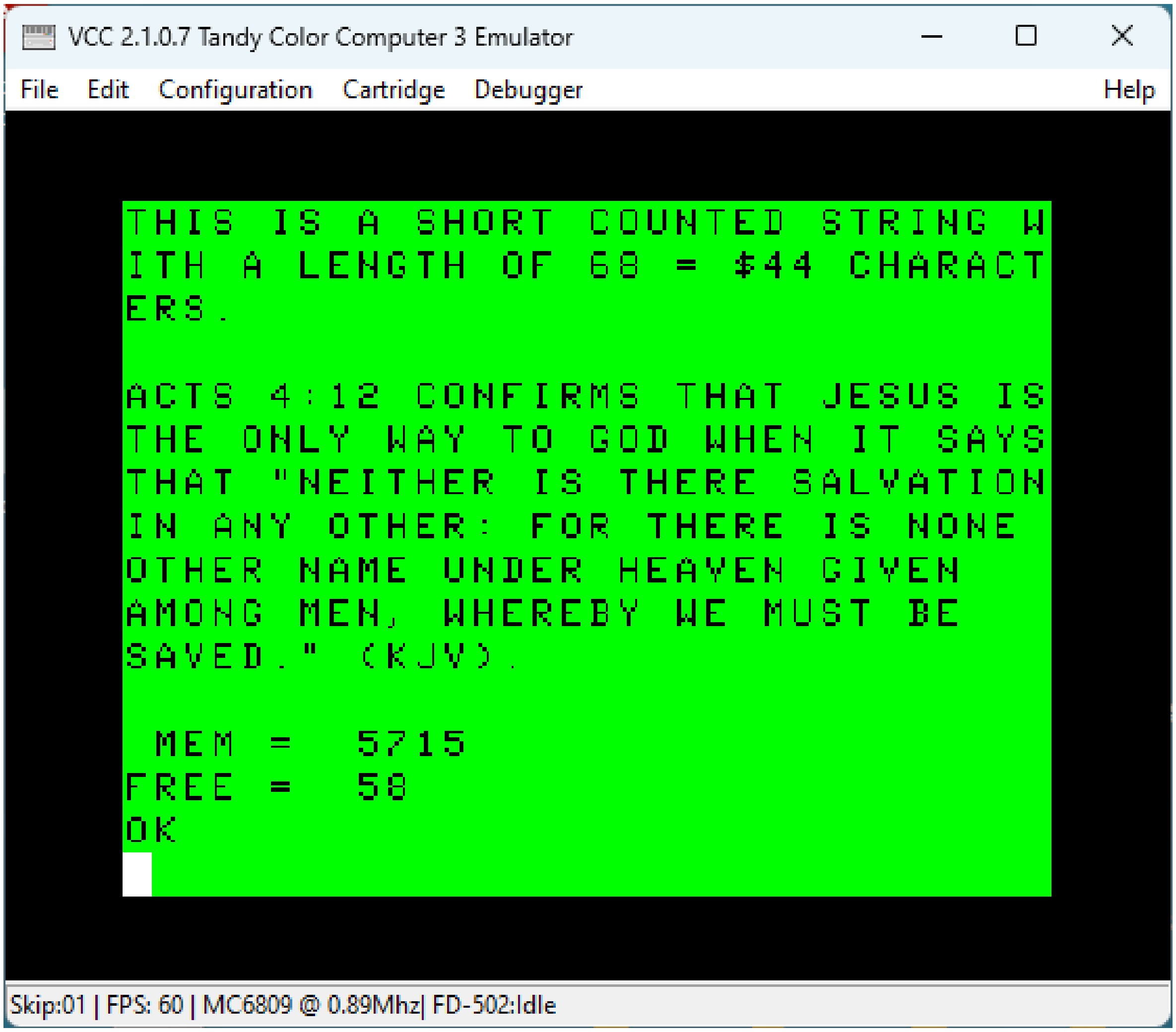
```
1000 '*****
1010 '*
1020 '* TEST0025.BAS
```

```
1030 '* MDJ 2023/02/15
1040 '*
1050 '* PRTSCS 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0025.BIN"
1330 '

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

```
32767 END
```

—

Result:



As expected.

Bible Note: Jesus testifies to the fact that He Himself is actually God; the Second Person of the Trinity, when He says in John 10:30, "I and my Father are one."

=====

DISKRW: Transfers Data Between a Specified Disk Sector And a Specified Buffer

DISKRW transfers data from a specified Disk Sector to a Specified 256-byte Buffer (READ Operation) or from a specified 256-byte Buffer to a specified Disk Sector (WRITE Operation).

In conformance with my CoCo Philosophy of trying to cram as much stuff as I can into the 96K of the 64K CoCo 2, DISKRW simply jumps into ROM and uses its DSKCON Routine. (cf. Appendix A below for the details of my CoCo Philosophy).

I developed many of the other Routines presented in this Paper entirely in Assembly Language in order to maximize speed while also minimizing memory space used.

But, with the Disk Drives, the limitations of the BASIC Language (Very Slow) are significantly overshadowed by the Drives' Mechanical Limitations (VERY VERY SLOW). There is thus no point in trying to write super-fast machine code for Disk Drive access. It is much more efficient to just use the ROM code: it immensely saves on RAM space, and won't materially affect overall system speed in normal use.

For details of DSKCON's internal operations, please refer to *Color Computer Disk System*, pp. 61-62.

```
00100 *****
00110 *
00120 * DISKRW.ASM
00130 * MDJ 2023/01/27
00140 *
00150 * READ OR WRITE
00160 * A DISK SECTOR
00170 *
00180 * WITHOUT ANY INTERNAL
00190 * ERROR TRAPPING
00200 *
00210 * ENTRY CONDITIONS:
00220 * $00EA = DCOPC
00230 *   2 = READ
00240 *   3 = WRITE
00250 * $00EB = DCDRV
00260 *   0 TO 3
00270 * $00EC = DCTRK
00280 *   0 TO 34
00290 * $00ED = DCSEC
```

```
                  00300 *   1 TO 18
                  00310 * $00EE = DCBPT
                  00320 *   DBUF0 = $0600
                  00330 *   DBUF1 = $O7OO
                  00340 *   OR ELSEWHERE
                  00350 *
                  00360 * EXIT CONDITIONS
                  00370 * $00F0 = DCSTA
                  00380 *   STATUS CODE
                  00390 *   0 = NO ERROR
                  00400 *
                  00410 *****
                  00420
                  00430 * RAMROM TRIGGER ADDRESS
         FFDE     00440 RAMROM  EQU     $FFDE
                  00450
                  00460 * ALLRAM TRIGGER ADDRESS
         FFDF     00470 ALLRAM  EQU     $FFDF
                  00480
                  00490 * ROM DSKCON JUMP ADDRESS
         D75F     00500 DSKCON  EQU     $D75F
                  00510
                  00520 * DSKCON MOTOR-OFF
                  00530 * TRIGGER ADDRESS
         FF40     00540 MOTOFF  EQU     $FF40
                  00550
4253              00560         ORG $4253
                  00570
4253 34  7F       00580 DISKRW  PSHS    A,B,X,Y,U,DP,CC
                  00590
                  00600 * SET RAMROM MODE
4255 B7  FFDE     00610         STA     RAMROM
                  00620
                  00630 * GO DO THE READ OR WRITE
4258 BD  D75F     00640         JSR     DSKCON
                  00650
                  00660 * SET ALLRAM MODE
425B B7  FFDF     00670         STA     ALLRAM
                  00680
                  00690 * TURN THE MOTOR OFF
425E 4F           00700         CLRA
425F B7  FF40     00710         STA     MOTOFF
                  00720
                  00730 * EXIT
4262 35  7F       00740         PULS    A,B,X,Y,U,DP,CC
4264 39           00750         RTS
                  00760
```

```
         0000     00770        END
```

The Assembly Language Test Routine. (Note that this Test Routine is significantly longer than previous ones):

```
                  00100 *****
                  00110 *
                  00120 * TEST0026.ASM
                  00130 * MDJ 2023/02/16
                  00140 *
                  00150 * DISKRW TEST
                  00160 *
                  00170 *****
                  00180
                  00190 * LOW RAM CURSOR ADDRESS
         0088     00200 CURPOS  EQU     $0088
                  00210
                  00220 * LOW RAM DSKCON ADDRESSES
         00EA     00230 DCOPC   EQU     $00EA
         00EB     00240 DCDRV   EQU     $00EB
         00EC     00250 DCTRK   EQU     $00EC
         00ED     00260 DCSEC   EQU     $00ED
         00EE     00270 DCBPT   EQU     $00EE
         00F0     00280 DCSTA   EQU     $00F0
         0600     00290 DBUF0   EQU     $0600
         0700     00300 DBUF1   EQU     $0700
                  00310
                  00320 * SCREEN ADDRESSES
         0400     00330 VIDRAM  EQU     $0400
         0600     00340 VIDEND  EQU     $0600
                  00350
                  00360 * ML FOUNDATION
                  00370 * CORE ADDRESSES
         400E     00380 VIDCLS  EQU     $400E
         40D7     00390 CRLF    EQU     $40D7
         4142     00400 POLCAT  EQU     $4142
         4218     00410 PRTS0D  EQU     $4218
         422B     00420 PRTSLS  EQU     $422B
         4253     00430 DISKRW  EQU     $4253
                  00440
7000              00450         ORG     $7000
                  00460
                  00470 * DISKRW TEST
                  00480
7000 34  73       00490         PSHS    A,X,Y,U,CC
```

```
7002 7E   72B6     00500          JMP     LBL001
7005      49       00510 BUFMSG   FCC     /IN JOHN 17:2-3, JESUS
PRAYED    /
          4E
          20
          4A
          4F
          48
          4E
          20
          31
          37
          3A
          32
          2D
          33
          2C
          20
          4A
          45
          53
          55
          53
          20
          50
          52
          41
          59
          45
          44
          20
          20
          20
          20
7025      46       00520          FCC     /FOR US WHEN HE SAID THAT
HIS    /
          4F
          52
          20
          55
          53
          20
          57
          48
          45
          4E
          20
```

48
45
20
53
41
49
44
20
54
48
41
54
20
48
49
53
20
20
20
20
7045 46 00530 FCC /FATHER HAD GIVEN "HIM
AUTHORITY /
41
54
48
45
52
20
48
41
44
20
47
49
56
45
4E
20
22
48
49
4D
20
41
55
54
48

```
        4F
        52
        49
        54
        59
        20
7065    4F      00540       FCC     /OVER ALL FLESH, TO GIVE
ETERNAL /
        56
        45
        52
        20
        41
        4C
        4C
        20
        46
        4C
        45
        53
        48
        2C
        20
        54
        4F
        20
        47
        49
        56
        45
        20
        45
        54
        45
        52
        4E
        41
        4C
        20
7085    4C      00550       FCC     /LIFE TO ALL WHOM YOU HAVE
GIVEN /
        49
        46
        45
        20
        54
        4F
```

```
            20
            41
            4C
            4C
            20
            57
            48
            4F
            4D
            20
            59
            4F
            55
            20
            48
            41
            56
            45
            20
            47
            49
            56
            45
            4E
            20
70A5        48        00560        FCC      /HIM... THAT THEY KNOW
YOU...      /
            49
            4D
            2E
            2E
            2E
            20
            54
            48
            41
            54
            20
            54
            48
            45
            59
            20
            4B
            4E
            4F
            57
```

```
          20
          59
          4F
          55
          2E
          2E
          2E
          20
          20
          20
          20
70C5      41        00570        FCC     /AND JESUS CHRIST WHOM YOU
HAVE  /
          4E
          44
          20
          4A
          45
          53
          55
          53
          20
          43
          48
          52
          49
          53
          54
          20
          57
          48
          4F
          4D
          20
          59
          4F
          55
          20
          48
          41
          56
          45
          20
          20
70E5      53        00580        FCC     /SENT." (ESV).
/
          45
```

```
            4E
            54
            2E
            22
            20
            28
            45
            53
            56
            29
            2E
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
            20
7105        50         00590 MSG00   FCC     /PUT SCRATCH DISK IN DRIVE
0/
            55
            54
            20
            53
            43
            52
            41
            54
            43
            48
            20
            44
            49
            53
            4B
```

```
          20
          49
          4E
          20
          44
          52
          49
          56
          45
          20
          30
7120      0D        00600           FCB     $0D     CR
7121      50        00610 MSG01     FCC     /PRESS ANY KEY WHEN READY/
          52
          45
          53
          53
          20
          41
          4E
          59
          20
          4B
          45
          59
          20
          57
          48
          45
          4E
          20
          52
          45
          41
          44
          59
7139      0D        00620           FCB     $0D     CR
713A      54        00630 MSG02     FCC     /THE BUFFER IS CURRENTLY
EMPTY./
          48
          45
          20
          42
          55
          46
          46
          45
```

```
          52
          20
          49
          53
          20
          43
          55
          52
          52
          45
          4E
          54
          4C
          59
          20
          45
          4D
          50
          54
          59
          2E
7158      0D        00640           FCB     $0D     CR
7159      50        00650 MSG03     FCC     /PRESS ANY KEY TO PREPARE
IT./
          52
          45
          53
          53
          20
          41
          4E
          59
          20
          4B
          45
          59
          20
          54
          4F
          20
          50
          52
          45
          50
          41
          52
          45
```

```
            20
            49
            54
            2E
7175        0D        00660          FCB      $0D      CR
7176        54        00670 MSG04    FCC      /THE BUFFER NOW CONTAINS:/
            48
            45
            20
            42
            55
            46
            46
            45
            52
            20
            4E
            4F
            57
            20
            43
            4F
            4E
            54
            41
            49
            4E
            53
            3A
718E        0D        00680          FCB      $0D      CR
718F        50        00690 MSG05    FCC      /PRESS KEY TO WRITE IT TO
DISK./
            52
            45
            53
            53
            20
            4B
            45
            59
            20
            54
            4F
            20
            57
            52
            49
```

```
          54
          45
          20
          49
          54
          20
          54
          4F
          20
          44
          49
          53
          4B
          2E
71AD      0D        00700          FCB      $0D      CR
71AE      50        00710 MSG06    FCC      /PRESS KEY TO READ IT FROM
DISK./
          52
          45
          53
          53
          20
          4B
          45
          59
          20
          54
          4F
          20
          52
          45
          41
          44
          20
          49
          54
          20
          46
          52
          4F
          4D
          20
          44
          49
          53
          4B
          2E
```

```
71CD      0D      00720          FCB     $0D     CR
71CE      52      00730 MSG07    FCC     /REPLACE SYSTEM DISK IN
DRIVE 0/
          45
          50
          4C
          41
          43
          45
          20
          53
          59
          53
          54
          45
          4D
          20
          44
          49
          53
          4B
          20
          49
          4E
          20
          44
          52
          49
          56
          45
          20
          30
71EC      0D      00740          FCB     $0D     CR
71ED      52      00750 RPTMSG   FCC     /REPORT:/
          45
          50
          4F
          52
          54
          3A
71F4      0D      00760          FCB     $0D     CR
71F5      44      00770 RPTMS1   FCC     /DISK WRITE WAS SUCCESSFUL./
          49
          53
          4B
          20
          57
```

```
            52
            49
            54
            45
            20
            57
            41
            53
            20
            53
            55
            43
            43
            45
            53
            53
            46
            55
            4C
            2E
720F        0D          00780          FCB     $0D     CR
7210        44          00790 RPTMS2   FCC     /DISK READ WAS SUCCESSFUL./
            49
            53
            4B
            20
            52
            45
            41
            44
            20
            57
            41
            53
            20
            53
            55
            43
            43
            45
            53
            53
            46
            55
            4C
            2E
7229        0D          00800          FCB     $0D     CR
```

```
722A      45      00810 RPTMS3  FCC     /ENTIRE TEST WAS
SUCCESSFUL./
          4E
          54
          49
          52
          45
          20
          54
          45
          53
          54
          20
          57
          41
          53
          20
          53
          55
          43
          43
          45
          53
          53
          46
          55
          4C
          2E
7245      0D      00820         FCB     $0D     CR
7246      2A      00830 RPTMS4  FCC     /*** TEST COMPLETED ***/
          2A
          2A
          20
          54
          45
          53
          54
          20
          43
          4F
          4D
          50
          4C
          45
          54
          45
          44
```

```
            20
            2A
            2A
            2A
725C        0D          00840          FCB      $0D      CR
725D        50          00850 RPTMS5   FCC      /PRESS ANY KEY TO EXIT/
            52
            45
            53
            53
            20
            41
            4E
            59
            20
            4B
            45
            59
            20
            54
            4F
            20
            45
            58
            49
            54
7272        0D          00860          FCB      $0D      CR
7273        45          00870 ERRMS1   FCC      /ERROR WRITING THE DISK/
            52
            52
            4F
            52
            20
            57
            52
            49
            54
            49
            4E
            47
            20
            54
            48
            45
            20
            44
            49
```

```
            53
            4B
7289        0D          00880           FCB     $0D     CR
728A        45          00890 ERRMS2    FCC     /ERROR READING THE DISK/
            52
            52
            4F
            52
            20
            52
            45
            41
            44
            49
            4E
            47
            20
            54
            48
            45
            20
            44
            49
            53
            4B
72A0        0D          00900           FCB     $0D     CR
72A1        2A          00910 ERRMS3    FCC     /*** TEST ABORTED ***/
            2A
            2A
            20
            54
            45
            53
            54
            20
            41
            42
            4F
            52
            54
            45
            44
            20
            2A
            2A
            2A
72B5        0D          00920           FCB     $0D     CR
```

```
                    00930
                    00940 *** FIRST SCREEN
72B6 BD   400E      00950 LBL001  JSR     VIDCLS
72B9 8E   0400      00960         LDX     #VIDRAM TOP OF SCREEN
72BC 9F   88        00970         STX     CURPOS  CURSOR
                    00980
                    00990 * DISPLAY MESSAGES
72BE 8E   7105      01000         LDX     #MSG00  SCRATCH DISK MESSAGE
72C1 BD   4218      01010         JSR     PRTS0D  PRINT THE MESSAGE
72C4 8E   7121      01020         LDX     #MSG01  KEYPRESS MESSAGE
72C7 BD   4218      01030         JSR     PRTS0D  PRINT THE MESSAGE
                    01040
                    01050 * WAIT FOR A KEYPRESS
72CA BD   4142      01060 LBL002  JSR     POLCAT  GET KEYPRESS
72CD 27   FB        01070         BEQ     LBL002  GO IF Z-BIT OF CC
SET
                    01080
                    01090 *** SECOND SCREEN
72CF BD   400E      01100         JSR     VIDCLS
72D2 8E   0400      01110         LDX     #VIDRAM TOP OF SCREEN
72D5 9F   88        01120         STX     CURPOS  CURSOR
                    01130
                    01140 * DISPLAY MESSAGES
72D7 8E   713A      01150         LDX     #MSG02  BUFFER EMPTY MESSAGE
72DA BD   4218      01160         JSR     PRTS0D  PRINT THE MESSAGE
72DD 8E   7159      01170         LDX     #MSG03  KEYPRESS MESSAGE
72E0 BD   4218      01180         JSR     PRTS0D  PRINT THE MESSAGE
                    01190
                    01200 * WAIT FOR A KEYPRESS
72E3 BD   4142      01210 LBL003  JSR     POLCAT  GET KEYPRESS
72E6 27   FB        01220         BEQ     LBL003  GO IF Z-BIT OF CC
SET
                    01230
                    01240 * PREPARE THE BUFFER
72E8 8E   7005      01250         LDX     #BUFMSG POINT TO MESSAGE
72EB CE   0600      01260         LDU     #DBUF0  BUFFER ADDRESS
72EE 108E 0100      01270         LDY     #256    COUNTER
72F2 A6   80        01280 LBL004  LDA     ,X+     GET CHARACTER
72F4 A7   C0        01290         STA     ,U+     PUT CHARACTER
72F6 31   3F        01300         LEAY    -1,Y    DECREMENT COUNTER
72F8 26   F8        01310         BNE     LBL004  GO IF NOT DONE
                    01320
                    01330 *** THIRD SCREEN
72FA BD   400E      01340         JSR     VIDCLS
72FD 8E   0400      01350         LDX     #VIDRAM TOP OF SCREEN
7300 9F   88        01360         STX     CURPOS  CURSOR
                    01370
```

```
                    01380 * DISPLAY MESSAGES
7302 8E   7176      01390        LDX     #MSG04  BUFFER CONTAINS MSG
7305 BD   4218      01400        JSR     PRTS0D  PRINT THE MESSAGE
7308 BD   40D7      01410        JSR     CRLF
730B 8E   0600      01420        LDX     #DBUF0  BUFFER POINTER
730E 108E 0100      01430        LDY     #256    LENGTH SPECIFICATION
7312 BD   422B      01440        JSR     PRTSLS  PRINT THE BUFFER
7315 BD   40D7      01450        JSR     CRLF
7318 8E   718F      01460        LDX     #MSG05  KEYPRESS MESSAGE
731B BD   4218      01470        JSR     PRTS0D  PRINT THE MESSAGE
                    01480
                    01490 * WAIT FOR A KEYPRESS
731E BD   4142      01500 LBL005 JSR     POLCAT  GET KEYPRESS
7321 27   FB        01510        BEQ     LBL005  GO IF Z-BIT OF CC
SET
                    01520
                    01530 *** FOURTH SCREEN
7323 BD   400E      01540        JSR     VIDCLS
7326 8E   0400      01550        LDX     #VIDRAM TOP OF SCREEN
7329 9F   88        01560        STX     CURPOS  CURSOR
                    01570
                    01580 * SETUP THE DISKRW PARAMETERS
732B 86   03        01590        LDA     #3      WRITE
732D 97   EA        01600        STA     DCOPC
732F 86   00        01610        LDA     #0      DRIVE
7331 97   EB        01620        STA     DCDRV
7333 86   00        01630        LDA     #0      TRACK
7335 97   EC        01640        STA     DCTRK
7337 86   01        01650        LDA     #1      SECTOR
7339 97   ED        01660        STA     DCSEC
733B 8E   0600      01670        LDX     #DBUF0  BUFFER ADDRESS
733E 9F   EE        01680        STX     DCBPT
                    01690
                    01700 * GO DO THE DISK WRITE
7340 BD   4253      01710        JSR     DISKRW  DO THE WRITE
                    01720
                    01730 * CLEAR THE BUFFER
7343 86   20        01740        LDA     #$20    SPACE
7345 8E   0600      01750        LDX     #DBUF0  BUFFER ADDRESS
7348 108E 0100      01760        LDY     #256    COUNTER
734C A7   80        01770 LBL006 STA     ,X+     PUT SPACE
734E 31   3F        01780        LEAY    -1,Y    DECREMENT COUNTER
7350 26   FA        01790        BNE     LBL006  GO IF NOT DONE
                    01800
                    01810 * GET THE DISKRW STATUS
7352 96   F0        01820        LDA     DCSTA   STATUS CODE
                    01830 * INSTEAD OF:
```

```
                  01840 *       BNE     LBLER1  GO TO WRITE ERROR
                  01850 * THESE TWO LINES AVOID BYTE OVERFLOW:
7354 27   03      01860         BEQ     LBLCNT  CONTINUE IF NO ERROR
7356 7E   7413    01870         JMP     LBLER1  GO TO WRITE ERROR
                  01880
                  01890 * REPORT THE RESULTS
7359 BD   40D7    01900 LBLCNT  JSR     CRLF
735C 8E   71ED    01910         LDX     #RPTMSG REPORT HEADER
MESSAGE
735F BD   4218    01920         JSR     PRTS0D  PRINT THE MESSAGE
7362 8E   71F5    01930         LDX     #RPTMS1 REPORT WRITE SUCCESS
7365 BD   40D7    01940         JSR     CRLF
                  01950
                  01960 * DISPLAY MESSAGES
7368 8E   713A    01970         LDX     #MSG02  BUFFER EMPTY MESSAGE
736B BD   4218    01980         JSR     PRTS0D  PRINT THE MESSAGE
736E 8E   7176    01990         LDX     #MSG04  BUFFER CONTAINS MSG
7371 BD   4218    02000         JSR     PRTS0D  PRINT THE MESSAGE
7374 BD   40D7    02010         JSR     CRLF
7377 8E   0600    02020         LDX     #DBUF0  BUFFER POINTER
737A 108E 0100    02030         LDY     #256    LENGTH SPECIFICATION
737E BD   422B    02040         JSR     PRTSLS  PRINT THE BUFFER
7381 BD   40D7    02050         JSR     CRLF
7384 8E   71AE    02060         LDX     #MSG06  KEYPRESS MESSAGE
7387 BD   4218    02070         JSR     PRTS0D  PRINT THE MESSAGE
                  02080
                  02090 * WAIT FOR A KEYPRESS
738A BD   4142    02100 LBL007  JSR     POLCAT  GET KEYPRESS
738D 27   FB      02110         BEQ     LBL007  GO IF Z-BIT OF CC
SET
                  02120
                  02130 *** FIFTH SCREEN
738F BD   400E    02140         JSR     VIDCLS
7392 8E   0400    02150         LDX     #VIDRAM TOP OF SCREEN
7395 9F   88      02160         STX     CURPOS  CURSOR
                  02170
                  02180 * SETUP THE DISKRW PARAMETERS
7397 86   02      02190         LDA     #2      READ
7399 97   EA      02200         STA     DCOPC
739B 86   00      02210         LDA     #0      DRIVE
739D 97   EB      02220         STA     DCDRV
739F 86   00      02230         LDA     #0      TRACK
73A1 97   EC      02240         STA     DCTRK
73A3 86   01      02250         LDA     #1      SECTOR
73A5 97   ED      02260         STA     DCSEC
73A7 8E   0600    02270         LDX     #DBUF0  BUFFER ADDRESS
73AA 9F   EE      02280         STX     DCBPT
```

```
                    02290
                    02300 * GO DO THE DISK READ
73AC BD   4253      02310        JSR    DISKRW  DO THE READ
                    02320
                    02330 * GET THE DISKRW STATUS
73AF 96   F0        02340        LDA    DCSTA   STATUS CODE
73B1 26   68        02350        BNE    LBLER2  GO TO READ ERROR
                    02360
                    02370 * REPORT THE RESULTS
73B3 BD   40D7      02380        JSR    CRLF
73B6 8E   71ED      02390        LDX    #RPTMSG REPORT HEADER
MESSAGE
73B9 BD   4218      02400        JSR    PRTS0D  PRINT THE MESSAGE
73BC 8E   7210      02410        LDX    #RPTMS2 REPORT READ SUCCESS
73BF BD   4218      02420        JSR    PRTS0D  PRINT THE MESSAGE
73C2 BD   40D7      02430        JSR    CRLF
73C5 8E   7176      02440        LDX    #MSG04  BUFFER CONTAINS MSG
73C8 BD   4218      02450        JSR    PRTS0D  PRINT THE MESSAGE
73CB BD   40D7      02460        JSR    CRLF
73CE 8E   0600      02470        LDX    #DBUF0  BUFFER POINTER
73D1 108E 0100      02480        LDY    #256    LENGTH SPECIFICATION
73D5 BD   422B      02490        JSR    PRTSLS  PRINT THE BUFFER
73D8 BD   40D7      02500        JSR    CRLF
73DB 8E   722A      02510        LDX    #RPTMS3 REPORT TEST SUCCESS
73DE BD   4218      02520        JSR    PRTS0D  PRINT THE MESSAGE
73E1 8E   7246      02530        LDX    #RPTMS4 REPORT END OF TEST
73E4 BD   4218      02540        JSR    PRTS0D  PRINT THE MESSAGE
73E7 BD   40D7      02550        JSR    CRLF
73EA 8E   725D      02560        LDX    #RPTMS5 KEYPRESS MESSAGE
73ED BD   4218      02570        JSR    PRTS0D  PRINT THE MESSAGE
                    02580
                    02590 * WAIT FOR A KEYPRESS
73F0 BD   4142      02600 LBL008 JSR    POLCAT  GET KEYPRESS
73F3 27   FB        02610        BEQ    LBL008  GO IF Z-BIT OF CC
SET
                    02620
                    02630 *** SIXTH SCREEN
73F5 BD   400E      02640        JSR    VIDCLS
73F8 8E   0400      02650        LDX    #VIDRAM TOP OF SCREEN
73FB 9F   88        02660        STX    CURPOS  CURSOR
                    02670
                    02680 * DISPLAY MESSAGES
73FD 8E   71CE      02690        LDX    #MSG07  SYSTEM DISK MESSAGE
7400 BD   4218      02700        JSR    PRTS0D  PRINT THE MESSAGE
7403 8E   7121      02710        LDX    #MSG01  KEYPRESS MESSAGE
7406 BD   4218      02720        JSR    PRTS0D  PRINT THE MESSAGE
                    02730
```

```
                  02740 * WAIT FOR A KEYPRESS
7409 BD   4142    02750 LBL009  JSR     POLCAT  GET KEYPRESS
740C 27   FB      02760         BEQ     LBL009  GO IF Z-BIT OF CC
SET
740E BD   40D7    02770         JSR     CRLF
7411 20   1D      02780         BRA     LBLEXI  GO TO EXIT
                  02790
                  02800 * ERROR REPORT
7413 8E   7273    02810 LBLER1  LDX     #ERRMS1 WRITE ERROR MESSAGE
7416 BD   4218    02820         JSR     PRTS0D  PRINT THE MESSAGE
7419 20   06      02830         BRA     LBLER3
741B 8E   728A    02840 LBLER2  LDX     #ERRMS2 READ ERROR MESSAGE
741E BD   4218    02850         JSR     PRTS0D  PRINT THE MESSAGE
7421 BD   40D7    02860 LBLER3  JSR     CRLF
7424 BD   40D7    02870         JSR     CRLF
7427 8E   72A1    02880         LDX     #ERRMS3 ABORT MESSAGE
742A BD   4218    02890         JSR     PRTS0D  PRINT THE MESSAGE
742D BD   40D7    02900         JSR     CRLF
                  02910
                  02920 * EXIT
7430 35   73      02930 LBLEXI  PULS    A,X,Y,U,CC
7432 39           02940         RTS
                  02950
          0000    02960         END
```

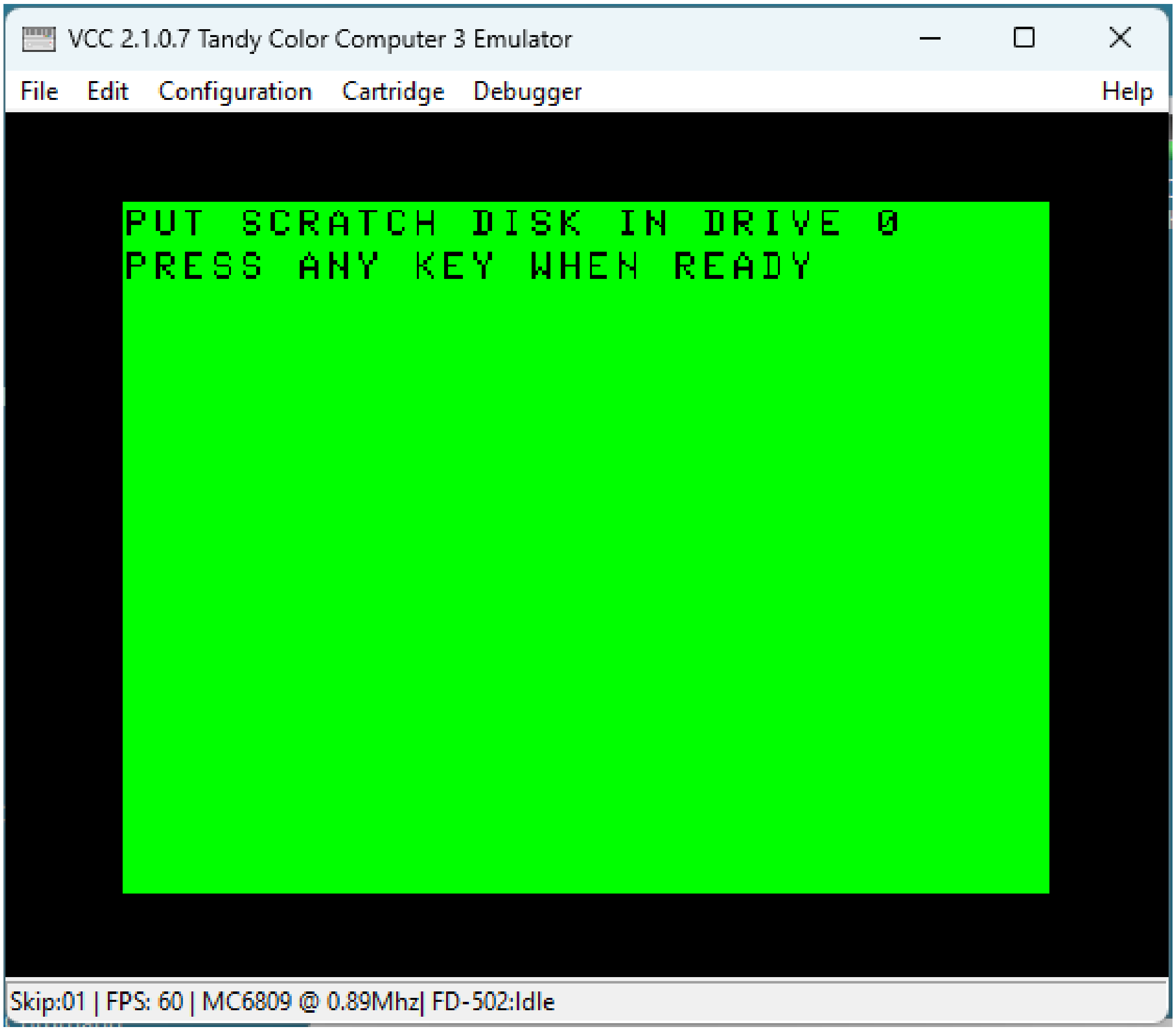
The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0026.BAS
1030 '* MDJ 2023/02/16
1040 '*
1050 '* DISKRW 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"
```

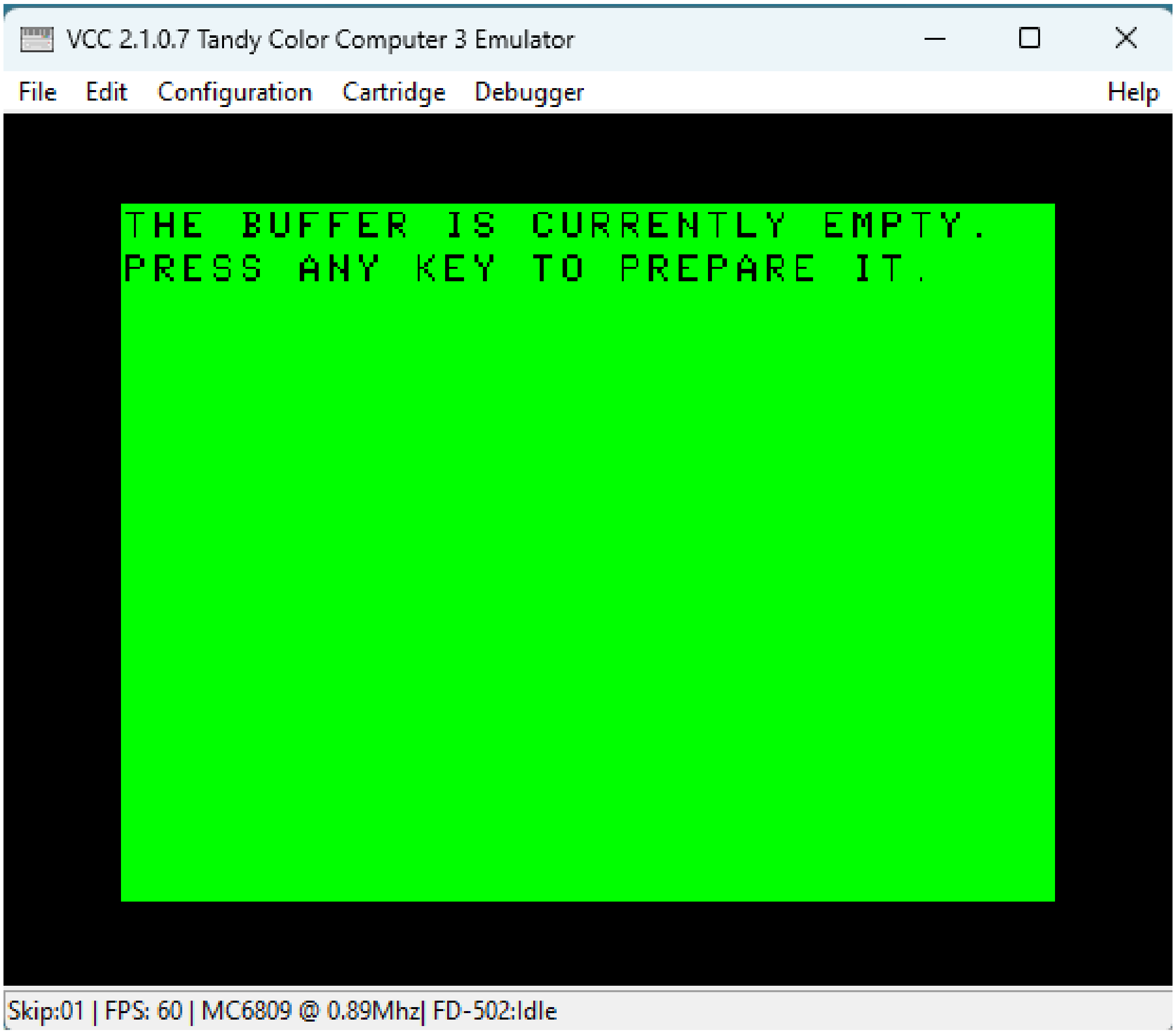
```
1230 '

1300 'LOAD THE
1310 'ML TEST ROUTINE
1320 LOADM "TEST0026.BIN"
1330 '

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
```

For this test, the results report requires six consecutive screens.

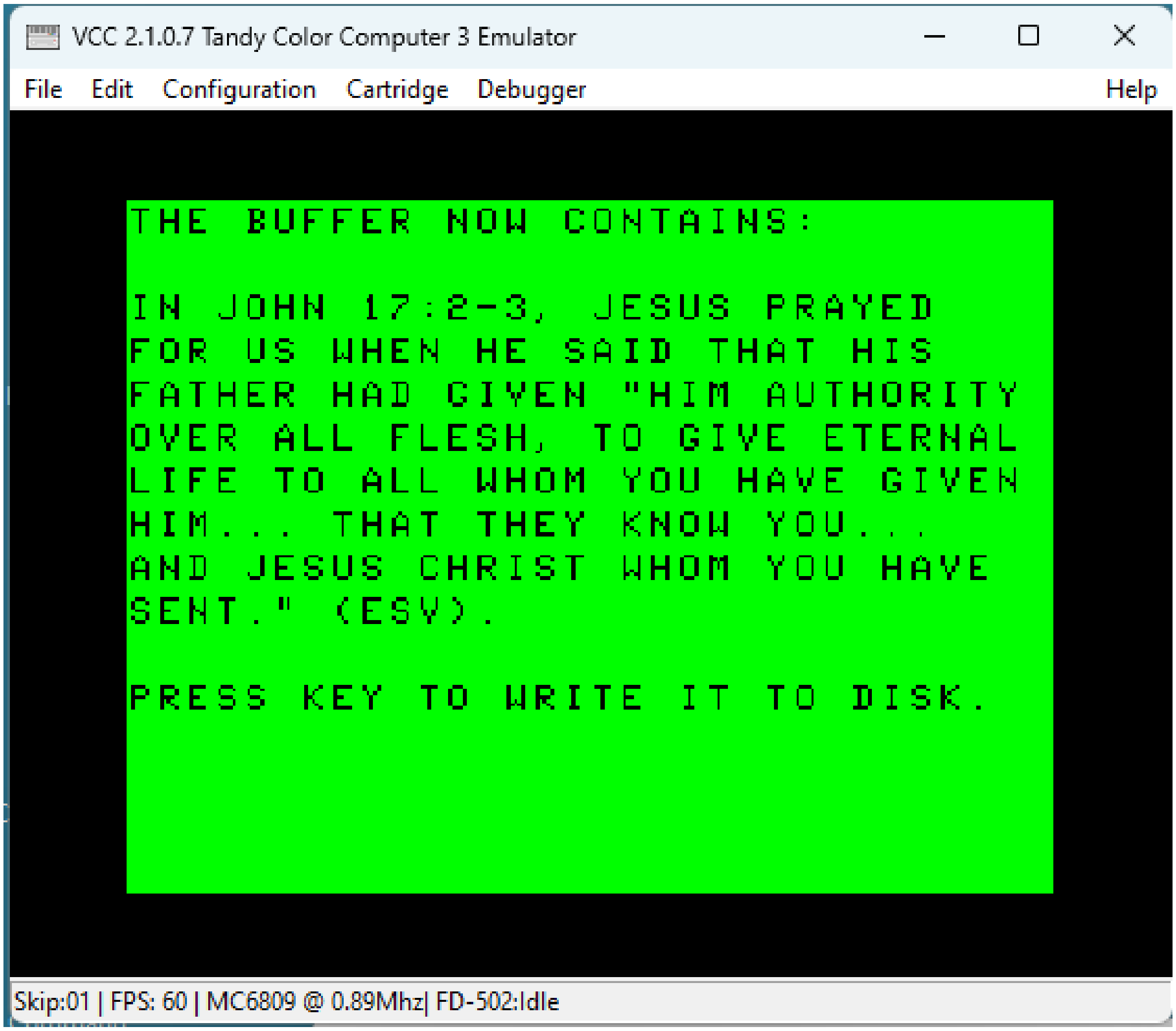
Result Screen One of Six:



Result Screen Two of Six:

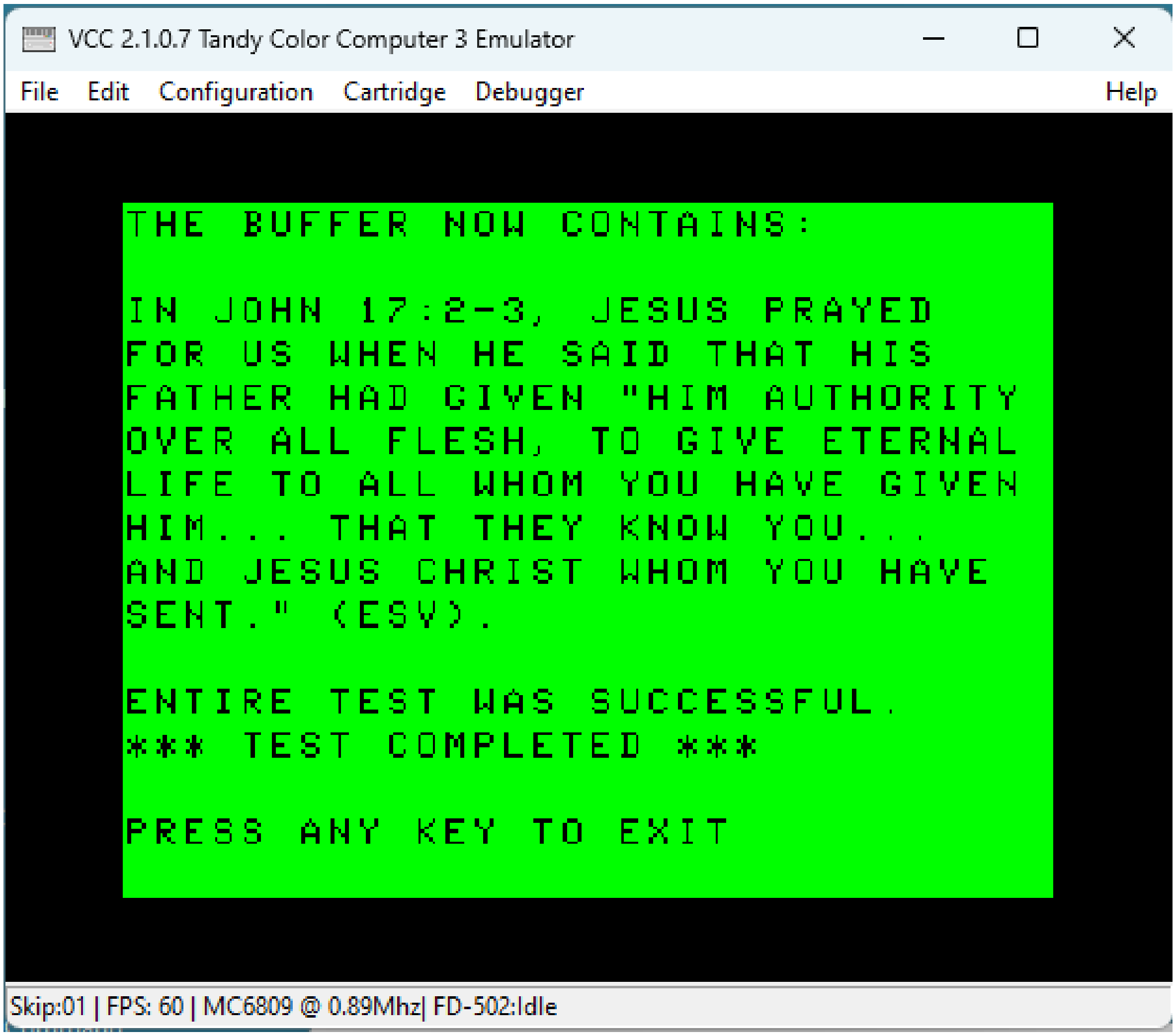


Result Screen Three of Six:

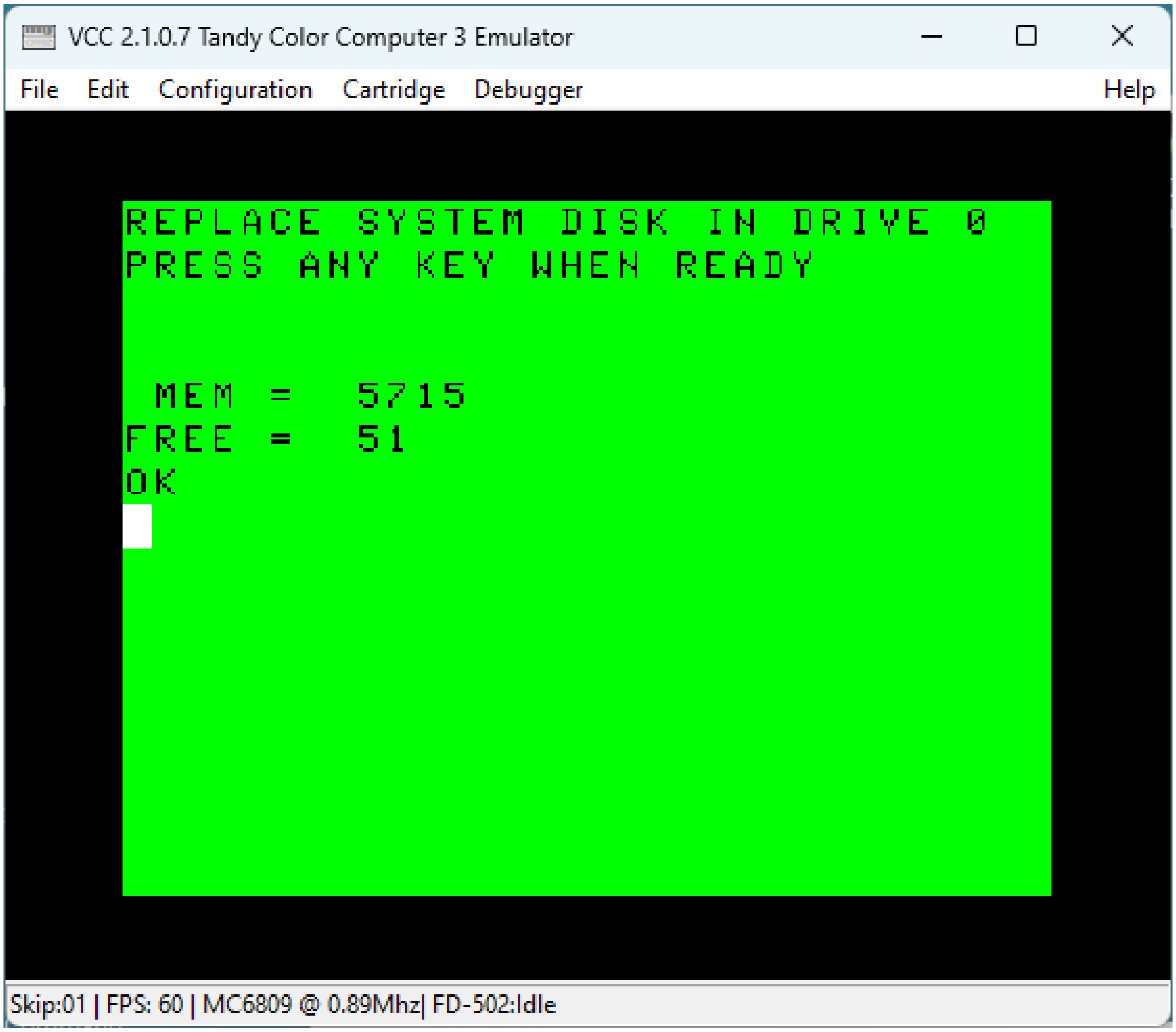


Result Screen Four of Six:

Result Screen Five of Six:



Result Screen Six of Six:



All as expected.

Bible Note: The full passage at John 17:1-5 is:

> 1 When Jesus had spoken these words, he lifted up his eyes to heaven, and said, “Father, the hour has come; glorify your Son that the Son may glorify you, 2 since you have given him authority over all flesh, to give eternal life to all whom you have given him. 3 And this is eternal life, that they know you, the only true God, and Jesus Christ whom you have sent. 4 I glorified you on earth, having accomplished the work that you gave me to do. 5 And now, Father, glorify me in your own presence with the glory that I had with you before the world existed.” (ESV).

And Jesus specifically applies these words to us by what He says in John 17:20-21.

[20] "I do not ask for these only, but also for those who will believe in me through their word, [21] that they may all be one, just as you, Father, are in me, and I in you, that they also may be in us, so that the world may believe that you have sent me." (ESV).

=====

MU0808: 8-bit by 8-bit Unsigned Multiply

This multiplies an 8-bit unsigned byte by another 8-bit unsigned byte and returns a 16-bit unsigned result.

This is just a wrapper for the MC6809 MUL Instruction and is provided simply for convenience and uniformity of presentation. You can save four bytes and nine MPU cycles (3 bytes and 4 MPU cycles for the JSR to this routine, and 1 byte and 5 MPU cycles for the RTS) by simply executing MUL instead.

```
                    00100 *****
                    00110 *
                    00120 * MU0808.ASM
                    00130 * MDJ 2023/02/06
                    00140 *
                    00150 * 8-BIT BY 8-BIT
                    00160 * UNSIGNED MULTIPLY
                    00170 *
                    00180 * ENTRY CONDITIONS:
                    00190 * A = MULTIPLICAND #1
                    00200 * B = MULTIPLICAND #2
                    00210 *
                    00220 * EXIT CONDITIONS:
                    00230 * D = RESULT
                    00240 *
                    00250 *****
                    00260 *
                    00270 * THIS IS JUST A WRAPPER
                    00280 * FOR MUL - IT'S PROVIDED
                    00290 * SIMPLY FOR UNIFORMITY
                    00300 *
                    00310 *****
                    00320
4265                00330         ORG     $4265
                    00340
4265 3D             00350 MU0808  MUL
                    00360
4266 39             00370         RTS
                    00380
          0000      00390         END
```

No testing of this wrapper was performed.

=====

MU1608: 16-bit by 8-bit Unsigned Multiply

This multiplies a 16-bit unsigned word by an 8-bit unsigned byte and returns a 32-bit unsigned result.

Although the result is provided as a 32-bit unsigned double-word, it is actually a 24-bit unsigned number: Upon return, the high 8-bits of the X-Register are always zero.

```
                   *****
                   *
                   * MU1608.ASM
                   * MDJ 2023/02/06
                   *
                   * 16-BIT BY 8-BIT
                   * UNSIGNED MULTIPLY
                   *
                   * ENTRY CONDITIONS
                   * X = 16-BIT MULTIPLICAND
                   * B =  8-BIT MULTIPLICAND
                   *
                   * EXIT CONDITIONS:
                   * X = HIGH 16-BITS
                   * Y = LOW 16-BITS
                   *     OF RESULT
                   *
                   *****

4267                       ORG     $4267

4267 34   26       MU1608  PSHS    A,B,Y   NOTE: THESE ARE NOT ACTUALLY PRESERVED

                   * AT THIS POINT, THE STACK CONTAINS:
                   *    5,S RTS LOCATION LOW BYTE
                   *    4,S RTS LOCATION HIGH BYTE
                   *    3,S REGISTER Y LOW BYTE
                   *    2,S REGISTER Y HIGH BYTE
                   *    1,S REGISTER B
                   *     ,S REGISTER A

                   * MAKE SPACE ON THE STACK FOR:
                   *   14,S RTS LOCATION LOW BYTE
                   *   13,S RTS LOCATION HIGH BYTE
```

```
                   00440 *   12,S REGISTER Y LOW BYTE
                   00450 *   11,S REGISTER Y HIGH BYTE
                   00460 *   10,S REGISTER B
                   00470 *    9,S REGISTER A
                   00480 *    7,S HI1 * LO2 RESULT LOW BYTE
                   00490 *    6,S HI1 * LO2 RESULT HIGH BYTE
                   00500 *    5,S LO1 * LO2 RESULT LOW BYTE
                   00510 *    4,S LO1 * LO2 RESULT HIGH BYTE
                   00520 *    3,S 8-BIT SEX MULTIPLICAND LOW BYTE
                   00530 *          AND RESULT BYTE #0
                   00540 *    2,S 8-BIT SEX MULTIPLICAND HIGH BYTE
                   00550 *          ( = #$00 )
                   00560 *          AND RESULT BYTE #1
                   00570 *    1,S 16-BIT MULTIPLICAND LOW BYTE
                   00580 *          AND RESULT BYTE #2
                   00590 *     ,S 16-BIT MULTIPLICAND HIGH BYTE
                   00600 *          AND RESULT BYTE #3
                   00610 *          ( = #$00 )
4269 32   78       00620        LEAS    -8,S
                   00630
                   00640 * MOVE THE 8-BIT MULTIPLICAND TO Y
426B 4F            00650        CLRA
426C 1F   02       00660        TFR     D,Y
                   00670
                   00680 * PUT THE MULTIPLICANDS ON THE STACK
426E AF   E4       00690        STX     ,S
4270 10AF 62       00700        STY     2,S
                   00710
                   00720 * DO LO1 * LO2
4273 A6   61       00730        LDA     1,S
4275 E6   63       00740        LDB     3,S
4277 3D            00750        MUL
4278 ED   64       00760        STD     4,S
                   00770
                   00780 * DO HI1 * LO2
427A A6   E4       00790        LDA     ,S
427C E6   63       00800        LDB     3,S
427E 3D            00810        MUL
427F ED   66       00820        STD     6,S
                   00830
                   00840 * FORM RESULT BYTE #0
4281 A6   65       00850        LDA     5,S     LO1 * LO2 LOW BYTE
4283 A7   63       00860        STA     3,S
                   00870
                   00880 * FORM RESULT BYTE #1
4285 A6   64       00890        LDA     4,S     LO1 * LO2 HIGH BYTE
4287 AB   67       00900        ADDA    7,S     HI1 * LO2 LOW BYTE
```

```
4289 A7   62       00910        STA     2,S
                   00920
                   00930 * FORM RESULT BYTE #2
428B A6   66       00940        LDA     6,S     HI1 * LO2 HIGH BYTE
428D 89   00       00950        ADCA    #$00    ACCOUNT FOR POSSIBLE
                   00960 *                      CARRY FROM RESULT
                   00970 *                      BYTE #1
428F A7   61       00980        STA     1,S
                   00990
                   01000 * FORM RESULT BYTE #3 = #$00
4291 86   00       01010        LDA     #$00
4293 A7   E4       01020        STA     ,S
                   01030
                   01040 * LOAD THE RESULTS
4295 AE   E4       01050        LDX     ,S
4297 10AE 62       01060        LDY     2,S
                   01070
                   01080 * CLEAR THE STACK
                   01090 * INCLUDING REGISTERS A, B, AND Y
429A 32   6C       01100        LEAS    12,S
                   01110
                   01120 * EXIT
429C 39            01130        RTS
                   01140
          0000     01150        END
```

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0027.ASM
                   00130 * MDJ 2023/02/17
                   00140 *
                   00150 * MU1608 TEST
                   00160 *
                   00170 *****
                   00180
                   00190 * BASIC/ML TRANSFER
                   00200 * VARIABLES
          4001     00210 REGB   EQU     $4001
          4002     00220 REGX   EQU     $4002
                   00230
                   00240 * ML FOUNDATION
                   00250 * CORE ADDRESSES
          40D7     00260 CRLF   EQU     $40D7
```

```
            4178      00270 PUTWRA  EQU     $4178
            4267      00280 MU1608  EQU     $4267
                      00290
7000                  00300         ORG     $7000
                      00310
                      00320 * MU1608 TEST
                      00330
7000 34     36        00340         PSHS    A,B,X,Y
                      00350
7002 BE     4002      00360         LDX     REGX    GET THE TEST VALUES
7005 F6     4001      00370         LDB     REGB    FROM THE XFER
VARIABLES
                      00380
7008 BD     4267      00390         JSR     MU1608  GO DO THE MULTIPLY
                      00400
700B 1F     10        00410         TFR     X,D     PRINT THE 32-BIT
RESULT
700D BD     4178      00420         JSR     PUTWRA
7010 1F     20        00430         TFR     Y,D
7012 BD     4178      00440         JSR     PUTWRA
7015 BD     40D7      00450         JSR     CRLF
                      00460
                      00470 * EXIT
7018 35     36        00480         PULS    A,B,X,Y
701A 39               00490         RTS
                      00500
            0000      00510         END
```

____

The BASIC Language Control Program, First Run:

```
1000 '*****
1010 '*
1020 '* TEST0027.BAS
1030 '* MDJ 2023/02/16
1040 '*
1050 '* MU1608 TEST
1060 '*
1070 '*****
1080 '

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

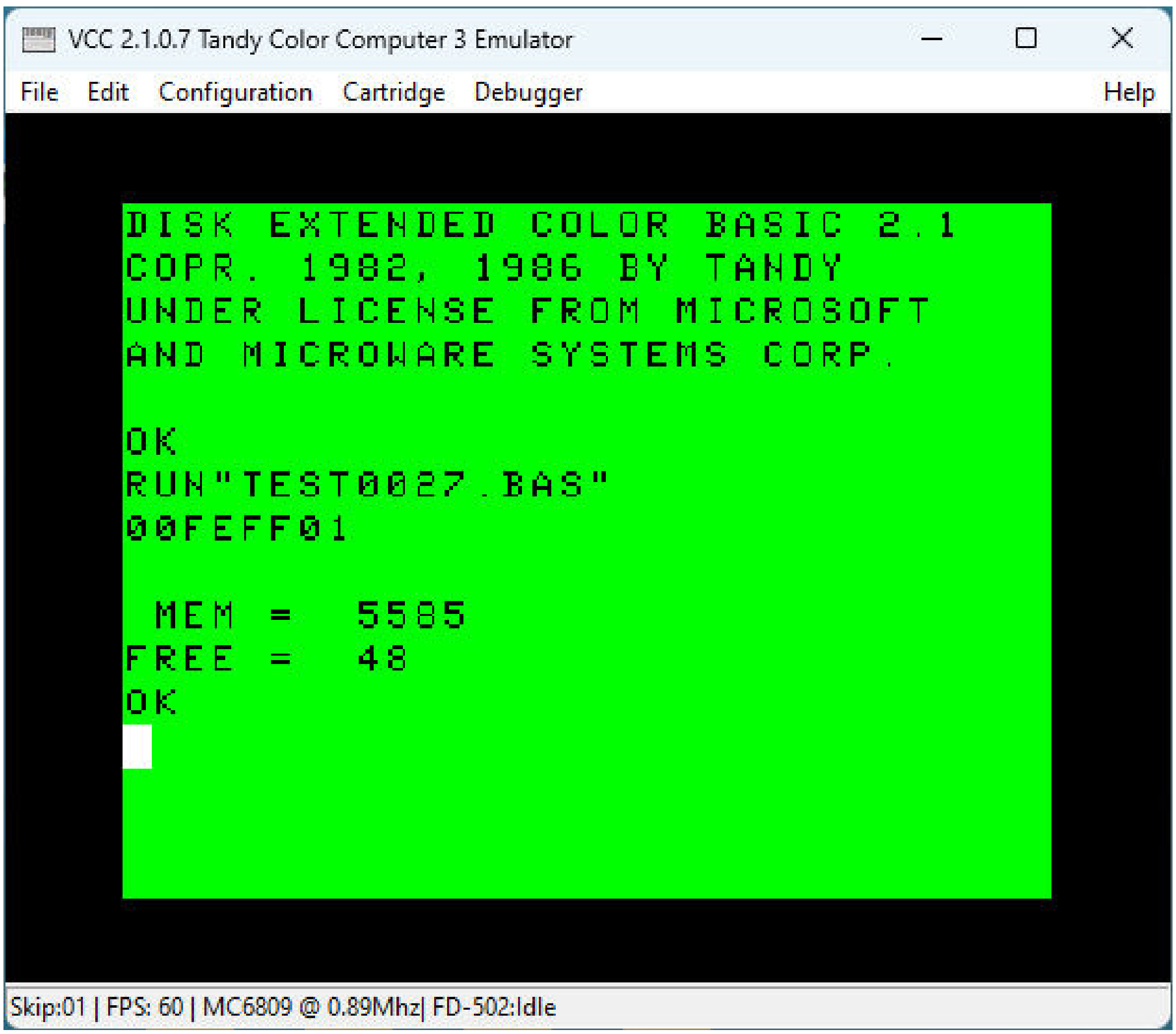
```
1200 'LOAD THE
1210 'ML FOUNDATION CORE
1220 LOADM "MLCORE.BIN"
1230 '

1300 'LOAD THE
1310 'ML TEST ROUTINE
1320 LOADM "TEST0027.BIN"
1330 '

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 '

4000 'LOAD THE TEST DATA
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B

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
```

With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B
```

—

Result: $FFFF x $FF = $FEFF01



As expected.

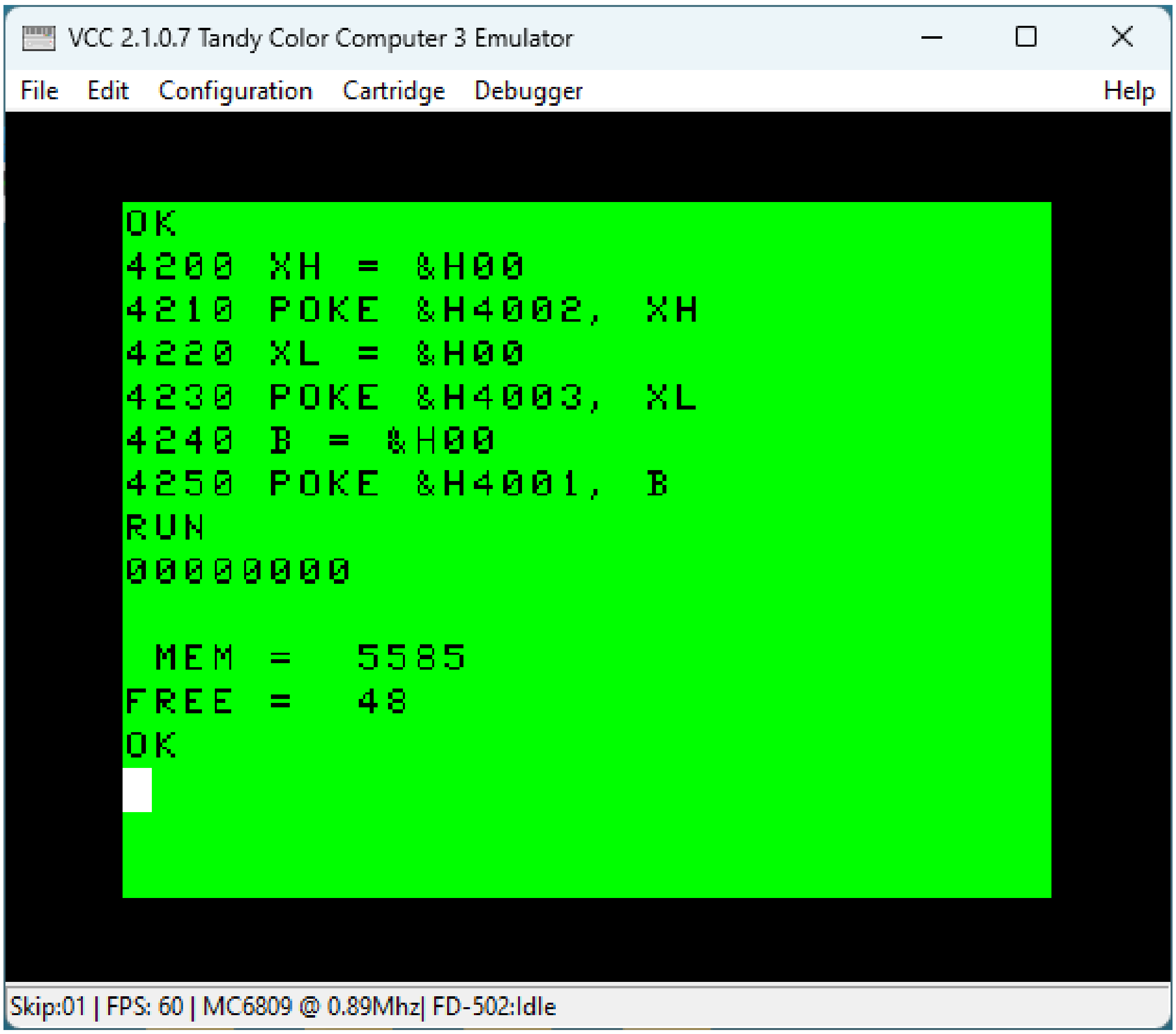
—

Second Run: - With Test Data:

```
4200 XH = &H00
4210 POKE &H4002, XH
4220 XL = &H00
4230 POKE &H4003, XL
4240 B = &H00
4250 POKE &H4001, B
```

—

Result: 0 x 0 = 0



As expected.

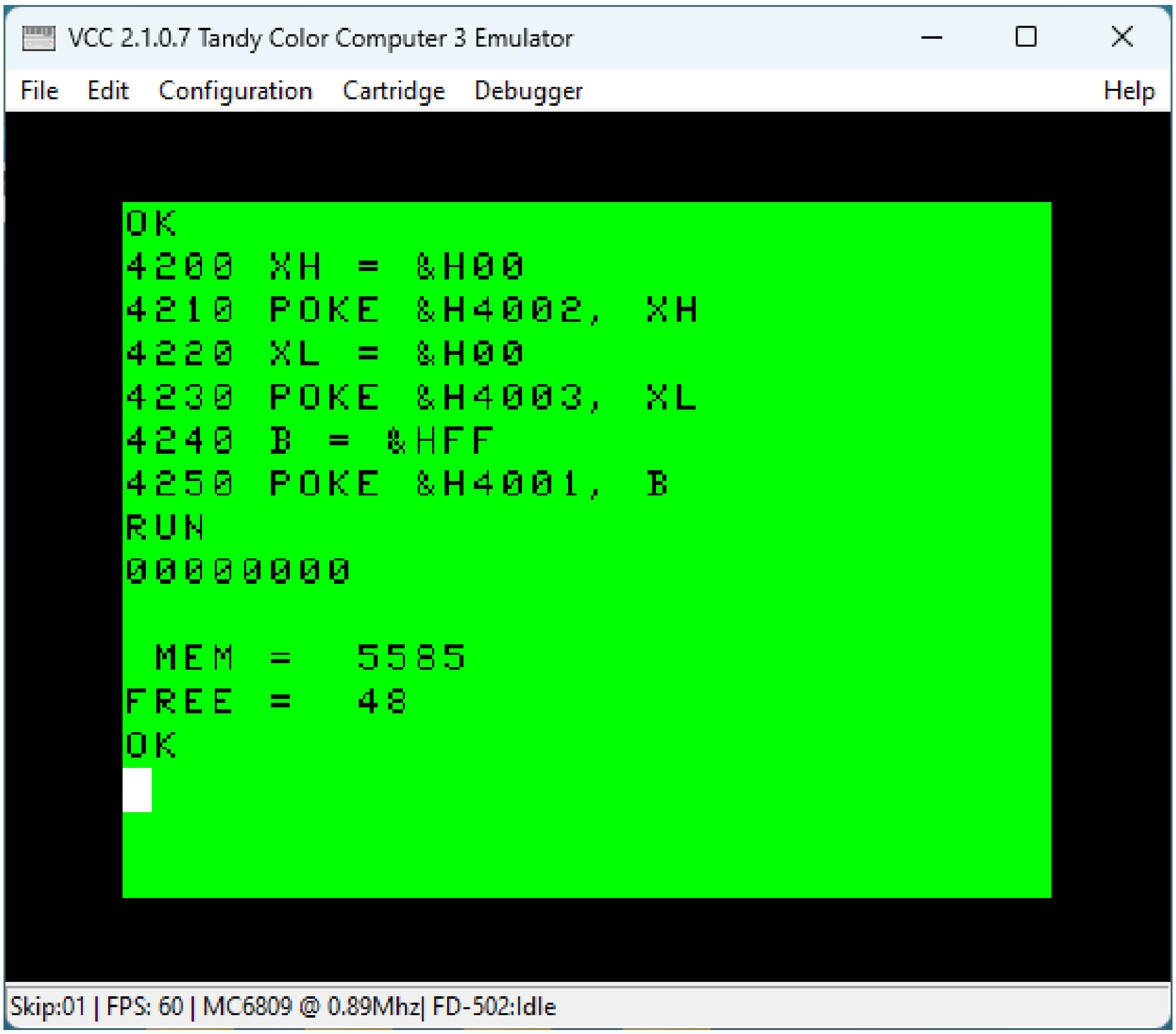
—

Third Run: - With Test Data:

```
4200 XH = &H00
4210 POKE &H4002, XH
4220 XL = &H00
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B
```

—

Result: 0 x $FF = 0



As expected.

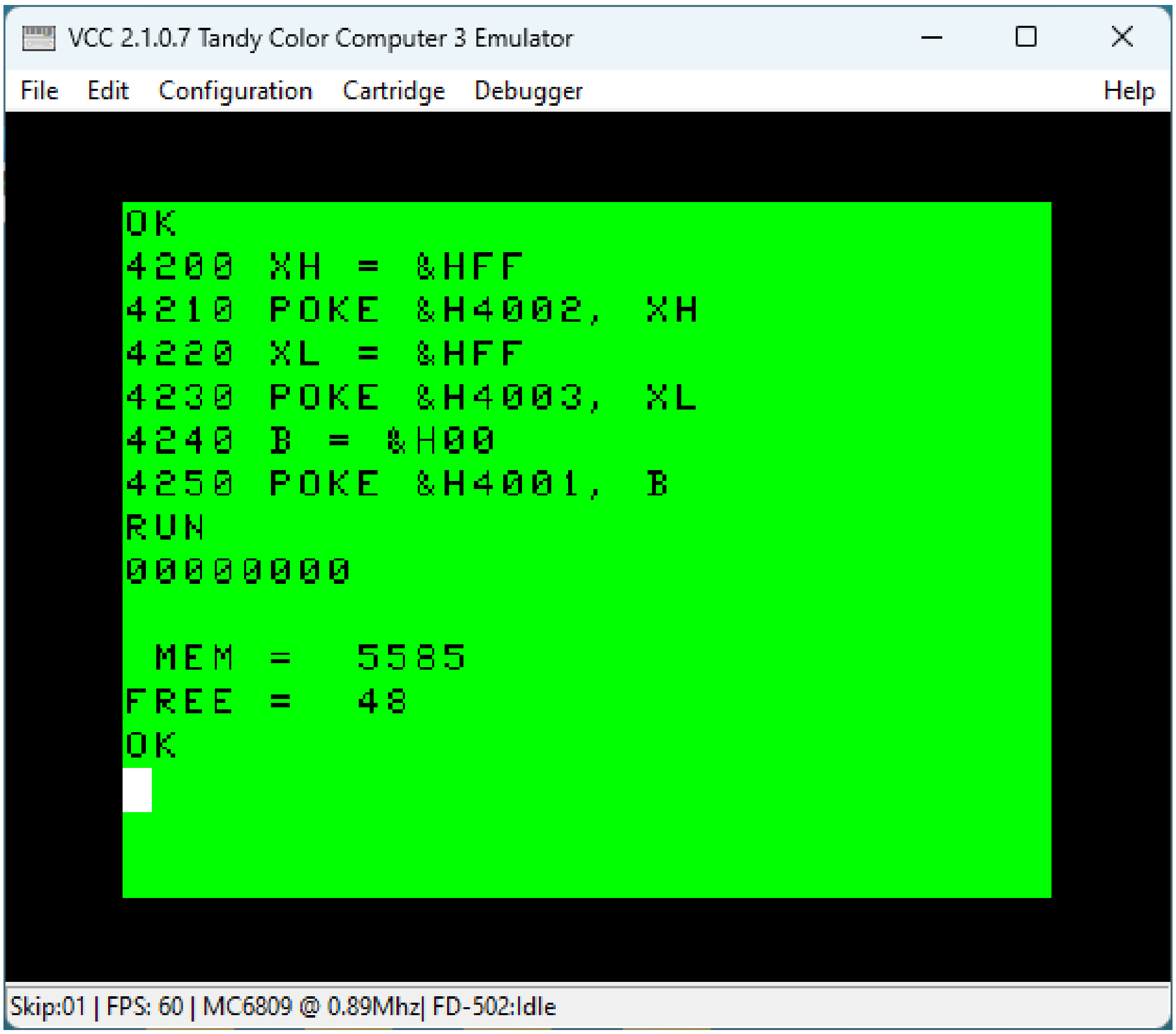
—

Fourth Run: - With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &H00
4250 POKE &H4001, B
```

—

Result: $FFFF x 0 = 0



As expected.

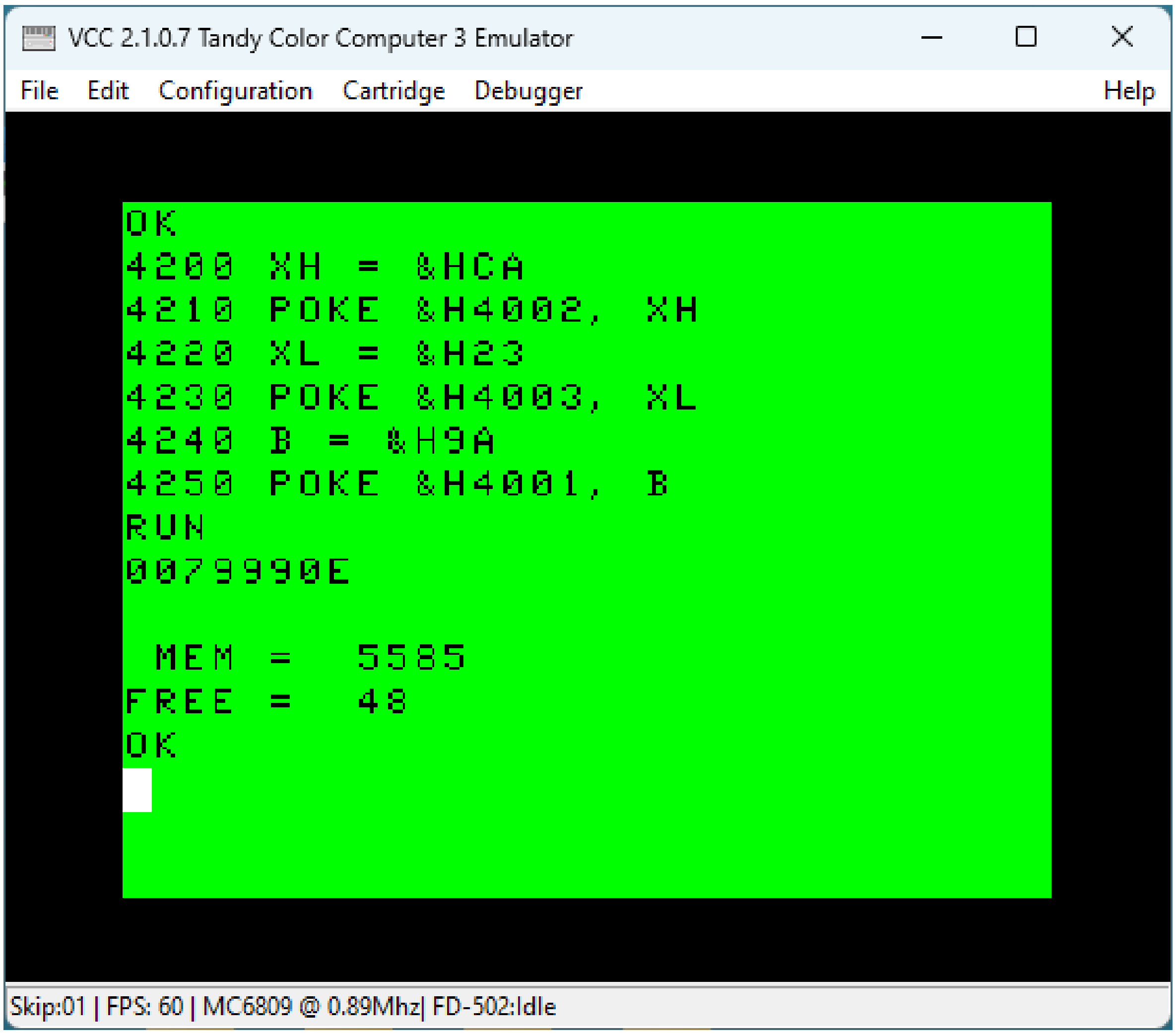
—

Fifth Run: - With Test Data:

```
4200 XH = &HCA
4210 POKE &H4002, XH
4220 XL = &H23
4230 POKE &H4003, XL
4240 B = &H9A
4250 POKE &H4001, B
```

—

Result: \$CA23 x \$9A = \$79990E



As expected.

=====

MU1616: 16-bit by 16-bit Unsigned Multiply

This multiplies a 16-bit unsigned word by another 16-bit unsigned word and returns a 32-bit unsigned result.

```
                   *****
                   *
                   * MU1616.ASM
                   * MDJ 2023/02/03
                   *
                   * 16-BIT BY 16-BIT
                   * UNSIGNED MULTIPLY
                   *
                   * ENTRY CONDITIONS
                   * X = MULTIPLICAND #1
                   * Y = MULTIPLICAND #2
                   *
                   * EXIT CONDITIONS:
                   * X = HIGH 16-BITS
                   * Y = LOW 16-BITS
                   *     OF RESULT
                   *
                   *****

429D                ORG     $429D

429D 34   06       MU1616  PSHS    A,B     NOTE: THESE ARE NOT ACTUALLY PRESERVED

                   * AT THIS POINT, THE STACK CONTAINS:
                   *    3,S RTS LOCATION LOW BYTE
                   *    2,S RTS LOCATION HIGH BYTE
                   *    1,S REGISTER B
                   *     ,S REGISTER A

                   * MAKE SPACE ON THE STACK FOR:
                   *   16,S RTS LOCATION LOW BYTE
                   *   15,S RTS LOCATION HIGH BYTE
                   *   14,S REGISTER B
                   *   13,S REGISTER A
                   *   12,S CARRIES-HOLD BYTE
                   *   11,S HI1 * HI2 RESULT LOW BYTE
                   *   10,S HI1 * HI2 RESULT HIGH BYTE
```

```
                    00470 *    9,S HI1 * LO2 RESULT LOW BYTE
                    00480 *    8,S HI1 * LO2 RESULT HIGH BYTE
                    00490 *    7,S LO1 * HI2 RESULT LOW BYTE
                    00500 *    6,S LO1 * HI2 RESULT HIGH BYTE
                    00510 *    5,S LO1 * LO2 RESULT LOW BYTE
                    00520 *    4,S LO1 * LO2 RESULT HIGH BYTE
                    00530 *    3,S MULTIPLICAND #2 LOW BYTE
                    00540 *          AND RESULT BYTE #0
                    00550 *    2,S MULTIPLICAND #2 HIGH BYTE
                    00560 *          AND RESULT BYTE #1
                    00570 *    1,S MULTIPLICAND #1 LOW BYTE
                    00580 *          AND RESULT BYTE #2
                    00590 *     ,S MULTIPLICAND #1 HIGH BYTE
                    00600 *          AND RESULT BYTE #3
429F 32   73        00610        LEAS    -13,S
                    00620
                    00630 * PUT THE MULTIPLICANDS ON THE STACK
42A1 AF   E4        00640        STX     ,S
42A3 10AF 62        00650        STY     2,S
                    00660
                    00670 * DO LO1 * LO2
42A6 A6   61        00680        LDA     1,S
42A8 E6   63        00690        LDB     3,S
42AA 3D             00700        MUL
42AB ED   64        00710        STD     4,S
                    00720
                    00730 * DO LO1 * HI2
42AD A6   61        00740        LDA     1,S
42AF E6   62        00750        LDB     2,S
42B1 3D             00760        MUL
42B2 ED   66        00770        STD     6,S
                    00780
                    00790 * DO HI1 * LO2
42B4 A6   E4        00800        LDA     ,S
42B6 E6   63        00810        LDB     3,S
42B8 3D             00820        MUL
42B9 ED   68        00830        STD     8,S
                    00840
                    00850 * DO HI1 * HI2
42BB A6   E4        00860        LDA     ,S
42BD E6   62        00870        LDB     2,S
42BF 3D             00880        MUL
42C0 ED   6A        00890        STD     10,S
                    00900
                    00910 * FORM RESULT BYTE #0
42C2 A6   65        00920        LDA     5,S     LO1 * LO2 LOW BYTE
42C4 A7   63        00930        STA     3,S
```

```
                    00940
                    00950 * FORM RESULT BYTE #1
42C6 6F   6C        00960         CLR     12,S    CLEAR CARRIES-HOLD
42C8 A6   64        00970         LDA     4,S     LO1 * LO2 HIGH BYTE
42CA AB   67        00980         ADDA    7,S     LO1 * HI2 LOW BYTE
42CC 24   02        00990         BCC     LBL001  GO IF NO CARRY
42CE 6C   6C        01000         INC     12,S    INCREMENT CARRIES-
HOLD
42D0 AB   69        01010 LBL001  ADDA    9,S     HI1 * LO2 LOW BYTE
42D2 24   02        01020         BCC     LBL002  GO IF NO CARRY
42D4 6C   6C        01030         INC     12,S    INCREMENT CARRIES-
HOLD
42D6 A7   62        01040 LBL002  STA     2,S
                    01050
                    01060 * FORM RESULT BYTE #2
42D8 A6   6C        01070         LDA     12,S    GET CARRIES-HOLD
42DA 6F   6C        01080         CLR     12,S    CLEAR CARRIES-HOLD
42DC AB   66        01090         ADDA    6,S     LO1 * HI2 HIGH BYTE
42DE 24   02        01100         BCC     LBL003  GO IF NO CARRY
42E0 6C   6C        01110         INC     12,S    INCREMENT CARRIES-
HOLD
42E2 AB   68        01120 LBL003  ADDA    8,S     HI1 * LO2 HIGH BYTE
42E4 24   02        01130         BCC     LBL004  GO IF NO CARRY
42E6 6C   6C        01140         INC     12,S    INCREMENT CARRIES-
HOLD
42E8 AB   6B        01150 LBL004  ADDA    11,S    HI1 * HI2 LO BYTE
42EA 24   02        01160         BCC     LBL005  GO IF NO CARRY
42EC 6C   6C        01170         INC     12,S    INCREMENT CARRIES-
HOLD
42EE A7   61        01180 LBL005  STA     1,S
                    01190
                    01200 * FORM RESULT BYTE #3
42F0 A6   6C        01210         LDA     12,S    GET CARRIES-HOLD
42F2 AB   6A        01220         ADDA    10,S    HI1 * HI2 HIGH BYTE
42F4 A7   E4        01230         STA     ,S
                    01240
                    01250 * LOAD THE RESULTS
42F6 AE   E4        01260         LDX     ,S
42F8 10AE 62        01270         LDY     2,S
                    01280
                    01290 * CLEAR THE STACK
                    01300 * INCLUDING REGISTERS A AND B
42FB 32   6F        01310         LEAS    15,S
                    01320
                    01330 * EXIT
42FD 39             01340         RTS
                    01350
```

```
        0000      01360           END
```

The Assembly Language Test Routine:

```
                  00100 *****
                  00110 *
                  00120 * TEST0028.ASM
                  00130 * MDJ 2023/02/17
                  00140 *
                  00150 * MU1616 TEST
                  00160 *
                  00170 *****
                  00180
                  00190 * BASIC/ML TRANSFER
                  00200 * VARIABLES
          4002    00210 REGX    EQU     $4002
          4004    00220 REGY    EQU     $4004
                  00230
                  00240 * ML FOUNDATION
                  00250 * CORE ADDRESSES
          40D7    00260 CRLF    EQU     $40D7
          4178    00270 PUTWRA  EQU     $4178
          429D    00280 MU1616  EQU     $429D
                  00290
7000              00300         ORG     $7000
                  00310
                  00320 * MU1616 TEST
                  00330
7000 34   36      00340         PSHS    A,B,X,Y
                  00350
7002 BE   4002    00360         LDX     REGX    GET THE TEST VALUES
7005 10BE 4004    00370         LDY     REGY    FROM THE XFER
VARIABLES
                  00380
7009 BD   429D    00390         JSR     MU1616  GO DO THE MULTIPLY
                  00400
700C 1F   10      00410         TFR     X,D     PRINT THE 32-BIT
RESULT
700E BD   4178    00420         JSR     PUTWRA
7011 1F   20      00430         TFR     Y,D
7013 BD   4178    00440         JSR     PUTWRA
7016 BD   40D7    00450         JSR     CRLF
                  00460
                  00470 * EXIT
7019 35   36      00480         PULS    A,B,X,Y
```

```
701B 39             00490          RTS
                    00500
          0000      00510          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0028.BAS
1030 '* MDJ 2023/02/17
1040 '*
1050 '* MU1616 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0028.BIN"
1330 '

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 '
```

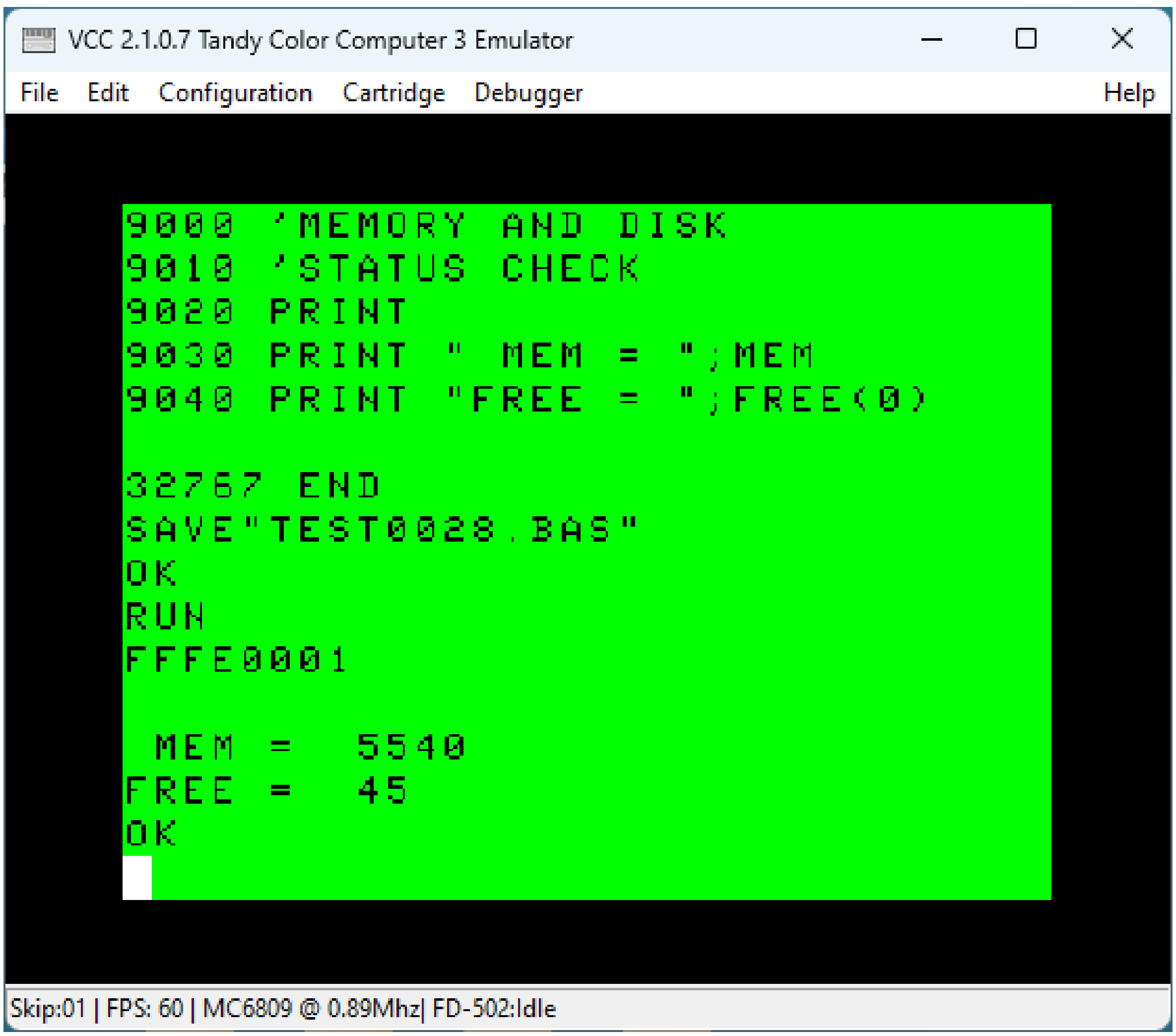
```
4000 'LOAD THE TEST DATA
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &HFF
4250 POKE &H4004, YH
4260 YL = &HFF
4270 POKE &H4005, YL

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
```

With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &HFF
4250 POKE &H4004, YH
4260 YL = &HFF
4270 POKE &H4005, YL
```

—

Result: $FFFF x $FFFF = $FFFE0001



As expected.

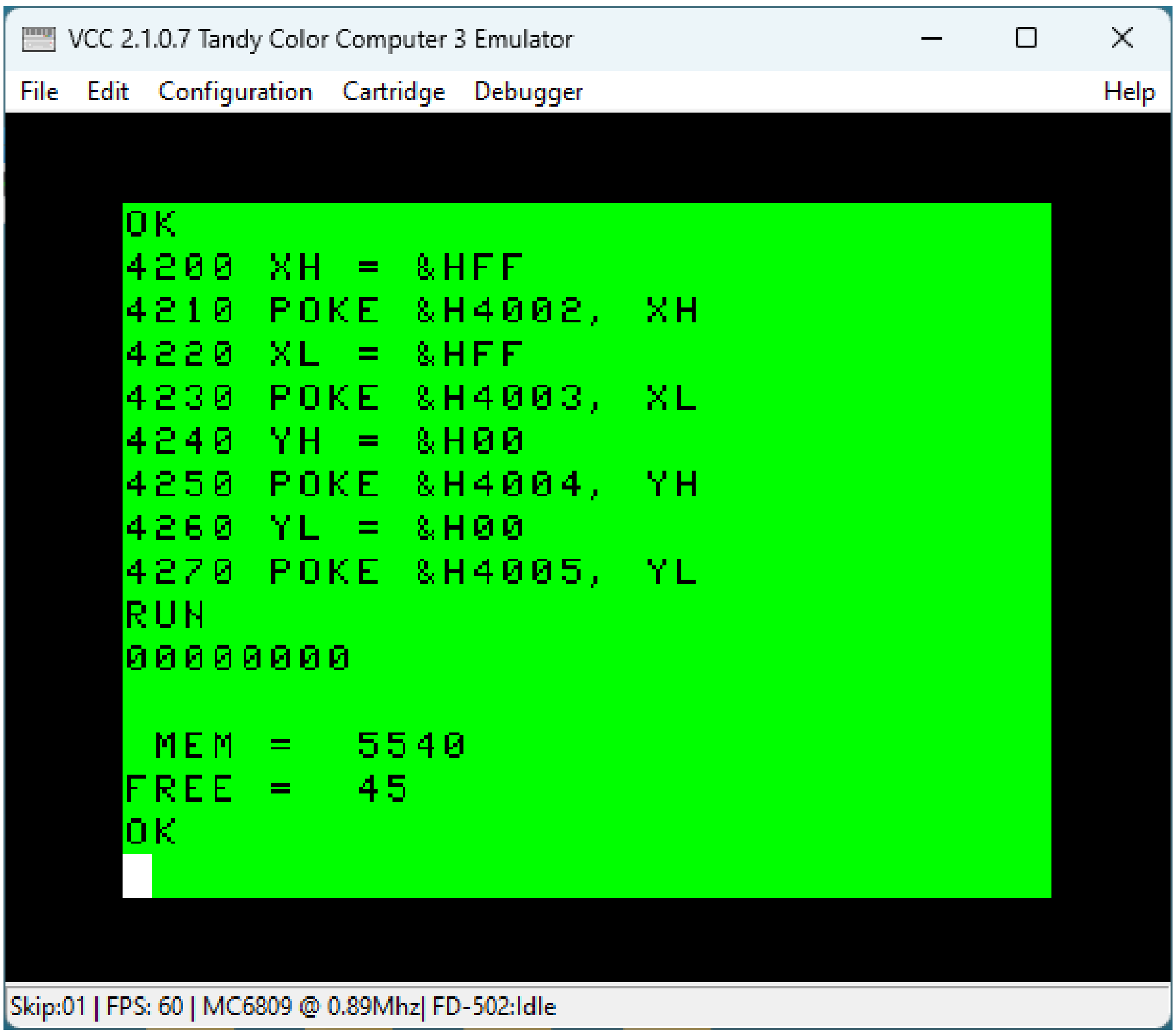
—

Second Run: - With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &H00
4250 POKE &H4004, YH
4260 YL = &H00
4270 POKE &H4005, YL
```

—

Result: $FFFF x 0 = 0



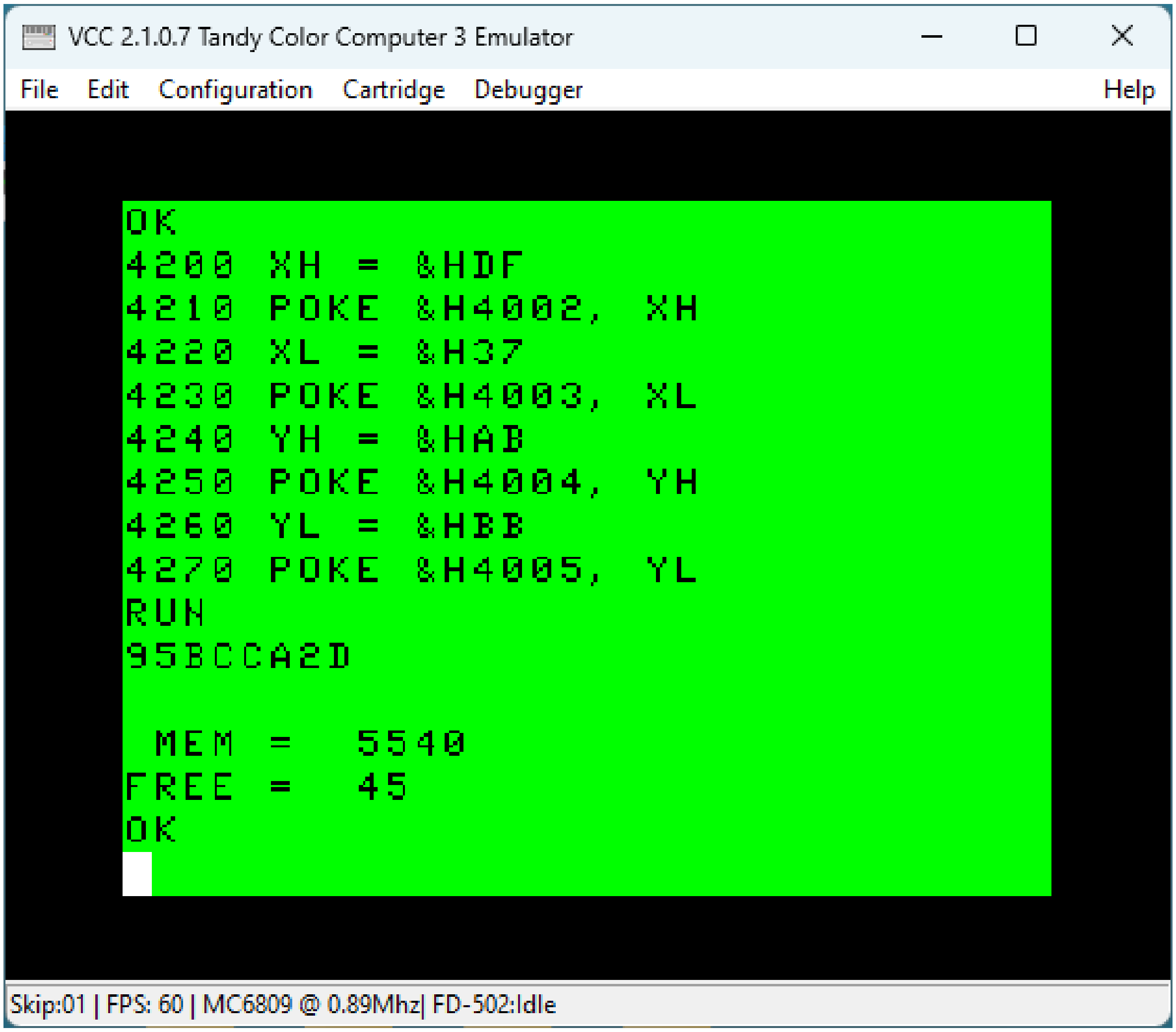
As expected.

Third Run: - With Test Data:

```
4200 XH = &HDF
4210 POKE &H4002, XH
4220 XL = &H37
4230 POKE &H4003, XL
4240 YH = &HAB
4250 POKE &H4004, YH
4260 YL = &HBB
4270 POKE &H4005, YL
```

—

Result: $DF37 x $ABBB = $95BCCA2D



As expected.

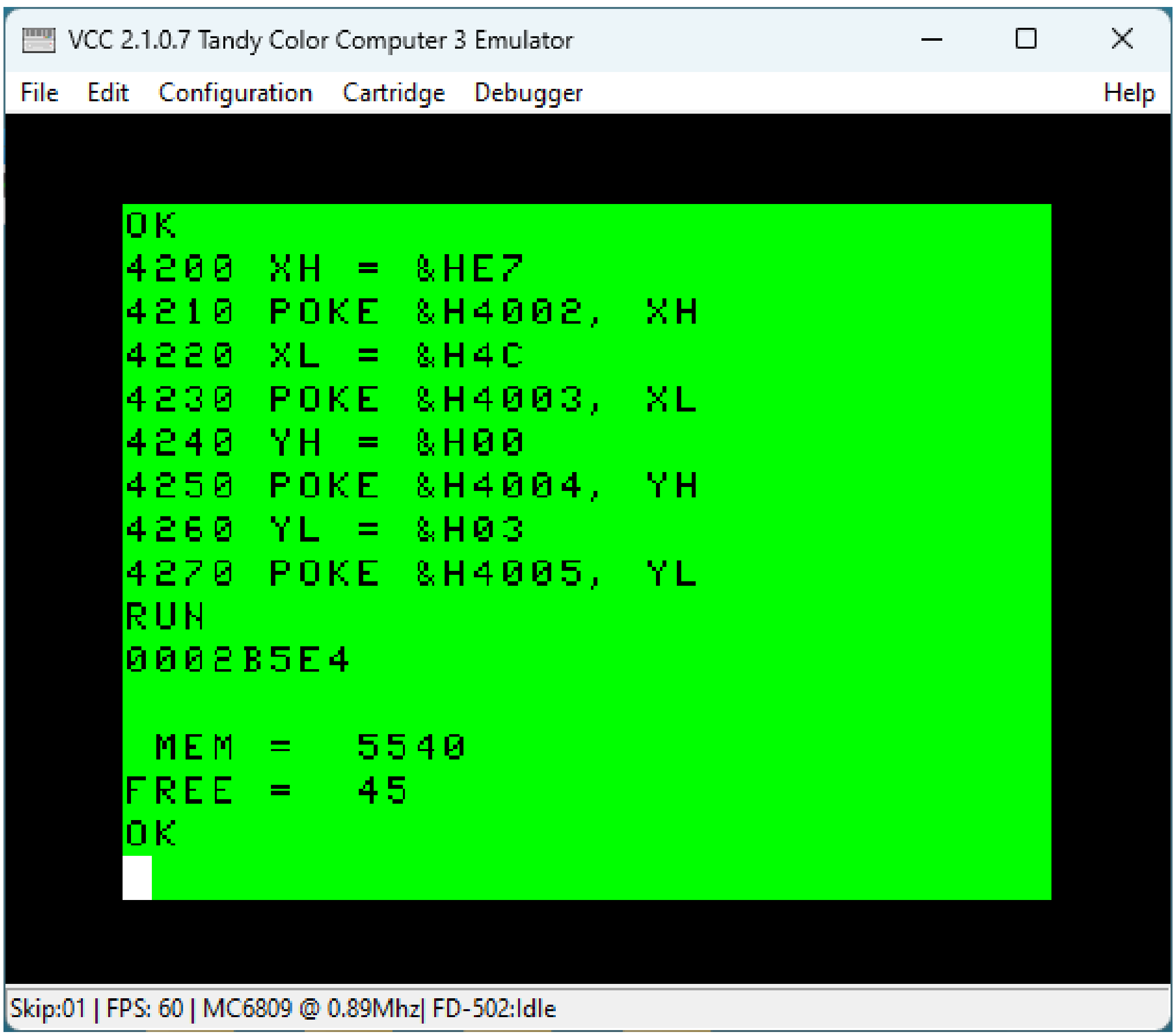
—

Fourth Run: - With Test Data:

```
4200 XH = &HE7
4210 POKE &H4002, XH
4220 XL = &H4C
4230 POKE &H4003, XL
4240 YH = &H00
4250 POKE &H4004, YH
4260 YL = &H03
4270 POKE &H4005, YL
```

—

Result: $E74C x 3 = $2B5E4



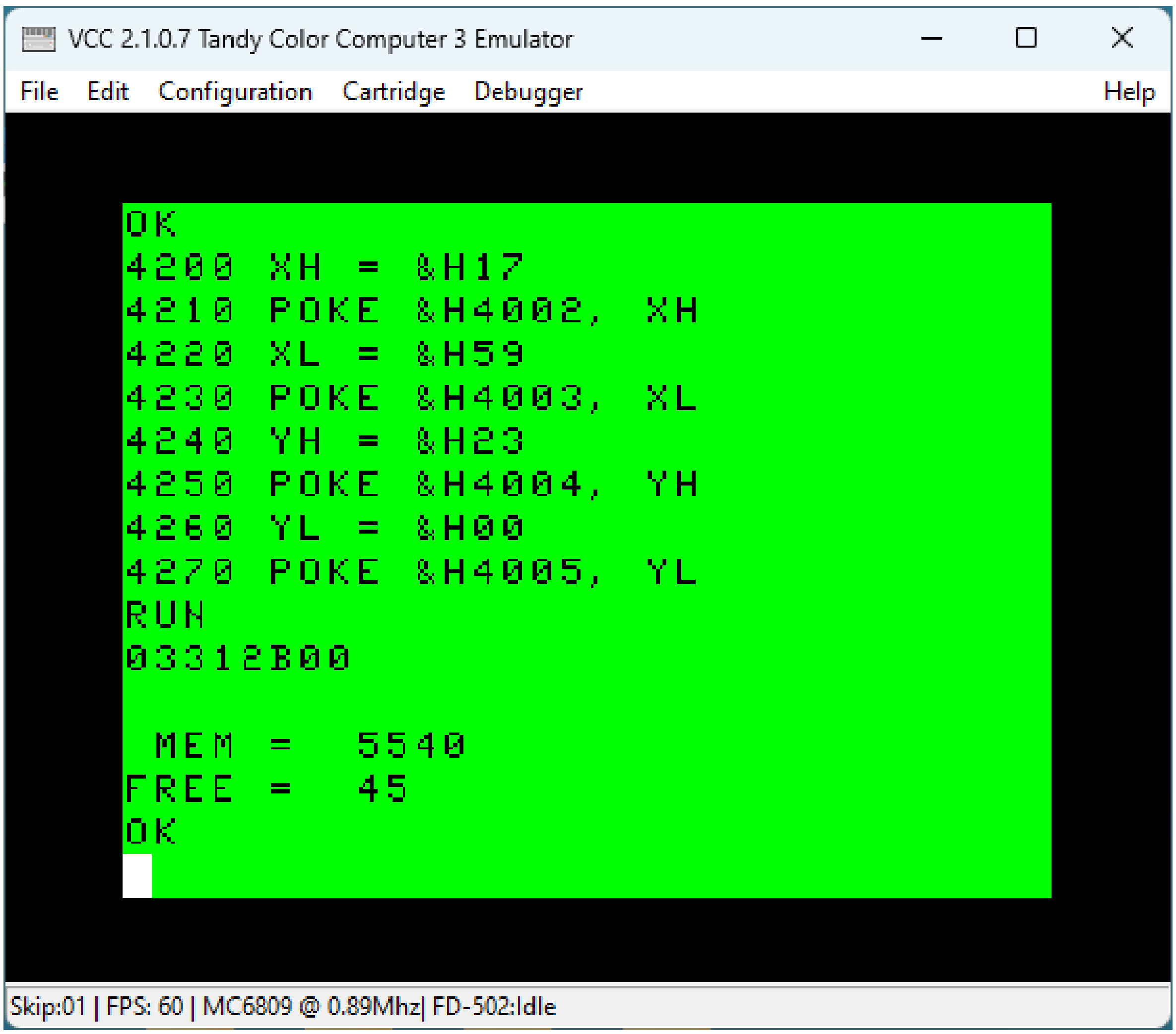
As expected.

Fifth Run: - With Test Data:

```
4200 XH = &H17
4210 POKE &H4002, XH
4220 XL = &H59
4230 POKE &H4003, XL
4240 YH = &H23
4250 POKE &H4004, YH
4260 YL = &H00
4270 POKE &H4005, YL
```

—

Result: $1759 x $2300 = $3312B00



As expected.

=====

DU0808: 8-bit by 8-bit Unsigned Divide

This divides an 8-bit unsigned byte by another 8-bit unsigned byte and returns an 8-bit unsigned quotient and an 8-bit unsigned remainder.

```
00100 *****
00110 *
00120 * DU0808.ASM
00130 * MDJ 2023/02/06
00140 *
00150 * 8-BIT BY 8-BIT
00160 * UNSIGNED DIVIDE
00170 *
00180 * ENTRY CONDITIONS:
00190 * A = DIVIDEND
00200 * B = DIVISOR
00210 *
00220 * EXIT CONDITIONS:
00230 * A = QUOTIENT
00240 * B = REMAINDER
00250 *
00260 * ON EXIT, IF
00270 * DIVIDE BY ZERO
00280 * ERROR:
00290 * A = #$FF
00300 * B = #$FF
00310 * AN IMPOSSIBLE RESULT
00320 *
00330 *****
00340 *
00350 * THIS FUNCTION WAS
00360 * ADAPTED FROM:
00370 * BARROW, DAVID (1984).
00380 * 6809 MACHINE CODE
00390 * PROGRAMMING.
00400 * LONDON: GRANADA
00410 * TECHNICAL BOOKS,
00420 * PAGES 29-33.
00430 *
00440 * *** EXCEPT ***
00450 * BARROW'S SIXTH LINE
00460 * IN DIVAB ON PAGE 30:
00470 *   RORB
```

```
                   00480 * SHOULD BE:
                   00490 *   ROLB
                   00500 *
                   00510 *****
                   00520
42FE               00530        ORG     $42FE
                   00540
42FE 34   05       00550 DU0808 PSHS    B,CC
4300 C1   00       00560        CMPB    #0      IS IT DIVIDE-BY-
ZERO?
4302 27   1A       00570        BEQ     LBL003  GO IF YES (ERROR)
4304 C6   08       00580        LDB     #8      NUMBER OF 8-BIT
SHIFTS
4306 34   04       00590        PSHS    B       SAVE THE COUNT
4308 5F            00600        CLRB            CLEAR B/REMAINDER
                   00610
                   00620 * 8 SHIFTS - TRY TO SUBTRACT DIVISOR AT EACH
SHIFT,
                   00630 * FORMING QUOTIENT ONE BIT AT A TIME.
                   00640 * QUOTIENT SHIFTS IN AS DIVIDEND SHIFTS OUT.
4309 48            00650 LBL001 ASLA            SHIFT NEXT DIVIDEND
BIT THROUGH

430A 59            00660        ROLB            INTO REMAINDER (B),
CLEARING
                   00670 *                      NEXT QUOTIENT BIT
430B E1   62       00680        CMPB    2,S     CAN DIVISOR BE
SUBTRACTED?
430D 25   03       00690        BLO     LBL002  GO IF NO
430F E0   62       00700        SUBB    2,S     SUBTRACT AND SET
QUOTIENT BIT
4311 4C            00710        INCA            AT CORRESPONDING BIT
LOCATION
4312 6A   E4       00720 LBL002 DEC     ,S      REPEAT TIL ENTIRE
DIVIDEND SHIF
TED
4314 26   F3       00730        BNE     LBL001  (B IS NOW THE
REMAINDER, AND)
                   00740 *                      (A IS NOW THE
QUOTIENT)
                   00750
                   00760 * PUT REMAINDER INTO STACKED B.
                   00770 * CLEAR COUNT BYTE OFF STACK.
                   00780 * PULL RESULTS AND CLEAN THE STACK
4316 E7   62       00790        STB     2,S     REMAINDER TO STACKED
B
```

```
4318 32   61        00800         LEAS    1,S     REMOVE COUNT FROM
STACK
431A 35   05        00810         PULS    B,CC
431C 20   06        00820         BRA     LBL004  GO TO EXIT
                    00830
                    00840 * ERROR HANDLING
431E 35   05        00850 LBL003  PULS    B,CC
4320 86   FF        00860         LDA     #$FF    SET ERROR CODING
4322 C6   FF        00870         LDB     #$FF
                    00880
                    00890 * EXIT
4324 39             00900 LBL004  RTS
                    00910
          0000      00920         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0029.ASM
                    00130 * MDJ 2023/02/17
                    00140 *
                    00150 * DU0808 TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * BASIC/ML TRANSFER
                    00200 * VARIABLES
          4000      00210 REGA    EQU     $4000
          4001      00220 REGB    EQU     $4001
                    00230
                    00240 * ML FOUNDATION
                    00250 * CORE ADDRESSES
          409E      00260 PUTBYA  EQU     $409E
          40D7      00270 CRLF    EQU     $40D7
          41DB      00280 PRTCHA  EQU     $41DB
          42FE      00290 DU0808  EQU     $42FE
                    00300
7000                00310         ORG     $7000
                    00320
                    00330 * DU0808 TEST
                    00340
7000 34   06        00350         PSHS    A,B
                    00360
7002 B6   4000      00370         LDA     REGA    GET THE TEST VALUES
```

```
7005 F6   4001      00380          LDB     REGB    FROM THE XFER
VARIABLES
                    00390
7008 BD   409E      00400          JSR     PUTBYA  A
700B 86   2F        00410          LDA     #$2F    /
700D BD   41DB      00420          JSR     PRTCHA
7010 1F   98        00430          TFR     B,A
7012 BD   409E      00440          JSR     PUTBYA  B
7015 86   20        00450          LDA     #$20    SPACE
7017 BD   41DB      00460          JSR     PRTCHA
701A 86   3D        00470          LDA     #$3D    =
701C BD   41DB      00480          JSR     PRTCHA
701F 86   20        00490          LDA     #$20    SPACE
7021 BD   41DB      00500          JSR     PRTCHA
7024 BD   40D7      00510          JSR     CRLF
                    00520
7027 B6   4000      00530          LDA     REGA    GET THE TEST VALUES
702A F6   4001      00540          LDB     REGB    AGAIN
                    00550
702D BD   42FE      00560          JSR     DU0808  GO DO THE DIVIDE
                    00570
7030 BD   409E      00580          JSR     PUTBYA  QUOTIENT
7033 86   20        00590          LDA     #$20    SPACE
7035 BD   41DB      00600          JSR     PRTCHA
7038 1F   98        00610          TFR     B,A
703A BD   409E      00620          JSR     PUTBYA  REMAINDER
703D BD   40D7      00630          JSR     CRLF
                    00640
                    00650 * EXIT
7040 35   06        00660          PULS    A,B
7042 39             00670          RTS
                    00680
          0000      00690          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0029.BAS
1030 '* MDJ 2023/02/17
1040 '*
1050 '* DU0808 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0029.BIN"
1330 '

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 '

4000 'LOAD THE TEST DATA
4200 A = &HFF
4210 POKE &H4000, A
4220 B = &HFF
4230 POKE &H4001, B

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
```

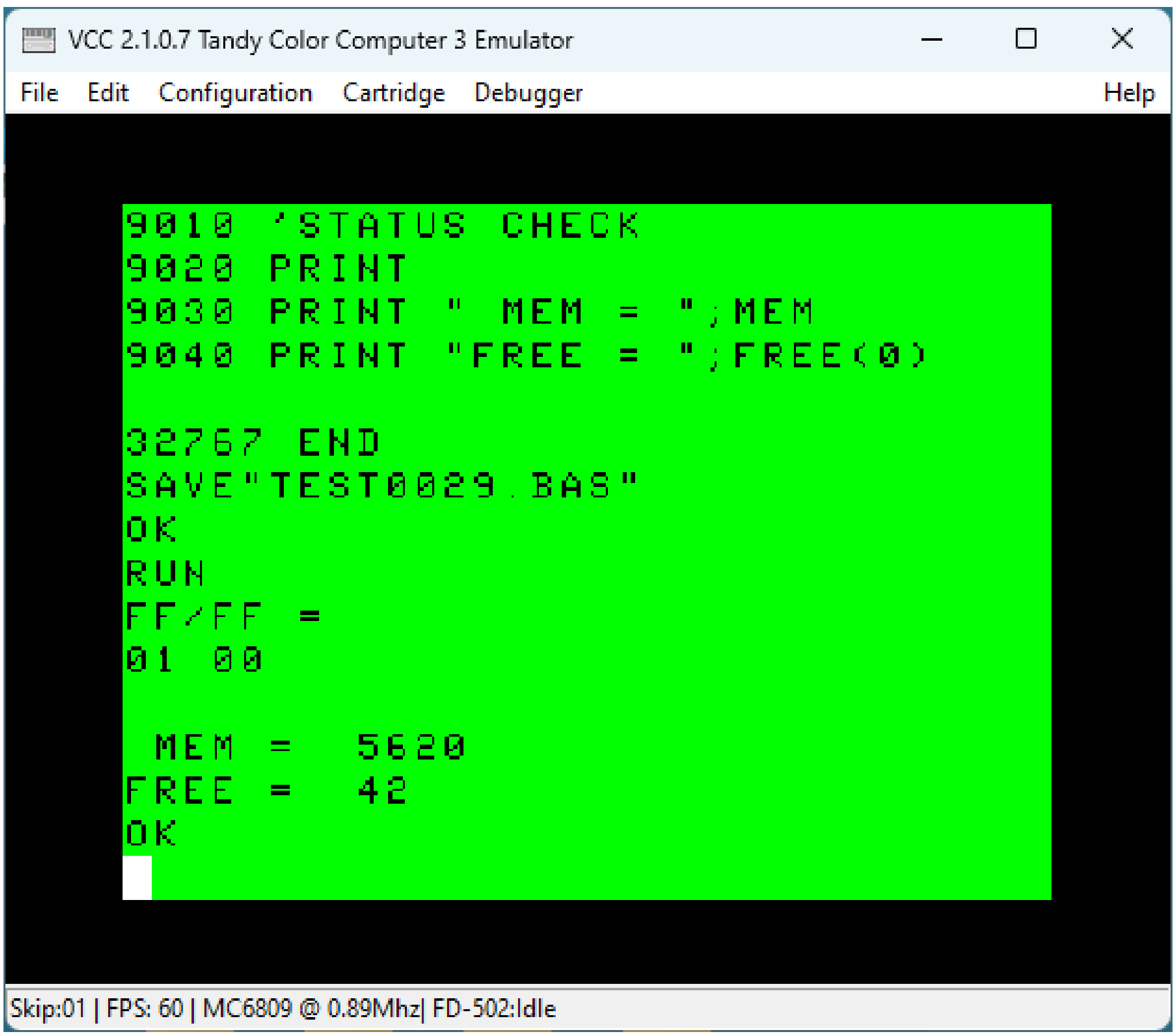
—

With Test Data:

```
4200 A = &HFF
4210 POKE &H4000, A
4220 B = &HFF
4230 POKE &H4001, B
```

—

Result: `$FF/$FF —> QUOTIENT  = 1`
`                REMAINDER = 0`



As expected.

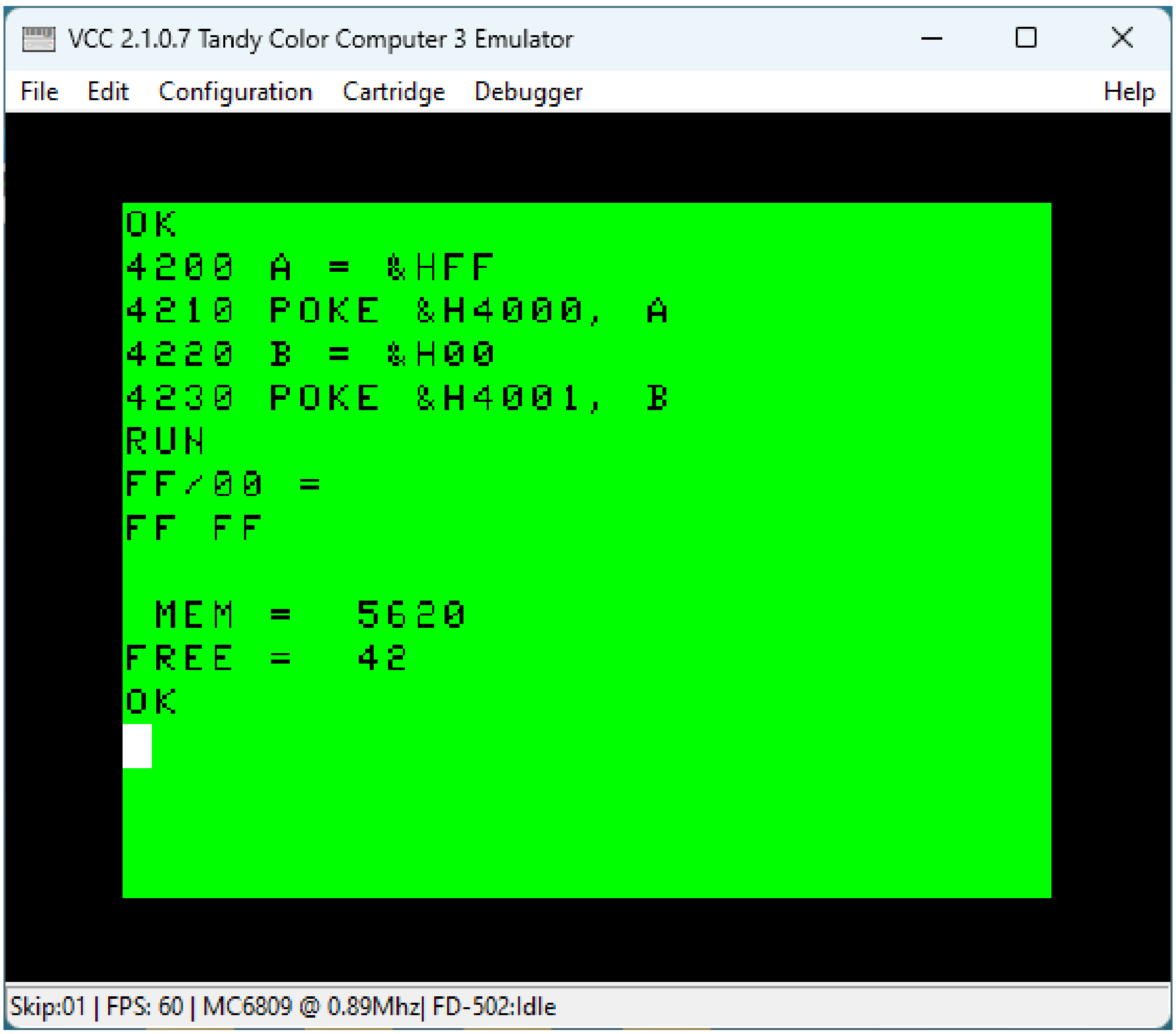
—

Second Run: - With Test Data:

```
4200 A = &HFF
4210 POKE &H4000, A
4220 B = &H00
4230 POKE &H4001, B
```

—

Result: `$FF/$00 —> QUOTIENT  = $FF`
`REMAINDER = $FF`
`INDICATING DIVIDE BY ZERO ERROR`



As expected.

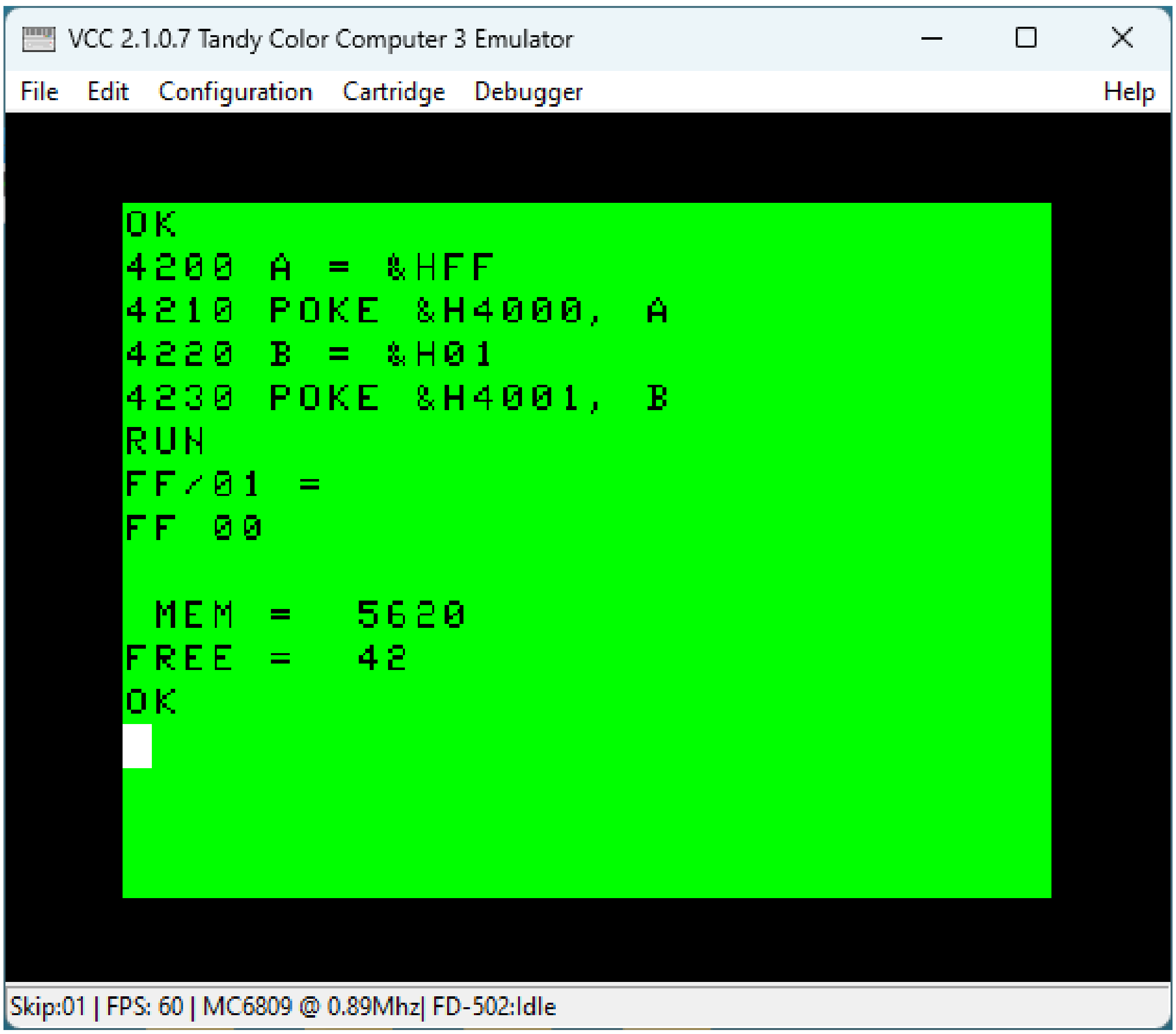
—

Third Run: - With Test Data:

```
4200 A = &HFF
4210 POKE &H4000, A
4220 B = &H01
4230 POKE &H4001, B
```

—

Result: `$FF/$01 —> QUOTIENT  = $FF`
`                  REMAINDER = 0`



As expected.

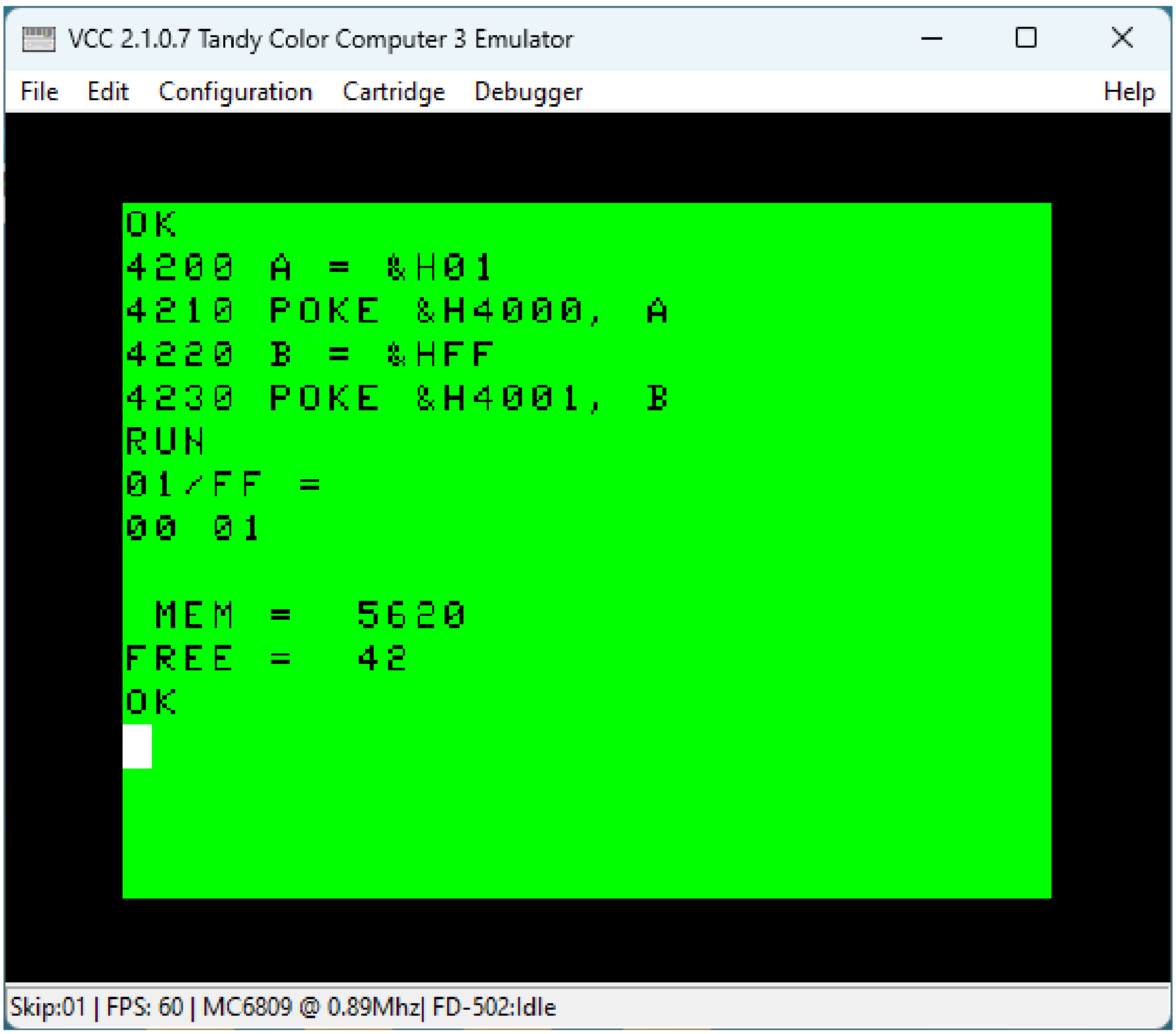
—

Fourth Run: - With Test Data:

```
4200 A = &H01
4210 POKE &H4000, A
4220 B = &HFF
4230 POKE &H4001, B
```

—

Result: `$01/$FF -> QUOTIENT  = 0`
`                REMAINDER = 1`



As expected.

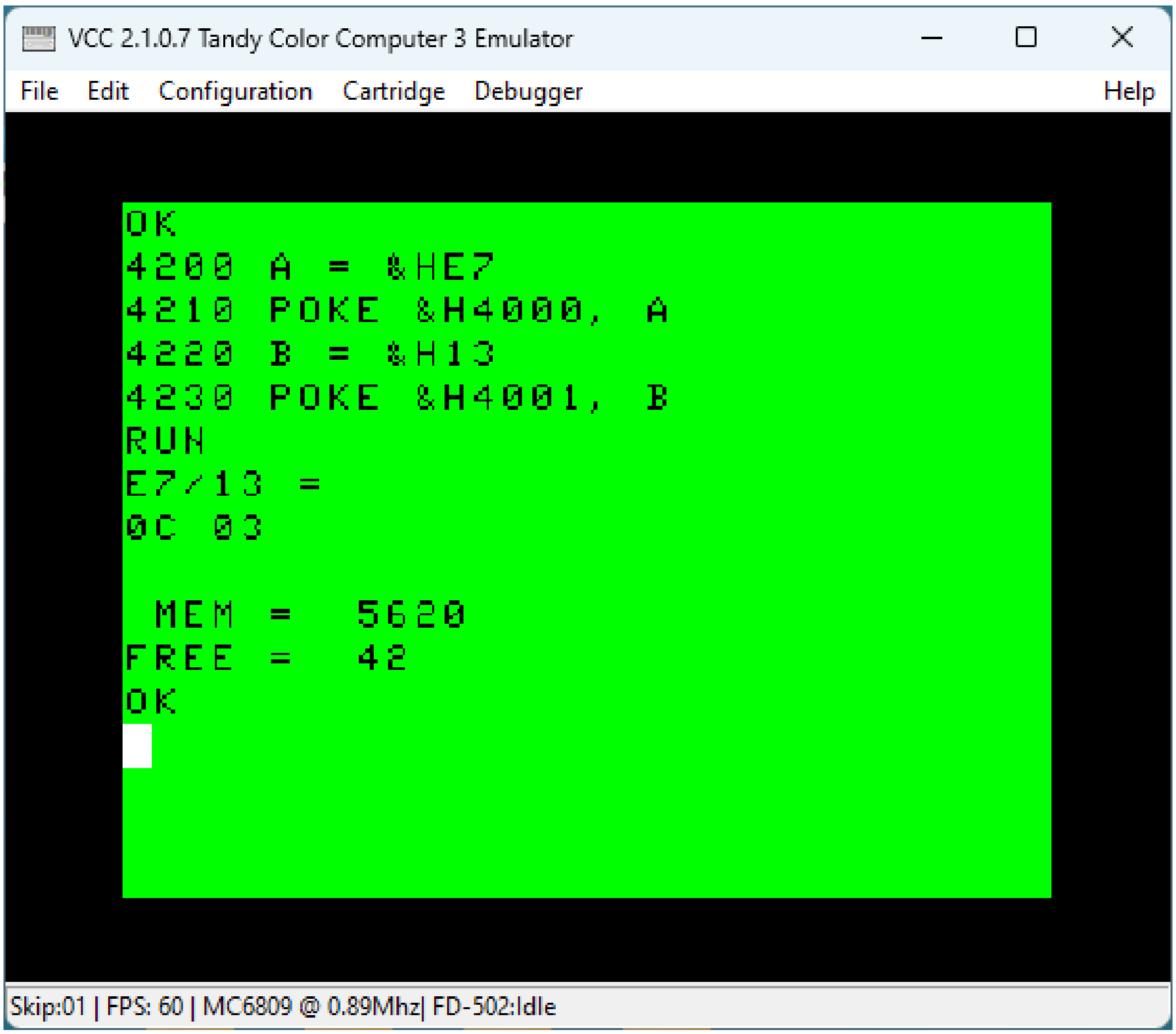
—

Fifth Run: - With Test Data:

```
4200 A = &HE7
4210 POKE &H4000, A
4220 B = &H13
4230 POKE &H4001, B
```

—

Result: `$E7/$13 —> QUOTIENT  = $0C`
`REMAINDER = $03`



As expected.

=====

DU1616: 16-bit by 16-bit Unsigned Divide

This divides a 16-bit unsigned word by another 16-bit unsigned word and returns a 16-bit unsigned quotient and a 16-bit unsigned remainder.

```
*****
*
* DU1616.ASM
* MDJ 2023/02/02
*
* 16-BIT BY 16-BIT
* UNSIGNED DIVIDE
*
* ENTRY CONDITIONS:
* X = DIVIDEND
* Y = DIVISOR
*
* EXIT CONDITIONS:
* X = QUOTIENT
* Y = REMAINDER
*
* ON EXIT, IF
* DIVIDE BY ZERO
* ERROR:
* X = #$FFFF
* Y = #$FFFF
* AN IMPOSSIBLE RESULT
*
*****
*
* THIS FUNCTION WAS
* ADAPTED FROM:
* BARROW, DAVID (1984).
* 6809 MACHINE CODE
* PROGRAMMING.
* LONDON: GRANADA
* TECHNICAL BOOKS,
* PAGES 29-33.
*
*****

4325                    ORG     $4325

```

```
4325 34   37        00480 DU1616  PSHS    A,B,X,Y,CC
4327 108C 0000      00490         CMPY    #0      IS IT DIVIDE-BY-
ZERO?
432B 27   21        00500         BEQ     LBL003  GO IF YES (ERROR)
432D C6   10        00510         LDB     #16     NUMBER OF 16-BIT
SHIFTS
432F 34   04        00520         PSHS    B       SAVE THE COUNT
4331 4F             00530         CLRA            CLEAR ACCUMULATOR D
4332 5F             00540         CLRB
                    00550
                    00560 * 16 SHIFTS - TRY TO SUBTRACT DIVISOR AT
EACH SHIFT,
                    00570 * FORMING QUOTIENT ONE BIT AT A TIME.
                    00580 * QUOTIENT SHIFTS IN AS DIVIDEND SHIFTS OUT.
4333 68   65        00590 LBL001  ASL     5,S     SHIFT NEXT DIVIDEND
BIT THROUGH

4335 69   64        00600         ROL     4,S     INTO REMAINDER (D),
CLEARING
4337 59             00610         ROLB            NEXT QUOTIENT BIT AT
BIT 0 5,S
4338 49             00620         ROLA
4339 10A3 66        00630         CMPD    6,S     CAN DIVISOR BE
SUBTRACTED?
433C 25   04        00640         BLO     LBL002  GO IF NO
433E A3   66        00650         SUBD    6,S     SUBTRACT AND SET
QUOTIENT BIT
4340 6C   65        00660         INC     5,S     AT CORRESPONDING BIT
LOCATION
4342 6A   E4        00670 LBL002  DEC     ,S      REPEAT TIL ENTIRE
DIVIDEND SHIF
TED
4344 26   ED        00680         BNE     LBL001  (D IS NOW THE
REMAINDER)
                    00690
                    00700 * PUT REMAINDER INTO STACKED Y.
                    00710 * CLEAR COUNT BYTE OFF STACK.
                    00720 * PULL RESULTS (X AND Y) AND CLEAN THE STACK
4346 ED   66        00730         STD     6,S     REMAINDER TO STACKED
Y
4348 32   61        00740         LEAS    1,S     REMOVE COUNT FROM
STACK
434A 35   37        00750         PULS    A,B,X,Y,CC
434C 20   09        00760         BRA     LBL004  GO TO EXIT
                    00770
                    00780 * ERROR HANDLING
434E 35   37        00790 LBL003  PULS    A,B,X,Y,CC
```

```
4350 8E   FFFF     00800        LDX     #$FFFF  SET ERROR CODING
4353 108E FFFF     00810        LDY     #$FFFF
                   00820
                   00830 * EXIT
4357 39            00840 LBL004  RTS
                   00850
          0000     00860        END
```

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0030.ASM
                   00130 * MDJ 2023/02/17
                   00140 *
                   00150 * DU1616 TEST
                   00160 *
                   00170 *****
                   00180
                   00190 * BASIC/ML TRANSFER
                   00200 * VARIABLES
          4002     00210 REGX    EQU     $4002
          4004     00220 REGY    EQU     $4004
                   00230
                   00240 * ML FOUNDATION
                   00250 * CORE ADDRESSES
          40D7     00260 CRLF    EQU     $40D7
          4178     00270 PUTWRA  EQU     $4178
          41DB     00280 PRTCHA  EQU     $41DB
          4325     00290 DU1616  EQU     $4325
                   00300
7000               00310         ORG     $7000
                   00320
                   00330 * DU1616 TEST
                   00340
7000 34   36       00350         PSHS    A,B,X,Y
                   00360
7002 BE   4002     00370         LDX     REGX    GET THE TEST VALUES
7005 10BE 4004     00380         LDY     REGY    FROM THE XFER
VARIABLES
                   00390
7009 1F   10       00400         TFR     X,D
700B BD   4178     00410         JSR     PUTWRA  X
700E 86   2F       00420         LDA     #$2F    /
7010 BD   41DB     00430         JSR     PRTCHA
```

```
7013 1F   20      00440          TFR    Y,D
7015 BD   4178    00450          JSR    PUTWRA  Y
7018 86   20      00460          LDA    #$20    SPACE
701A BD   41DB    00470          JSR    PRTCHA
701D 86   3D      00480          LDA    #$3D    =
701F BD   41DB    00490          JSR    PRTCHA
7022 86   20      00500          LDA    #$20    SPACE
7024 BD   41DB    00510          JSR    PRTCHA
7027 BD   40D7    00520          JSR    CRLF
                  00530
702A BE   4002    00540          LDX    REGX    GET THE TEST VALUES
702D 10BE 4004    00550          LDY    REGY    AGAIN
                  00560
7031 BD   4325    00570          JSR    DU1616  GO DO THE DIVIDE
                  00580
7034 1F   10      00590          TFR    X,D
7036 BD   4178    00600          JSR    PUTWRA  QUOTIENT
7039 86   20      00610          LDA    #$20    SPACE
703B BD   41DB    00620          JSR    PRTCHA
703E 1F   20      00630          TFR    Y,D
7040 BD   4178    00640          JSR    PUTWRA  REMAINDER
                  00660
                  00670 * EXIT
7046 35   36      00680          PULS   A,B,X,Y
7048 39           00690          RTS
                  00700
          0000    00710          END
```

—

The BASIC Language Control Program:

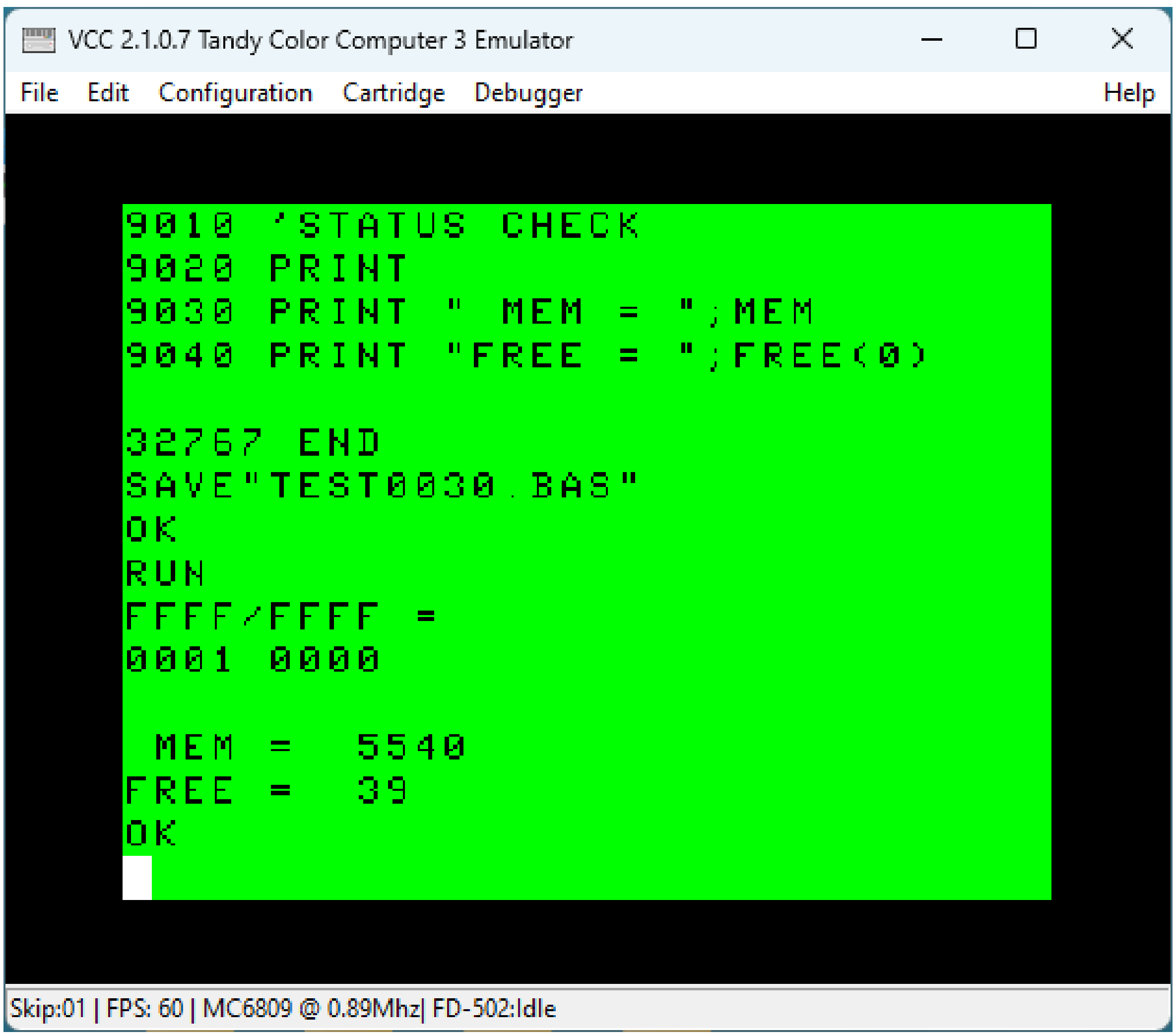
```
1000 '*****
1010 '*
1020 '* TEST0030.BAS
1030 '* MDJ 2023/02/17
1040 '*
1050 '* DU1616 TEST
1060 '*
1070 '*****
1080 '

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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0030.BIN"
1330 '

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 '

4000 'LOAD THE TEST DATA
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &HFF
4250 POKE &H4004, YH
4260 YL = &HFF
4270 POKE &H4005, YL

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
```

With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &HFF
4250 POKE &H4004, YH
4260 YL = &HFF
4270 POKE &H4005, YL
```

Result: `$FFFF/$FFFF –> QUOTIENT  = $0001`
`                         REMAINDER = $0000`



As expected.

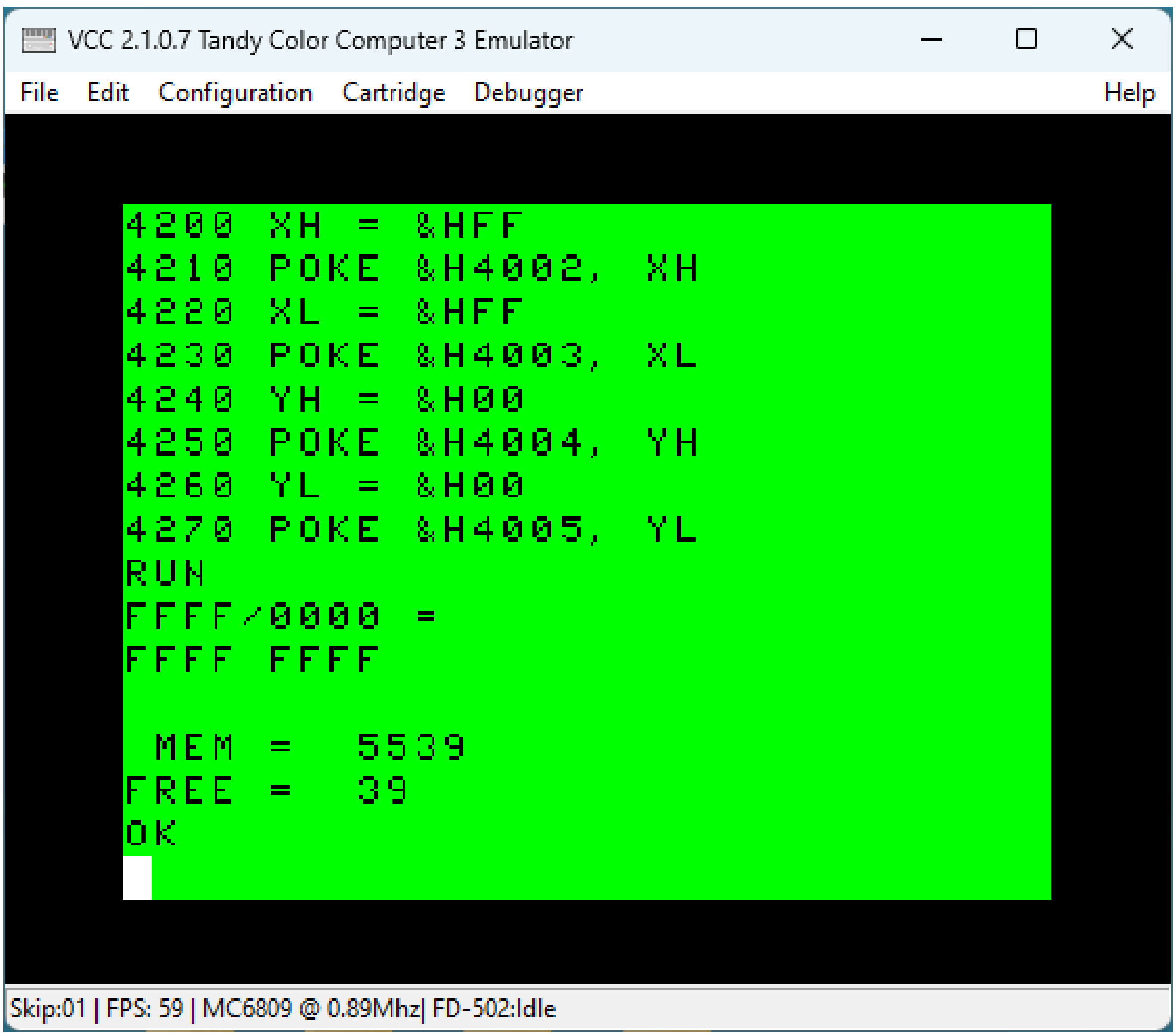
Second Run: - With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &H00
4250 POKE &H4004, YH
4260 YL = &H00
4270 POKE &H4005, YL
```

—

Result: `$FFFF/$0000 –> QUOTIENT  = $FFFF`

```
                 REMAINDER = $FFFF
                 INDICATING DIVIDE BY ZERO ERROR
```

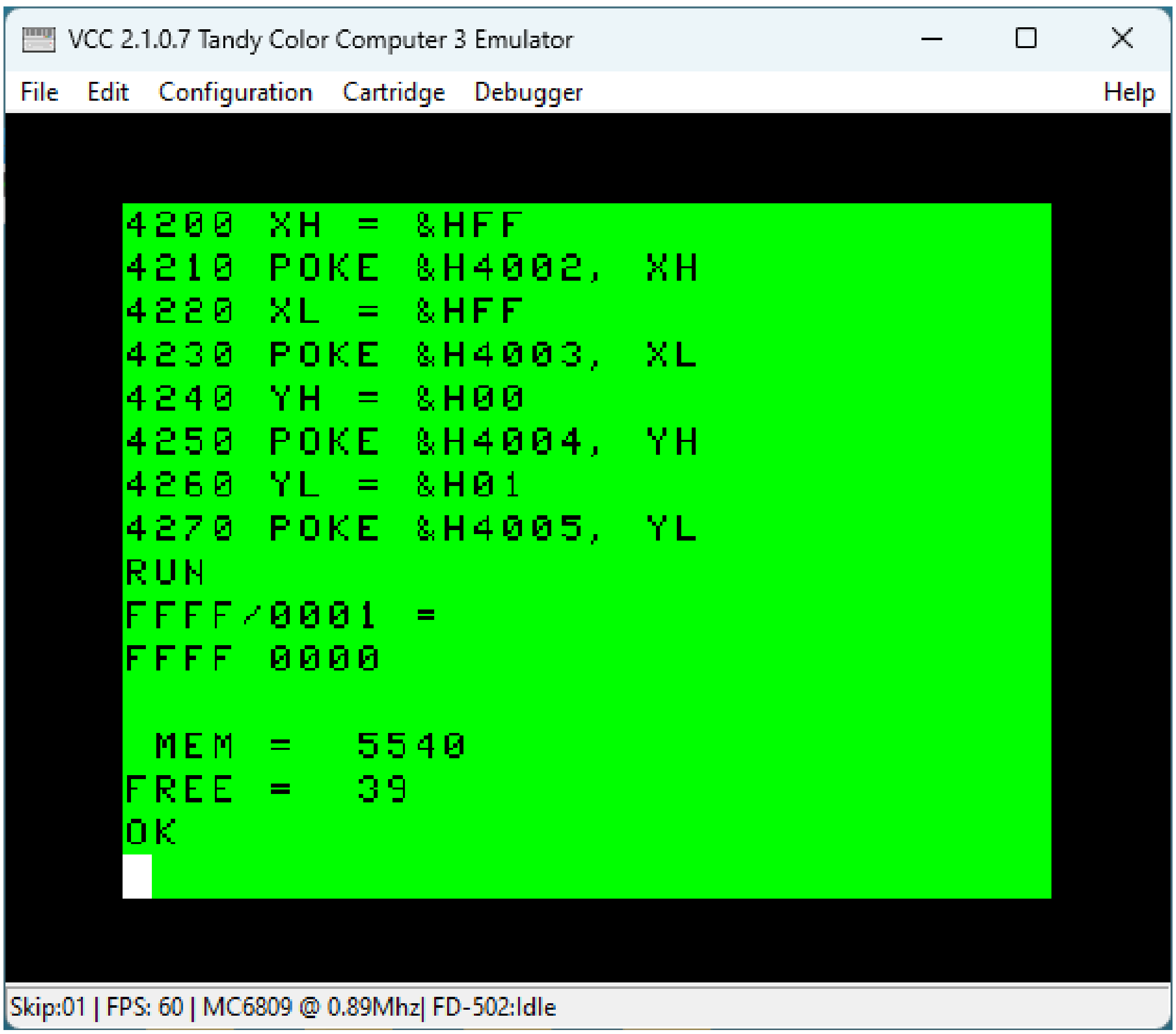


As expected.

Third Run: - With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 YH = &H00
4250 POKE &H4004, YH
4260 YL = &H01
4270 POKE &H4005, YL
```

Result `$FFFF/$0001 –> QUOTIENT  = $FFFF`
`REMAINDER = $0000:`

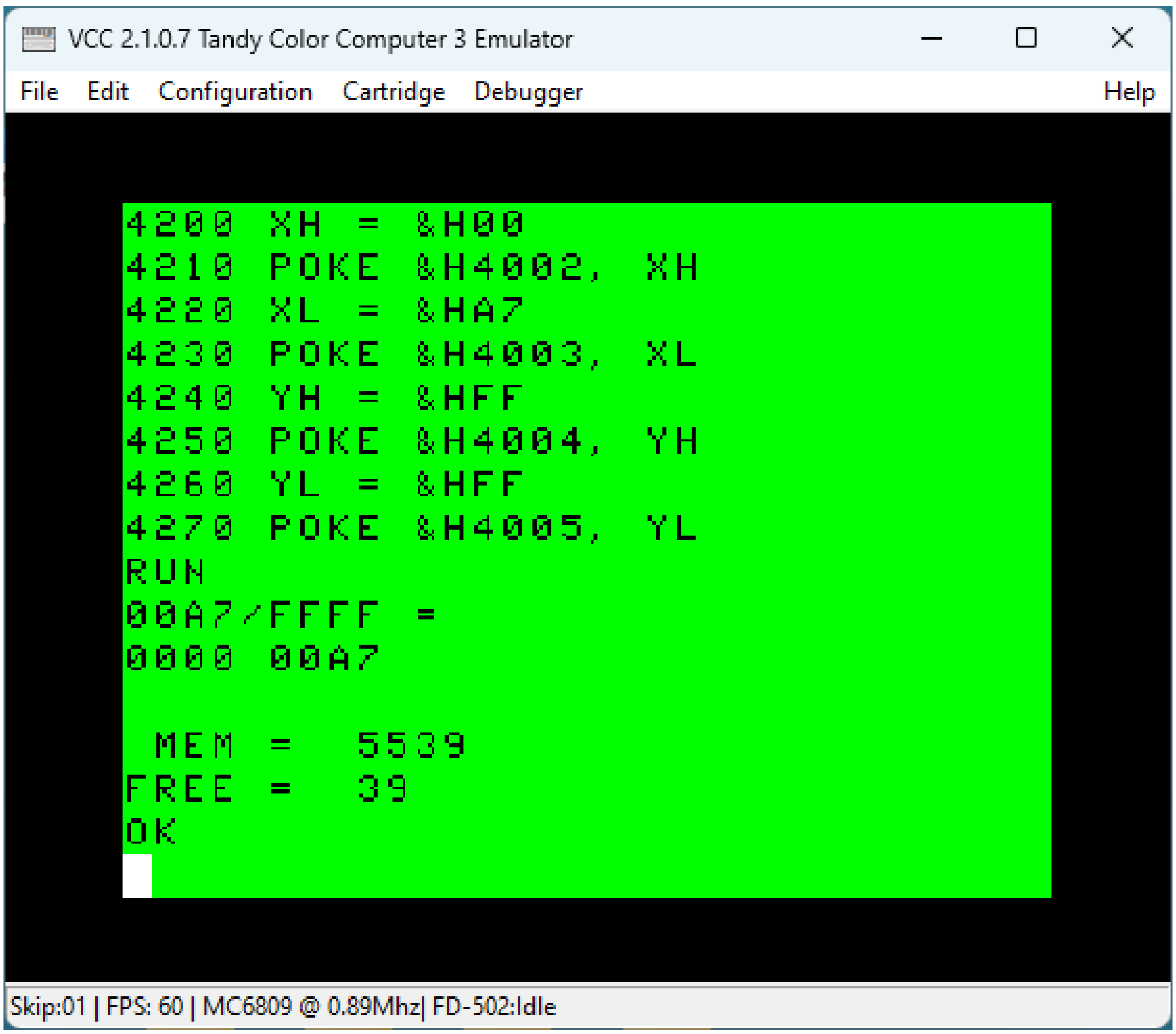


As expected.

Fourth Run: - With Test Data:

```
4200 XH = &H00
4210 POKE &H4002, XH
4220 XL = &HA7
4230 POKE &H4003, XL
4240 YH = &HFF
4250 POKE &H4004, YH
4260 YL = &HFF
4270 POKE &H4005, YL
```

Result: `$00A7/$FFFF —> QUOTIENT  = $0000`
`REMAINDER = $00A7`



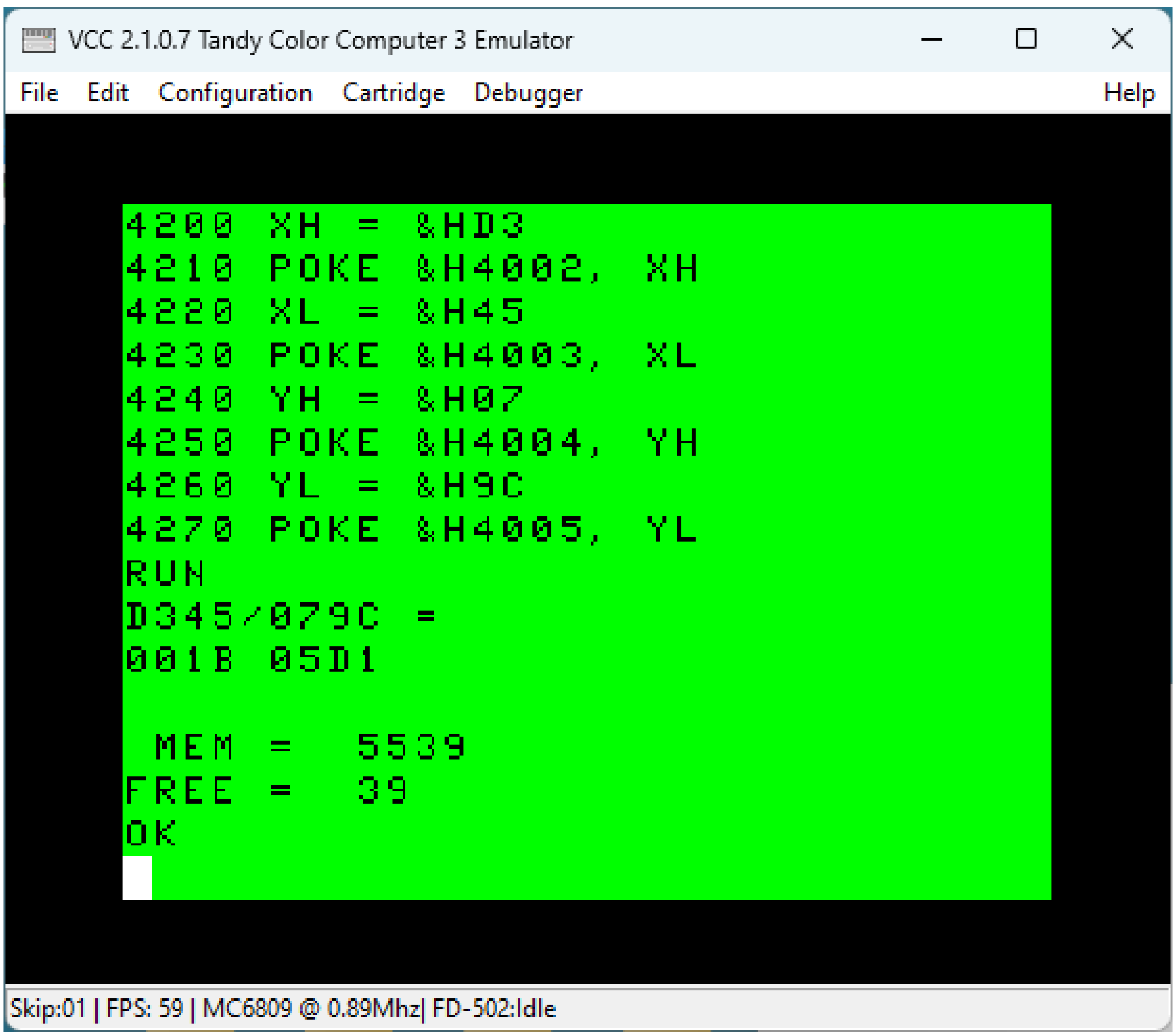
As expected.

Fifth Run: - With Test Data:

```
4200 XH = &HD3
4210 POKE &H4002, XH
4220 XL = &H45
4230 POKE &H4003, XL
4240 YH = &H07
4250 POKE &H4004, YH
4260 YL = &H9C
4270 POKE &H4005, YL
```

—

Result: `$D345/$079C –> QUOTIENT   = $001B`
`                         REMAINDER = $05D1`



As expected.

=====

DU1608: 16-bit by 8-bit Unsigned Divide

This divides a 16-bit unsigned word by an 8-bit unsigned byte and returns a 16-bit unsigned quotient and a 16-bit unsigned remainder.

This routine performs its task by simply extending the 8-bit divisor to 16-bits and then calling DU1616.

```
                  *****
                  *
                  * DU1608.ASM
                  * MDJ 2023/02/06
                  *
                  * 16-BIT BY 8-BIT
                  * UNSIGNED DIVIDE
                  *
                  * ENTRY CONDITIONS:
                  * X = 16-BIT DIVIDEND
                  * B = 8-BIT DIVISOR
                  *
                  * EXIT CONDITIONS:
                  * X = QUOTIENT
                  * Y = REMAINDER
                  *
                  * ON EXIT, IF
                  * DIVIDE BY ZERO
                  * ERROR:
                  * X = #$FFFF
                  * Y = #$FFFF
                  * AN IMPOSSIBLE RESULT
                  *
                  *****

                  * EXTERNAL ROUTINE
                  * ADDRESS
          4325    DU1616  EQU     $4325

4358                      ORG     $4358

4358 34   06      DU1608  PSHS    A,B

                  * MOVE THE 8-BIT DIVISOR TO Y
435A 86   00              LDA     #$00
```

```
435C 1F   02        00450         TFR     D,Y
                    00460
                    00470 * GO DO 16-BIT BY 16-BIT
                    00480 * UNSIGNED DIVIDE
435E BD   4325      00490         JSR     DU1616
                    00500
4361 35   06        00510         PULS    A,B
4363 39             00520         RTS
                    00530
          0000      00540         END
```

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0031.ASM
                    00130 * MDJ 2023/02/17
                    00140 *
                    00150 * DU1608 TEST
                    00160 *
                    00170 *****
                    00180
                    00190 * BASIC/ML TRANSFER
                    00200 * VARIABLES
          4001      00210 REGB    EQU     $4001
          4002      00220 REGX    EQU     $4002
                    00230
                    00240 * ML FOUNDATION
                    00250 * CORE ADDRESSES
          409E      00260 PUTBYA  EQU     $409E
          40D7      00270 CRLF    EQU     $40D7
          4178      00280 PUTWRA  EQU     $4178
          41DB      00290 PRTCHA  EQU     $41DB
          4358      00300 DU1608  EQU     $4358
                    00310
7000                00320         ORG     $7000
                    00330
                    00340 * DU1608 TEST
                    00350
7000 34   36        00360         PSHS    A,B,X,Y
                    00370
7002 BE   4002      00380         LDX     REGX    GET THE TEST VALUES
7005 F6   4001      00390         LDB     REGB    FROM THE XFER
VARIABLES
                    00400
```

```
7008 1F   10        00410          TFR      X,D
700A BD   4178      00420          JSR      PUTWRA  X
700D 86   2F        00430          LDA      #$2F    /
700F BD   41DB      00440          JSR      PRTCHA
7012 F6   4001      00450          LDB      REGB
7015 1F   98        00460          TFR      B,A
7017 BD   409E      00470          JSR      PUTBYA  B
701A 86   20        00480          LDA      #$20    SPACE
701C BD   41DB      00490          JSR      PRTCHA
701F 86   3D        00500          LDA      #$3D    =
7021 BD   41DB      00510          JSR      PRTCHA
7024 86   20        00520          LDA      #$20    SPACE
7026 BD   41DB      00530          JSR      PRTCHA
7029 BD   40D7      00540          JSR      CRLF
                    00550
702C BE   4002      00560          LDX      REGX    GET THE TEST VALUES
702F F6   4001      00570          LDB      REGB    AGAIN
                    00580
7032 BD   4358      00590          JSR      DU1608  GO DO THE DIVIDE
                    00600
7035 1F   10        00610          TFR      X,D
7037 BD   4178      00620          JSR      PUTWRA  QUOTIENT
703A 86   20        00630          LDA      #$20    SPACE
703C BD   41DB      00640          JSR      PRTCHA
703F 1F   20        00650          TFR      Y,D
7041 BD   4178      00660          JSR      PUTWRA  REMAINDER
7044 BD   40D7      00670          JSR      CRLF
                    00680
                    00690 * EXIT
7047 35   36        00700          PULS     A,B,X,Y
7049 39             00710          RTS
                    00720
          0000      00730          END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0031.BAS
1030 '* MDJ 2023/02/17
1040 '*
1050 '* DU1608 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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0031.BIN"
1330 '

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 '

4000 'LOAD THE TEST DATA
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B

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

9000 'MEMORY AND DISK
9010 'STATUS CHECK
9020 PRINT
```

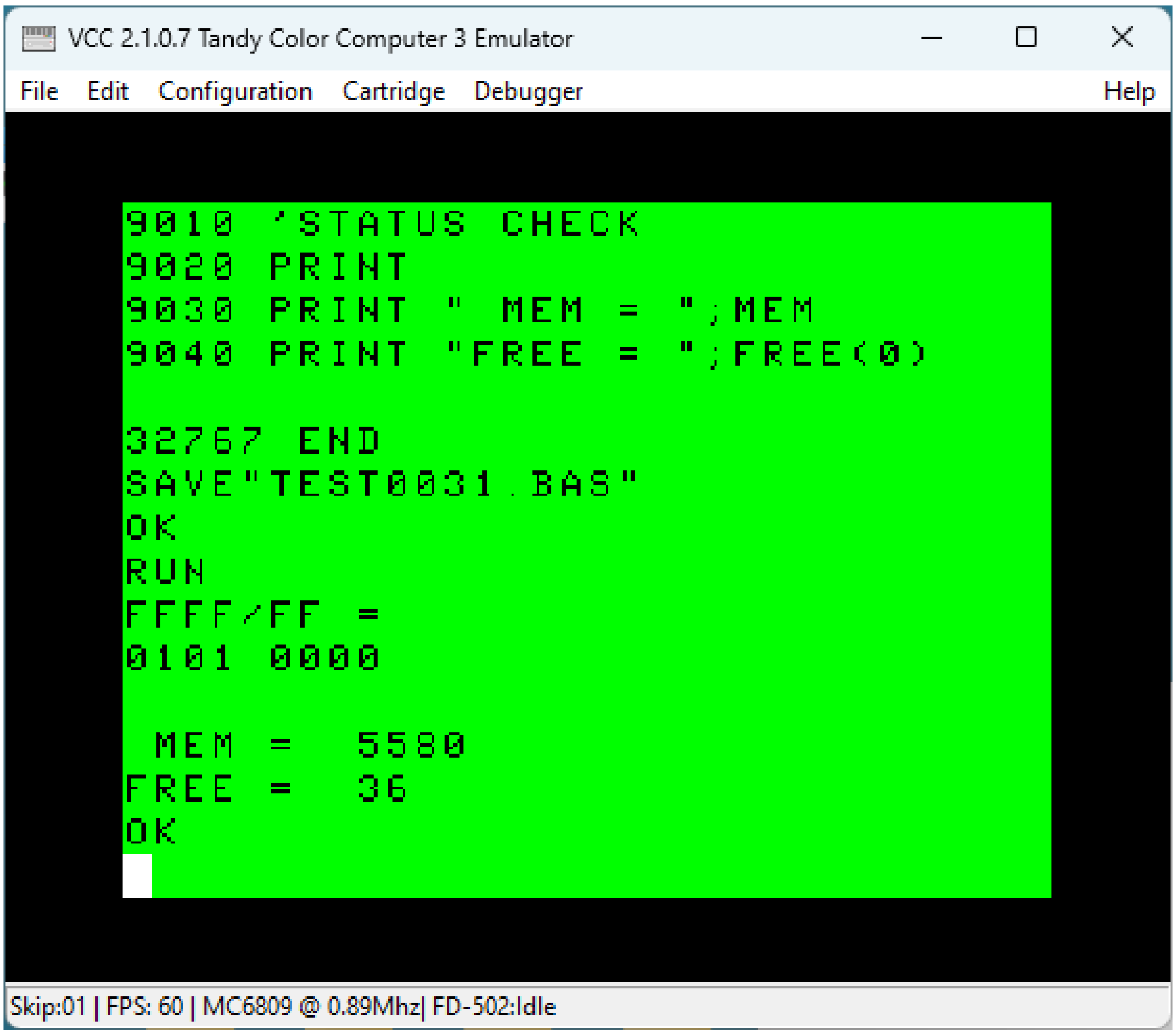
```
9030 PRINT " MEM = ";MEM
9040 PRINT "FREE = ";FREE(0)

32767 END
```

With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B
```

—

Result: `$FFFF/$FF –> QUOTIENT  = $0101`
`                        REMAINDER = $0000`



As expected.

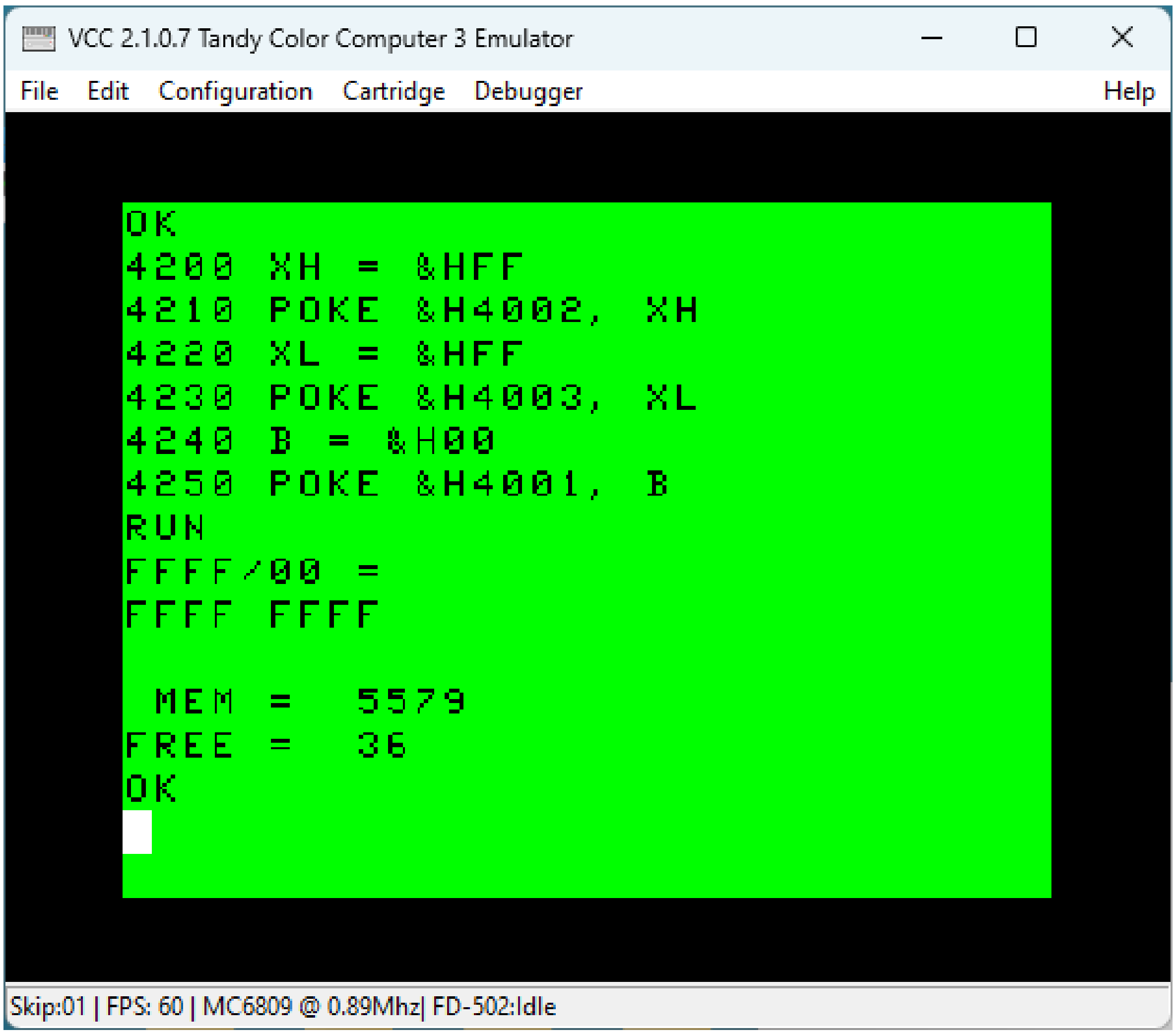
—

Second Run: - With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &H00
4250 POKE &H4001, B
```

—

Result: `$FFFF/$00 -> QUOTIENT  = $FFFF`
```
                   REMAINDER = $FFFF
                   INDICATING DIVIDE BY ZERO ERROR
```



As expected.

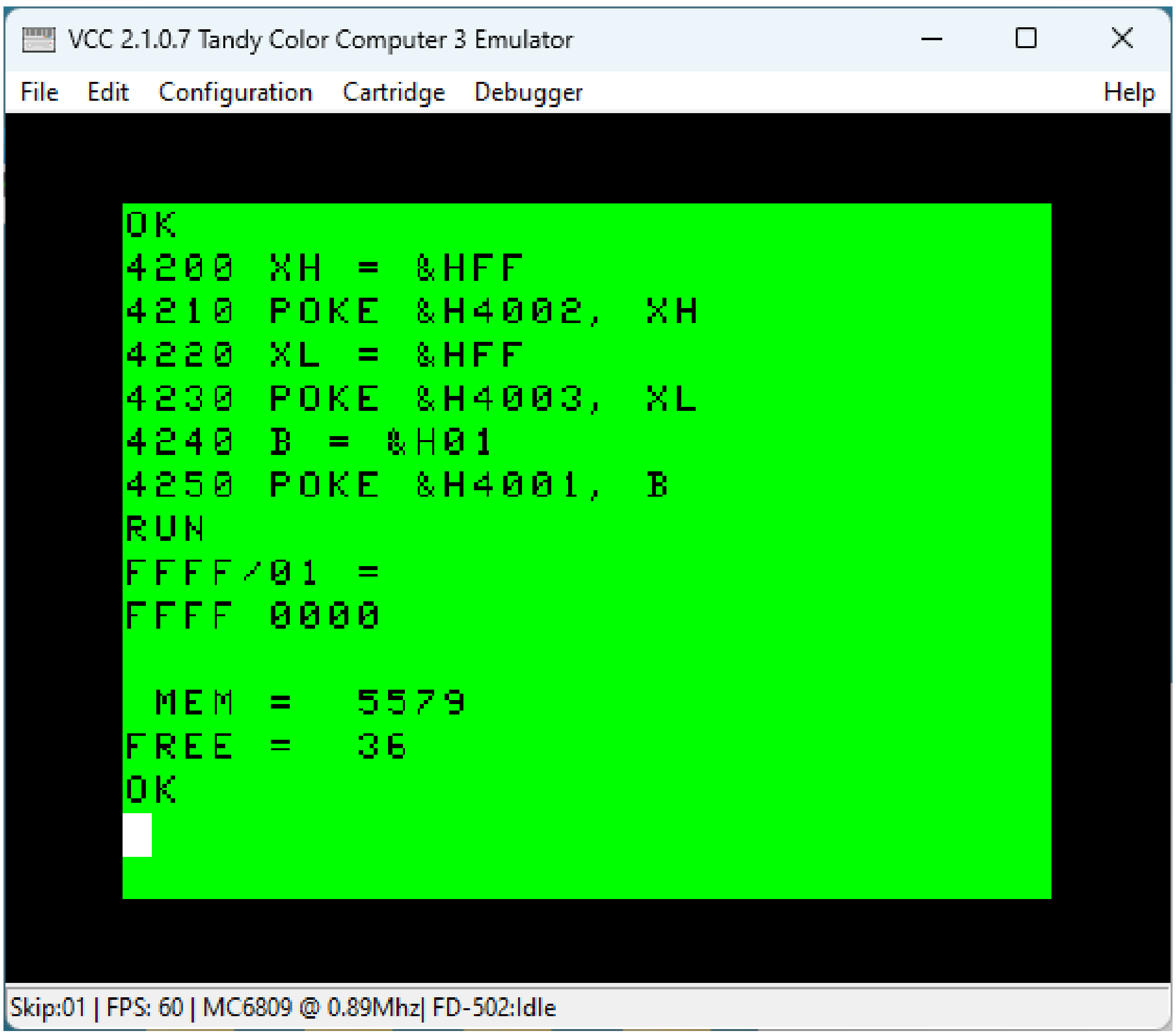
—

Third Run: - With Test Data:

```
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &H01
4250 POKE &H4001, B
```

—

Result: `$FFFF/$01 —> QUOTIENT  = $FFFF`
`                 REMAINDER = $0000`



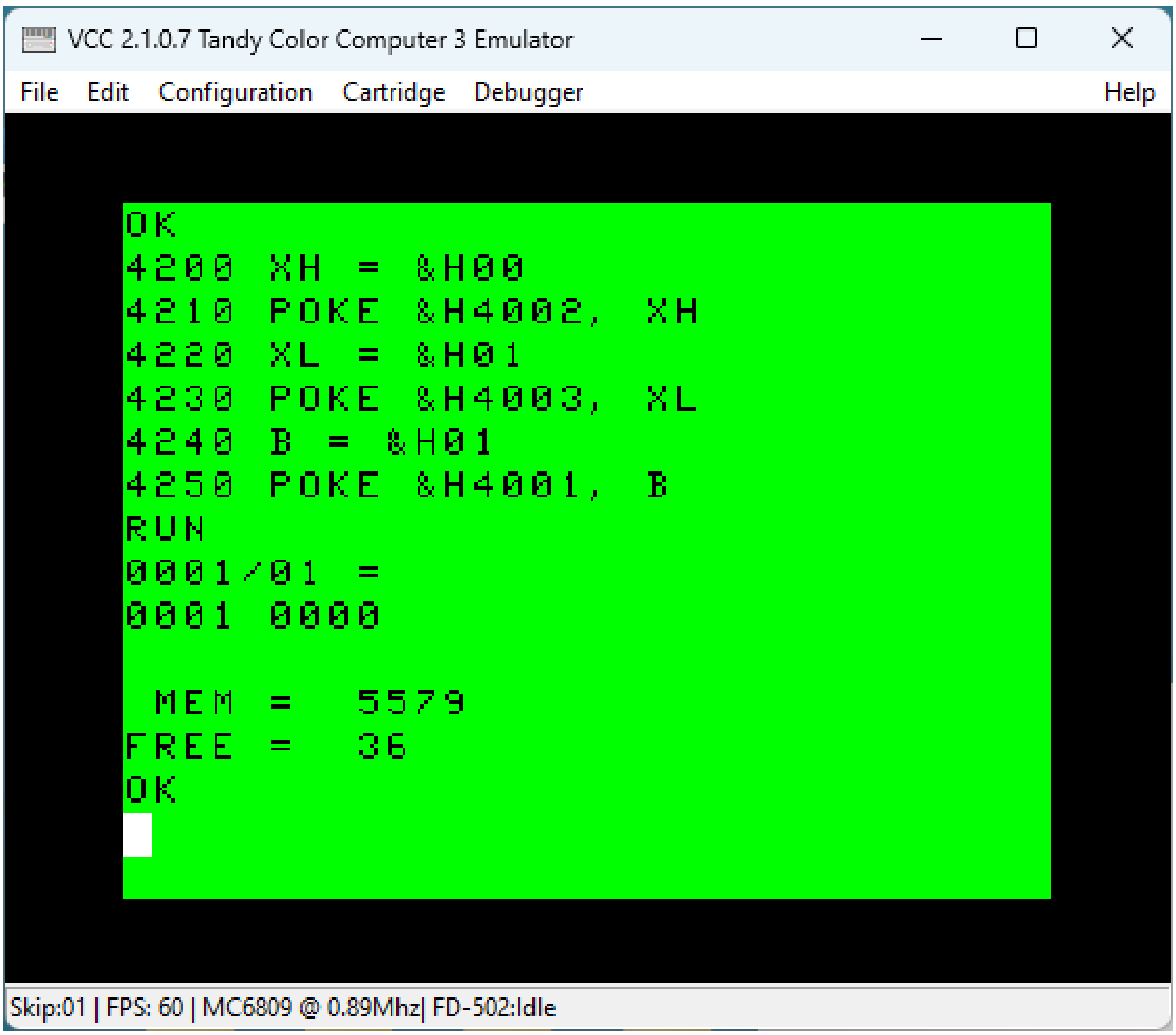
As expected.

—

Fourth Run: - With Test Data:

```
4200 XH = &H00
4210 POKE &H4002, XH
4220 XL = &H01
4230 POKE &H4003, XL
4240 B = &H01
4250 POKE &H4001, B
```

Result: `$0001/$01 –> QUOTIENT  = $0001`
`REMAINDER = $0000`



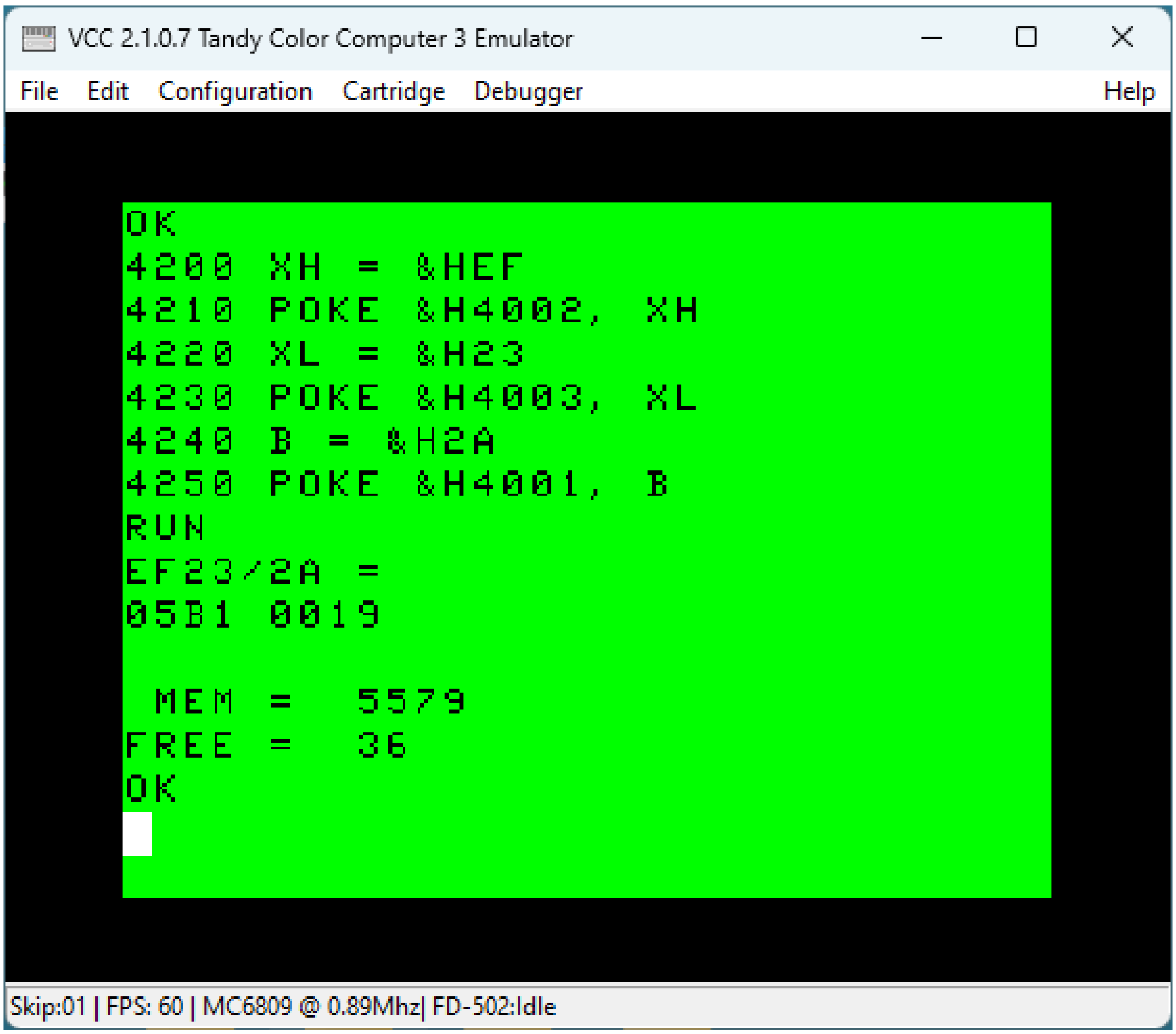
As expected.

Fifth Run: - With Test Data:

```
4200 XH = &HEF
4210 POKE &H4002, XH
4220 XL = &H23
4230 POKE &H4003, XL
4240 B = &H2A
4250 POKE &H4001, B
```

—

Result: `$EF23/$2A —> QUOTIENT  = $05B1`
`                  REMAINDER = $0019`



As expected.

=====

NIRQS: New Interrupts

The existing CoCo Interrupts jump to locations in ROM. Since we will be jumping back-and-forth between ALLRAM Mode and RAMROM Mode, and spending most of our time in ALLRAM Mode, we need to put our Interrupt Handling Routines in low RAM so that they will function correctly regardless of which Mode we are in at the time an interrupt occurs.

Once these new handlers are established in low RAM, we execute the NIRQS Setup routine, SNIRQS located at $43B6 to switch from the old interrupt handlers to the new interrupt handlers.

```
00100 *****
00110 *
00120 * NIRQS.ASM
00130 * MDJ 2023/02/08
00140 *
00150 * NEW IRQS, I.E.
00160 *   NNMI
00170 *   NFIRQ
00180 *   NIRQ
00190 *
00200 * WE MOVE THE IRQS'
00210 * CODE TO THE LOW RAM
00220 * SO THAT THE SYSTEM
00230 * WON'T CARE WHETHER
00240 * IT'S IN RAMROM MODE
00250 * OR ALLRAM MODE.
00260 *
00270 * WE IGNORE ALL OF THE
00280 * 63.5 MICROSECOND
00290 * CODE SINCE ALL COCOS
00300 * RUN ONLY AND ALWAYS
00310 * AT 60 HZ
00320 *
00330 * WE ALSO NOTE THAT, IN
00340 * THE INTERRUPT ROUTINES
00350 * THEMSELVES, WE DON'T
00360 * HAVE TO PSHS ANY
00370 * REGISTERS BECAUSE
00380 * THAT IS TAKEN CARE OF
00390 * BY THE INTERRUPT/RTI
00400 * MECHANISM ITSELF.
00410 *
00420 * IN THE NEW FIRQ ROUTINE,
00430 * IF IT IS A CARTRIDGE
00440 * INTERRUPT, WE JUST
```

```
                  00450 * IMMEDIATELY FORCE A
                  00460 * COLD START
                  00470 *
                  00480 * IN THE NEW IRQ ROUTINE,
                  00490 * WE DO NOT REDIRECT THE
                  00500 * RTI INTO THE PLAY COMMAND
                  00510 *
                  00520 * WE DO NOT EXECUTE ANY
                  00530 * OF THIS CODE. INSTEAD,
                  00540 * WE JUST EXECUTE:
                  00550 *   SNIRQS AT $43B6
                  00560 * TO SET UP THE NEW IRQS
                  00570 *
                  00580 *****
                  00590 *
                  00600 * ALL OF THIS WAS ADAPTED
                  00610 * FROM THE UNRAVELLED
                  00620 * SERIES AUTHORED BY:
                  00630 * WALTER K ZYDHEK
                  00640 *
                  00650 *****
                  00660
                  00670 * EXTERNAL ROUTINE
                  00680 * AND VARIABLE
                  00690 * ADDRESSES
          008D    00700 SNDDUR  EQU     $008D   INTERRUPT TIMER
(SOUND COMMAND)

          00D5    00710 VD5     EQU     $00D5   SCRATCHPAD VARIABLE
('PLAY' INT
ERVAL)
          00E3    00720 PLYTMR  EQU     $00E3   THE PLAY TIMER
          010A    00730 NMIJV   EQU     $010A   NMI JUMP VECTOR
          010D    00740 IRQJV   EQU     $010D   IRQ JUMP VECTOR
          0110    00750 FIRQJV  EQU     $0110   FIRQ JUMP VECTOR
          0112    00760 TIMVAL  EQU     $0112   REAL TIME CLOCK
          0982    00770 NMIFLG  EQU     $0982   NMI FLAG: 0=DON'T
VECTOR <>0=YE
CTOR OUT
          0983    00780 DNMISV  EQU     $0983   NMI VECTOR: WHERE TO
JUMP FOLLO
WING AN NMI
          0985    00790 RDYTMR  EQU     $0985   MOTOR TURN OFF TIMER
          0986    00800 DRGRAM  EQU     $0986   RAM IMAGE OF DSKREG
($FF40)
          41A2    00810 COLD    EQU     $41A2   ML FOUNDATION CORE
COLD START
```

```
          FF00     00820 PIA0    EQU     $FF00   PERIPHERAL INTERFACE
ADAPTER ON
E
          FF20     00830 PIA1    EQU     $FF20   PERIPHERAL INTERFACE
ADAPTER TW
O
          FF40     00840 DSKREG  EQU     $FF40   DISK CONTROL
REGISTER
                   00850
4364               00860         ORG     $4364
                   00870
                   00880 * NEW NON-MASKABLE INTERRUPT
                   00890 * TO REPLACE NMI
4364 B6   0982     00900 NNMI    LDA    NMIFLG  GET NMI FLAG
4367 27   08       00910         BEQ    LBL001  RETURN IF NOT ACTIVE
4369 BE   0983     00920         LDX    DNMISV  GET NEW RETURN VECTOR
436C AF   6A       00930         STX    10,S    STORE AT STACKED PC
SLOT ON STAC
K
436E 7F   0982     00940         CLR    NMIFLG  RESET NMI FLAG
4371 3B            00950 LBL001  RTI
                   00960
                   00970 * NEW FAST INTERRUPT
                   00980 * TO REPLACE FIRQ
4372 7D   FF23     00990 NFIRQ   TST     PIA1+3  CARTRIDGE INTERRUPT?
4375 2B   01       01000         BMI     LBL002  YES
4377 3B            01010         RTI
4378 7E   41A2     01020 LBL002  JMP     COLD    GO DO MLF CORE COLD
START
                   01030
                   01040 * NEW INTERRUPT
                   01050 * TO REPLACE IRQ
437B B6   FF02     01060 NIRQ    LDA     PIA0+2  RESET PIA INTERRUPT
FLAG
437E B6   0985     01070         LDA     RDYTMR  GET TIMER
4381 27   11       01080         BEQ     LBL003  BRANCH IF NOT ACTIVE
4383 4A            01090         DECA            DECREMENT THE TIMER
4384 B7   0985     01100         STA     RDYTMR  SAVE IT
4387 26   0B       01110         BNE     LBL003  BRANCH IF NOT TIME
TO TURN OFF
DISK MOTORS
4389 B6   0986     01120         LDA     DRGRAM  = GET DSKREG IMAGE
438C 84   B0       01130         ANDA    #$B0    = TURN ALL MOTORS
AND DRIVE SEL
ECTS OFF
438E B7   0986     01140         STA     DRGRAM  = PUT IT BACK IN RAM
IMAGE
```

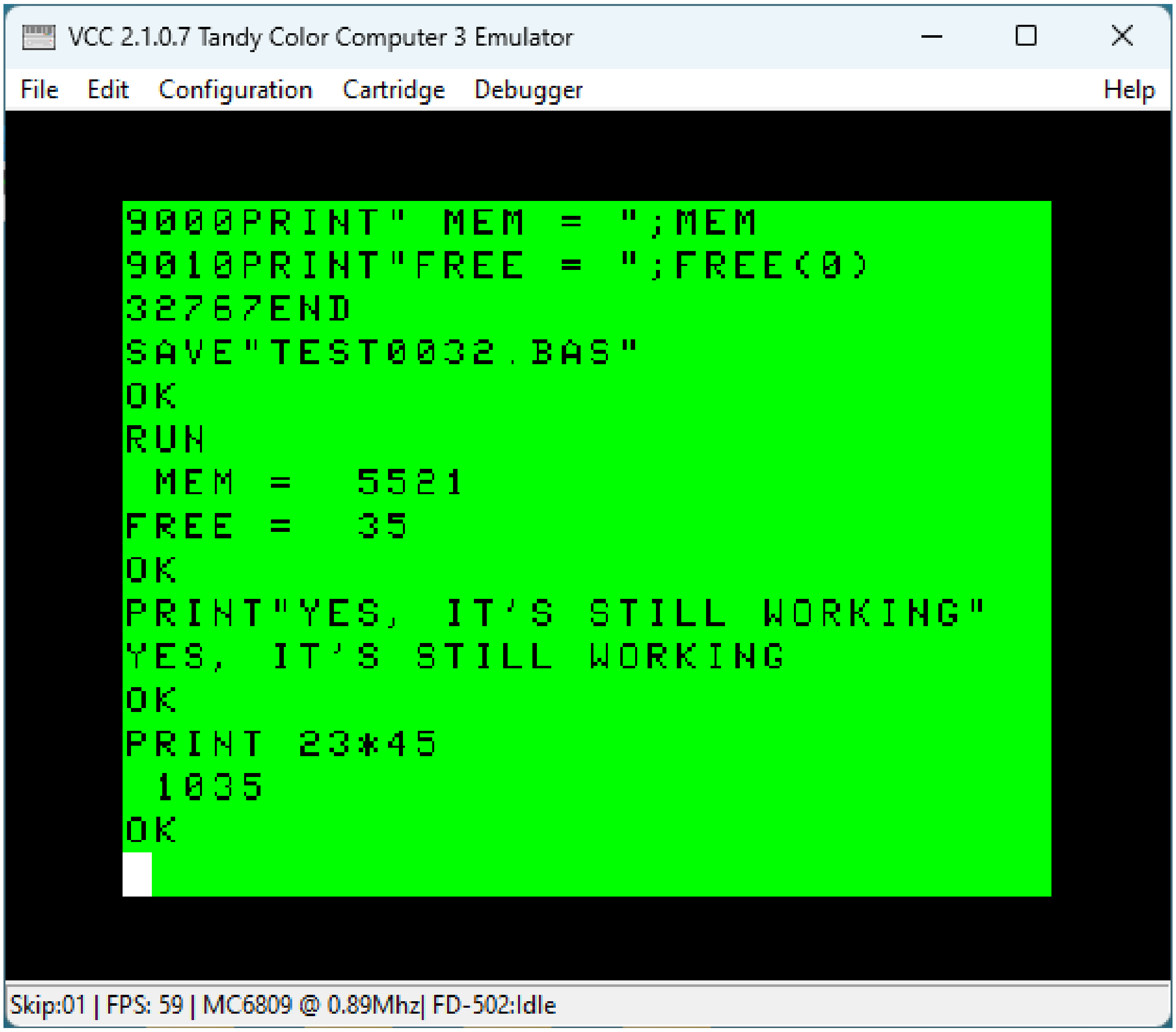
```
4391 B7   FF40      01150          STA     DSKREG  SEND TO CONTROL
REGISTER (MOTOR
S OFF)
4394 BE   0112      01160 LBL003  LDX     TIMVAL  GET REAL TIME CLOCK
4397 30   01        01170          LEAX    $01,X   INCREMENT IT
4399 BF   0112      01180          STX     TIMVAL  SAVE IT
439C 4F             01190          CLRA            CLEAR ACCA
439D 1F   8B        01200          TFR     A,DP    SET THE DIRECT PAGE
TO ZERO
439F DC   E3        01210          LDD     PLYTMR  GET THE PLAY TIMER
43A1 27   0A        01220          BEQ     LBL004
43A3 93   D5        01230          SUBD    VD5     SUBTRACT OUT PLAY
INTERVAL
43A5 DD   E3        01240          STD     PLYTMR  SAVE THE NEW TIMER
VALUE
43A7 22   04        01250          BHI     LBL004  BRANCH IF PLAY
COMMAND NOT DONE

43A9 0F   E3        01260          CLR     PLYTMR  RESET MSB OF PLAY
TIMER IF DONE

43AB 0F   E4        01270          CLR     PLYTMR+1 RESET LSB OF PLAY
TIMER
43AD 9E   8D        01280 LBL004  LDX     SNDDUR  GET INTERRUPT TIMER
(SOUND COMM
AND)
43AF 27   04        01290          BEQ     LBL005  RETURN IF TIMER = 0
43B1 30   1F        01300          LEAX    -1,X    DECREMENT TIMER IF
NOT = 0
43B3 9F   8D        01310          STX     SNDDUR  SAVE NEW TIMER VALUE
43B5 3B             01320 LBL005  RTI
                    01330
                    01340 * SET UP THE NEW INTERRUPTS
                    01350 * NOTE THAT WE MAKE THE NNMI
                    01360 * CHANGE FIRST; SO AS TO
                    01370 * MINIMIZE THE AMOUNT OF TIME
                    01380 * THAT THE INTERRUPTS ARE
                    01390 * MASKED.
43B6 34   07        01400 SNIRQS  PSHS    A,B,CC
43B8 CC   4364      01410          LDD     #NNMI   NEW NMI VECTOR
43BB FD   010A      01420          STD     NMIJV   NEW NMI JUMP VECTOR
LOCATION
43BE 1A   50        01430          ORCC    #$50    MASK IRQ & FIRQ
INTERRUPTS
43C0 CC   4372      01440          LDD     #NFIRQ  NEW FIRQ VECTOR
43C3 FD   0110      01450          STD     FIRQJV  NEW FIRQ JUMP VECTOR
LOCATION
```

```
43C6 CC   437B      01460          LDD      #NIRQ    NEW IRQ VECTOR
43C9 FD   010D      01470          STD      IRQJV    NEW IRQ JUMP VECTOR
LOCATION
43CC 1C   AF        01480          ANDCC    #$AF     UNMASK IRQ & FIRQ
INTERRUPTS
43CE 35   07        01490          PULS     A,B,CC
                    01500
43D0 39             01510          RTS
                    01520
          0000      01530          END
```

There is no Assembly Language Test Routine for NIRQS. To test the handlers, we simply use a tiny BASIC program to execute SNIRQS while still in RAMROM Mode, and then we simply execute some BASIC commands to see that they're still working.

```
1000'TEST0032.BAS
1210CLEAR1000,&H4000
1220PCLEAR4
1300LOADM"MLCORE.BIN"
5010EXEC&H43B6
9000PRINT" MEM = ";MEM
9010PRINT"FREE = ";FREE(0)
32767END
```

Result:

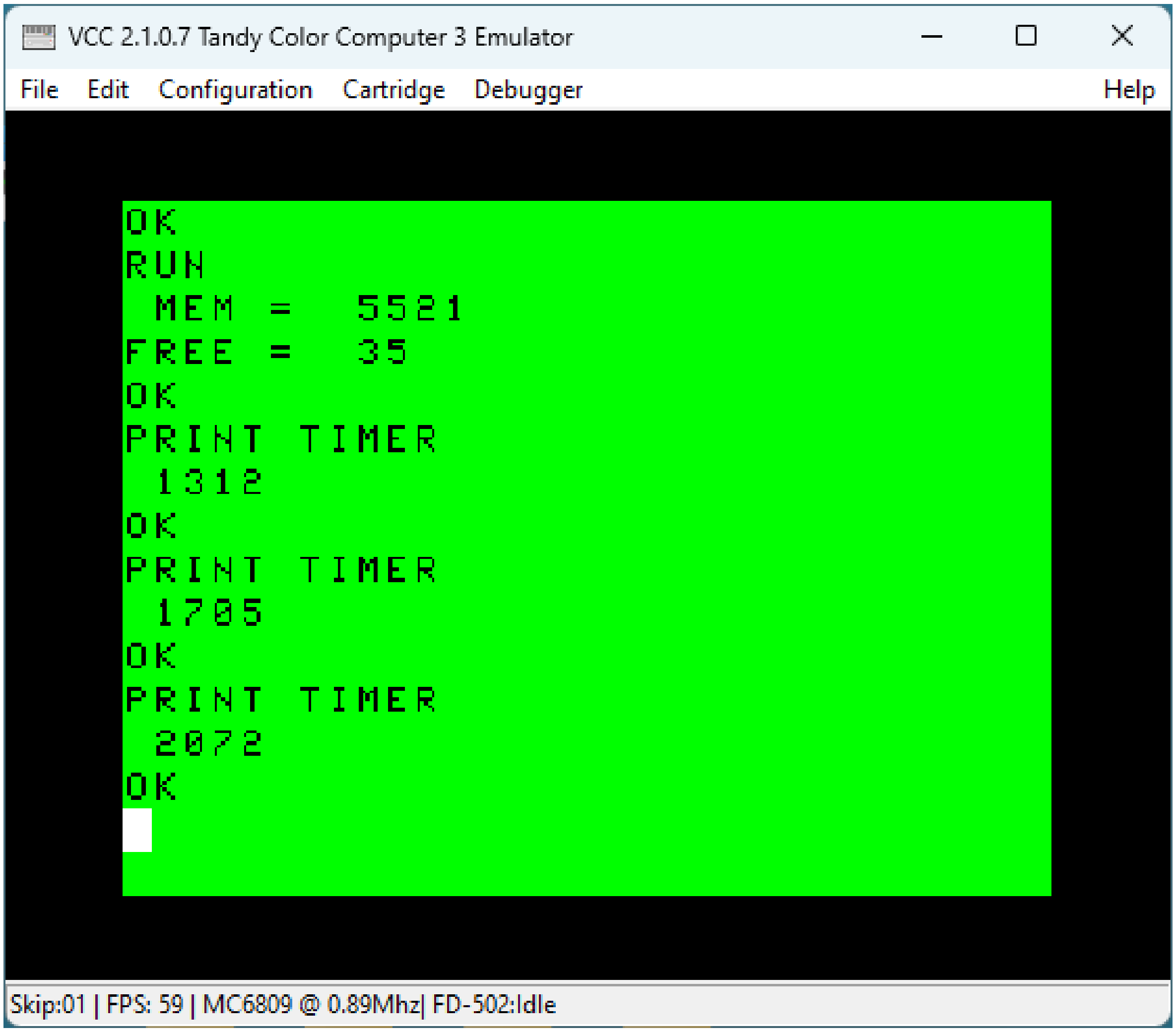


As expected.

———

As a second simple test, we run TEST0032.BAS again and just check that the TIMER Interrupt is being properly processed.

Result:



As expected.

____

Additional testing of these new Interrupt Handling Routines, in ALLRAM Mode, is performed in the RNDU16 Chapter below.

=====

SEED: 16-bit Unsigned Pseudo-Random Number Seed Memory Location

This is simply a two-byte dedicated memory location.

```
                    00100 *****
                    00110 *
                    00120 * SEED.ASM
                    00130 * MDJ 2023/02/07
                    00140 *
                    00150 * 16-BIT UNSIGNED
                    00160 * SEED FOR THE
                    00170 * RANDOM NUMBER
                    00180 * GENERATOR
                    00190 *
                    00200 *****
                    00210
43D1                00220         ORG     $43D1
                    00230
43D1                00240 SEED    RMB     2
                    00250
          0000      00260         END
```

——-

See the RNDU16 Chapter for testing.

=====

SSEED: Set a Specified 16-bit Unsigned Pseudo-Random Number Seed Value

This is used for setting the Random Seed to a specific value and is primarily used for when you want to repeatedly test something using a known sequence of 16-bit unsigned numbers.

For example, if you set the Seed to $1234, the sequence will be 4CAD, F7EA, 567B, C330, 8A19 …. every time.

```
                    *****
                    *
                    * SSEED.ASM
                    * MDJ 2023/02/07
                    *
                    * SET A SELECTED
                    * 16-BIT UNSIGNED
                    * SEED FOR THE
                    * RANDOM NUMBER
                    * GENERATOR
                    *
                    * ENTRY CONDITIONS:
                    * D = THE SELECTED SEED
                    *
                    * EXIT CONDITIONS:
                    * NONE
                    *
                    *****

                    * EXTERNAL VARIABLE
                    * ADDRESS
          43D1      SEED    EQU     $43D1

43D3                        ORG     $43D3

43D3 FD   43D1      SSEED   STD     SEED

43D6 39                     RTS

          0000              END
```

See the RNDU16 Chapter for testing.

=====

RSEED: Set a Random 16-bit Unsigned Pseudo-Random Number Seed Value

This is used for setting the Random Seed to a random value and is primarily used for when you want to simulate a random sequence which will be different each time you run it, e.g. as during game processing.

```
                    00100 *****
                    00110 *
                    00120 * RSEED.ASM
                    00130 * MDJ 2023/02/07
                    00140 *
                    00150 * SET A RANDOM
                    00160 * 16-BIT UNSIGNED
                    00170 * SEED FOR THE
                    00180 * RANDOM NUMBER
                    00190 * GENERATOR
                    00200 *
                    00210 * ENTRY CONDITIONS:
                    00220 * NONE
                    00230 *
                    00240 * EXIT CONDITIONS:
                    00250 * NONE
                    00260 *
                    00270 *****
                    00280
                    00290 * EXTERNAL VARIABLE
                    00300 * ADDRESSES
          0112      00310 TIMVAL  EQU     $0112
          43D1      00320 SEED    EQU     $43D1
                    00330
43D7                00340         ORG     $43D7
                    00350
43D7 34   06        00360 RSEED   PSHS    A,B
43D9 FC   0112      00370         LDD     TIMVAL
43DC FD   43D1      00380         STD     SEED
43DF 35   06        00390         PULS    A,B
                    00400
43E1 39             00410         RTS
                    00420
```

```
     0000     00430          END
```

See the RNDU16 Chapter for testing.

=====

RNDU16: Returns a 16-bit Unsigned Pseudo-Random Number

This is the Pseudo-Random Number Generator itself.

```
00100 *****
00110 *
00120 * RNDU16.ASM
00130 * MDJ 2023/02/08
00140 *
00150 * 16-BIT UNSIGNED
00160 * RANDOM NUMBER
00170 * GENERATOR
00180 *
00190 * ENTRY CONDITIONS:
00200 * NONE
00210 *
00220 * EXIT CONDITIONS:
00230 * D = RANDOM NUMBER
00240 *
00250 * THE RANDOM NUMBER
00260 * IS ALSO STORED
00270 * IN THE SEED, READY
00280 * FOR THE NEXT
00290 * ITERATION
00300 *
00310 *****
00320 *
00330 * THIS FUNCTION WAS
00340 * ADAPTED FROM:
00350 * BARROW, DAVID (1984).
00360 * 6809 MACHINE CODE
00370 * PROGRAMMING.
00380 * LONDON: GRANADA
00390 * TECHNICAL BOOKS,
00400 * PAGES 29-33.
00410 *
00420 *****
00430 *
00440 * EQUATION:
00450 * R2 = ((1509 * R1) + 41) MOD (65536)
00460 * USING 1509 = (6 * 256) - 27
00470 * AND THEN DOING SHIFT AND ADDITION
00480 * INSTEAD OF MULTIPLICATION
```

```
                   00490 *   =  75 CLOCK CYCLES INSTEAD OF
                   00500 *     333 CLOCK CYCLES USING MU1616
                   00510 *
                   00520 *****
                   00530
                   00540 * EXTERNAL VARIABLE
                   00550 * ADDRESS
          43D1     00560 SEED    EQU     $43D1
                   00570
43E2               00580         ORG     $43E2
                   00590
43E2 FC   43D1     00600 RNDU16  LDD     SEED
43E5 34   06       00610         PSHS    D
43E7 58            00620         ASLB
43E8 49            00630         ROLA
43E9 E3   E4       00640         ADDD    ,S
43EB ED   E4       00650         STD     ,S
43ED 58            00660         ASLB
43EE 49            00670         ROLA
43EF 34   04       00680         PSHS    B
43F1 58            00690         ASLB
43F2 49            00700         ROLA
43F3 58            00710         ASLB
43F4 49            00720         ROLA
43F5 E3   61       00730         ADDD    1,S
43F7 ED   61       00740         STD     1,S
43F9 35   02       00750         PULS    A
43FB C6   29       00760         LDB     #41
43FD A3   E1       00770         SUBD    ,S++
43FF FD   43D1     00780         STD     SEED
                   00790
4402 39            00800         RTS
                   00810
          0000     00820         END
```

There are two Assembly Language Test Routines for testing RNDU16. The first is with a Selected Seed, and the Second is with a Random Seed.

The First Assembly Language Test Routine:

```
00100 *****
00110 *
00120 * TEST0033.ASM
00130 * MDJ 2023/02/17
00140 *
```

```
                  00150 * RNDU16 TEST:
                  00160 * SELECTED SEED
                  00170 *
                  00180 *****
                  00190
                  00200 * RAMROM TRIGGER ADDRESS
          FFDE    00210 RAMROM  EQU     $FFDE
                  00220
                  00230 * ALLRAM TRIGGER ADDRESS
          FFDF    00240 ALLRAM  EQU     $FFDF
                  00250
                  00260 * ML FOUNDATION
                  00270 * CORE ADDRESSES
          40D7    00280 CRLF    EQU     $40D7
          4178    00290 PUTWRA  EQU     $4178
          43B6    00300 SNIRQS  EQU     $43B6
          43D1    00310 SEED    EQU     $43D1
          43D3    00320 SSEED   EQU     $43D3
          43E2    00330 RNDU16  EQU     $43E2
                  00340
7000              00350         ORG     $7000
                  00360
                  00370 * RNDU16 TEST
                  00380
7000 34   07      00390         PSHS    A,B,CC
                  00400
                  00410 * SETUP THE NEW INTERRUPT HANDLERS
                  00420 * AND ENTER ALLRAM MODE
7002 BD   43B6    00430         JSR     SNIRQS
7005 B7   FFDF    00440         STA     ALLRAM
                  00450
7008 CC   1234    00460         LDD     #$1234  SET SEED
700B BD   43D3    00470         JSR     SSEED
700E BD   40D7    00480         JSR     CRLF
                  00490
                  00500 * FIVE RANDOM NUMBERS
7011 BD   43E2    00510         JSR     RNDU16  GET A RANDOM NUMBER
7014 BD   4178    00520         JSR     PUTWRA  DISPLAY IT
7017 BD   40D7    00530         JSR     CRLF
701A BD   43E2    00540         JSR     RNDU16  GET A RANDOM NUMBER
701D BD   4178    00550         JSR     PUTWRA  DISPLAY IT
7020 BD   40D7    00560         JSR     CRLF
7023 BD   43E2    00570         JSR     RNDU16  GET A RANDOM NUMBER
7026 BD   4178    00580         JSR     PUTWRA  DISPLAY IT
7029 BD   40D7    00590         JSR     CRLF
702C BD   43E2    00600         JSR     RNDU16  GET A RANDOM NUMBER
702F BD   4178    00610         JSR     PUTWRA  DISPLAY IT
```

```
7032 BD   40D7      00620          JSR      CRLF
7035 BD   43E2      00630          JSR      RNDU16  GET A RANDOM NUMBER
7038 BD   4178      00640          JSR      PUTWRA  DISPLAY IT
703B BD   40D7      00650          JSR      CRLF
                    00660
                    00670 * RETURN TO RAMROM MODE AND EXIT
703E B7   FFDE      00680 LBL002   STA      RAMROM
7041 35   07        00690          PULS     A,B,CC
7043 39             00700          RTS
                    00710
          0000      00720          END
```

———

The Second Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0034.ASM
                    00130 * MDJ 2023/02/17
                    00140 *
                    00150 * RNDU16 TEST:
                    00160 * RANDOM SEED
                    00170 *
                    00180 *****
                    00190
                    00200 * RAMROM TRIGGER ADDRESS
          FFDE      00210 RAMROM   EQU      $FFDE
                    00220
                    00230 * ALLRAM TRIGGER ADDRESS
          FFDF      00240 ALLRAM   EQU      $FFDF
                    00250
                    00260 * ML FOUNDATION
                    00270 * CORE ADDRESSES
          40D7      00280 CRLF     EQU      $40D7
          4178      00290 PUTWRA   EQU      $4178
          43B6      00300 SNIRQS   EQU      $43B6
          43D1      00310 SEED     EQU      $43D1
          43D7      00320 RSEED    EQU      $43D7
          43E2      00330 RNDU16   EQU      $43E2
                    00340
7000                00350          ORG      $7000
                    00360
                    00370 * RNDU16 TEST
                    00380
7000 34   07        00390          PSHS     A,B,CC
                    00400
```

```
                    00410 * SETUP THE NEW INTERRUPT HANDLERS
                    00420 * AND ENTER ALLRAM MODE
7002 BD   43B6      00430         JSR     SNIRQS
7005 B7   FFDF      00440         STA     ALLRAM
                    00450
7008 BD   43D7      00460         JSR     RSEED   RANDOM SEED
700B BD   40D7      00470         JSR     CRLF
                    00480
                    00490 * FIVE RANDOM NUMBERS
700E BD   43E2      00500         JSR     RNDU16  GET A RANDOM NUMBER
7011 BD   4178      00510         JSR     PUTWRA  DISPLAY IT
7014 BD   40D7      00520         JSR     CRLF
7017 BD   43E2      00530         JSR     RNDU16  GET A RANDOM NUMBER
701A BD   4178      00540         JSR     PUTWRA  DISPLAY IT
701D BD   40D7      00550         JSR     CRLF
7020 BD   43E2      00560         JSR     RNDU16  GET A RANDOM NUMBER
7023 BD   4178      00570         JSR     PUTWRA  DISPLAY IT
7026 BD   40D7      00580         JSR     CRLF
7029 BD   43E2      00590         JSR     RNDU16  GET A RANDOM NUMBER
702C BD   4178      00600         JSR     PUTWRA  DISPLAY IT
702F BD   40D7      00610         JSR     CRLF
7032 BD   43E2      00620         JSR     RNDU16  GET A RANDOM NUMBER
7035 BD   4178      00630         JSR     PUTWRA  DISPLAY IT
7038 BD   40D7      00640         JSR     CRLF
                    00650
                    00660 * RETURN TO RAMROM MODE AND EXIT
703B B7   FFDE      00670 LBL002  STA     RAMROM
703E 35   07        00680         PULS    A,B,CC
7040 39             00690         RTS
                    00700
          0000      00710         END
```

There are also two BASIC Language Control Programs for testing RNDU16. The first is with a Selected Seed, and the Second is with a Random Seed. Each is run twice; the first set of two runs to verify that the Selected Seed generates the same series each time it is run, and the second set of two runs to verify that the Random Seed generates a different series each time it is run.

The First BASIC Language Control Program (Selected Seed):

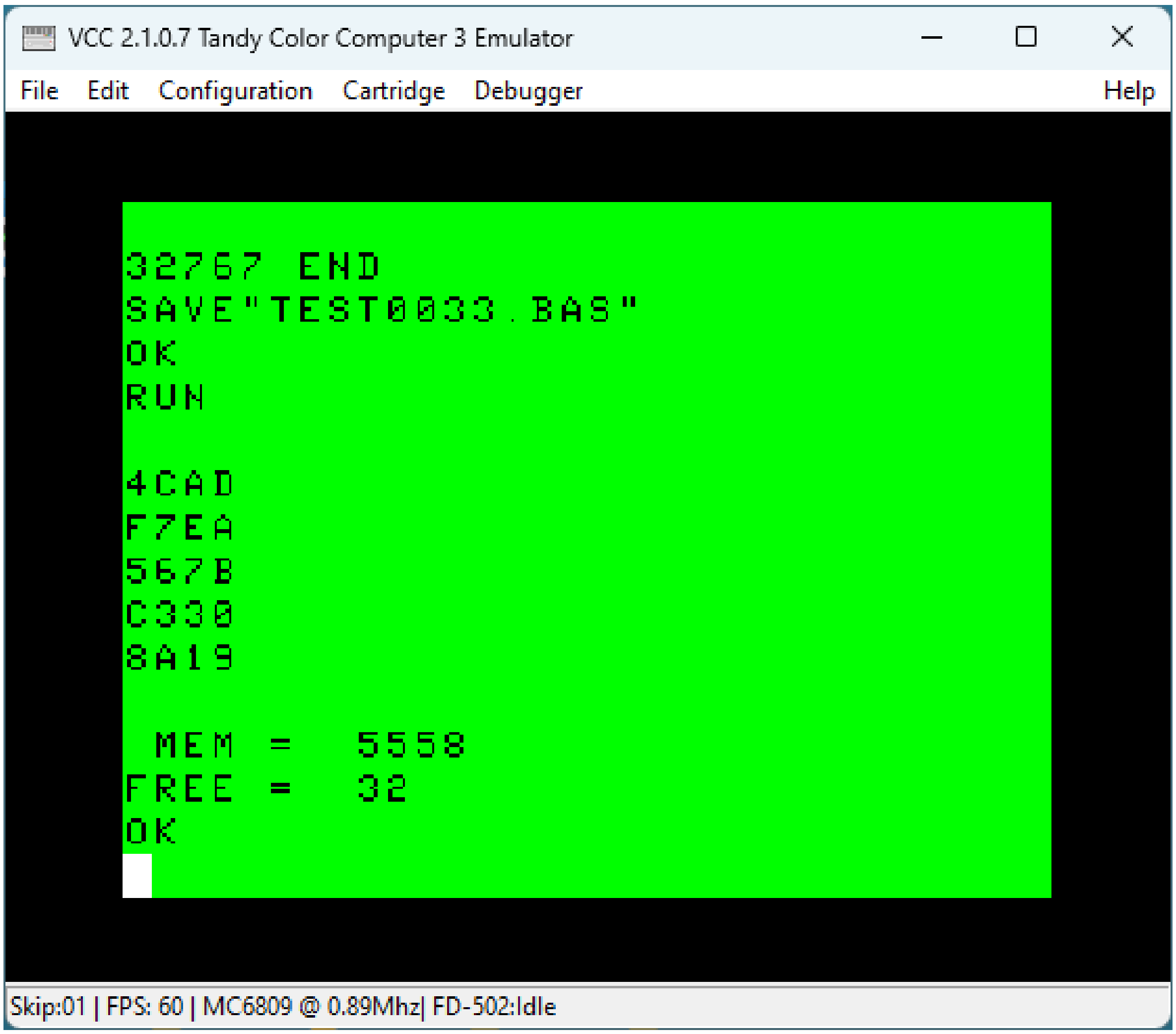
```
1000 '*****
1010 '*
1020 '* TEST0033.BAS
1030 '* MDJ 2023/02/17
1040 '*
1050 '* RNDU16 TEST
```

```
1055 '* SELECTED SEED
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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0033.BIN"
1330 '

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 '

4000 'LOAD THE TEST DATA
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B

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
```

—

First Run Result:



As expected.

—

Second Run Result:

As expected.

—

The Second BASIC Language Control Program (Random Seed):

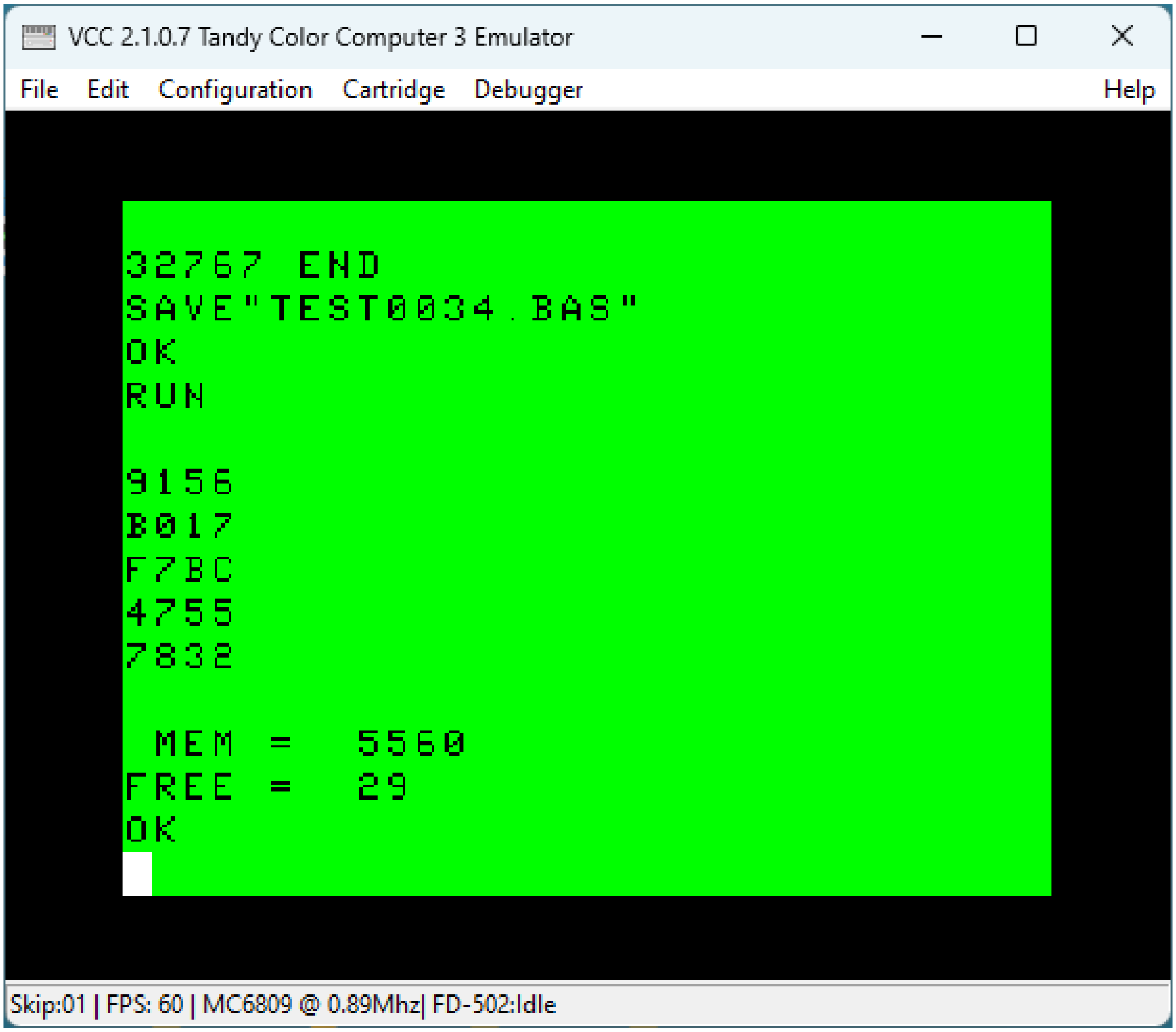
```
1000 '*****
1010 '*
1020 '* TEST0034.BAS
1030 '* MDJ 2023/02/17
1040 '*
1050 '* RNDU16 TEST
1055 '* RANDOM SEED
1060 '*
1070 '*****
1080 '
```

```
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
1310 'ML TEST ROUTINE
1320 LOADM "TEST0034.BIN"
1330 '

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 '

4000 'LOAD THE TEST DATA
4200 XH = &HFF
4210 POKE &H4002, XH
4220 XL = &HFF
4230 POKE &H4003, XL
4240 B = &HFF
4250 POKE &H4001, B

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
```

—

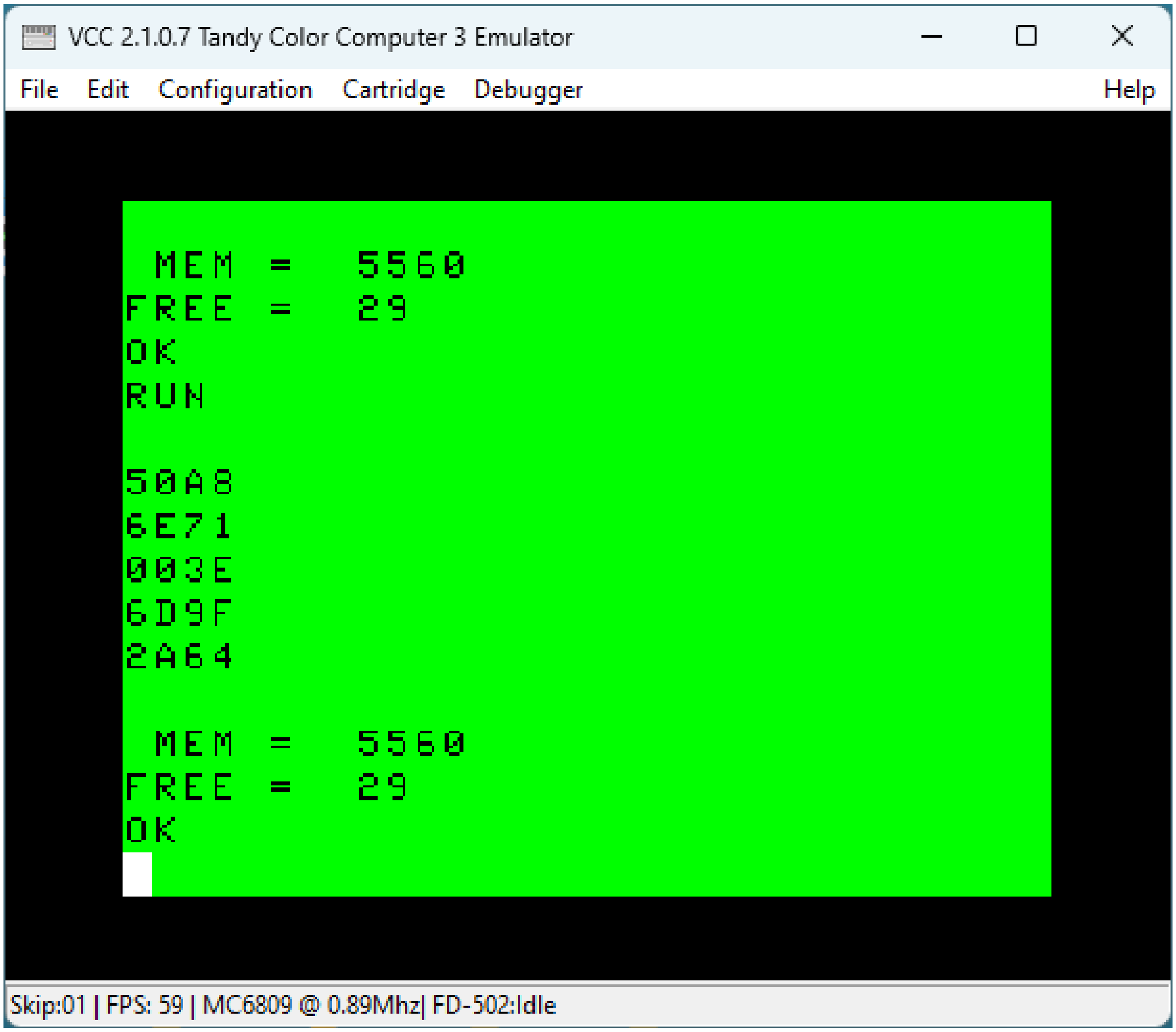
First Run Result:



As expected.

—

Second Run Result:



As expected.

=====

STRTUP: The ML Foundation Core Startup Routine

This is the entry point for the ML Foundation Core..

```
                    *****
                    *
                    * STRTUP.ASM
                    * MDJ 2023/02/09
                    *
                    * ML CORE
                    * STARTUP ROUTINE
                    *
                    *****

                    * EXTERNAL ROUTINE
                    * AND VARIABLE
                    * ADDRESSES
          400A      REGPC   EQU     $400A
          43B6      SNIRQS  EQU     $43B6

                    * ALLRAM TRIGGER ADDRESS
          FFDF      ALLRAM  EQU     $FFDF

                    * ADDRESS OF
                    * TOP OF HIGH RAM
          FEFF      TPHRAM  EQU     $FEFF

4403                        ORG     $4403

4403 34   50        STRTUP  PSHS    X,U

                    * GET THE RUN ADDRESS
                    * FROM THE BASIC/ML
                    * REGXFR'S REGPC
4405 BE   400A              LDX     REGPC

                    * GO SET THE NEW INTERRUPTS
4408 BD   43B6              JSR     SNIRQS

                    * SET ALLRAM MODE
440B B7   FFDF              STA     ALLRAM
```

```
                00470
                00480 * PUT THE USER STACK
                00490 * AT THE TOP OF HIGH RAM
440E CE   FEFF  00500         LDU     #TPHRAM
                00510
                00520 * GO TO THE RUN ADDRESS
4411 AD   84    00530         JSR     ,X
                00540
4413 35   50    00550         PULS    X,U
                00560
4415 39         00570         RTS
                00580
          0000  00590         END
```

———-

No specific testing is required for this routine because it has been inherently tested in many of the tests performed throughout this paper. The BASIC Language Control Programs' lines

```
6020 EXEC &H4403
```

jump into this STRTUP Routine, and that line is present in almost every BASIC Language Control Program used in this paper.

=====

Results

The development and preliminary testing of these components of the The ML Foundation Core are now complete and correct to the best of my knowledge and ability.

This does not preclude the possibility that errors in these routines may be encountered during the continued development of the Core and the rest of The ML Foundation.

Nor does it preclude the possibility that you may discover some errors which I’ve missed. If you do, please let me know.

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

=====

Conclusions and Future Work

At this point, The ML Foundation Core is deemed to be complete and correct.

Please take it, play with it, experiment with it, try to break it, etc. Let me know how you fare, if you will.

Meanwhile, I will continue with further development of The ML Foundation. The work will be split into several parallel paths which will proceed informally and somewhat concurrently:

1. A "False Disk" system which will provide for the linear sequential numbering of Disk Sectors in games and other simple software systems. For example, instead of referring to Track 13, Sector 17; Track 13, Sector 18; Track 14, Sector 1; and Track 14, Sector 2 - we would simply refer to Linear Sectors 251, 252, 253, and 254.

2. A "Fake Text" system which will divide the 256 x 192 PMODE 4 Screen into 32 x 16 cells (each cell being 8 pixels wide and 12 pixels high) and being associated with a set of 256 (8 x 12 pixel) graphic "characters" for use in developing simple games (especially mazes).

3. A High-RAM-resident Stack Engine.

4. Semi-Graphics and PMODE Graphics.

5. All-Assembly Language Routines where speed is a major factor, and

6. Routines that jump into ROM to handle inherently slower tasks.

7. Other stuff that may occasionally come to mind.

Among items to be considered under item 5 above might be Assembly Language Routines analogous to BASIC Language commands and functions such as **ABS, ASC, ATN, CHR$, CONT, COS, CVN, EXP, FIELD, FILES, INSTR, INT, LEN, LIST, LOF, LOG, LSET, MEM, MKN$, MID$, NEW, PLAY, RIGHT$, RSET, SGN, SIN, SQR, STOP, STRING$, STR$, TAN, TIMER**, and **VAL.**

Among items to be considered under item 6 above might be Assembly Language Routines to access ROM code for BASIC Language commands and functions such as **AUDIO, BACKUP, CLOADM, CLOSE, COPY, CSAVEM, DIR, DRIVE, DSKINI, EOF, FREE, GET, INPUT#, JOYSTK, KILL, LINE INPUT, LLIST, LOADM, LOC, MOTOR, OPEN, POS, PRINT#, PRINT# USING, PUT#, RENAME, SAVEM, SKIPF, UNLOAD, VERIFY OFF, VERIFY ON,** and **WRITE.**

Among items which might be considered under item 7 above would be Assembly Language Routines for Signed 8-bit and 16-bit integer math; Signed and Unsigned 32-bit and 64-bit integer math; IEEE Standard 754 32-bit and 64-bit floating-point math, etc.

=====

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: Truly Relocatable Code

Jack Ganssle writes:

> Relocatable code is software whose execution address can be changed. A relocatable program might run at address 0 in one instance, and at 10000 in another.
>
> Just to confuse the issue, partially built programs are composed of object modules unfortunately called "relocatables". Linkers combine multiple relocatables to one final program. The word "relocatable" is applicable, since each is assembled at a pseudo-address of 0. The linker corrects all address references to the proper execution values. Once linked, the code is frequently no longer relocatable, since it can typically run only at a single address.
>
> Obviously, this sort of relocation is an important consideration for linking multi-module programs. Without it we'd be doomed to giving absolute start addresses to each of the modules before assembly, a mind-numbing prospect since changing the length of any one module may necessitate changing the origin of all of them.

First, an opening caveat: I use Disk EDTASM. I like Disk EDTASM. The fact that Disk EDTASM doesn't include a Linker is not a bother to me, because I also like the closer control I can maintain by writing short, somewhat independent modules of code that can be "relocated" simply (though not trivially) via re-assembly.

Yeah, I know: I'm only working with 64K here. But, if I was on an 3.6 GHz Intel Core i7, I wouldn't be likely to be writing code in assembly language anyway.

That being said, even with a Linker, I wouldn't prefer truly relocatable code because of the cost in bytes, cycles, and code complexity that such relocatability would incur.

For example, consider the following code which compares PC-relative addressing with the usual simple addressing, in order to compare truly relocatable code with non-relocatable code (although even that core is relocatable via re-assembly):

```
00100 *****
00110 *
00120 * PCRTST.ASM
00130 * MDJ 2023/02/04
00140 *
00150 * COMPARATIVE TEST OF
00160 * PC-RELATIVE ADDRESSING
00170 * VS. THE USUAL ADDRESSING
00180 *
00190 * TO DETERMINE WHETHER OR
00200 * NOT RELOCATABLE CODE
00210 * IS WORTH THE COST
00220 *
00230 *****
```

```
                * PUT THIS UP HIGH SO IT
                * WON'T INTERFERE WITH
                * ANYTHING ELSE.
                * (NOT THAT IT REALLY
                *  MATTERS SINCE WE
                *  WON'T ACTUALLY BE
                *  RUNNING THIS CODE)
5000                    ORG     $5000

                * CALL THE SUBROUTINE
                * USING THE USUAL
                * ADDRESSING METHOD
5000 BD   600A          JSR     SBRT
                * NOTE THAT THIS TAKES
                * UP THREE BYTES:
                *   BD 600A
                * AND PAGE 88 OF
                * THE MC6809 COOKBOOK
                * INDICATES THAT BD
                * USES 8 MPU CYCLES

                * CALL THE SUBROUTINE
                * USING PC-RELATIVE
                * ADDRESSING
5003 AD   8D 1003       JSR     SBRT,PCR
                * NOTE THAT THIS TAKES
                * UP FOUR BYTES:
                *   AD 8D 1003
                * AND PAGE 88 OF
                * THE MC6809 COOKBOOK
                * INDICATES THAT AD
                * USES 7+ MPU CYCLES;
                * AND TABLE 3-7 ON
                * PAGE 47 INDICATES
                * THAT THE TOTAL FOR
                * THIS 16-BIT PCR
                * OFFSET = 7+5 = 12
                * MPU CYCLES

                *****
                *****
                **
                ** SO, THE PC-RELATIVE
                ** ADDRESSING HERE COSTS
                **   1 ADDITIONAL BYTE AND
                **   4 ADDITIONAL MPU CYCLES
```

```
                   00710 **
                   00720 ** THEREFORE:
                   00730 **   WE WILL NOT USE
                   00740 **   PC-RELATIVE ADDRESSING
                   00750 **
                   00760 ** IN OUR (MDJ'S) OPINION,
                   00770 ** RELOCATABLE CODE ISN'T
                   00780 ** WORTH THE COST.
                   00790 **
                   00800 *****
                   00810 *****
                   00820
                   00830 * MAKE SOME USELESS
                   00840 * INTERVENING STUFF
                   00850 * JUST TO TAKE UP SPACE
5007 7E   600B     00860         JMP     EXIT
500A               00870 STUFF   RMB     $1000
                   00880
                   00890 * THE SUBROUTINE
600A 39            00900 SBRT    RTS
                   00910
                   00920 * END OF TEST
600B 39            00930 EXIT    RTS
                   00940
          0000     00950         END
```

Appendix D: Stack Testing

In developing the Multiplication and Division routines for the ML Foundation Core, I deemed it advisable to learn more about how to use the stack for temporary storage from within any given routine.

Accordingly, I performed the following five tests which should be fairly self-explanatory.

Test No. 1: THE FIRST TEST SUITE

```
                  00100 *****
                  00110 *
                  00120 * STKTST1.ASM
                  00130 * MDJ 2023/02/02
                  00140 *
                  00150 * A TESTING TOOL TO
                  00160 * HELP LEARN ABOUT
                  00170 * STACK OPERATIONS
                  00180 *
                  00190 *****
                  00200
                  00210 * BASIC/ML TRANSFER
                  00220 * VARIABLES
         4000     00230 REGA    EQU     $4000
         4001     00240 REGB    EQU     $4001
         4002     00250 REGX    EQU     $4002
         4004     00260 REGY    EQU     $4004
         4006     00270 REGS    EQU     $4006
         4008     00280 REGU    EQU     $4008
         400A     00290 REGPC   EQU     $400A
         400C     00300 REGDP   EQU     $400C
         400D     00310 REGCC   EQU     $400D
                  00320
                  00330 * EXTERNAL ROUTINE
                  00340 * ADDRESSES
         409E     00350 PUTBYA  EQU     $409E
         40D7     00360 CRLF    EQU     $40D7
         4178     00370 PUTWRA  EQU     $4178
         41DB     00380 PRTCHA  EQU     $41DB
                  00390
                  00400 * PUT THIS UP IN HIGH
                  00410 * MEMORY SO IT WON'T
                  00420 * INTERFERE WITH
                  00430 * ANYTHING ELSE
7000              00440         ORG     $7000
                  00450
                  00460 * SAVE THE REGISTERS EXCEPT S
```

```
                      00470 * ON THE STACK
7000 34   7F          00480        PSHS A,B,X,Y,U,DP,CC
                      00490
                      00500 * SAVE THE STACK POINTER
                      00510 * SO WE CAN MOVE IT FOR
                      00520 * TESTING PURPOSES
7002 10FF 4006        00530        STS     REGS
                      00540
                      00550 * POINT STACK POINTER
                      00560 * TO TOP OF RAM
7006 10CE 7FFF        00570        LDS     #$7FFF
                      00580
                      00590 * PLACE A SENTINEL AT
                      00600 * THE TOP OF RAM
700A 86   FF          00610        LDA     #$FF
700C B7   7FFF        00620        STA     $7FFF
                      00630
                      00640 * SETUP FIRST TEST SUITE
700F 86   00          00650        LDA     #$00
7011 34   02          00660        PSHS    A
7013 86   01          00670        LDA     #$01
7015 34   02          00680        PSHS    A
7017 86   02          00690        LDA     #$02
7019 34   02          00700        PSHS    A
701B 86   03          00710        LDA     #$03
701D 34   02          00720        PSHS    A
701F 86   04          00730        LDA     #$04
7021 34   02          00740        PSHS    A
7023 86   05          00750        LDA     #$05
7025 34   02          00760        PSHS    A
7027 86   06          00770        LDA     #$06
7029 34   02          00780        PSHS    A
702B 86   07          00790        LDA     #$07
702D 34   02          00800        PSHS    A
702F 86   08          00810        LDA     #$08
7031 34   02          00820        PSHS    A
7033 86   09          00830        LDA     #$09
7035 34   02          00840        PSHS    A
                      00850
                      00860 * DISPLAY THE FIRST TEST SUITE
7037 BD   703D        00870        JSR     STKDSP
                      00880
703A 7E   72A3        00890        JMP     LBL001
                      00900
                      00910 *****
                      00920 *
                      00930 * STKDSP
```

```
                    00940 * SUBROUTINE
                    00950 * TO DISPLAY THE STACK
                    00960 *
                    00970 *****
                    00980
                    00990 * SAVE REGISTERS
703D B7   4000      01000 STKDSP  STA     REGA
7040 F7   4001      01010         STB     REGB
7043 BF   4002      01020         STX     REGX
                    01030
                    01040 * STACK LOCATION 12
7046 86   31        01050         LDA     #$31    1
7048 BD   41DB      01060         JSR     PRTCHA
704B 86   32        01070         LDA     #$32    2
704D BD   41DB      01080         JSR     PRTCHA
7050 86   2C        01090         LDA     #$2C    ,
7052 BD   41DB      01100         JSR     PRTCHA
7055 86   53        01110         LDA     #$53    S
7057 BD   41DB      01120         JSR     PRTCHA
705A 86   20        01130         LDA     #$20    SPACE
705C BD   41DB      01140         JSR     PRTCHA
705F 1F   40        01150         TFR     S,D
7061 C3   000C      01160         ADDD    #$000C
7064 BD   4178      01170         JSR     PUTWRA  ADDRESS
7067 86   20        01180         LDA     #$20    SPACE
7069 BD   41DB      01190         JSR     PRTCHA
706C A6   6C        01200         LDA     12,S
706E BD   409E      01210         JSR     PUTBYA  VALUE
7071 BD   40D7      01220         JSR     CRLF
                    01230
                    01240 * STACK LOCATION 11
7074 86   31        01250         LDA     #$31    1
7076 BD   41DB      01260         JSR     PRTCHA
7079 86   31        01270         LDA     #$31    1
707B BD   41DB      01280         JSR     PRTCHA
707E 86   2C        01290         LDA     #$2C    ,
7080 BD   41DB      01300         JSR     PRTCHA
7083 86   53        01310         LDA     #$53    S
7085 BD   41DB      01320         JSR     PRTCHA
7088 86   20        01330         LDA     #$20    SPACE
708A BD   41DB      01340         JSR     PRTCHA
708D 1F   40        01350         TFR     S,D
708F C3   000B      01360         ADDD    #$000B
7092 BD   4178      01370         JSR     PUTWRA  ADDRESS
7095 86   20        01380         LDA     #$20    SPACE
7097 BD   41DB      01390         JSR     PRTCHA
709A A6   6B        01400         LDA     11,S
```

```
709C BD   409E    01410         JSR     PUTBYA  VALUE
709F BD   40D7    01420         JSR     CRLF
                  01430
                  01440 * STACK LOCATION 10
70A2 86   31      01450         LDA     #$31    1
70A4 BD   41DB    01460         JSR     PRTCHA
70A7 86   30      01470         LDA     #$30    0
70A9 BD   41DB    01480         JSR     PRTCHA
70AC 86   2C      01490         LDA     #$2C    ,
70AE BD   41DB    01500         JSR     PRTCHA
70B1 86   53      01510         LDA     #$53    S
70B3 BD   41DB    01520         JSR     PRTCHA
70B6 86   20      01530         LDA     #$20    SPACE
70B8 BD   41DB    01540         JSR     PRTCHA
70BB 1F   40      01550         TFR     S,D
70BD C3   000A    01560         ADDD    #$000A
70C0 BD   4178    01570         JSR     PUTWRA  ADDRESS
70C3 86   20      01580         LDA     #$20    SPACE
70C5 BD   41DB    01590         JSR     PRTCHA
70C8 A6   6A      01600         LDA     10,S
70CA BD   409E    01610         JSR     PUTBYA  VALUE
70CD BD   40D7    01620         JSR     CRLF
                  01630
                  01640 * STACK LOCATION 9
70D0 86   20      01650         LDA     #$20    SPACE
70D2 BD   41DB    01660         JSR     PRTCHA
70D5 86   39      01670         LDA     #$39    9
70D7 BD   41DB    01680         JSR     PRTCHA
70DA 86   2C      01690         LDA     #$2C    ,
70DC BD   41DB    01700         JSR     PRTCHA
70DF 86   53      01710         LDA     #$53    S
70E1 BD   41DB    01720         JSR     PRTCHA
70E4 86   20      01730         LDA     #$20    SPACE
70E6 BD   41DB    01740         JSR     PRTCHA
70E9 1F   40      01750         TFR     S,D
70EB C3   0009    01760         ADDD    #$0009
70EE BD   4178    01770         JSR     PUTWRA  ADDRESS
70F1 86   20      01780         LDA     #$20    SPACE
70F3 BD   41DB    01790         JSR     PRTCHA
70F6 A6   69      01800         LDA     9,S
70F8 BD   409E    01810         JSR     PUTBYA  VALUE
70FB BD   40D7    01820         JSR     CRLF
                  01830
                  01840 * STACK LOCATION 8
70FE 86   20      01850         LDA     #$20    SPACE
7100 BD   41DB    01860         JSR     PRTCHA
7103 86   38      01870         LDA     #$38    8
```

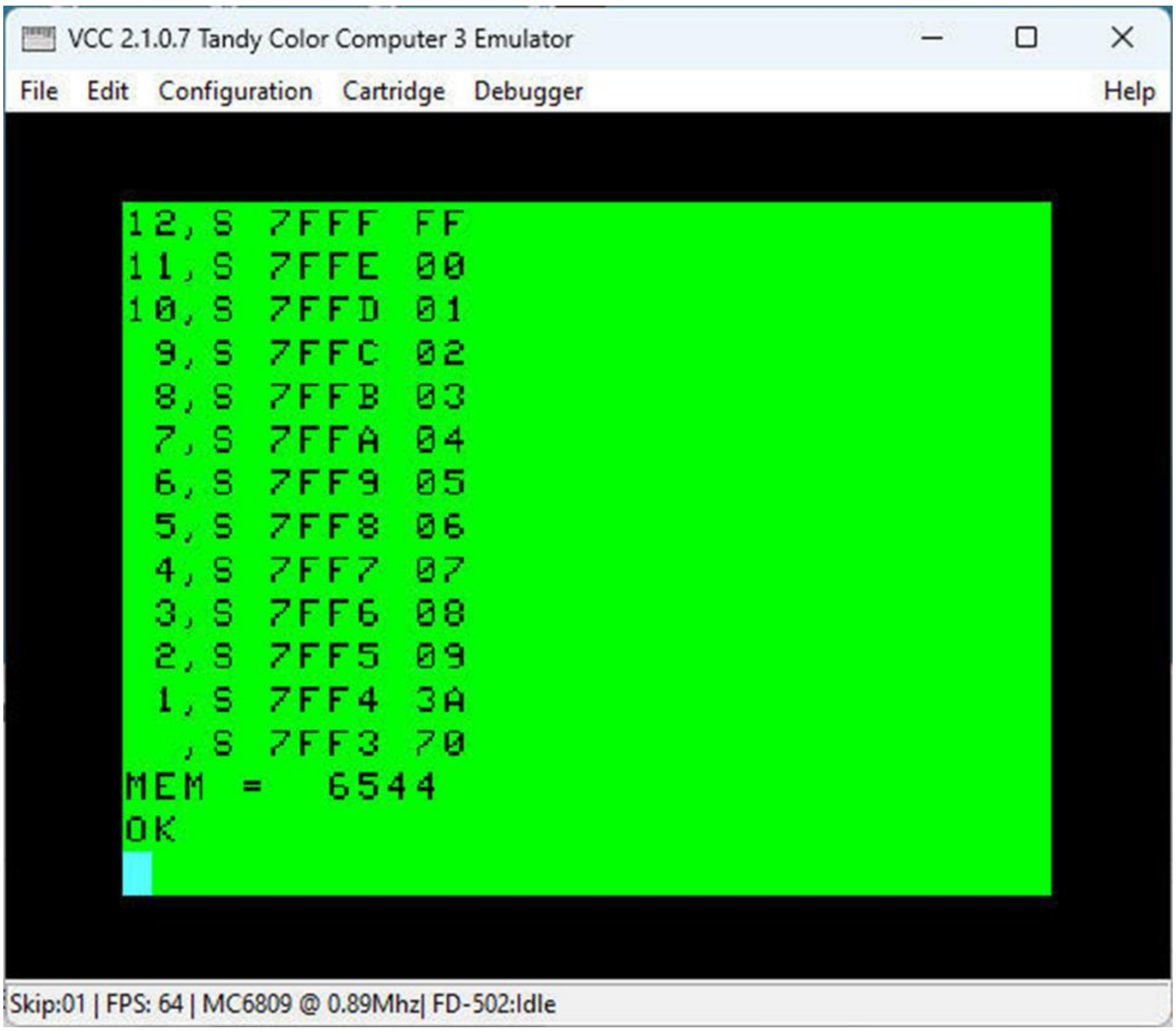
```
7105 BD   41DB     01880          JSR     PRTCHA
7108 86   2C       01890          LDA     #$2C    ,
710A BD   41DB     01900          JSR     PRTCHA
710D 86   53       01910          LDA     #$53    S
710F BD   41DB     01920          JSR     PRTCHA
7112 86   20       01930          LDA     #$20    SPACE
7114 BD   41DB     01940          JSR     PRTCHA
7117 1F   40       01950          TFR     S,D
7119 C3   0008     01960          ADDD    #$0008
711C BD   4178     01970          JSR     PUTWRA  ADDRESS
711F 86   20       01980          LDA     #$20    SPACE
7121 BD   41DB     01990          JSR     PRTCHA
7124 A6   68       02000          LDA     8,S
7126 BD   409E     02010          JSR     PUTBYA  VALUE
7129 BD   40D7     02020          JSR     CRLF
                   02030
                   02040 * STACK LOCATION 7
712C 86   20       02050          LDA     #$20    SPACE
712E BD   41DB     02060          JSR     PRTCHA
7131 86   37       02070          LDA     #$37    7
7133 BD   41DB     02080          JSR     PRTCHA
7136 86   2C       02090          LDA     #$2C    ,
7138 BD   41DB     02100          JSR     PRTCHA
713B 86   53       02110          LDA     #$53    S
713D BD   41DB     02120          JSR     PRTCHA
7140 86   20       02130          LDA     #$20    SPACE
7142 BD   41DB     02140          JSR     PRTCHA
7145 1F   40       02150          TFR     S,D
7147 C3   0007     02160          ADDD    #$0007
714A BD   4178     02170          JSR     PUTWRA  ADDRESS
714D 86   20       02180          LDA     #$20    SPACE
714F BD   41DB     02190          JSR     PRTCHA
7152 A6   67       02200          LDA     7,S
7154 BD   409E     02210          JSR     PUTBYA  VALUE
7157 BD   40D7     02220          JSR     CRLF
                   02230
                   02240 * STACK LOCATION 6
715A 86   20       02250          LDA     #$20    SPACE
715C BD   41DB     02260          JSR     PRTCHA
715F 86   36       02270          LDA     #$36    6
7161 BD   41DB     02280          JSR     PRTCHA
7164 86   2C       02290          LDA     #$2C    ,
7166 BD   41DB     02300          JSR     PRTCHA
7169 86   53       02310          LDA     #$53    S
716B BD   41DB     02320          JSR     PRTCHA
716E 86   20       02330          LDA     #$20    SPACE
7170 BD   41DB     02340          JSR     PRTCHA
```

```
7173 1F   40      02350         TFR     S,D
7175 C3   0006    02360         ADDD    #$0006
7178 BD   4178    02370         JSR     PUTWRA  ADDRESS
717B 86   20      02380         LDA     #$20    SPACE
717D BD   41DB    02390         JSR     PRTCHA
7180 A6   66      02400         LDA     6,S
7182 BD   409E    02410         JSR     PUTBYA  VALUE
7185 BD   40D7    02420         JSR     CRLF
                  02430
                  02440 * STACK LOCATION 5
7188 86   20      02450         LDA     #$20    SPACE
718A BD   41DB    02460         JSR     PRTCHA
718D 86   35      02470         LDA     #$35    5
718F BD   41DB    02480         JSR     PRTCHA
7192 86   2C      02490         LDA     #$2C    ,
7194 BD   41DB    02500         JSR     PRTCHA
7197 86   53      02510         LDA     #$53    S
7199 BD   41DB    02520         JSR     PRTCHA
719C 86   20      02530         LDA     #$20    SPACE
719E BD   41DB    02540         JSR     PRTCHA
71A1 1F   40      02550         TFR     S,D
71A3 C3   0005    02560         ADDD    #$0005
71A6 BD   4178    02570         JSR     PUTWRA  ADDRESS
71A9 86   20      02580         LDA     #$20    SPACE
71AB BD   41DB    02590         JSR     PRTCHA
71AE A6   65      02600         LDA     5,S
71B0 BD   409E    02610         JSR     PUTBYA  VALUE
71B3 BD   40D7    02620         JSR     CRLF
                  02630
                  02640 * STACK LOCATION 4
71B6 86   20      02650         LDA     #$20    SPACE
71B8 BD   41DB    02660         JSR     PRTCHA
71BB 86   34      02670         LDA     #$34    4
71BD BD   41DB    02680         JSR     PRTCHA
71C0 86   2C      02690         LDA     #$2C    ,
71C2 BD   41DB    02700         JSR     PRTCHA
71C5 86   53      02710         LDA     #$53    S
71C7 BD   41DB    02720         JSR     PRTCHA
71CA 86   20      02730         LDA     #$20    SPACE
71CC BD   41DB    02740         JSR     PRTCHA
71CF 1F   40      02750         TFR     S,D
71D1 C3   0004    02760         ADDD    #$0004
71D4 BD   4178    02770         JSR     PUTWRA  ADDRESS
71D7 86   20      02780         LDA     #$20    SPACE
71D9 BD   41DB    02790         JSR     PRTCHA
71DC A6   64      02800         LDA     4,S
71DE BD   409E    02810         JSR     PUTBYA  VALUE
```

```
71E1 BD   40D7     02820          JSR     CRLF
                   02830
                   02840 * STACK LOCATION 3
71E4 86   20       02850          LDA     #$20    SPACE
71E6 BD   41DB     02860          JSR     PRTCHA
71E9 86   33       02870          LDA     #$33    3
71EB BD   41DB     02880          JSR     PRTCHA
71EE 86   2C       02890          LDA     #$2C    ,
71F0 BD   41DB     02900          JSR     PRTCHA
71F3 86   53       02910          LDA     #$53    S
71F5 BD   41DB     02920          JSR     PRTCHA
71F8 86   20       02930          LDA     #$20    SPACE
71FA BD   41DB     02940          JSR     PRTCHA
71FD 1F   40       02950          TFR     S,D
71FF C3   0003     02960          ADDD    #$0003
7202 BD   4178     02970          JSR     PUTWRA  ADDRESS
7205 86   20       02980          LDA     #$20    SPACE
7207 BD   41DB     02990          JSR     PRTCHA
720A A6   63       03000          LDA     3,S
720C BD   409E     03010          JSR     PUTBYA  VALUE
720F BD   40D7     03020          JSR     CRLF
                   03030
                   03040 * STACK LOCATION 2
7212 86   20       03050          LDA     #$20    SPACE
7214 BD   41DB     03060          JSR     PRTCHA
7217 86   32       03070          LDA     #$32    2
7219 BD   41DB     03080          JSR     PRTCHA
721C 86   2C       03090          LDA     #$2C    ,
721E BD   41DB     03100          JSR     PRTCHA
7221 86   53       03110          LDA     #$53    S
7223 BD   41DB     03120          JSR     PRTCHA
7226 86   20       03130          LDA     #$20    SPACE
7228 BD   41DB     03140          JSR     PRTCHA
722B 1F   40       03150          TFR     S,D
722D C3   0002     03160          ADDD    #$0002
7230 BD   4178     03170          JSR     PUTWRA  ADDRESS
7233 86   20       03180          LDA     #$20    SPACE
7235 BD   41DB     03190          JSR     PRTCHA
7238 A6   62       03200          LDA     2,S
723A BD   409E     03210          JSR     PUTBYA  VALUE
723D BD   40D7     03220          JSR     CRLF
                   03230
                   03240 * STACK LOCATION 1
7240 86   20       03250          LDA     #$20    SPACE
7242 BD   41DB     03260          JSR     PRTCHA
7245 86   31       03270          LDA     #$31    1
7247 BD   41DB     03280          JSR     PRTCHA
```

```
724A 86   2C       03290        LDA     #$2C    ,
724C BD   41DB     03300        JSR     PRTCHA
724F 86   53       03310        LDA     #$53    S
7251 BD   41DB     03320        JSR     PRTCHA
7254 86   20       03330        LDA     #$20    SPACE
7256 BD   41DB     03340        JSR     PRTCHA
7259 1F   40       03350        TFR     S,D
725B C3   0001     03360        ADDD    #$0001
725E BD   4178     03370        JSR     PUTWRA  ADDRESS
7261 86   20       03380        LDA     #$20    SPACE
7263 BD   41DB     03390        JSR     PRTCHA
7266 A6   61       03400        LDA     1,S
7268 BD   409E     03410        JSR     PUTBYA  VALUE
726B BD   40D7     03420        JSR     CRLF
                   03430
                   03440 * STACK LOCATION 0
726E 86   20       03450        LDA     #$20    SPACE
7270 BD   41DB     03460        JSR     PRTCHA
7273 86   20       03470        LDA     #$20    SPACE
7275 BD   41DB     03480        JSR     PRTCHA
7278 86   2C       03490        LDA     #$2C    ,
727A BD   41DB     03500        JSR     PRTCHA
727D 86   53       03510        LDA     #$53    S
727F BD   41DB     03520        JSR     PRTCHA
7282 86   20       03530        LDA     #$20    SPACE
7284 BD   41DB     03540        JSR     PRTCHA
7287 1F   40       03550        TFR     S,D
7289 BD   4178     03560        JSR     PUTWRA  ADDRESS
728C 86   20       03570        LDA     #$20    SPACE
728E BD   41DB     03580        JSR     PRTCHA
7291 A6   E4       03590        LDA     ,S
7293 BD   409E     03600        JSR     PUTBYA  VALUE
7296 BD   40D7     03610        JSR     CRLF
                   03620
                   03630 * RESTORE REGISTERS
7299 B6   4000     03640        LDA     REGA
729C F6   4001     03650        LDB     REGB
729F BE   4002     03660        LDX     REGX
                   03670
                   03680 * EXIT
72A2 39            03690        RTS
                   03700
                   03710 *****
                   03720 *
                   03730 * END OF SUBROUTINE
                   03740 *
                   03750 *****
```

```
                      
                      * LEAVE THE TEST
                      
                      * RESTORE THE STACK POINTER
72A3 10FE 4006        LBL001   LDS     REGS
                      
                      * RESTORE THE REGISTERS EXCEPT S
                      * FROM THE STACK
72A7 35   7F                   PULS A,B,X,Y,U,DP,CC
                      
72A9 39                        RTS
                      
          0000                 END
```



Test No. 2: THE SECOND TEST SUITE

```
                    00100 *****
                    00110 *
                    00120 * STKTST2.ASM
                    00130 * MDJ 2023/02/02
                    00140 *
                    00150 * A TESTING TOOL TO
                    00160 * HELP LEARN ABOUT
                    00170 * STACK OPERATIONS
                    00180 *
                    00190 *****
                    00200
                    00210 * BASIC/ML TRANSFER
                    00220 * VARIABLES
          4000      00230 REGA    EQU     $4000
          4001      00240 REGB    EQU     $4001
          4002      00250 REGX    EQU     $4002
          4004      00260 REGY    EQU     $4004
          4006      00270 REGS    EQU     $4006
          4008      00280 REGU    EQU     $4008
          400A      00290 REGPC   EQU     $400A
          400C      00300 REGDP   EQU     $400C
          400D      00310 REGCC   EQU     $400D
                    00320
                    00330 * EXTERNAL ROUTINE
                    00340 * ADDRESSES
          409E      00350 PUTBYA  EQU     $409E
          40D7      00360 CRLF    EQU     $40D7
          4178      00370 PUTWRA  EQU     $4178
          41DB      00380 PRTCHA  EQU     $41DB
                    00390
                    00400 * PUT THIS UP IN HIGH
                    00410 * MEMORY SO IT WON'T
                    00420 * INTERFERE WITH
                    00430 * ANYTHING ELSE
7000                00440         ORG     $7000
                    00450
                    00460 * SAVE THE REGISTERS EXCEPT S
                    00470 * ON THE STACK
7000 34   7F        00480         PSHS A,B,X,Y,U,DP,CC
                    00490
                    00500 * SAVE THE STACK POINTER
                    00510 * SO WE CAN MOVE IT FOR
                    00520 * TESTING PURPOSES
7002 10FF 4006      00530         STS     REGS
                    00540
```

```
                   00550 * POINT STACK POINTER
                   00560 * TO TOP OF RAM
7006 10CE 7FFF     00570        LDS     #$7FFF
                   00580
                   00590 * PLACE A SENTINEL AT
                   00600 * THE TOP OF RAM
700A 86   FF       00610        LDA     #$FF
700C B7   7FFF     00620        STA     $7FFF
                   00630
                   00640 * SETUP FIRST TEST SUITE
                   00650 *        LDA     #$00
                   00660 *        PSHS    A
                   00670 *        LDA     #$01
                   00680 *        PSHS    A
                   00690 *        LDA     #$02
                   00700 *        PSHS    A
                   00710 *        LDA     #$03
                   00720 *        PSHS    A
                   00730 *        LDA     #$04
                   00740 *        PSHS    A
                   00750 *        LDA     #$05
                   00760 *        PSHS    A
                   00770 *        LDA     #$06
                   00780 *        PSHS    A
                   00790 *        LDA     #$07
                   00800 *        PSHS    A
                   00810 *        LDA     #$08
                   00820 *        PSHS    A
                   00830 *        LDA     #$09
                   00840 *        PSHS    A
                   00850
                   00860 * DISPLAY THE FIRST TEST SUITE
                   00870 *        JSR     STKDSP
                   00880
                   00890 *        JMP     LBL001
                   00900
                   00910 * SETUP SECOND TEST SUITE
700F 8E   0001     00920        LDX     #$0001
7012 34   10       00930        PSHS    X
7014 8E   0203     00940        LDX     #$0203
7017 34   10       00950        PSHS    X
7019 8E   0405     00960        LDX     #$0405
701C 34   10       00970        PSHS    X
701E 8E   0607     00980        LDX     #$0607
7021 34   10       00990        PSHS    X
7023 8E   0809     01000        LDX     #$0809
7026 34   10       01010        PSHS    X
```

```
                    01020
                    01030 * DISPLAY THE SECOND TEST SUITE
7028 BD   702E      01040        JSR     STKDSP
                    01050
702B 7E   7294      01060        JMP     LBL001
                    01070
                    01080 *****
                    01090 *
                    01100 * STKDSP
                    01110 * SUBROUTINE
                    01120 * TO DISPLAY THE STACK
                    01130 *
                    01140 *****
                    01150
                    01160 * SAVE REGISTERS
702E B7   4000      01170 STKDSP STA     REGA
7031 F7   4001      01180        STB     REGB
7034 BF   4002      01190        STX     REGX
                    01200
                    01210 * STACK LOCATION 12
7037 86   31        01220        LDA     #$31    1
7039 BD   41DB      01230        JSR     PRTCHA
703C 86   32        01240        LDA     #$32    2
703E BD   41DB      01250        JSR     PRTCHA
7041 86   2C        01260        LDA     #$2C    ,
7043 BD   41DB      01270        JSR     PRTCHA
7046 86   53        01280        LDA     #$53    S
7048 BD   41DB      01290        JSR     PRTCHA
704B 86   20        01300        LDA     #$20    SPACE
704D BD   41DB      01310        JSR     PRTCHA
7050 1F   40        01320        TFR     S,D
7052 C3   000C      01330        ADDD    #$000C
7055 BD   4178      01340        JSR     PUTWRA  ADDRESS
7058 86   20        01350        LDA     #$20    SPACE
705A BD   41DB      01360        JSR     PRTCHA
705D A6   6C        01370        LDA     12,S
705F BD   409E      01380        JSR     PUTBYA  VALUE
7062 BD   40D7      01390        JSR     CRLF
                    01400
                    01410 * STACK LOCATION 11
7065 86   31        01420        LDA     #$31    1
7067 BD   41DB      01430        JSR     PRTCHA
706A 86   31        01440        LDA     #$31    1
706C BD   41DB      01450        JSR     PRTCHA
706F 86   2C        01460        LDA     #$2C    ,
7071 BD   41DB      01470        JSR     PRTCHA
7074 86   53        01480        LDA     #$53    S
```

```
7076 BD   41DB     01490          JSR      PRTCHA
7079 86   20       01500          LDA      #$20    SPACE
707B BD   41DB     01510          JSR      PRTCHA
707E 1F   40       01520          TFR      S,D
7080 C3   000B     01530          ADDD     #$000B
7083 BD   4178     01540          JSR      PUTWRA  ADDRESS
7086 86   20       01550          LDA      #$20    SPACE
7088 BD   41DB     01560          JSR      PRTCHA
708B A6   6B       01570          LDA      11,S
708D BD   409E     01580          JSR      PUTBYA  VALUE
7090 BD   40D7     01590          JSR      CRLF
                   01600
                   01610 * STACK LOCATION 10
7093 86   31       01620          LDA      #$31    1
7095 BD   41DB     01630          JSR      PRTCHA
7098 86   30       01640          LDA      #$30    0
709A BD   41DB     01650          JSR      PRTCHA
709D 86   2C       01660          LDA      #$2C    ,
709F BD   41DB     01670          JSR      PRTCHA
70A2 86   53       01680          LDA      #$53    S
70A4 BD   41DB     01690          JSR      PRTCHA
70A7 86   20       01700          LDA      #$20    SPACE
70A9 BD   41DB     01710          JSR      PRTCHA
70AC 1F   40       01720          TFR      S,D
70AE C3   000A     01730          ADDD     #$000A
70B1 BD   4178     01740          JSR      PUTWRA  ADDRESS
70B4 86   20       01750          LDA      #$20    SPACE
70B6 BD   41DB     01760          JSR      PRTCHA
70B9 A6   6A       01770          LDA      10,S
70BB BD   409E     01780          JSR      PUTBYA  VALUE
70BE BD   40D7     01790          JSR      CRLF
                   01800
                   01810 * STACK LOCATION 9
70C1 86   20       01820          LDA      #$20    SPACE
70C3 BD   41DB     01830          JSR      PRTCHA
70C6 86   39       01840          LDA      #$39    9
70C8 BD   41DB     01850          JSR      PRTCHA
70CB 86   2C       01860          LDA      #$2C    ,
70CD BD   41DB     01870          JSR      PRTCHA
70D0 86   53       01880          LDA      #$53    S
70D2 BD   41DB     01890          JSR      PRTCHA
70D5 86   20       01900          LDA      #$20    SPACE
70D7 BD   41DB     01910          JSR      PRTCHA
70DA 1F   40       01920          TFR      S,D
70DC C3   0009     01930          ADDD     #$0009
70DF BD   4178     01940          JSR      PUTWRA  ADDRESS
70E2 86   20       01950          LDA      #$20    SPACE
```

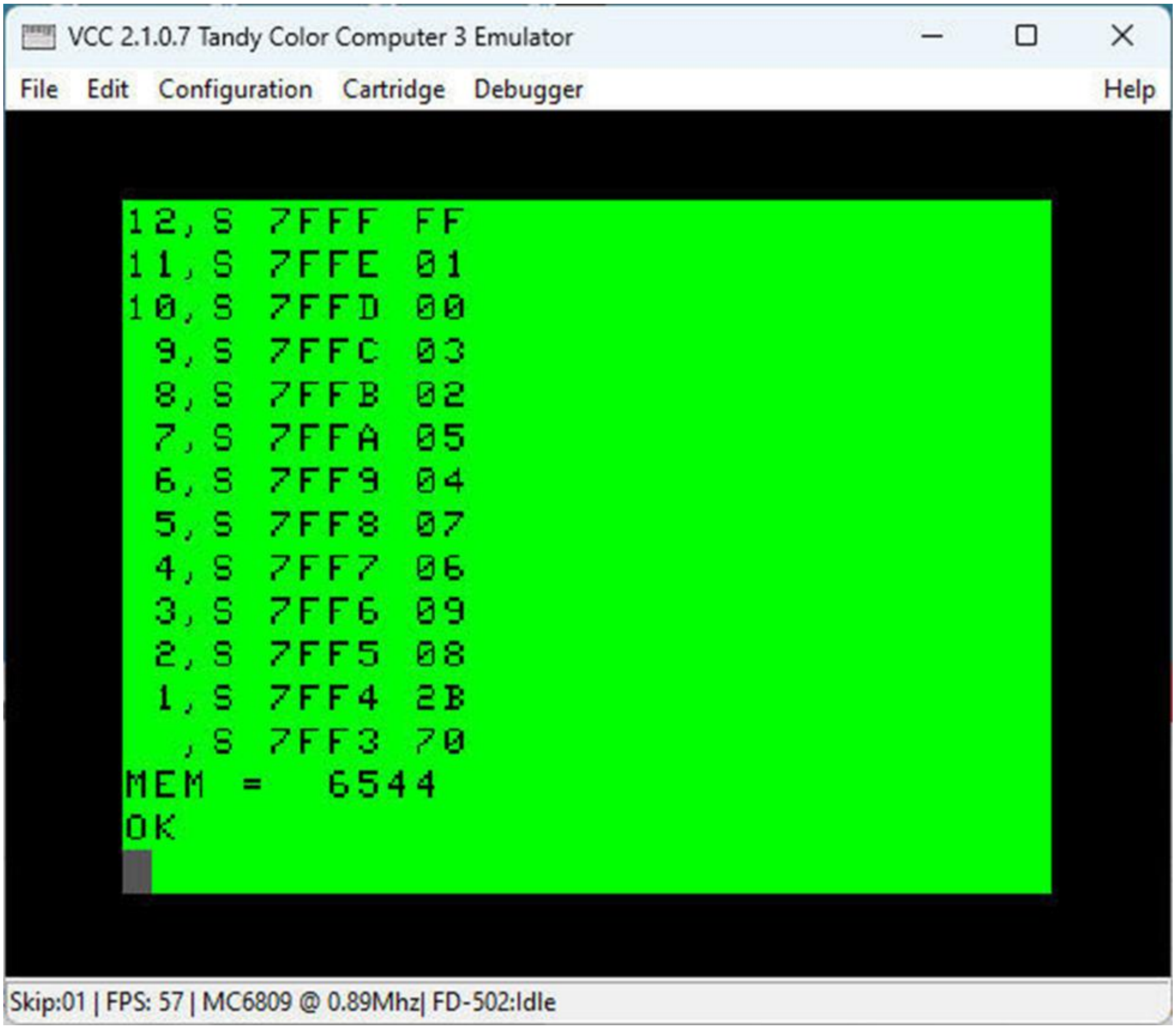
```
70E4 BD   41DB     01960          JSR     PRTCHA
70E7 A6   69       01970          LDA     9,S
70E9 BD   409E     01980          JSR     PUTBYA  VALUE
70EC BD   40D7     01990          JSR     CRLF
                   02000
                   02010 * STACK LOCATION 8
70EF 86   20       02020          LDA     #$20    SPACE
70F1 BD   41DB     02030          JSR     PRTCHA
70F4 86   38       02040          LDA     #$38    8
70F6 BD   41DB     02050          JSR     PRTCHA
70F9 86   2C       02060          LDA     #$2C    ,
70FB BD   41DB     02070          JSR     PRTCHA
70FE 86   53       02080          LDA     #$53    S
7100 BD   41DB     02090          JSR     PRTCHA
7103 86   20       02100          LDA     #$20    SPACE
7105 BD   41DB     02110          JSR     PRTCHA
7108 1F   40       02120          TFR     S,D
710A C3   0008     02130          ADDD    #$0008
710D BD   4178     02140          JSR     PUTWRA  ADDRESS
7110 86   20       02150          LDA     #$20    SPACE
7112 BD   41DB     02160          JSR     PRTCHA
7115 A6   68       02170          LDA     8,S
7117 BD   409E     02180          JSR     PUTBYA  VALUE
711A BD   40D7     02190          JSR     CRLF
                   02200
                   02210 * STACK LOCATION 7
711D 86   20       02220          LDA     #$20    SPACE
711F BD   41DB     02230          JSR     PRTCHA
7122 86   37       02240          LDA     #$37    7
7124 BD   41DB     02250          JSR     PRTCHA
7127 86   2C       02260          LDA     #$2C    ,
7129 BD   41DB     02270          JSR     PRTCHA
712C 86   53       02280          LDA     #$53    S
712E BD   41DB     02290          JSR     PRTCHA
7131 86   20       02300          LDA     #$20    SPACE
7133 BD   41DB     02310          JSR     PRTCHA
7136 1F   40       02320          TFR     S,D
7138 C3   0007     02330          ADDD    #$0007
713B BD   4178     02340          JSR     PUTWRA  ADDRESS
713E 86   20       02350          LDA     #$20    SPACE
7140 BD   41DB     02360          JSR     PRTCHA
7143 A6   67       02370          LDA     7,S
7145 BD   409E     02380          JSR     PUTBYA  VALUE
7148 BD   40D7     02390          JSR     CRLF
                   02400
                   02410 * STACK LOCATION 6
714B 86   20       02420          LDA     #$20    SPACE
```

```
714D BD   41DB     02430          JSR     PRTCHA
7150 86   36       02440          LDA     #$36    6
7152 BD   41DB     02450          JSR     PRTCHA
7155 86   2C       02460          LDA     #$2C    ,
7157 BD   41DB     02470          JSR     PRTCHA
715A 86   53       02480          LDA     #$53    S
715C BD   41DB     02490          JSR     PRTCHA
715F 86   20       02500          LDA     #$20    SPACE
7161 BD   41DB     02510          JSR     PRTCHA
7164 1F   40       02520          TFR     S,D
7166 C3   0006     02530          ADDD    #$0006
7169 BD   4178     02540          JSR     PUTWRA  ADDRESS
716C 86   20       02550          LDA     #$20    SPACE
716E BD   41DB     02560          JSR     PRTCHA
7171 A6   66       02570          LDA     6,S
7173 BD   409E     02580          JSR     PUTBYA  VALUE
7176 BD   40D7     02590          JSR     CRLF
                   02600
                   02610 * STACK LOCATION 5
7179 86   20       02620          LDA     #$20    SPACE
717B BD   41DB     02630          JSR     PRTCHA
717E 86   35       02640          LDA     #$35    5
7180 BD   41DB     02650          JSR     PRTCHA
7183 86   2C       02660          LDA     #$2C    ,
7185 BD   41DB     02670          JSR     PRTCHA
7188 86   53       02680          LDA     #$53    S
718A BD   41DB     02690          JSR     PRTCHA
718D 86   20       02700          LDA     #$20    SPACE
718F BD   41DB     02710          JSR     PRTCHA
7192 1F   40       02720          TFR     S,D
7194 C3   0005     02730          ADDD    #$0005
7197 BD   4178     02740          JSR     PUTWRA  ADDRESS
719A 86   20       02750          LDA     #$20    SPACE
719C BD   41DB     02760          JSR     PRTCHA
719F A6   65       02770          LDA     5,S
71A1 BD   409E     02780          JSR     PUTBYA  VALUE
71A4 BD   40D7     02790          JSR     CRLF
                   02800
                   02810 * STACK LOCATION 4
71A7 86   20       02820          LDA     #$20    SPACE
71A9 BD   41DB     02830          JSR     PRTCHA
71AC 86   34       02840          LDA     #$34    4
71AE BD   41DB     02850          JSR     PRTCHA
71B1 86   2C       02860          LDA     #$2C    ,
71B3 BD   41DB     02870          JSR     PRTCHA
71B6 86   53       02880          LDA     #$53    S
71B8 BD   41DB     02890          JSR     PRTCHA
```

```
71BB 86   20        02900          LDA    #$20    SPACE
71BD BD   41DB      02910          JSR    PRTCHA
71C0 1F   40        02920          TFR    S,D
71C2 C3   0004      02930          ADDD   #$0004
71C5 BD   4178      02940          JSR    PUTWRA  ADDRESS
71C8 86   20        02950          LDA    #$20    SPACE
71CA BD   41DB      02960          JSR    PRTCHA
71CD A6   64        02970          LDA    4,S
71CF BD   409E      02980          JSR    PUTBYA  VALUE
71D2 BD   40D7      02990          JSR    CRLF
                    03000
                    03010 * STACK LOCATION 3
71D5 86   20        03020          LDA    #$20    SPACE
71D7 BD   41DB      03030          JSR    PRTCHA
71DA 86   33        03040          LDA    #$33    3
71DC BD   41DB      03050          JSR    PRTCHA
71DF 86   2C        03060          LDA    #$2C    ,
71E1 BD   41DB      03070          JSR    PRTCHA
71E4 86   53        03080          LDA    #$53    S
71E6 BD   41DB      03090          JSR    PRTCHA
71E9 86   20        03100          LDA    #$20    SPACE
71EB BD   41DB      03110          JSR    PRTCHA
71EE 1F   40        03120          TFR    S,D
71F0 C3   0003      03130          ADDD   #$0003
71F3 BD   4178      03140          JSR    PUTWRA  ADDRESS
71F6 86   20        03150          LDA    #$20    SPACE
71F8 BD   41DB      03160          JSR    PRTCHA
71FB A6   63        03170          LDA    3,S
71FD BD   409E      03180          JSR    PUTBYA  VALUE
7200 BD   40D7      03190          JSR    CRLF
                    03200
                    03210 * STACK LOCATION 2
7203 86   20        03220          LDA    #$20    SPACE
7205 BD   41DB      03230          JSR    PRTCHA
7208 86   32        03240          LDA    #$32    2
720A BD   41DB      03250          JSR    PRTCHA
720D 86   2C        03260          LDA    #$2C    ,
720F BD   41DB      03270          JSR    PRTCHA
7212 86   53        03280          LDA    #$53    S
7214 BD   41DB      03290          JSR    PRTCHA
7217 86   20        03300          LDA    #$20    SPACE
7219 BD   41DB      03310          JSR    PRTCHA
721C 1F   40        03320          TFR    S,D
721E C3   0002      03330          ADDD   #$0002
7221 BD   4178      03340          JSR    PUTWRA  ADDRESS
7224 86   20        03350          LDA    #$20    SPACE
7226 BD   41DB      03360          JSR    PRTCHA
```

```
7229 A6   62         03370          LDA      2,S
722B BD   409E       03380          JSR      PUTBYA  VALUE
722E BD   40D7       03390          JSR      CRLF
                     03400
                     03410 * STACK LOCATION 1
7231 86   20         03420          LDA      #$20    SPACE
7233 BD   41DB       03430          JSR      PRTCHA
7236 86   31         03440          LDA      #$31    1
7238 BD   41DB       03450          JSR      PRTCHA
723B 86   2C         03460          LDA      #$2C    ,
723D BD   41DB       03470          JSR      PRTCHA
7240 86   53         03480          LDA      #$53    S
7242 BD   41DB       03490          JSR      PRTCHA
7245 86   20         03500          LDA      #$20    SPACE
7247 BD   41DB       03510          JSR      PRTCHA
724A 1F   40         03520          TFR      S,D
724C C3   0001       03530          ADDD     #$0001
724F BD   4178       03540          JSR      PUTWRA  ADDRESS
7252 86   20         03550          LDA      #$20    SPACE
7254 BD   41DB       03560          JSR      PRTCHA
7257 A6   61         03570          LDA      1,S
7259 BD   409E       03580          JSR      PUTBYA  VALUE
725C BD   40D7       03590          JSR      CRLF
                     03600
                     03610 * STACK LOCATION 0
725F 86   20         03620          LDA      #$20    SPACE
7261 BD   41DB       03630          JSR      PRTCHA
7264 86   20         03640          LDA      #$20    SPACE
7266 BD   41DB       03650          JSR      PRTCHA
7269 86   2C         03660          LDA      #$2C    ,
726B BD   41DB       03670          JSR      PRTCHA
726E 86   53         03680          LDA      #$53    S
7270 BD   41DB       03690          JSR      PRTCHA
7273 86   20         03700          LDA      #$20    SPACE
7275 BD   41DB       03710          JSR      PRTCHA
7278 1F   40         03720          TFR      S,D
727A BD   4178       03730          JSR      PUTWRA  ADDRESS
727D 86   20         03740          LDA      #$20    SPACE
727F BD   41DB       03750          JSR      PRTCHA
7282 A6   E4         03760          LDA      ,S
7284 BD   409E       03770          JSR      PUTBYA  VALUE
7287 BD   40D7       03780          JSR      CRLF
                     03790
                     03800 * RESTORE REGISTERS
728A B6   4000       03810          LDA      REGA
728D F6   4001       03820          LDB      REGB
7290 BE   4002       03830          LDX      REGX
```

```
                    03840
                    03850 * EXIT
7293 39             03860         RTS
                    03870
                    03880 *****
                    03890 *
                    03900 * END OF SUBROUTINE
                    03910 *
                    03920 *****
                    03930
                    03940 * LEAVE THE TEST
                    03950
                    03960 * RESTORE THE STACK POINTER
7294 10FE 4006      03970 LBL001  LDS     REGS
                    03980
                    03990 * RESTORE THE REGISTERS EXCEPT S
                    04000 * FROM THE STACK
7298 35   7F        04010         PULS A,B,X,Y,U,DP,CC
                    04020
729A 39             04030         RTS
                    04040
          0000      04050         END
```


VCC 2.1.0.7 Tandy Color Computer 3 Emulator
File Edit Configuration Cartridge Debugger Help
12,S 7FFF FF
11,S 7FFE 01
10,S 7FFD 00
9,S 7FFC 03
8,S 7FFB 02
7,S 7FFA 05
6,S 7FF9 04
5,S 7FF8 07
4,S 7FF7 06
3,S 7FF6 09
2,S 7FF5 08
1,S 7FF4 2B
,S 7FF3 70
MEM = 6544
OK
Skip:01 | FPS: 57 | MC6809 @ 0.89Mhz| FD-502:Idle

Test No. 3: THE SECOND TEST SUITE CHECK

```
                    00100 *****
                    00110 *
                    00120 * STKTST2C.ASM
                    00130 * MDJ 2023/02/02
                    00140 *
                    00150 * A TESTING TOOL TO
                    00160 * HELP LEARN ABOUT
                    00170 * STACK OPERATIONS
                    00180 *
                    00190 *****
                    00200
                    00210 * BASIC/ML TRANSFER
                    00220 * VARIABLES
          4000      00230 REGA    EQU     $4000
          4001      00240 REGB    EQU     $4001
          4002      00250 REGX    EQU     $4002
          4004      00260 REGY    EQU     $4004
          4006      00270 REGS    EQU     $4006
          4008      00280 REGU    EQU     $4008
          400A      00290 REGPC   EQU     $400A
          400C      00300 REGDP   EQU     $400C
          400D      00310 REGCC   EQU     $400D
                    00320
                    00330 * EXTERNAL ROUTINE
                    00340 * ADDRESSES
          409E      00350 PUTBYA  EQU     $409E
          40D7      00360 CRLF    EQU     $40D7
          4178      00370 PUTWRA  EQU     $4178
          41DB      00380 PRTCHA  EQU     $41DB
                    00390
                    00400 * PUT THIS UP IN HIGH
                    00410 * MEMORY SO IT WON'T
                    00420 * INTERFERE WITH
                    00430 * ANYTHING ELSE
7000                00440         ORG     $7000
                    00450
                    00460 * SAVE THE REGISTERS EXCEPT S
                    00470 * ON THE STACK
7000 34   7F        00480         PSHS A,B,X,Y,U,DP,CC
                    00490
                    00500 * SAVE THE STACK POINTER
                    00510 * SO WE CAN MOVE IT FOR
                    00520 * TESTING PURPOSES
7002 10FF 4006      00530         STS     REGS
                    00540
```

```
                   00550 * POINT STACK POINTER
                   00560 * TO TOP OF RAM
7006 10CE 7FFF     00570        LDS     #$7FFF
                   00580
                   00590 * PLACE A SENTINEL AT
                   00600 * THE TOP OF RAM
700A 86   FF       00610        LDA     #$FF
700C B7   7FFF     00620        STA     $7FFF
                   00630
                   00640 * SETUP FIRST TEST SUITE
                   00650 *       LDA     #$00
                   00660 *       PSHS    A
                   00670 *       LDA     #$01
                   00680 *       PSHS    A
                   00690 *       LDA     #$02
                   00700 *       PSHS    A
                   00710 *       LDA     #$03
                   00720 *       PSHS    A
                   00730 *       LDA     #$04
                   00740 *       PSHS    A
                   00750 *       LDA     #$05
                   00760 *       PSHS    A
                   00770 *       LDA     #$06
                   00780 *       PSHS    A
                   00790 *       LDA     #$07
                   00800 *       PSHS    A
                   00810 *       LDA     #$08
                   00820 *       PSHS    A
                   00830 *       LDA     #$09
                   00840 *       PSHS    A
                   00850
                   00860 * DISPLAY THE FIRST TEST SUITE
                   00870 *       JSR     STKDSP
                   00880
                   00890 *       JMP     LBL001
                   00900
                   00910 * SETUP SECOND TEST SUITE
700F 8E   0001     00920        LDX     #$0001
7012 34   10       00930        PSHS    X
7014 8E   0203     00940        LDX     #$0203
7017 34   10       00950        PSHS    X
7019 8E   0405     00960        LDX     #$0405
701C 34   10       00970        PSHS    X
701E 8E   0607     00980        LDX     #$0607
7021 34   10       00990        PSHS    X
7023 8E   0809     01000        LDX     #$0809
7026 34   10       01010        PSHS    X
```

```
                    01020
                    01030 * DISPLAY THE SECOND TEST SUITE
7028 BD   7059      01040         JSR     STKDSP
                    01050
                    01060 * SECOND TEST SUITE CHECK
702B BD   40D7      01070         JSR     CRLF
702E EC   6A        01080         LDD     10,S
7030 BD   4178      01090         JSR     PUTWRA
7033 BD   40D7      01100         JSR     CRLF
7036 EC   68        01110         LDD     8,S
7038 BD   4178      01120         JSR     PUTWRA
703B BD   40D7      01130         JSR     CRLF
703E EC   66        01140         LDD     6,S
7040 BD   4178      01150         JSR     PUTWRA
7043 BD   40D7      01160         JSR     CRLF
7046 EC   64        01170         LDD     4,S
7048 BD   4178      01180         JSR     PUTWRA
704B BD   40D7      01190         JSR     CRLF
704E EC   62        01200         LDD     2,S
7050 BD   4178      01210         JSR     PUTWRA
7053 BD   40D7      01220         JSR     CRLF
                    01230
7056 7E   72BF      01240         JMP     LBL001
                    01250
                    01260 *****
                    01270 *
                    01280 * STKDSP
                    01290 * SUBROUTINE
                    01300 * TO DISPLAY THE STACK
                    01310 *
                    01320 *****
                    01330
                    01340 * SAVE REGISTERS
7059 B7   4000      01350 STKDSP  STA     REGA
705C F7   4001      01360         STB     REGB
705F BF   4002      01370         STX     REGX
                    01380
                    01390 * STACK LOCATION 12
7062 86   31        01400         LDA     #$31    1
7064 BD   41DB      01410         JSR     PRTCHA
7067 86   32        01420         LDA     #$32    2
7069 BD   41DB      01430         JSR     PRTCHA
706C 86   2C        01440         LDA     #$2C    ,
706E BD   41DB      01450         JSR     PRTCHA
7071 86   53        01460         LDA     #$53    S
7073 BD   41DB      01470         JSR     PRTCHA
7076 86   20        01480         LDA     #$20    SPACE
```

```
7078 BD   41DB     01490          JSR     PRTCHA
707B 1F   40       01500          TFR     S,D
707D C3   000C     01510          ADDD    #$000C
7080 BD   4178     01520          JSR     PUTWRA  ADDRESS
7083 86   20       01530          LDA     #$20    SPACE
7085 BD   41DB     01540          JSR     PRTCHA
7088 A6   6C       01550          LDA     12,S
708A BD   409E     01560          JSR     PUTBYA  VALUE
708D BD   40D7     01570          JSR     CRLF
                   01580
                   01590 * STACK LOCATION 11
7090 86   31       01600          LDA     #$31    1
7092 BD   41DB     01610          JSR     PRTCHA
7095 86   31       01620          LDA     #$31    1
7097 BD   41DB     01630          JSR     PRTCHA
709A 86   2C       01640          LDA     #$2C    ,
709C BD   41DB     01650          JSR     PRTCHA
709F 86   53       01660          LDA     #$53    S
70A1 BD   41DB     01670          JSR     PRTCHA
70A4 86   20       01680          LDA     #$20    SPACE
70A6 BD   41DB     01690          JSR     PRTCHA
70A9 1F   40       01700          TFR     S,D
70AB C3   000B     01710          ADDD    #$000B
70AE BD   4178     01720          JSR     PUTWRA  ADDRESS
70B1 86   20       01730          LDA     #$20    SPACE
70B3 BD   41DB     01740          JSR     PRTCHA
70B6 A6   6B       01750          LDA     11,S
70B8 BD   409E     01760          JSR     PUTBYA  VALUE
70BB BD   40D7     01770          JSR     CRLF
                   01780
                   01790 * STACK LOCATION 10
70BE 86   31       01800          LDA     #$31    1
70C0 BD   41DB     01810          JSR     PRTCHA
70C3 86   30       01820          LDA     #$30    0
70C5 BD   41DB     01830          JSR     PRTCHA
70C8 86   2C       01840          LDA     #$2C    ,
70CA BD   41DB     01850          JSR     PRTCHA
70CD 86   53       01860          LDA     #$53    S
70CF BD   41DB     01870          JSR     PRTCHA
70D2 86   20       01880          LDA     #$20    SPACE
70D4 BD   41DB     01890          JSR     PRTCHA
70D7 1F   40       01900          TFR     S,D
70D9 C3   000A     01910          ADDD    #$000A
70DC BD   4178     01920          JSR     PUTWRA  ADDRESS
70DF 86   20       01930          LDA     #$20    SPACE
70E1 BD   41DB     01940          JSR     PRTCHA
70E4 A6   6A       01950          LDA     10,S
```

```
70E6 BD   409E     01960          JSR     PUTBYA  VALUE
70E9 BD   40D7     01970          JSR     CRLF
                   01980
                   01990 * STACK LOCATION 9
70EC 86   20       02000          LDA     #$20    SPACE
70EE BD   41DB     02010          JSR     PRTCHA
70F1 86   39       02020          LDA     #$39    9
70F3 BD   41DB     02030          JSR     PRTCHA
70F6 86   2C       02040          LDA     #$2C    ,
70F8 BD   41DB     02050          JSR     PRTCHA
70FB 86   53       02060          LDA     #$53    S
70FD BD   41DB     02070          JSR     PRTCHA
7100 86   20       02080          LDA     #$20    SPACE
7102 BD   41DB     02090          JSR     PRTCHA
7105 1F   40       02100          TFR     S,D
7107 C3   0009     02110          ADDD    #$0009
710A BD   4178     02120          JSR     PUTWRA  ADDRESS
710D 86   20       02130          LDA     #$20    SPACE
710F BD   41DB     02140          JSR     PRTCHA
7112 A6   69       02150          LDA     9,S
7114 BD   409E     02160          JSR     PUTBYA  VALUE
7117 BD   40D7     02170          JSR     CRLF
                   02180
                   02190 * STACK LOCATION 8
711A 86   20       02200          LDA     #$20    SPACE
711C BD   41DB     02210          JSR     PRTCHA
711F 86   38       02220          LDA     #$38    8
7121 BD   41DB     02230          JSR     PRTCHA
7124 86   2C       02240          LDA     #$2C    ,
7126 BD   41DB     02250          JSR     PRTCHA
7129 86   53       02260          LDA     #$53    S
712B BD   41DB     02270          JSR     PRTCHA
712E 86   20       02280          LDA     #$20    SPACE
7130 BD   41DB     02290          JSR     PRTCHA
7133 1F   40       02300          TFR     S,D
7135 C3   0008     02310          ADDD    #$0008
7138 BD   4178     02320          JSR     PUTWRA  ADDRESS
713B 86   20       02330          LDA     #$20    SPACE
713D BD   41DB     02340          JSR     PRTCHA
7140 A6   68       02350          LDA     8,S
7142 BD   409E     02360          JSR     PUTBYA  VALUE
7145 BD   40D7     02370          JSR     CRLF
                   02380
                   02390 * STACK LOCATION 7
7148 86   20       02400          LDA     #$20    SPACE
714A BD   41DB     02410          JSR     PRTCHA
714D 86   37       02420          LDA     #$37    7
```

```
714F BD   41DB     02430          JSR     PRTCHA
7152 86   2C       02440          LDA     #$2C    ,
7154 BD   41DB     02450          JSR     PRTCHA
7157 86   53       02460          LDA     #$53    S
7159 BD   41DB     02470          JSR     PRTCHA
715C 86   20       02480          LDA     #$20    SPACE
715E BD   41DB     02490          JSR     PRTCHA
7161 1F   40       02500          TFR     S,D
7163 C3   0007     02510          ADDD    #$0007
7166 BD   4178     02520          JSR     PUTWRA  ADDRESS
7169 86   20       02530          LDA     #$20    SPACE
716B BD   41DB     02540          JSR     PRTCHA
716E A6   67       02550          LDA     7,S
7170 BD   409E     02560          JSR     PUTBYA  VALUE
7173 BD   40D7     02570          JSR     CRLF
                   02580
                   02590 * STACK LOCATION 6
7176 86   20       02600          LDA     #$20    SPACE
7178 BD   41DB     02610          JSR     PRTCHA
717B 86   36       02620          LDA     #$36    6
717D BD   41DB     02630          JSR     PRTCHA
7180 86   2C       02640          LDA     #$2C    ,
7182 BD   41DB     02650          JSR     PRTCHA
7185 86   53       02660          LDA     #$53    S
7187 BD   41DB     02670          JSR     PRTCHA
718A 86   20       02680          LDA     #$20    SPACE
718C BD   41DB     02690          JSR     PRTCHA
718F 1F   40       02700          TFR     S,D
7191 C3   0006     02710          ADDD    #$0006
7194 BD   4178     02720          JSR     PUTWRA  ADDRESS
7197 86   20       02730          LDA     #$20    SPACE
7199 BD   41DB     02740          JSR     PRTCHA
719C A6   66       02750          LDA     6,S
719E BD   409E     02760          JSR     PUTBYA  VALUE
71A1 BD   40D7     02770          JSR     CRLF
                   02780
                   02790 * STACK LOCATION 5
71A4 86   20       02800          LDA     #$20    SPACE
71A6 BD   41DB     02810          JSR     PRTCHA
71A9 86   35       02820          LDA     #$35    5
71AB BD   41DB     02830          JSR     PRTCHA
71AE 86   2C       02840          LDA     #$2C    ,
71B0 BD   41DB     02850          JSR     PRTCHA
71B3 86   53       02860          LDA     #$53    S
71B5 BD   41DB     02870          JSR     PRTCHA
71B8 86   20       02880          LDA     #$20    SPACE
71BA BD   41DB     02890          JSR     PRTCHA
```

```
71BD 1F   40      02900          TFR     S,D
71BF C3   0005    02910          ADDD    #$0005
71C2 BD   4178    02920          JSR     PUTWRA  ADDRESS
71C5 86   20      02930          LDA     #$20    SPACE
71C7 BD   41DB    02940          JSR     PRTCHA
71CA A6   65      02950          LDA     5,S
71CC BD   409E    02960          JSR     PUTBYA  VALUE
71CF BD   40D7    02970          JSR     CRLF
                  02980
                  02990 * STACK LOCATION 4
71D2 86   20      03000          LDA     #$20    SPACE
71D4 BD   41DB    03010          JSR     PRTCHA
71D7 86   34      03020          LDA     #$34    4
71D9 BD   41DB    03030          JSR     PRTCHA
71DC 86   2C      03040          LDA     #$2C    ,
71DE BD   41DB    03050          JSR     PRTCHA
71E1 86   53      03060          LDA     #$53    S
71E3 BD   41DB    03070          JSR     PRTCHA
71E6 86   20      03080          LDA     #$20    SPACE
71E8 BD   41DB    03090          JSR     PRTCHA
71EB 1F   40      03100          TFR     S,D
71ED C3   0004    03110          ADDD    #$0004
71F0 BD   4178    03120          JSR     PUTWRA  ADDRESS
71F3 86   20      03130          LDA     #$20    SPACE
71F5 BD   41DB    03140          JSR     PRTCHA
71F8 A6   64      03150          LDA     4,S
71FA BD   409E    03160          JSR     PUTBYA  VALUE
71FD BD   40D7    03170          JSR     CRLF
                  03180
                  03190 * STACK LOCATION 3
7200 86   20      03200          LDA     #$20    SPACE
7202 BD   41DB    03210          JSR     PRTCHA
7205 86   33      03220          LDA     #$33    3
7207 BD   41DB    03230          JSR     PRTCHA
720A 86   2C      03240          LDA     #$2C    ,
720C BD   41DB    03250          JSR     PRTCHA
720F 86   53      03260          LDA     #$53    S
7211 BD   41DB    03270          JSR     PRTCHA
7214 86   20      03280          LDA     #$20    SPACE
7216 BD   41DB    03290          JSR     PRTCHA
7219 1F   40      03300          TFR     S,D
721B C3   0003    03310          ADDD    #$0003
721E BD   4178    03320          JSR     PUTWRA  ADDRESS
7221 86   20      03330          LDA     #$20    SPACE
7223 BD   41DB    03340          JSR     PRTCHA
7226 A6   63      03350          LDA     3,S
7228 BD   409E    03360          JSR     PUTBYA  VALUE
```

```
722B BD   40D7     03370          JSR     CRLF
                   03380
                   03390 * STACK LOCATION 2
722E 86   20       03400          LDA     #$20    SPACE
7230 BD   41DB     03410          JSR     PRTCHA
7233 86   32       03420          LDA     #$32    2
7235 BD   41DB     03430          JSR     PRTCHA
7238 86   2C       03440          LDA     #$2C    ,
723A BD   41DB     03450          JSR     PRTCHA
723D 86   53       03460          LDA     #$53    S
723F BD   41DB     03470          JSR     PRTCHA
7242 86   20       03480          LDA     #$20    SPACE
7244 BD   41DB     03490          JSR     PRTCHA
7247 1F   40       03500          TFR     S,D
7249 C3   0002     03510          ADDD    #$0002
724C BD   4178     03520          JSR     PUTWRA  ADDRESS
724F 86   20       03530          LDA     #$20    SPACE
7251 BD   41DB     03540          JSR     PRTCHA
7254 A6   62       03550          LDA     2,S
7256 BD   409E     03560          JSR     PUTBYA  VALUE
7259 BD   40D7     03570          JSR     CRLF
                   03580
                   03590 * STACK LOCATION 1
725C 86   20       03600          LDA     #$20    SPACE
725E BD   41DB     03610          JSR     PRTCHA
7261 86   31       03620          LDA     #$31    1
7263 BD   41DB     03630          JSR     PRTCHA
7266 86   2C       03640          LDA     #$2C    ,
7268 BD   41DB     03650          JSR     PRTCHA
726B 86   53       03660          LDA     #$53    S
726D BD   41DB     03670          JSR     PRTCHA
7270 86   20       03680          LDA     #$20    SPACE
7272 BD   41DB     03690          JSR     PRTCHA
7275 1F   40       03700          TFR     S,D
7277 C3   0001     03710          ADDD    #$0001
727A BD   4178     03720          JSR     PUTWRA  ADDRESS
727D 86   20       03730          LDA     #$20    SPACE
727F BD   41DB     03740          JSR     PRTCHA
7282 A6   61       03750          LDA     1,S
7284 BD   409E     03760          JSR     PUTBYA  VALUE
7287 BD   40D7     03770          JSR     CRLF
                   03780
                   03790 * STACK LOCATION 0
728A 86   20       03800          LDA     #$20    SPACE
728C BD   41DB     03810          JSR     PRTCHA
728F 86   20       03820          LDA     #$20    SPACE
7291 BD   41DB     03830          JSR     PRTCHA
```

```
7294 86   2C       03840        LDA    #$2C    ,
7296 BD   41DB     03850        JSR    PRTCHA
7299 86   53       03860        LDA    #$53    S
729B BD   41DB     03870        JSR    PRTCHA
729E 86   20       03880        LDA    #$20    SPACE
72A0 BD   41DB     03890        JSR    PRTCHA
72A3 1F   40       03900        TFR    S,D
72A5 BD   4178     03910        JSR    PUTWRA  ADDRESS
72A8 86   20       03920        LDA    #$20    SPACE
72AA BD   41DB     03930        JSR    PRTCHA
72AD A6   E4       03940        LDA    ,S
72AF BD   409E     03950        JSR    PUTBYA  VALUE
72B2 BD   40D7     03960        JSR    CRLF
                   03970
                   03980 * RESTORE REGISTERS
72B5 B6   4000     03990        LDA    REGA
72B8 F6   4001     04000        LDB    REGB
72BB BE   4002     04010        LDX    REGX
                   04020
                   04030 * EXIT
72BE 39            04040        RTS
                   04050
                   04060 *****
                   04070 *
                   04080 * END OF SUBROUTINE
                   04090 *
                   04100 *****
                   04110
                   04120 * LEAVE THE TEST
                   04130
                   04140 * RESTORE THE STACK POINTER
72BF 10FE 4006     04150 LBL001 LDS    REGS
                   04160
                   04170 * RESTORE THE REGISTERS EXCEPT S
                   04180 * FROM THE STACK
72C3 35   7F       04190        PULS A,B,X,Y,U,DP,CC
                   04200
72C5 39            04210        RTS
                   04220
          0000     04230        END
```


VCC 2.1.0.7 Tandy Color Computer 3 Emulator
File Edit Configuration Cartridge Debugger Help
6,S 7FF9 04
5,S 7FF8 07
4,S 7FF7 06
3,S 7FF6 09
2,S 7FF5 08
1,S 7FF4 2B
,S 7FF3 70
0001
0203
0405
0607
0809
MEM = 6544
OK
Skip:01 | FPS: 64 | MC6809 @ 0.89Mhz| FD-502:Idle

Test No. 4: THE THIRD TEST SUITE

```
                    00100 *****
                    00110 *
                    00120 * STKTST3.ASM
                    00130 * MDJ 2023/02/02
                    00140 *
                    00150 * A TESTING TOOL TO
                    00160 * HELP LEARN ABOUT
                    00170 * STACK OPERATIONS
                    00180 *
                    00190 *****
                    00200
                    00210 * BASIC/ML TRANSFER
                    00220 * VARIABLES
          4000      00230 REGA    EQU     $4000
          4001      00240 REGB    EQU     $4001
          4002      00250 REGX    EQU     $4002
          4004      00260 REGY    EQU     $4004
          4006      00270 REGS    EQU     $4006
          4008      00280 REGU    EQU     $4008
          400A      00290 REGPC   EQU     $400A
          400C      00300 REGDP   EQU     $400C
          400D      00310 REGCC   EQU     $400D
                    00320
                    00330 * EXTERNAL ROUTINE
                    00340 * ADDRESSES
          409E      00350 PUTBYA  EQU     $409E
          40D7      00360 CRLF    EQU     $40D7
          4178      00370 PUTWRA  EQU     $4178
          41DB      00380 PRTCHA  EQU     $41DB
                    00390
                    00400 * PUT THIS UP IN HIGH
                    00410 * MEMORY SO IT WON'T
                    00420 * INTERFERE WITH
                    00430 * ANYTHING ELSE
7000                00440         ORG     $7000
                    00450
                    00460 * SAVE THE REGISTERS EXCEPT S
                    00470 * ON THE STACK
7000 34   7F        00480         PSHS A,B,X,Y,U,DP,CC
                    00490
                    00500 * SAVE THE STACK POINTER
                    00510 * SO WE CAN MOVE IT FOR
                    00520 * TESTING PURPOSES
7002 10FF 4006      00530         STS     REGS
                    00540
```

```
                   00550 * POINT STACK POINTER
                   00560 * TO TOP OF RAM
7006 10CE 7FFF     00570         LDS     #$7FFF
                   00580
                   00590 * PLACE A SENTINEL AT
                   00600 * THE TOP OF RAM
700A 86   FF       00610         LDA     #$FF
700C B7   7FFF     00620         STA     $7FFF
                   00630
                   00640 * SETUP FIRST TEST SUITE
                   00650 *         LDA     #$00
                   00660 *         PSHS    A
                   00670 *         LDA     #$01
                   00680 *         PSHS    A
                   00690 *         LDA     #$02
                   00700 *         PSHS    A
                   00710 *         LDA     #$03
                   00720 *         PSHS    A
                   00730 *         LDA     #$04
                   00740 *         PSHS    A
                   00750 *         LDA     #$05
                   00760 *         PSHS    A
                   00770 *         LDA     #$06
                   00780 *         PSHS    A
                   00790 *         LDA     #$07
                   00800 *         PSHS    A
                   00810 *         LDA     #$08
                   00820 *         PSHS    A
                   00830 *         LDA     #$09
                   00840 *         PSHS    A
                   00850
                   00860 * DISPLAY THE FIRST TEST SUITE
                   00870 *         JSR     STKDSP
                   00880
                   00890 *         JMP     LBL001
                   00900
                   00910 * SETUP SECOND TEST SUITE
                   00920 *         LDX     #$0001
                   00930 *         PSHS    X
                   00940 *         LDX     #$0203
                   00950 *         PSHS    X
                   00960 *         LDX     #$0405
                   00970 *         PSHS    X
                   00980 *         LDX     #$0607
                   00990 *         PSHS    X
                   01000 *         LDX     #$0809
                   01010 *         PSHS    X
```

```
                    01020
                    01030 * DISPLAY THE SECOND TEST SUITE
                    01040 *        JSR     STKDSP
                    01050
                    01060 * SETUP THIRD TEST SUITE
700F 86   00        01070         LDA     #$00
7011 34   02        01080         PSHS    A
7013 8E   0102      01090         LDX     #$0102
7016 34   10        01100         PSHS    X
7018 86   03        01110         LDA     #$03
701A 34   02        01120         PSHS    A
701C 86   04        01130         LDA     #$04
701E 34   02        01140         PSHS    A
7020 8E   0506      01150         LDX     #$0506
7023 34   10        01160         PSHS    X
7025 86   07        01170         LDA     #$07
7027 34   02        01180         PSHS    A
7029 8E   0809      01190         LDX     #$0809
702C 34   10        01200         PSHS    X
                    01210
                    01220 * DISPLAY THE THIRD TEST SUITE
702E BD   7034      01230         JSR     STKDSP
                    01240
7031 7E   729A      01250         JMP     LBL001
                    01260
                    01270 *****
                    01280 *
                    01290 * STKDSP
                    01300 * SUBROUTINE
                    01310 * TO DISPLAY THE STACK
                    01320 *
                    01330 *****
                    01340
                    01350 * SAVE REGISTERS
7034 B7   4000      01360 STKDSP  STA     REGA
7037 F7   4001      01370         STB     REGB
703A BF   4002      01380         STX     REGX
                    01390
                    01400 * STACK LOCATION 12
703D 86   31        01410         LDA     #$31    1
703F BD   41DB      01420         JSR     PRTCHA
7042 86   32        01430         LDA     #$32    2
7044 BD   41DB      01440         JSR     PRTCHA
7047 86   2C        01450         LDA     #$2C    ,
7049 BD   41DB      01460         JSR     PRTCHA
704C 86   53        01470         LDA     #$53    S
704E BD   41DB      01480         JSR     PRTCHA
```

```
7051 86   20             LDA     #$20    SPACE
7053 BD   41DB           JSR     PRTCHA
7056 1F   40             TFR     S,D
7058 C3   000C           ADDD    #$000C
705B BD   4178           JSR     PUTWRA  ADDRESS
705E 86   20             LDA     #$20    SPACE
7060 BD   41DB           JSR     PRTCHA
7063 A6   6C             LDA     12,S
7065 BD   409E           JSR     PUTBYA  VALUE
7068 BD   40D7           JSR     CRLF

               * STACK LOCATION 11
706B 86   31             LDA     #$31    1
706D BD   41DB           JSR     PRTCHA
7070 86   31             LDA     #$31    1
7072 BD   41DB           JSR     PRTCHA
7075 86   2C             LDA     #$2C    ,
7077 BD   41DB           JSR     PRTCHA
707A 86   53             LDA     #$53    S
707C BD   41DB           JSR     PRTCHA
707F 86   20             LDA     #$20    SPACE
7081 BD   41DB           JSR     PRTCHA
7084 1F   40             TFR     S,D
7086 C3   000B           ADDD    #$000B
7089 BD   4178           JSR     PUTWRA  ADDRESS
708C 86   20             LDA     #$20    SPACE
708E BD   41DB           JSR     PRTCHA
7091 A6   6B             LDA     11,S
7093 BD   409E           JSR     PUTBYA  VALUE
7096 BD   40D7           JSR     CRLF

               * STACK LOCATION 10
7099 86   31             LDA     #$31    1
709B BD   41DB           JSR     PRTCHA
709E 86   30             LDA     #$30    0
70A0 BD   41DB           JSR     PRTCHA
70A3 86   2C             LDA     #$2C    ,
70A5 BD   41DB           JSR     PRTCHA
70A8 86   53             LDA     #$53    S
70AA BD   41DB           JSR     PRTCHA
70AD 86   20             LDA     #$20    SPACE
70AF BD   41DB           JSR     PRTCHA
70B2 1F   40             TFR     S,D
70B4 C3   000A           ADDD    #$000A
70B7 BD   4178           JSR     PUTWRA  ADDRESS
70BA 86   20             LDA     #$20    SPACE
70BC BD   41DB           JSR     PRTCHA
```

```
70BF A6    6A       01960          LDA     10,S
70C1 BD    409E     01970          JSR     PUTBYA  VALUE
70C4 BD    40D7     01980          JSR     CRLF
                    01990
                    02000 * STACK LOCATION 9
70C7 86    20       02010          LDA     #$20    SPACE
70C9 BD    41DB     02020          JSR     PRTCHA
70CC 86    39       02030          LDA     #$39    9
70CE BD    41DB     02040          JSR     PRTCHA
70D1 86    2C       02050          LDA     #$2C    ,
70D3 BD    41DB     02060          JSR     PRTCHA
70D6 86    53       02070          LDA     #$53    S
70D8 BD    41DB     02080          JSR     PRTCHA
70DB 86    20       02090          LDA     #$20    SPACE
70DD BD    41DB     02100          JSR     PRTCHA
70E0 1F    40       02110          TFR     S,D
70E2 C3    0009     02120          ADDD    #$0009
70E5 BD    4178     02130          JSR     PUTWRA  ADDRESS
70E8 86    20       02140          LDA     #$20    SPACE
70EA BD    41DB     02150          JSR     PRTCHA
70ED A6    69       02160          LDA     9,S
70EF BD    409E     02170          JSR     PUTBYA  VALUE
70F2 BD    40D7     02180          JSR     CRLF
                    02190
                    02200 * STACK LOCATION 8
70F5 86    20       02210          LDA     #$20    SPACE
70F7 BD    41DB     02220          JSR     PRTCHA
70FA 86    38       02230          LDA     #$38    8
70FC BD    41DB     02240          JSR     PRTCHA
70FF 86    2C       02250          LDA     #$2C    ,
7101 BD    41DB     02260          JSR     PRTCHA
7104 86    53       02270          LDA     #$53    S
7106 BD    41DB     02280          JSR     PRTCHA
7109 86    20       02290          LDA     #$20    SPACE
710B BD    41DB     02300          JSR     PRTCHA
710E 1F    40       02310          TFR     S,D
7110 C3    0008     02320          ADDD    #$0008
7113 BD    4178     02330          JSR     PUTWRA  ADDRESS
7116 86    20       02340          LDA     #$20    SPACE
7118 BD    41DB     02350          JSR     PRTCHA
711B A6    68       02360          LDA     8,S
711D BD    409E     02370          JSR     PUTBYA  VALUE
7120 BD    40D7     02380          JSR     CRLF
                    02390
                    02400 * STACK LOCATION 7
7123 86    20       02410          LDA     #$20    SPACE
7125 BD    41DB     02420          JSR     PRTCHA
```

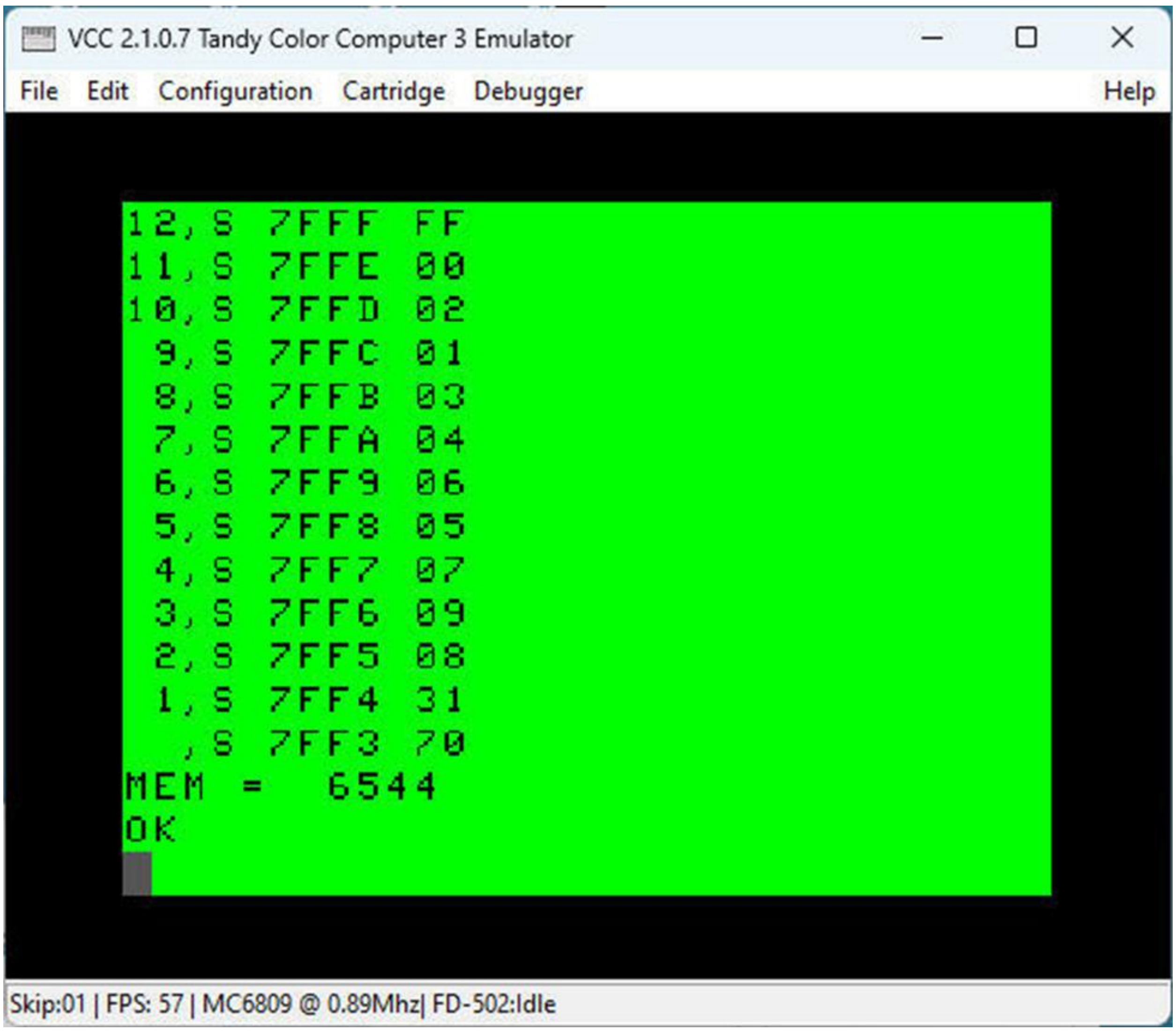
```
7128 86   37        02430          LDA     #$37    7
712A BD   41DB      02440          JSR     PRTCHA
712D 86   2C        02450          LDA     #$2C    ,
712F BD   41DB      02460          JSR     PRTCHA
7132 86   53        02470          LDA     #$53    S
7134 BD   41DB      02480          JSR     PRTCHA
7137 86   20        02490          LDA     #$20    SPACE
7139 BD   41DB      02500          JSR     PRTCHA
713C 1F   40        02510          TFR     S,D
713E C3   0007      02520          ADDD    #$0007
7141 BD   4178      02530          JSR     PUTWRA  ADDRESS
7144 86   20        02540          LDA     #$20    SPACE
7146 BD   41DB      02550          JSR     PRTCHA
7149 A6   67        02560          LDA     7,S
714B BD   409E      02570          JSR     PUTBYA  VALUE
714E BD   40D7      02580          JSR     CRLF
                    02590
                    02600 * STACK LOCATION 6
7151 86   20        02610          LDA     #$20    SPACE
7153 BD   41DB      02620          JSR     PRTCHA
7156 86   36        02630          LDA     #$36    6
7158 BD   41DB      02640          JSR     PRTCHA
715B 86   2C        02650          LDA     #$2C    ,
715D BD   41DB      02660          JSR     PRTCHA
7160 86   53        02670          LDA     #$53    S
7162 BD   41DB      02680          JSR     PRTCHA
7165 86   20        02690          LDA     #$20    SPACE
7167 BD   41DB      02700          JSR     PRTCHA
716A 1F   40        02710          TFR     S,D
716C C3   0006      02720          ADDD    #$0006
716F BD   4178      02730          JSR     PUTWRA  ADDRESS
7172 86   20        02740          LDA     #$20    SPACE
7174 BD   41DB      02750          JSR     PRTCHA
7177 A6   66        02760          LDA     6,S
7179 BD   409E      02770          JSR     PUTBYA  VALUE
717C BD   40D7      02780          JSR     CRLF
                    02790
                    02800 * STACK LOCATION 5
717F 86   20        02810          LDA     #$20    SPACE
7181 BD   41DB      02820          JSR     PRTCHA
7184 86   35        02830          LDA     #$35    5
7186 BD   41DB      02840          JSR     PRTCHA
7189 86   2C        02850          LDA     #$2C    ,
718B BD   41DB      02860          JSR     PRTCHA
718E 86   53        02870          LDA     #$53    S
7190 BD   41DB      02880          JSR     PRTCHA
7193 86   20        02890          LDA     #$20    SPACE
```

```
7195 BD   41DB     02900         JSR     PRTCHA
7198 1F   40       02910         TFR     S,D
719A C3   0005     02920         ADDD    #$0005
719D BD   4178     02930         JSR     PUTWRA  ADDRESS
71A0 86   20       02940         LDA     #$20    SPACE
71A2 BD   41DB     02950         JSR     PRTCHA
71A5 A6   65       02960         LDA     5,S
71A7 BD   409E     02970         JSR     PUTBYA  VALUE
71AA BD   40D7     02980         JSR     CRLF
                   02990
                   03000 * STACK LOCATION 4
71AD 86   20       03010         LDA     #$20    SPACE
71AF BD   41DB     03020         JSR     PRTCHA
71B2 86   34       03030         LDA     #$34    4
71B4 BD   41DB     03040         JSR     PRTCHA
71B7 86   2C       03050         LDA     #$2C    ,
71B9 BD   41DB     03060         JSR     PRTCHA
71BC 86   53       03070         LDA     #$53    S
71BE BD   41DB     03080         JSR     PRTCHA
71C1 86   20       03090         LDA     #$20    SPACE
71C3 BD   41DB     03100         JSR     PRTCHA
71C6 1F   40       03110         TFR     S,D
71C8 C3   0004     03120         ADDD    #$0004
71CB BD   4178     03130         JSR     PUTWRA  ADDRESS
71CE 86   20       03140         LDA     #$20    SPACE
71D0 BD   41DB     03150         JSR     PRTCHA
71D3 A6   64       03160         LDA     4,S
71D5 BD   409E     03170         JSR     PUTBYA  VALUE
71D8 BD   40D7     03180         JSR     CRLF
                   03190
                   03200 * STACK LOCATION 3
71DB 86   20       03210         LDA     #$20    SPACE
71DD BD   41DB     03220         JSR     PRTCHA
71E0 86   33       03230         LDA     #$33    3
71E2 BD   41DB     03240         JSR     PRTCHA
71E5 86   2C       03250         LDA     #$2C    ,
71E7 BD   41DB     03260         JSR     PRTCHA
71EA 86   53       03270         LDA     #$53    S
71EC BD   41DB     03280         JSR     PRTCHA
71EF 86   20       03290         LDA     #$20    SPACE
71F1 BD   41DB     03300         JSR     PRTCHA
71F4 1F   40       03310         TFR     S,D
71F6 C3   0003     03320         ADDD    #$0003
71F9 BD   4178     03330         JSR     PUTWRA  ADDRESS
71FC 86   20       03340         LDA     #$20    SPACE
71FE BD   41DB     03350         JSR     PRTCHA
7201 A6   63       03360         LDA     3,S
```

```
7203 BD   409E     03370          JSR     PUTBYA  VALUE
7206 BD   40D7     03380          JSR     CRLF
                   03390
                   03400 * STACK LOCATION 2
7209 86   20       03410          LDA     #$20    SPACE
720B BD   41DB     03420          JSR     PRTCHA
720E 86   32       03430          LDA     #$32    2
7210 BD   41DB     03440          JSR     PRTCHA
7213 86   2C       03450          LDA     #$2C    ,
7215 BD   41DB     03460          JSR     PRTCHA
7218 86   53       03470          LDA     #$53    S
721A BD   41DB     03480          JSR     PRTCHA
721D 86   20       03490          LDA     #$20    SPACE
721F BD   41DB     03500          JSR     PRTCHA
7222 1F   40       03510          TFR     S,D
7224 C3   0002     03520          ADDD    #$0002
7227 BD   4178     03530          JSR     PUTWRA  ADDRESS
722A 86   20       03540          LDA     #$20    SPACE
722C BD   41DB     03550          JSR     PRTCHA
722F A6   62       03560          LDA     2,S
7231 BD   409E     03570          JSR     PUTBYA  VALUE
7234 BD   40D7     03580          JSR     CRLF
                   03590
                   03600 * STACK LOCATION 1
7237 86   20       03610          LDA     #$20    SPACE
7239 BD   41DB     03620          JSR     PRTCHA
723C 86   31       03630          LDA     #$31    1
723E BD   41DB     03640          JSR     PRTCHA
7241 86   2C       03650          LDA     #$2C    ,
7243 BD   41DB     03660          JSR     PRTCHA
7246 86   53       03670          LDA     #$53    S
7248 BD   41DB     03680          JSR     PRTCHA
724B 86   20       03690          LDA     #$20    SPACE
724D BD   41DB     03700          JSR     PRTCHA
7250 1F   40       03710          TFR     S,D
7252 C3   0001     03720          ADDD    #$0001
7255 BD   4178     03730          JSR     PUTWRA  ADDRESS
7258 86   20       03740          LDA     #$20    SPACE
725A BD   41DB     03750          JSR     PRTCHA
725D A6   61       03760          LDA     1,S
725F BD   409E     03770          JSR     PUTBYA  VALUE
7262 BD   40D7     03780          JSR     CRLF
                   03790
                   03800 * STACK LOCATION 0
7265 86   20       03810          LDA     #$20    SPACE
7267 BD   41DB     03820          JSR     PRTCHA
726A 86   20       03830          LDA     #$20    SPACE
```

```
726C BD   41DB     03840         JSR     PRTCHA
726F 86   2C       03850         LDA     #$2C    ,
7271 BD   41DB     03860         JSR     PRTCHA
7274 86   53       03870         LDA     #$53    S
7276 BD   41DB     03880         JSR     PRTCHA
7279 86   20       03890         LDA     #$20    SPACE
727B BD   41DB     03900         JSR     PRTCHA
727E 1F   40       03910         TFR     S,D
7280 BD   4178     03920         JSR     PUTWRA  ADDRESS
7283 86   20       03930         LDA     #$20    SPACE
7285 BD   41DB     03940         JSR     PRTCHA
7288 A6   E4       03950         LDA     ,S
728A BD   409E     03960         JSR     PUTBYA  VALUE
728D BD   40D7     03970         JSR     CRLF
                   03980
                   03990 * SECOND TEST SUITE CHECK
                   04000 *         JSR     CRLF
                   04010 *         LDD     10,S
                   04020 *         JSR     PUTWRA
                   04030 *         JSR     CRLF
                   04040 *         LDD     8,S
                   04050 *         JSR     PUTWRA
                   04060 *         JSR     CRLF
                   04070 *         LDD     6,S
                   04080 *         JSR     PUTWRA
                   04090 *         JSR     CRLF
                   04100 *         LDD     4,S
                   04110 *         JSR     PUTWRA
                   04120 *         JSR     CRLF
                   04130 *         LDD     2,S
                   04140 *         JSR     PUTWRA
                   04150 *         JSR     CRLF
                   04160
                   04170 * RESTORE REGISTERS
7290 B6   4000     04180         LDA     REGA
7293 F6   4001     04190         LDB     REGB
7296 BE   4002     04200         LDX     REGX
                   04210
                   04220 * EXIT
7299 39            04230         RTS
                   04240
                   04250 *****
                   04260 *
                   04270 * END OF SUBROUTINE
                   04280 *
                   04290 *****
                   04300
```

```
                   04310 * LEAVE THE TEST
                   04320
                   04330 * RESTORE THE STACK POINTER
729A 10FE 4006     04340 LBL001   LDS     REGS
                   04350
                   04360 * RESTORE THE REGISTERS EXCEPT S
                   04370 * FROM THE STACK
729E 35   7F       04380          PULS A,B,X,Y,U,DP,CC
                   04390
72A0 39            04400          RTS
                   04410
          0000     04420          END
```



Test No. 5: THE THIRD TEST SUITE CHECK

```
                    00100 *****
                    00110 *
                    00120 * STKTST3C.ASM
                    00130 * MDJ 2023/02/02
                    00140 *
                    00150 * A TESTING TOOL TO
                    00160 * HELP LEARN ABOUT
                    00170 * STACK OPERATIONS
                    00180 *
                    00190 *****
                    00200
                    00210 * BASIC/ML TRANSFER
                    00220 * VARIABLES
          4000      00230 REGA    EQU     $4000
          4001      00240 REGB    EQU     $4001
          4002      00250 REGX    EQU     $4002
          4004      00260 REGY    EQU     $4004
          4006      00270 REGS    EQU     $4006
          4008      00280 REGU    EQU     $4008
          400A      00290 REGPC   EQU     $400A
          400C      00300 REGDP   EQU     $400C
          400D      00310 REGCC   EQU     $400D
                    00320
                    00330 * EXTERNAL ROUTINE
                    00340 * ADDRESSES
          409E      00350 PUTBYA  EQU     $409E
          40D7      00360 CRLF    EQU     $40D7
          4178      00370 PUTWRA  EQU     $4178
          41DB      00380 PRTCHA  EQU     $41DB
                    00390
                    00400 * PUT THIS UP IN HIGH
                    00410 * MEMORY SO IT WON'T
                    00420 * INTERFERE WITH
                    00430 * ANYTHING ELSE
7000                00440         ORG     $7000
                    00450
                    00460 * SAVE THE REGISTERS EXCEPT S
                    00470 * ON THE STACK
7000 34   7F        00480         PSHS A,B,X,Y,U,DP,CC
                    00490
                    00500 * SAVE THE STACK POINTER
                    00510 * SO WE CAN MOVE IT FOR
                    00520 * TESTING PURPOSES
7002 10FF 4006      00530         STS     REGS
                    00540
```

```
                  * POINT STACK POINTER
                  * TO TOP OF RAM
7006 10CE 7FFF            LDS     #$7FFF

                  * PLACE A SENTINEL AT
                  * THE TOP OF RAM
700A 86   FF              LDA     #$FF
700C B7   7FFF            STA     $7FFF

                  * SETUP FIRST TEST SUITE
                  *        LDA     #$00
                  *        PSHS    A
                  *        LDA     #$01
                  *        PSHS    A
                  *        LDA     #$02
                  *        PSHS    A
                  *        LDA     #$03
                  *        PSHS    A
                  *        LDA     #$04
                  *        PSHS    A
                  *        LDA     #$05
                  *        PSHS    A
                  *        LDA     #$06
                  *        PSHS    A
                  *        LDA     #$07
                  *        PSHS    A
                  *        LDA     #$08
                  *        PSHS    A
                  *        LDA     #$09
                  *        PSHS    A

                  * DISPLAY THE FIRST TEST SUITE
                  *        JSR     STKDSP

                  *        JMP     LBL001

                  * SETUP SECOND TEST SUITE
                  *        LDX     #$0001
                  *        PSHS    X
                  *        LDX     #$0203
                  *        PSHS    X
                  *        LDX     #$0405
                  *        PSHS    X
                  *        LDX     #$0607
                  *        PSHS    X
                  *        LDX     #$0809
                  *        PSHS    X
```

```
                    01020
                    01030 * DISPLAY THE SECOND TEST SUITE
                    01040 *        JSR     STKDSP
                    01050
                    01060 * SETUP THIRD TEST SUITE
700F 86   00        01070         LDA     #$00
7011 34   02        01080         PSHS    A
7013 8E   0102      01090         LDX     #$0102
7016 34   10        01100         PSHS    X
7018 86   03        01110         LDA     #$03
701A 34   02        01120         PSHS    A
701C 86   04        01130         LDA     #$04
701E 34   02        01140         PSHS    A
7020 8E   0506      01150         LDX     #$0506
7023 34   10        01160         PSHS    X
7025 86   07        01170         LDA     #$07
7027 34   02        01180         PSHS    A
7029 8E   0809      01190         LDX     #$0809
702C 34   10        01200         PSHS    X
                    01210
                    01220 * DISPLAY THE THIRD TEST SUITE
702E BD   7034      01230         JSR     STKDSP
                    01240
7031 7E   72D5      01250         JMP     LBL001
                    01260
                    01270 *****
                    01280 *
                    01290 * STKDSP
                    01300 * SUBROUTINE
                    01310 * TO DISPLAY THE STACK
                    01320 *
                    01330 *****
                    01340
                    01350 * SAVE REGISTERS
7034 B7   4000      01360 STKDSP  STA     REGA
7037 F7   4001      01370         STB     REGB
703A BF   4002      01380         STX     REGX
                    01390
                    01400 * STACK LOCATION 12
703D 86   31        01410         LDA     #$31    1
703F BD   41DB      01420         JSR     PRTCHA
7042 86   32        01430         LDA     #$32    2
7044 BD   41DB      01440         JSR     PRTCHA
7047 86   2C        01450         LDA     #$2C    ,
7049 BD   41DB      01460         JSR     PRTCHA
704C 86   53        01470         LDA     #$53    S
704E BD   41DB      01480         JSR     PRTCHA
```

```
7051 86   20      01490         LDA     #$20    SPACE
7053 BD   41DB    01500         JSR     PRTCHA
7056 1F   40      01510         TFR     S,D
7058 C3   000C    01520         ADDD    #$000C
705B BD   4178    01530         JSR     PUTWRA  ADDRESS
705E 86   20      01540         LDA     #$20    SPACE
7060 BD   41DB    01550         JSR     PRTCHA
7063 A6   6C      01560         LDA     12,S
7065 BD   409E    01570         JSR     PUTBYA  VALUE
7068 BD   40D7    01580         JSR     CRLF
                  01590
                  01600 * STACK LOCATION 11
706B 86   31      01610         LDA     #$31    1
706D BD   41DB    01620         JSR     PRTCHA
7070 86   31      01630         LDA     #$31    1
7072 BD   41DB    01640         JSR     PRTCHA
7075 86   2C      01650         LDA     #$2C    ,
7077 BD   41DB    01660         JSR     PRTCHA
707A 86   53      01670         LDA     #$53    S
707C BD   41DB    01680         JSR     PRTCHA
707F 86   20      01690         LDA     #$20    SPACE
7081 BD   41DB    01700         JSR     PRTCHA
7084 1F   40      01710         TFR     S,D
7086 C3   000B    01720         ADDD    #$000B
7089 BD   4178    01730         JSR     PUTWRA  ADDRESS
708C 86   20      01740         LDA     #$20    SPACE
708E BD   41DB    01750         JSR     PRTCHA
7091 A6   6B      01760         LDA     11,S
7093 BD   409E    01770         JSR     PUTBYA  VALUE
7096 BD   40D7    01780         JSR     CRLF
                  01790
                  01800 * STACK LOCATION 10
7099 86   31      01810         LDA     #$31    1
709B BD   41DB    01820         JSR     PRTCHA
709E 86   30      01830         LDA     #$30    0
70A0 BD   41DB    01840         JSR     PRTCHA
70A3 86   2C      01850         LDA     #$2C    ,
70A5 BD   41DB    01860         JSR     PRTCHA
70A8 86   53      01870         LDA     #$53    S
70AA BD   41DB    01880         JSR     PRTCHA
70AD 86   20      01890         LDA     #$20    SPACE
70AF BD   41DB    01900         JSR     PRTCHA
70B2 1F   40      01910         TFR     S,D
70B4 C3   000A    01920         ADDD    #$000A
70B7 BD   4178    01930         JSR     PUTWRA  ADDRESS
70BA 86   20      01940         LDA     #$20    SPACE
70BC BD   41DB    01950         JSR     PRTCHA
```

```
70BF A6   6A       01960          LDA     10,S
70C1 BD   409E     01970          JSR     PUTBYA  VALUE
70C4 BD   40D7     01980          JSR     CRLF
                   01990
                   02000 * STACK LOCATION 9
70C7 86   20       02010          LDA     #$20    SPACE
70C9 BD   41DB     02020          JSR     PRTCHA
70CC 86   39       02030          LDA     #$39    9
70CE BD   41DB     02040          JSR     PRTCHA
70D1 86   2C       02050          LDA     #$2C    ,
70D3 BD   41DB     02060          JSR     PRTCHA
70D6 86   53       02070          LDA     #$53    S
70D8 BD   41DB     02080          JSR     PRTCHA
70DB 86   20       02090          LDA     #$20    SPACE
70DD BD   41DB     02100          JSR     PRTCHA
70E0 1F   40       02110          TFR     S,D
70E2 C3   0009     02120          ADDD    #$0009
70E5 BD   4178     02130          JSR     PUTWRA  ADDRESS
70E8 86   20       02140          LDA     #$20    SPACE
70EA BD   41DB     02150          JSR     PRTCHA
70ED A6   69       02160          LDA     9,S
70EF BD   409E     02170          JSR     PUTBYA  VALUE
70F2 BD   40D7     02180          JSR     CRLF
                   02190
                   02200 * STACK LOCATION 8
70F5 86   20       02210          LDA     #$20    SPACE
70F7 BD   41DB     02220          JSR     PRTCHA
70FA 86   38       02230          LDA     #$38    8
70FC BD   41DB     02240          JSR     PRTCHA
70FF 86   2C       02250          LDA     #$2C    ,
7101 BD   41DB     02260          JSR     PRTCHA
7104 86   53       02270          LDA     #$53    S
7106 BD   41DB     02280          JSR     PRTCHA
7109 86   20       02290          LDA     #$20    SPACE
710B BD   41DB     02300          JSR     PRTCHA
710E 1F   40       02310          TFR     S,D
7110 C3   0008     02320          ADDD    #$0008
7113 BD   4178     02330          JSR     PUTWRA  ADDRESS
7116 86   20       02340          LDA     #$20    SPACE
7118 BD   41DB     02350          JSR     PRTCHA
711B A6   68       02360          LDA     8,S
711D BD   409E     02370          JSR     PUTBYA  VALUE
7120 BD   40D7     02380          JSR     CRLF
                   02390
                   02400 * STACK LOCATION 7
7123 86   20       02410          LDA     #$20    SPACE
7125 BD   41DB     02420          JSR     PRTCHA
```

```
7128 86   37        02430         LDA     #$37    7
712A BD   41DB      02440         JSR     PRTCHA
712D 86   2C        02450         LDA     #$2C    ,
712F BD   41DB      02460         JSR     PRTCHA
7132 86   53        02470         LDA     #$53    S
7134 BD   41DB      02480         JSR     PRTCHA
7137 86   20        02490         LDA     #$20    SPACE
7139 BD   41DB      02500         JSR     PRTCHA
713C 1F   40        02510         TFR     S,D
713E C3   0007      02520         ADDD    #$0007
7141 BD   4178      02530         JSR     PUTWRA  ADDRESS
7144 86   20        02540         LDA     #$20    SPACE
7146 BD   41DB      02550         JSR     PRTCHA
7149 A6   67        02560         LDA     7,S
714B BD   409E      02570         JSR     PUTBYA  VALUE
714E BD   40D7      02580         JSR     CRLF
                    02590
                    02600 * STACK LOCATION 6
7151 86   20        02610         LDA     #$20    SPACE
7153 BD   41DB      02620         JSR     PRTCHA
7156 86   36        02630         LDA     #$36    6
7158 BD   41DB      02640         JSR     PRTCHA
715B 86   2C        02650         LDA     #$2C    ,
715D BD   41DB      02660         JSR     PRTCHA
7160 86   53        02670         LDA     #$53    S
7162 BD   41DB      02680         JSR     PRTCHA
7165 86   20        02690         LDA     #$20    SPACE
7167 BD   41DB      02700         JSR     PRTCHA
716A 1F   40        02710         TFR     S,D
716C C3   0006      02720         ADDD    #$0006
716F BD   4178      02730         JSR     PUTWRA  ADDRESS
7172 86   20        02740         LDA     #$20    SPACE
7174 BD   41DB      02750         JSR     PRTCHA
7177 A6   66        02760         LDA     6,S
7179 BD   409E      02770         JSR     PUTBYA  VALUE
717C BD   40D7      02780         JSR     CRLF
                    02790
                    02800 * STACK LOCATION 5
717F 86   20        02810         LDA     #$20    SPACE
7181 BD   41DB      02820         JSR     PRTCHA
7184 86   35        02830         LDA     #$35    5
7186 BD   41DB      02840         JSR     PRTCHA
7189 86   2C        02850         LDA     #$2C    ,
718B BD   41DB      02860         JSR     PRTCHA
718E 86   53        02870         LDA     #$53    S
7190 BD   41DB      02880         JSR     PRTCHA
7193 86   20        02890         LDA     #$20    SPACE
```

```
7195 BD   41DB     02900          JSR     PRTCHA
7198 1F   40       02910          TFR     S,D
719A C3   0005     02920          ADDD    #$0005
719D BD   4178     02930          JSR     PUTWRA  ADDRESS
71A0 86   20       02940          LDA     #$20    SPACE
71A2 BD   41DB     02950          JSR     PRTCHA
71A5 A6   65       02960          LDA     5,S
71A7 BD   409E     02970          JSR     PUTBYA  VALUE
71AA BD   40D7     02980          JSR     CRLF
                   02990
                   03000 * STACK LOCATION 4
71AD 86   20       03010          LDA     #$20    SPACE
71AF BD   41DB     03020          JSR     PRTCHA
71B2 86   34       03030          LDA     #$34    4
71B4 BD   41DB     03040          JSR     PRTCHA
71B7 86   2C       03050          LDA     #$2C    ,
71B9 BD   41DB     03060          JSR     PRTCHA
71BC 86   53       03070          LDA     #$53    S
71BE BD   41DB     03080          JSR     PRTCHA
71C1 86   20       03090          LDA     #$20    SPACE
71C3 BD   41DB     03100          JSR     PRTCHA
71C6 1F   40       03110          TFR     S,D
71C8 C3   0004     03120          ADDD    #$0004
71CB BD   4178     03130          JSR     PUTWRA  ADDRESS
71CE 86   20       03140          LDA     #$20    SPACE
71D0 BD   41DB     03150          JSR     PRTCHA
71D3 A6   64       03160          LDA     4,S
71D5 BD   409E     03170          JSR     PUTBYA  VALUE
71D8 BD   40D7     03180          JSR     CRLF
                   03190
                   03200 * STACK LOCATION 3
71DB 86   20       03210          LDA     #$20    SPACE
71DD BD   41DB     03220          JSR     PRTCHA
71E0 86   33       03230          LDA     #$33    3
71E2 BD   41DB     03240          JSR     PRTCHA
71E5 86   2C       03250          LDA     #$2C    ,
71E7 BD   41DB     03260          JSR     PRTCHA
71EA 86   53       03270          LDA     #$53    S
71EC BD   41DB     03280          JSR     PRTCHA
71EF 86   20       03290          LDA     #$20    SPACE
71F1 BD   41DB     03300          JSR     PRTCHA
71F4 1F   40       03310          TFR     S,D
71F6 C3   0003     03320          ADDD    #$0003
71F9 BD   4178     03330          JSR     PUTWRA  ADDRESS
71FC 86   20       03340          LDA     #$20    SPACE
71FE BD   41DB     03350          JSR     PRTCHA
7201 A6   63       03360          LDA     3,S
```

```
7203 BD   409E     03370         JSR     PUTBYA  VALUE
7206 BD   40D7     03380         JSR     CRLF
                   03390
                   03400 * STACK LOCATION 2
7209 86   20       03410         LDA     #$20    SPACE
720B BD   41DB     03420         JSR     PRTCHA
720E 86   32       03430         LDA     #$32    2
7210 BD   41DB     03440         JSR     PRTCHA
7213 86   2C       03450         LDA     #$2C    ,
7215 BD   41DB     03460         JSR     PRTCHA
7218 86   53       03470         LDA     #$53    S
721A BD   41DB     03480         JSR     PRTCHA
721D 86   20       03490         LDA     #$20    SPACE
721F BD   41DB     03500         JSR     PRTCHA
7222 1F   40       03510         TFR     S,D
7224 C3   0002     03520         ADDD    #$0002
7227 BD   4178     03530         JSR     PUTWRA  ADDRESS
722A 86   20       03540         LDA     #$20    SPACE
722C BD   41DB     03550         JSR     PRTCHA
722F A6   62       03560         LDA     2,S
7231 BD   409E     03570         JSR     PUTBYA  VALUE
7234 BD   40D7     03580         JSR     CRLF
                   03590
                   03600 * STACK LOCATION 1
7237 86   20       03610         LDA     #$20    SPACE
7239 BD   41DB     03620         JSR     PRTCHA
723C 86   31       03630         LDA     #$31    1
723E BD   41DB     03640         JSR     PRTCHA
7241 86   2C       03650         LDA     #$2C    ,
7243 BD   41DB     03660         JSR     PRTCHA
7246 86   53       03670         LDA     #$53    S
7248 BD   41DB     03680         JSR     PRTCHA
724B 86   20       03690         LDA     #$20    SPACE
724D BD   41DB     03700         JSR     PRTCHA
7250 1F   40       03710         TFR     S,D
7252 C3   0001     03720         ADDD    #$0001
7255 BD   4178     03730         JSR     PUTWRA  ADDRESS
7258 86   20       03740         LDA     #$20    SPACE
725A BD   41DB     03750         JSR     PRTCHA
725D A6   61       03760         LDA     1,S
725F BD   409E     03770         JSR     PUTBYA  VALUE
7262 BD   40D7     03780         JSR     CRLF
                   03790
                   03800 * STACK LOCATION 0
7265 86   20       03810         LDA     #$20    SPACE
7267 BD   41DB     03820         JSR     PRTCHA
726A 86   20       03830         LDA     #$20    SPACE
```

```
726C BD   41DB     03840         JSR     PRTCHA
726F 86   2C       03850         LDA     #$2C    ,
7271 BD   41DB     03860         JSR     PRTCHA
7274 86   53       03870         LDA     #$53    S
7276 BD   41DB     03880         JSR     PRTCHA
7279 86   20       03890         LDA     #$20    SPACE
727B BD   41DB     03900         JSR     PRTCHA
727E 1F   40       03910         TFR     S,D
7280 BD   4178     03920         JSR     PUTWRA  ADDRESS
7283 86   20       03930         LDA     #$20    SPACE
7285 BD   41DB     03940         JSR     PRTCHA
7288 A6   E4       03950         LDA     ,S
728A BD   409E     03960         JSR     PUTBYA  VALUE
728D BD   40D7     03970         JSR     CRLF
                   03980
                   03990 * SECOND TEST SUITE CHECK
                   04000 *        JSR     CRLF
                   04010 *        LDD     10,S
                   04020 *        JSR     PUTWRA
                   04030 *        JSR     CRLF
                   04040 *        LDD     8,S
                   04050 *        JSR     PUTWRA
                   04060 *        JSR     CRLF
                   04070 *        LDD     6,S
                   04080 *        JSR     PUTWRA
                   04090 *        JSR     CRLF
                   04100 *        LDD     4,S
                   04110 *        JSR     PUTWRA
                   04120 *        JSR     CRLF
                   04130 *        LDD     2,S
                   04140 *        JSR     PUTWRA
                   04150 *        JSR     CRLF
                   04160
                   04170 * THIRD TEST SUITE CHECK
7290 BD   40D7     04180         JSR     CRLF
7293 A6   6B       04190         LDA     11,S
7295 BD   409E     04200         JSR     PUTBYA
7298 BD   40D7     04210         JSR     CRLF
729B EC   69       04220         LDD     9,S
729D BD   4178     04230         JSR     PUTWRA
72A0 BD   40D7     04240         JSR     CRLF
72A3 A6   68       04250         LDA     8,S
72A5 BD   409E     04260         JSR     PUTBYA
72A8 BD   40D7     04270         JSR     CRLF
72AB A6   67       04280         LDA     7,S
72AD BD   409E     04290         JSR     PUTBYA
72B0 BD   40D7     04300         JSR     CRLF
```

```
72B3 EC   65        04310         LDD     5,S
72B5 BD   4178      04320         JSR     PUTWRA
72B8 BD   40D7      04330         JSR     CRLF
72BB A6   64        04340         LDA     4,S
72BD BD   409E      04350         JSR     PUTBYA
72C0 BD   40D7      04360         JSR     CRLF
72C3 EC   62        04370         LDD     2,S
72C5 BD   4178      04380         JSR     PUTWRA
72C8 BD   40D7      04390         JSR     CRLF
                    04400
                    04410 * RESTORE REGISTERS
72CB B6   4000      04420         LDA     REGA
72CE F6   4001      04430         LDB     REGB
72D1 BE   4002      04440         LDX     REGX
                    04450
                    04460 * EXIT
72D4 39             04470         RTS
                    04480
                    04490 *****
                    04500 *
                    04510 * END OF SUBROUTINE
                    04520 *
                    04530 *****
                    04540
                    04550 * LEAVE THE TEST
                    04560
                    04570 * RESTORE THE STACK POINTER
72D5 10FE 4006      04580 LBL001  LDS     REGS
                    04590
                    04600 * RESTORE THE REGISTERS EXCEPT S
                    04610 * FROM THE STACK
72D9 35   7F        04620         PULS A,B,X,Y,U,DP,CC
                    04630
72DB 39             04640         RTS
                    04650
          0000      04660         END
```


VCC 2.1.0.7 Tandy Color Computer 3 Emulator
File Edit Configuration Cartridge Debugger Help
4,S 7FF7 07
3,S 7FF6 09
2,S 7FF5 08
1,S 7FF4 31
,S 7FF3 70
00
0102
03
04
0506
07
0809
MEM = 6544
OK
Skip:01 | FPS: 64 | MC6809 @ 0.89Mhz| FD-502:Idle

Appendix E: 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

"Bare-Metal Programming." *Techopedia*. Web. https://www.techopedia.com/definition/3745/bare-metal-programming

.

Barrow, David. *6809 Machine Code Programming*. London: Granada. 1984. Print.

Color Computer Disk System: Owners manual and Programming Guide. Fort Worth, TX: Tandy Corporation, 1981. Print.

Ganssle, Jack. *Writing Relocatable Code.* 1992. Web. http://www.ganssle.com/articles/arelocat.htm .
Getting Started With Extended Color BASIC. Fort Worth, TX: Radio Shack, 1983. Print.

Manchester, William and Reid, Paul. *Defender of the Realm 1940-1965*. New York: Little, Brown and Company, 2012. Print. The Last Lion.

[MDJ01] Johnson, M. David. *Key Codes and VIDRAM*. Glenview, IL: BDS Software, 2021. Web. http://www.bds-soft.com/cocoPapers.php .

[MDJ02] Johnson, M. David. *Towards a VCC Bundle*. Glenview, IL: BDS Software, 2021. Web. http://www.bds-soft.com/cocoPapers.php .

[MDJ03] Johnson, M. David. *Back To (Almost) Bare-Metal Programming*. Glenview, IL: BDS Software, 2021. Web. http://www.bds-soft.com/cocoPapers.php .

[MDJ04] Johnson, M. David. *Research Notes On Previously Initiated Projects*. Glenview, IL: BDS Software, 2022. Web. http://www.bds-soft.com/cocoPapers.php .

Microsoft. *Disk EDTASM+ 01.00.00*. Fort Worth, TX: Radio Shack, 1983. Print.

Perotti, James and Perotti, Victor. "Assembly 101." *Hot CoCo,* May 1985, 68. Peterborough, NH: CW Communications, 1985. Print.

Snider, Ed. "Assembly Programming." *The Zippster Zone*. Web. https://thezippsterzone.com/programming/ .

Warren, Carl. *MC6809 Cookbook, The*. Blue Ridge Summit, PA: TAB Books, 1980. Print.

=====