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

Preliminary
Graphics Control Routines
for the
ML Foundation System

by M. David Johnson

2023/04/22

Abstract

Preliminary PMODE 4 setup and PMODE 4 PSET Graphics Control Routines for the ML Foundation System are developed and presented to run on top of the ML Foundation Core. These Preliminary Routines will be required during the implementation of a yet-to-be-developed Fake Text (maze-like) game.

____

This paper and its associated code are available online at:

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

=====

Table of Contents

=====

Introduction

The first thing I wanted to do with the ML Foundation System was to see if I could build a PMODE 4 (maze-like) game using it. In order to do so, I had to first build and control the PMODE 4 screens. That is the subject of this paper.

The CoCo's Extended Color BASIC ROM seems to be structured around a maximal graphics services concept; i.e. it's set up so that programmers can select and use whatever PMODE they desire at any time, changing back and forth between PMODEs in whatever sequence they choose, and the ROM would handle it all seamlessly.

Furthermore, that ROM code is also fully integrated with the BASIC Interpreter so that whatever the programmer wants to do must first be filtered through the BASIC and Extended BASIC Language system to determine what part of the ROM it should be dispatched to. For example, given a line of code, the BASIC Interpreter must first determine that it is about graphics rather than text before it can even begin to decide where it should be dispatched.

This is convenient for a beginning (or even a comparatively accomplished) BASIC programmer, but it uses a lot of memory and is quite slow.

I instead determined to pursue a minimal graphics services concept; one where only PMODE 4 would be used, and wherein the Assembly Language code could immediately dispatch control to the proper handler with a minimum of code and maximum speed.

Ultimately, separate routines will be developed for each PMODE. This will allow the programmer to determine how many (or if any) of these graphics routines are needed for the project being developed by the programmer. Keeping the PMODES separate will thus further minimize memory usage and maximize speed.

For the purposes of this paper, however, my efforts are restricted to PMODE 4 only.

In addition to restricting the game design to PMODE 4; in order to provide increased speed and simplicity, I determined to develop the game using 256 8-pixel by 12-pixel Fake Text characters as if the PMODE 4 screen were a 32x16 Text screen. The details of that are a subject for a later paper, however.

For the purposes of this present paper, that fake text concept simply served as a guideline into how complex these PMODE 4 Graphics Routines for the ML Foundation System needed to be.

To even further increase speed and simplicity, I determined to utilize only unsigned integers in the game development. Signed integer arithmetic and floating point numerical operations will also be left for future work.

_____

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

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

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

=====

General Methodology

For any given PMODE (and for the Text screen as well), the CoCo's Extended BASIC ROM sets certain low-memory variables, including, but not limited to:

Hex Address	Abbrev.	Variable
00B2	FORCOL	Foreground Color
00B3	BAKCOL	Background Color
00B5	ALLCOL	All Pixels Color Byte
00B6	PMODE	The PMODE Value
00B7	ENDGRP	End of Current Graphic Page
00B9	HORBYT	# of Bytes/Horizontal Line
00BA	BEGGRP	Start of Current Graphic Page
00BC	GRPRAM	MSB of Start of Graphic RAM
00C1	CSSVAL	Screen's Color Set Argument
00DB	CHGFLG	Graphic Data Changed Flag

(cf. Zydhek, p. A-4).

Through some trial and error testing, I determined that by setting some of these variables and then setting-up the SAM and VDG chips appropriately, I could enter PMODE 4 quickly with a minimum of code usage.

Following that, I set-up simplified PCLS and PSET routines for PMODE 4.

More complex graphics code (e.g. LINE, BOX, FILL, etc.) are left for future work, as they were determined to not be necessary for the game I envisioned.

=====

BITWISE.BAS: Unsigned Integer Bitwise Operations in BASIC

In order to be able to check my work as I went along in the development, I decided that it would be convenient to be able to perform unsigned integer bitwise operations in BASIC, a task for which CoCo BASIC has not provided. Accordingly, I developed the following BITWISE.BAS program to perform unsigned 8-bit integer AND, OR, NOT, XOR, LSL, LSR, ROL, and ROR bitwise operations.

This is not part of the Graphics Control Routines, per se, but it is a self-contained utility which you might find useful in your own code development.

```
1000 '*****
1010 '*
1020 '* BITWISE.BAS
1030 '* MDJ 2023/03/14
1040 '*
1050 '* UNSIGNED INTEGER
1060 '* BITWISE OPERATIONS
1070 '* IN BASIC
1080 '*
1090 '*****
1100 CLS
1110 C = 0  'CARRY BIT
1120 'TEMPORARY VARIABLES
1130 ' = A8, A9, B8, B9, T
1140 '

1200 A = 0
1210 INPUT "ENTER 1ST UINT A (0-255)";A
1220 T = INT(A)
1230 IF (T < 0) THEN A = 0
1240 IF (T > 255) THEN A = 255
1250 PRINT
1260 '

1400 B = 0
1410 INPUT "ENTER 1ST UINT B (0-255)";B
1420 T = INT(B)
1430 IF (T < 0) THEN B = 0
1440 IF (T > 255) THEN B = 255
```

```
1450 PRINT
1460 GOTO 2200
1470 '

2000 A = 0
2010 C = 0  'CLEAR CARRY BIT
2020 INPUT "ENTER NEW UINT A (0-255)";A
2030 T = INT(A)
2040 IF (T < 0) THEN A = 0
2050 IF (T > 255) THEN A = 255
2060 PRINT
2070 GOTO 2200
2080 '

2100 B = 0
2110 INPUT "ENTER NEW UINT B (0-255)";B
2120 T = INT(B)
2130 IF (T < 0) THEN B = 0
2140 IF (T > 255) THEN B = 255
2150 PRINT
2160 '

2200 PRINT "SELECT OP:"
2210 PRINT "  A = INPUT NEW A"
2220 PRINT "  B = INPUT NEW B"
2230 PRINT "  Y = CHECK A"
2240 PRINT "  Z = CHECK B"
2250 PRINT "  S = SWAP:  A <--> B"
2260 PRINT "  N = NOT(A)    --> A"
2270 PRINT "  D = (A AND B) --> A"
2280 PRINT "  R = (A OR B)  --> A"
2290 PRINT "  X = (A XOR B) --> A"
2300 PRINT "  2 = GO TO SECOND MENU"
2310 PRINT "  E = EXIT ?";
2320 A$ = INKEY$
2330 IF (A$ = "")  GOTO 2320
2340 PRINT
2350 IF (A$ = "A") GOTO 2000
2360 IF (A$ = "B") GOTO 2100
2370 IF (A$ = "Y") GOTO 12500
2380 IF (A$ = "Z") GOTO 13000
2390 IF (A$ = "S") GOTO 13500
2400 IF (A$ = "N") GOTO 14000
2410 IF (A$ = "D") GOTO 14500
2420 IF (A$ = "R") GOTO 15000
2430 IF (A$ = "X") GOTO 15500
2440 IF (A$ = "2") GOTO 3000
```

```
2450 IF (A$ = "E") GOTO 32767
2460 GOTO 2200
2470 '

3000 PRINT
3010 PRINT
3020 PRINT "SELECT 2ND MENU OP:"
3030 PRINT "  4 = LSL"
3040 PRINT "  5 = LSR"
3050 PRINT "  6 = ROL"
3060 PRINT "  7 = ROR"
3070 PRINT "  1 = BACK TO FIRST MENU"
3080 PRINT "  E = EXIT ?";
3090 A$ = INKEY$
3100 IF (A$ = "")  GOTO 3090
3110 PRINT
3120 IF (A$ = "4") GOTO 16000
3130 IF (A$ = "5") GOTO 16500
3140 IF (A$ = "6") GOTO 17000
3150 IF (A$ = "7") GOTO 17500
3160 IF (A$ = "1") GOTO 2200
3170 IF (A$ = "E") GOTO 32767
3180 GOTO 3010
3190 '

12500 'OP Y = CHECK A
12510 PRINT "UINT: A = ";A
12520 GOSUB 20000               'A: UINT-->BITS
12530 PRINT "A BITS = ";A7;A6;A5;A4;A3;A2;A1;A0;
12540 PRINT "C = CARRY BIT = ";C
12550 GOSUB 20500               'A: BITS-->UINT
12560 PRINT "CHECK: A = ";A
12570 GOTO 2200
12580 '

13000 'OP Z = CHECK B
13010 PRINT "UINT: B = ";B
13020 GOSUB 21000               'B: UINT-->BITS
13030 PRINT "B BITS = ";B7;B6;B5;B4;B3;B2;B1;B0;
13040 GOSUB 20500               'B: BITS-->UINT
13050 PRINT "CHECK: B = ";B
13060 GOTO 2200
13070 '

13500 'OP S = SWAP: A <--> B
13510 GOSUB 22000
13520 PRINT "SWAPPED, NOW:"
```

```
13530 PRINT "  UINT: A = ";A
13540 PRINT "  UINT: B = ";B
13550 GOTO 2200
13560 '

14000 'OP N = NOT(A) --> A
14010 C = 0  'CLEAR CARRY BIT
14020 GOSUB 22500
14030 PRINT "A = NOT(A) = ";A
14040 GOTO 2200
14050 '

14500 'OP D = (A AND B) --> A
14510 C = 0  'CLEAR CARRY BIT
14520 GOSUB 23000
14530 PRINT "A = (A AND B) = ";A
14540 GOTO 2200
14550 '

15000 'OP R = (A OR B) --> A
15010 C = 0  'CLEAR CARRY BIT
15020 GOSUB 23500
15030 PRINT "A = (A OR B) = ";A
15040 GOTO 2200
15050 '

15500 'OP X = (A XOR B) --> A
15510 C = 0  'CLEAR CARRY BIT
15520 GOSUB 24000
15530 PRINT "A = (A XOR B) = ";A
15540 GOTO 2200
15050 '

16000 'OP 4 = LSL(A) --> A
16010 GOSUB 24500
16020 PRINT "A = LSL(A) = ";A
16030 GOTO 2200
16040 '

16500 'OP 5 = LSR(A) --> A
16510 GOSUB 25000
16520 PRINT "A = LSR(A) = ";A
16530 GOTO 2200
16540 '

17000 'OP 6 = ROL(A) --> A
17010 GOSUB 25500
```

```
17020 PRINT "A = ROL(A) = ";A
17030 GOTO 2200
17040 '

17500 'OP 7 = ROR(A) --> A
17510 GOSUB 26000
17520 PRINT "A = ROR(A) = ";A
17530 GOTO 2200
17540 '

20000 'CONVERT AN UNSIGNED INTEGER
20010 'IN REGISTER "A" TO INDIVIDUAL BITS.
20020 'ENTER WITH A = THE UNSIGNED INTEGER.
20030 'EXIT WITH A0-A7 = THE BITS.
20040 A8 = INT(A)
20050 A9 = INT(A8/2)
20060 A0 = A8 - (A9*2)
20070 A8 = A9
20080 A9 = INT(A8/2)
20090 A1 = A8 - (A9*2)
20100 A8 = A9
20110 A9 = INT(A8/2)
20120 A2 = A8 - (A9*2)
20130 A8 = A9
20140 A9 = INT(A8/2)
20150 A3 = A8 - (A9*2)
20160 A8 = A9
20170 A9 = INT(A8/2)
20180 A4 = A8 - (A9*2)
20190 A8 = A9
20200 A9 = INT(A8/2)
20210 A5 = A8 - (A9*2)
20220 A8 = A9
20230 A9 = INT(A8/2)
20240 A6 = A8 - (A9*2)
20250 A8 = A9
20260 A9 = INT(A8/2)
20270 A7 = A8 - (A9*2)
20280 RETURN
20290 '

20500 'CONVERT INDIVIDUAL BITS IN
20510 'REGISTER "A" TO AN UNSIGNED INTEGER.
20520 'ENTER WITH A0-A7 = THE BITS.
20530 'EXIT WITH A = THE UNSIGNED INTEGER.
20540 A = A7*128 + A6*64 + A5*32 + A4*16
20550 A = A + A3*8 + A2*4 + A1*2 + A0*1
```

```
20560 RETURN
20570 '

21000 'CONVERT AN UNSIGNED INTEGER
21010 'IN REGISTER "B" TO INDIVIDUAL BITS.
21020 'ENTER WITH B = THE UNSIGNED INTEGER.
21030 'EXIT WITH B0-B7 = THE BITS.
21040 B8 = INT(B)
21050 B9 = INT(B8/2)
21060 B0 = B8 - (B9*2)
21070 B8 = B9
21080 B9 = INT(B8/2)
21090 B1 = B8 - (B9*2)
21100 B8 = B9
21110 B9 = INT(B8/2)
21120 B2 = B8 - (B9*2)
21130 B8 = B9
21140 B9 = INT(B8/2)
21150 B3 = B8 - (B9*2)
21160 B8 = B9
21170 B9 = INT(B8/2)
21180 B4 = B8 - (B9*2)
21190 B8 = B9
21200 B9 = INT(B8/2)
21210 B5 = B8 - (B9*2)
21220 B8 = B9
21230 B9 = INT(B8/2)
21240 B6 = B8 - (B9*2)
21250 B8 = B9
21260 B9 = INT(B8/2)
21270 B7 = B8 - (B9*2)
21280 RETURN
21290 '

21500 'CONVERT INDIVIDUAL BITS IN
21510 'REGISTER "B" TO AN UNSIGNED INTEGER.
21520 'ENTER WITH B0-B7 = THE BITS.
21530 'EXIT WITH B = THE UNSIGNED INTEGER.
21540 B = B7*128 + B6*64 + B5*32 + B4*16
21550 B = B + B3*8 + B2*4 + B1*2 + B0*1
21560 RETURN
21570 '

22000 'SWAP REGISTERS
22010 T = A
22020 A = B
22030 B = T
```

```
22040 RETURN
22050 '

22500 'BITWISE NOT A
22510 GOSUB 20000                'A: UINT-->BITS
22520 IF (A0 = 0) THEN A0 = 1 ELSE A0 = 0
22530 IF (A1 = 0) THEN A1 = 1 ELSE A1 = 0
22540 IF (A2 = 0) THEN A2 = 1 ELSE A2 = 0
22550 IF (A3 = 0) THEN A3 = 1 ELSE A3 = 0
22560 IF (A4 = 0) THEN A4 = 1 ELSE A4 = 0
22570 IF (A5 = 0) THEN A5 = 1 ELSE A5 = 0
22580 IF (A6 = 0) THEN A6 = 1 ELSE A6 = 0
22590 IF (A7 = 0) THEN A7 = 1 ELSE A7 = 0
22600 GOSUB 20500                'A: BITS-->UINT
22610 RETURN
22620 '

23000 'BITWISE (A AND B) --> A
23010 GOSUB 20000                'A: UINT-->BITS
23020 GOSUB 21000                'B: UINT-->BITS
23030 IF ((A0 = 1) AND (B0 = 1)) THEN A0 = 1 ELSE A0 =
0
23040 IF ((A1 = 1) AND (B1 = 1)) THEN A1 = 1 ELSE A1 =
0
23050 IF ((A2 = 1) AND (B2 = 1)) THEN A2 = 1 ELSE A2 =
0
23060 IF ((A3 = 1) AND (B3 = 1)) THEN A3 = 1 ELSE A3 =
0
23070 IF ((A4 = 1) AND (B4 = 1)) THEN A4 = 1 ELSE A4 =
0
23080 IF ((A5 = 1) AND (B5 = 1)) THEN A5 = 1 ELSE A5 =
0
23090 IF ((A6 = 1) AND (B6 = 1)) THEN A6 = 1 ELSE A6 =
0
23100 IF ((A7 = 1) AND (B7 = 1)) THEN A7 = 1 ELSE A7 =
0
23110 GOSUB 20500                'A: BITS-->UINT
23120 RETURN
23130 '

23500 'BITWISE (A OR B) --> A
23510 GOSUB 20000                'A: UINT-->BITS
23520 GOSUB 21000                'B: UINT-->BITS
23530 IF ((A0 = 1) OR (B0 = 1)) THEN A0 = 1 ELSE A0 =
0
23540 IF ((A1 = 1) OR (B1 = 1)) THEN A1 = 1 ELSE A1 =
0
```

```
23550 IF ((A2 = 1) OR (B2 = 1)) THEN A2 = 1 ELSE A2 =
0
23560 IF ((A3 = 1) OR (B3 = 1)) THEN A3 = 1 ELSE A3 =
0
23570 IF ((A4 = 1) OR (B4 = 1)) THEN A4 = 1 ELSE A4 =
0
23580 IF ((A5 = 1) OR (B5 = 1)) THEN A5 = 1 ELSE A5 =
0
23590 IF ((A6 = 1) OR (B6 = 1)) THEN A6 = 1 ELSE A6 =
0
23600 IF ((A7 = 1) OR (B7 = 1)) THEN A7 = 1 ELSE A7 =
0
23610 GOSUB 20500               'A: BITS-->UINT
23620 RETURN
23630 '

24000 'BITWISE (A XOR B) --> A
24010 GOSUB 20000               'A: UINT-->BITS
24020 GOSUB 21000               'B: UINT-->BITS
24030 IF (((A0 = 1) AND (B0 = 0)) OR ((A0 = 0) AND (B0
= 1))) THEN A0 = 1 ELSE A0 = 0
24040 IF (((A1 = 1) AND (B1 = 0)) OR ((A1 = 0) AND (B1
= 1))) THEN A1 = 1 ELSE A1 = 0
24050 IF (((A2 = 1) AND (B2 = 0)) OR ((A2 = 0) AND (B2
= 1))) THEN A2 = 1 ELSE A2 = 0
24060 IF (((A3 = 1) AND (B3 = 0)) OR ((A3 = 0) AND (B3
= 1))) THEN A3 = 1 ELSE A3 = 0
24070 IF (((A4 = 1) AND (B4 = 0)) OR ((A4 = 0) AND (B4
= 1))) THEN A4 = 1 ELSE A4 = 0
24080 IF (((A5 = 1) AND (B5 = 0)) OR ((A5 = 0) AND (B5
= 1))) THEN A5 = 1 ELSE A5 = 0
24090 IF (((A6 = 1) AND (B6 = 0)) OR ((A6 = 0) AND (B6
= 1))) THEN A6 = 1 ELSE A6 = 0
24100 IF (((A7 = 1) AND (B7 = 0)) OR ((A7 = 0) AND (B7
= 1))) THEN A7 = 1 ELSE A7 = 0
24110 GOSUB 20500               'A: BITS-->UINT
24120 RETURN
24130 '

24500 'BITWISE LOGICAL SHIFT LEFT (LSL)
24510 GOSUB 20000               'A: UINT-->BITS
24520 C = A7
24530 A7 = A6
24540 A6 = A5
24550 A5 = A4
24560 A4 = A3
24570 A3 = A2
```

```
24580 A2 = A1
24590 A1 = A0
24600 A0 = 0
24610 GOSUB 20500                'A: BITS-->UINT
24620 RETURN
24630 '

25000 'BITWISE LOGICAL SHIFT RIGHT (LSR)
25010 GOSUB 20000                'A: UINT-->BITS
25020 C = A0
25030 A0 = A1
25040 A1 = A2
25050 A2 = A3
25060 A3 = A4
25070 A4 = A5
25080 A5 = A6
25090 A6 = A7
25100 A7 = 0
25110 GOSUB 20500                'A: BITS-->UINT
25120 RETURN
25130 '

25500 'BITWISE ROTATE LEFT THRU CARRY (ROL)
25510 GOSUB 20000                'A: UINT-->BITS
25520 T = C
25530 C = A7
25540 A7 = A6
25550 A6 = A5
25560 A5 = A4
25570 A4 = A3
25580 A3 = A2
25590 A2 = A1
25600 A1 = A0
25610 A0 = T
25620 GOSUB 20500                'A: BITS-->UINT
25630 RETURN
25640 '

26000 'BITWISE ROTATE RIGHT THRU CARRY (ROR)
26010 GOSUB 20000                'A: UINT-->BITS
26020 T = C
26030 C = A0
26040 A0 = A1
26050 A1 = A2
26060 A2 = A3
26070 A3 = A4
26080 A4 = A5
```

```
26090 A5 = A6
26100 A6 = A7
26110 A7 = T
26120 GOSUB 20500                'A: BITS-->UINT
26130 RETURN
26140 '

32767 END
```

=====

Graphics Parameters

As mentioned under General Methodology above, for any given PMODE (and for the Text screen as well), the CoCo's Extended BASIC ROM sets certain low-memory variables.

To determine and record those variables' values under the various possible PMODE, Start Page, and Color Set combinations, I prepared the BASIC MLGCTST2.BAS program which appears in Appendix B.

The resulting lists of parameters are recorded in Appendix C.

For the purposes of this paper, we will be utilizing only the following three lists of parameters:

```
BASE: TEXT MODE
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 1400
B9 HORBYT = 10
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

*****

PMODE= 4 , PAGE= 1 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 4
B7 ENDGRP = 2600
B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = F0
POSTCALC  = 7
ENDCALC   = 6
```

*****

PMODE= 4 , PAGE= 5 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 4
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = F0
POSTCALC = 13
ENDCALC   = 6

*****

=====

SVT: Set the SAM and VDG Registers for Text Mode

SVT is used to return to Text Mode from either of the PMODE 4 Graphics screens. This is a self-contained routine which sets the appropriate low-memory variables and also sets-up the SAM and VDG chips for Text Mode. Although SVT physically appears in memory after STSAMG (ORG $458F vs. ORG $4574 respectively), it appears here because it is logically prior to STSAMG in the Graphics Control System.

```
                    00100 *****
                    00110 *
                    00120 * SVT.ASM
                    00130 * MDJ 2023/03/15
                    00140 *
                    00150 * SET THE SAM AND
                    00160 * VDG REGISTERS FOR
                    00170 * TEXT MODE
                    00180 *
                    00190 * CF. EXTENDED BASIC
                    00200 *     UNRAVELLED,
                    00210 *     PAGE B-31
                    00220 *
                    00230 * ENTRY CONDITIONS:
                    00240 * NONE
                    00250 *
                    00260 * EXIT CONDITIONS:
                    00270 * NONE
                    00280 *
                    00290 * SEE MLGCTST2.BAS
                    00300 * RESULTS FOR LOW MEMORY
                    00310 * VARIABLE VALUES
                    00320 *
                    00330 *****
                    00340
                    00350 * LOW MEMORY
                    00360 * GRAPHICS
                    00370 * VARIABLES
          00B2      00380 FORCOL  EQU     $00B2   FOREGROUND
COLOR
          00B3      00390 BAKCOL  EQU     $00B3   BACKGROUND
COLOR
```

```
          00B5     00400 ALLCOL  EQU     $00B5   PIXEL MASK
          00B6     00410 PMODE   EQU     $00B6   PMODE (0-4)
          00B7     00420 ENDGRP  EQU     $00B7   END OF GRAPHIC
PAGE
          00B9     00430 HORBYT  EQU     $00B9   # OF
BYTES/HORIZONTAL LINE
          00BA     00440 BEGGRP  EQU     $00BA   START OF
GRAPHIC PAGE
          00BC     00450 GRPRAM  EQU     $00BC   MSB OF START OF
GRAPHIC RAM
          00C1     00460 CSSVAL  EQU     $00C1   VDG CSS RAM
IMAGE
          00DB     00470 CHGFLG  EQU     $00DB   CHANGE FLAG
                   00480
                   00490 * HIGH MEMORY
                   00500 * SAM AND VDG REGISTER
                   00510 * ADDRESSES
          FF20     00520 PIA1    EQU     $FF20
          FFC0     00530 SAM     EQU     $FFC0
                   00540
458F               00550         ORG     $458F
                   00560
458F 34   16       00570 SVT     PSHS    A,B,X
4591 86   0E       00580         LDA     #$0E
4593 97   BC       00590         STA     GRPRAM
4595 CC   0E00     00600         LDD     #$0E00
4598 DD   BA       00610         STD     BEGGRP
459A CC   1400     00620         LDD     #$1400
459D DD   B7       00630         STD     ENDGRP
459F 86   10       00640         LDA     #$10
45A1 97   B9       00650         STA     HORBYT
45A3 86   03       00660         LDA     #$03
45A5 97   B2       00670         STA     FORCOL
45A7 4F            00680         CLRA
45A8 97   B6       00690         STA     PMODE
45AA 97   B3       00700         STA     BAKCOL
45AC 97   B5       00710         STA     ALLCOL
45AE 97   C1       00720         STA     CSSVAL
45B0 97   DB       00730         STA     CHGFLG
                   00740
45B2 8E   FFC8     00750         LDX     #SAM+8  POINT TO THE
MIDDLE OF THE SAM
REGISTER
45B5 A7   0A       00760         STA     10,X
45B7 A7   08       00770         STA     8,X
45B9 A7   06       00780         STA     6,X
```

```
45BB A7   04       00790          STA     4,X     RESET SAM
DISPLAY PAGE TO $400
45BD A7   02       00800          STA     2,X
45BF A7   01       00810          STA     1,X
45C1 A7   1E       00820          STA     -2,X
45C3 A7   1C       00830          STA     -4,X
45C5 A7   1A       00840          STA     -6,X    RESET SAM S VDG
TO TEXT MODE
45C7 A7   18       00850          STA     -8,X
45C9 B6   FF22     00860          LDA     PIA1+2  PORT B DATA
45CC 84   07       00870          ANDA    #$07    FORCE ALL BITS
TO ZERO:
                   00880 *                        KEEP ONLY CSS
DATA
45CE B7   FF22     00890          STA     PIA1+2  PUT THE VDG
INTO TEXT MODE
45D1 35   16       00900          PULS    A,B,X
45D3 39            00910          RTS
                   00920
          0000     00930          END
```

See the PCLS4 Chapter below for testing.

=====

STSAMG: Set the SAM Register for Graphics Modes

STSAMG is a subroutine called by both SVG410 and SVG450. It sets the SAM Register for either of the two PMODE 4 setups (Start Page 1 or Start Page 5). Although STSAMG physically appears in memory before SVT (ORG $4574 vs. ORG $458F respectively), it appears here because it is logically subsequent to SVT in the Graphics Control System.

```
                    00100 *****
                    00110 *
                    00120 * STSAMG.ASM
                    00130 * MDJ 2023/03/12
                    00140 *
                    00150 * SET THE
                    00160 * SAM REGISTER FOR
                    00170 * GRAPHICS MODES
                    00180 *
                    00190 * CF. EXTENDED BASIC
                    00200 *     UNRAVELLED,
                    00210 *     PAGE B-31
                    00220 *
                    00230 * ENTRY CONDITIONS:
                    00240 * NONE
                    00250 *
                    00260 * EXIT CONDITIONS:
                    00270 * NONE
                    00280 *
                    00290 *****
                    00300
                    00310 * HIGH MEMORY
                    00320 * SAM REGISTER
                    00330 * ADDRESS
          FFC0      00340 SAM     EQU     $FFC0
                    00350
4574                00360         ORG     $4574
                    00370
4574 C6   03        00380 STSAMG  LDB     #3      FIRST STSAMG
ENTRY POINT
                    00390 *                       3 BITS IN SAM
VDG CONTROL REGIS
TER
```

```
4576 8E   FFC0     00400          LDX     #SAM    POINT TO SAM
CONTROL REGISTER
4579 46            00410 LBL001   RORA            PUT A BIT INTO
THE CARRY FLAG
457A 24   04       00420          BCC     LBL002  GO IF BIT WAS
ZERO
457C A7   01       00430          STA     1,X     SET SAM
REGISTER BIT
457E 20   02       00440          BRA     LBL003  DO NEXT BIT
4580 A7   84       00450 LBL002   STA     ,X      CLEAR SAM
REGISTER
4582 30   02       00460 LBL003   LEAX    2,X     NEXT BIT IN
REGISTER
4584 5A            00470          DECB            HAVE WE DONE
ALL BITS?
4585 26   F2       00480          BNE     LBL001  GO IF NO
4587 39            00490          RTS
4588 C6   07       00500 STSAM2   LDB     #7      SECOND STSAMG
ENTRY POINT
                   00510 *                        7 BITS IN SAM
DISPLAY PAGE REGI
STER
458A 8E   FFC6     00520          LDX     #SAM+6  POINT TO SAM
DISPLAY PAGE REGIS
TER
458D 20   EA       00530          BRA     LBL001  GO SET THE
REGISTER
                   00540
          0000     00550          END
```

See the PCLS4 Chapter below for testing.

=====

SVG410: Set the SAM and VDG Registers for Graphics Mode PMODE 4, Start Page 1, Color Set 0

SVG410 is used to go into PMODE 4 at Start Page 1. This routine sets the appropriate low-memory variables and also sets-up the VDG chip; it then calls STSAMG to setup the SAM chip.

```
00100 *****
00110 *
00120 * SVG410.ASM
00130 * MDJ 2023/03/15
00140 *
00150 * SET THE SAM AND
00160 * VDG REGISTERS FOR
00170 * GRAPHICS MODE
00180 *
00190 * PMODE 4
00200 * START PAGE 1
00210 * COLOR SET 0
00220 *
00230 * CF. EXTENDED BASIC
00240 *     UNRAVELLED,
00250 *     PAGE B-31
00260 *
00270 * ENTRY CONDITIONS:
00280 * NONE
00290 *
00300 * EXIT CONDITIONS:
00310 * NONE
00320 *
00330 * SEE MLGCTST2.BAS
00340 * RESULTS FOR LOW MEMORY
00350 * VARIABLE VALUES AND:
00360 *   MIDCALC
00370 *   POSTCALC
00380 *   ENDCALC
00390 *
00400 *****
00410
```

```
                    00420 * LOW MEMORY
                    00430 * GRAPHICS
                    00440 * VARIABLES
          00B2      00450 FORCOL  EQU     $00B2   FOREGROUND
COLOR
          00B3      00460 BAKCOL  EQU     $00B3   BACKGROUND
COLOR
          00B5      00470 ALLCOL  EQU     $00B5   PIXEL MASK
          00B6      00480 PMODE   EQU     $00B6   PMODE (0-4)
          00B7      00490 ENDGRP  EQU     $00B7   END OF GRAPHIC
PAGE
          00B9      00500 HORBYT  EQU     $00B9   # OF
BYTES/HORIZONTAL LINE
          00BA      00510 BEGGRP  EQU     $00BA   START OF
GRAPHIC PAGE
          00BC      00520 GRPRAM  EQU     $00BC   MSB OF START OF
GRAPHIC RAM
          00C1      00530 CSSVAL  EQU     $00C1   VDG CSS RAM
IMAGE
          00DB      00540 CHGFLG  EQU     $00DB   CHANGE FLAG
                    00550
                    00560 * SAM REGISTER CONTROL
                    00570 * FIRST ENTRY POINT
          4574      00580 STSAMG  EQU     $4574
                    00590
                    00600 * SAM REGISTER CONTROL
                    00610 * SECOND ENTRY POINT
          4588      00620 STSAM2  EQU     $4588
                    00630
                    00640 * HIGH MEMORY
                    00650 * VDG REGISTER
                    00660 * ADDRESS
          FF20      00670 PIA1    EQU     $FF20
                    00680
45D4                00690         ORG     $45D4
                    00700
45D4 34   16        00710 SVG410  PSHS    A,B,X
45D6 86   0E        00720         LDA     #$0E
45D8 97   BC        00730         STA     GRPRAM
45DA CC   0E00      00740         LDD     #$0E00
45DD DD   BA        00750         STD     BEGGRP
45DF CC   2600      00760         LDD     #$2600
45E2 DD   B7        00770         STD     ENDGRP
45E4 86   20        00780         LDA     #$20
45E6 97   B9        00790         STA     HORBYT
45E8 86   04        00800         LDA     #$04
45EA 97   B6        00810         STA     PMODE
```

```
45EC 86   03       00820          LDA     #$03
45EE 97   B2       00830          STA     FORCOL
45F0 4F            00840          CLRA
45F1 97   B3       00850          STA     BAKCOL
45F3 97   B5       00860          STA     ALLCOL
45F5 97   C1       00870          STA     CSSVAL
45F7 97   DB       00880          STA     CHGFLG
                   00890
45F9 86   F0       00900          LDA     #$F0    MIDCALC
45FB B7   FF22     00910          STA     PIA1+2
45FE 86   07       00920          LDA     #$07    POSTCALC
4600 BD   4588     00930          JSR     STSAM2
4603 86   06       00940          LDA     #$06    ENDCALC
4605 BD   4574     00950          JSR     STSAMG
4608 35   16       00960          PULS    A,B,X
460A 39            00970          RTS
                   00980
          0000     00990          END
```

See the PCLS4 Chapter below for testing.

=====

SVG450: Set the SAM and VDG Registers for Graphics Mode PMODE 4, Start Page 5, Color Set 0

SVG450 is used to go into PMODE 4 at Start Page 5. This routine sets the appropriate low-memory variables and also sets-up the VDG chip; it then calls STSAMG to setup the SAM chip.

```
00100 *****
00110 *
00120 * SVG450.ASM
00130 * MDJ 2023/03/15
00140 *
00150 * SET THE SAM AND
00160 * VDG REGISTERS FOR
00170 * GRAPHICS MODE
00180 *
00190 * PMODE 4
00200 * START PAGE 5
00210 * COLOR SET 0
00220 *
00230 * CF. EXTENDED BASIC
00240 *     UNRAVELLED,
00250 *     PAGE B-31
00260 *
00270 * ENTRY CONDITIONS:
00280 * NONE
00290 *
00300 * EXIT CONDITIONS:
00310 * NONE
00320 *
00330 * SEE MLGCTST2.BAS
00340 * RESULTS FOR LOW MEMORY
00350 * VARIABLE VALUES AND:
00360 *   MIDCALC
00370 *   POSTCALC
00380 *   ENDCALC
00390 *
00400 *****
00410
```

```
                    00420 * LOW MEMORY
                    00430 * GRAPHICS
                    00440 * VARIABLES
          00B2      00450 FORCOL  EQU     $00B2   FOREGROUND
COLOR
          00B3      00460 BAKCOL  EQU     $00B3   BACKGROUND
COLOR
          00B5      00470 ALLCOL  EQU     $00B5   PIXEL MASK
          00B6      00480 PMODE   EQU     $00B6   PMODE (0-4)
          00B7      00490 ENDGRP  EQU     $00B7   END OF GRAPHIC
PAGE
          00B9      00500 HORBYT  EQU     $00B9   # OF
BYTES/HORIZONTAL LINE
          00BA      00510 BEGGRP  EQU     $00BA   START OF
GRAPHIC PAGE
          00BC      00520 GRPRAM  EQU     $00BC   MSB OF START OF
GRAPHIC RAM
          00C1      00530 CSSVAL  EQU     $00C1   VDG CSS RAM
IMAGE
          00DB      00540 CHGFLG  EQU     $00DB   CHANGE FLAG
                    00550
                    00560 * SAM REGISTER CONTROL
                    00570 * FIRST ENTRY POINT
          4574      00580 STSAMG  EQU     $4574
                    00590
                    00600 * SAM REGISTER CONTROL
                    00610 * SECOND ENTRY POINT
          4588      00620 STSAM2  EQU     $4588
                    00630
                    00640 * HIGH MEMORY
                    00650 * VDG REGISTER
                    00660 * ADDRESS
          FF20      00670 PIA1    EQU     $FF20
                    00680
460B                00690         ORG     $460B
                    00700
460B 34   16        00710 SVG410  PSHS    A,B,X
460D 86   0E        00720         LDA     #$0E
460F 97   BC        00730         STA     GRPRAM
4611 CC   2600      00740         LDD     #$2600
4614 DD   BA        00750         STD     BEGGRP
4616 CC   3E00      00760         LDD     #$3E00
4619 DD   B7        00770         STD     ENDGRP
461B 86   20        00780         LDA     #$20
461D 97   B9        00790         STA     HORBYT
461F 86   04        00800         LDA     #$04
4621 97   B6        00810         STA     PMODE
```

```
4623 86   03       00820          LDA     #$03
4625 97   B2       00830          STA     FORCOL
4627 4F            00840          CLRA
4628 97   B3       00850          STA     BAKCOL
462A 97   B5       00860          STA     ALLCOL
462C 97   C1       00870          STA     CSSVAL
462E 97   DB       00880          STA     CHGFLG
                   00890
4630 86   F0       00900          LDA     #$F0    MIDCALC
4632 B7   FF22     00910          STA     PIA1+2
4635 86   13       00920          LDA     #$13    POSTCALC
4637 BD   4588     00930          JSR     STSAM2
463A 86   06       00940          LDA     #$06    ENDCALC
463C BD   4574     00950          JSR     STSAMG
463F 35   16       00960          PULS    A,B,X
4641 39            00970          RTS
                   00980
          0000     00990          END
```

See the PCLS4 Chapter below for testing.

=====

PCLS4: PMODE 4 PCLS

PCLS4 clears the PMODE 4 screen to either Black or Green.

```
                    00100 *****
                    00110 *
                    00120 * PCLS4.ASM
                    00130 * MDJ 2023/03/15
                    00140 *
                    00150 * PMODE4 PCLS
                    00160 *
                    00170 * ENTRY CONDITIONS:
                    00180 * A = COLOR CODE
                    00190 *      0 = BLACK -->
                    00200 *          SET A = #$00
                    00210 *          (ALL ZEROES)
                    00220 *      1 = GREEN -->
                    00230 *          SET A = #$FF
                    00240 *          (ALL ONES)
                    00250 * DEFAULTS TO GREEN
                    00260 * IF ANYTHING ELSE
                    00270 *
                    00280 * EXIT CONDITIONS:
                    00290 * NONE
                    00300 *
                    00310 *****
                    00320
                    00330 * LOW MEMORY
                    00340 * GRAPHICS
                    00350 * VARIABLES
          00B7      00360 ENDGRP  EQU     $00B7   END OF GRAPHIC
PAGE
          00BA      00370 BEGGRP  EQU     $00BA   START OF
GRAPHIC PAGE
                    00380
4642                00390         ORG     $4642
                    00400
4642 34   14        00410 PCLS4   PSHS    B,X
4644 81   00        00420         CMPA    #$00    IS COLOR 0 =
BLACK?
4646 27   02        00430         BEQ     LBL001  GO IF YES
4648 86   FF        00440         LDA     #$FF    DEFAULT TO 1 =
GREEN
464A 1F   89        00450 LBL001  TFR     A,B     D = A:A
```

```
464C 9E   BA       00460         LDX     BEGGRP  START ADDRESS
464E ED   81       00470 LBL002  STD     ,X++    SET BYTES TO
COLOR
4650 9C   B7       00480         CMPX    ENDGRP  AT END ADDRESS?
4652 26   FA       00490         BNE     LBL002  GO IF NO
4654 35   14       00500         PULS    B,X
4656 39            00510         RTS
                   00520
          0000     00530         END
```

The Assembly Language Test Routine:

```
                   00100 *****
                   00110 *
                   00120 * TEST0201.ASM
                   00130 * MDJ 2023/03/15
                   00140 *
                   00150 * INITIAL PMODE 4
                   00160 * GRAPHICS TEST
                   00170 *
                   00180 *****
                   00190
                   00200 * MLCORE ADDRESS
          4142     00210 POLCAT  EQU     $4142
                   00220
                   00230 * GRAPHICS ROUTINES
                   00240 * ADDRESSES
          4574     00250 STSAMG  EQU     $4574
          4588     00260 STSAM2  EQU     $4588
          458F     00270 SVT     EQU     $458F
          45D4     00280 SVG410  EQU     $45D4
          4608     00290 SVG450  EQU     $4608
          4642     00300 PCLS4   EQU     $4642
                   00310
7000               00320         ORG     $7000
                   00330
7000 34   02       00340         PSHS    A
                   00350
                   00360 * SET PMODE 4 PAGE 1
7002 BD   45D4     00370         JSR     SVG410
                   00380
                   00390 * CLEAR THE SCREEN TO GREEN
7005 86   01       00400         LDA     #$01
7007 BD   4642     00410         JSR     PCLS4
                   00420
```

```
                    00430 * WAIT FOR A KEYPRESS
700A 34   03      00440         PSHS    A,CC
700C BD   4142    00450 LBL001  JSR     POLCAT
700F 27   FB      00460         BEQ     LBL001  NO KEYPRESS
7011 35   03      00470         PULS    A,CC
                  00480
                  00490 * CLEAR THE SCREEN TO BLACK
7013 86   00      00500         LDA     #$00
7015 BD   4642    00510         JSR     PCLS4
                  00520
                  00530 * WAIT FOR A KEYPRESS
7018 34   03      00540         PSHS    A,CC
701A BD   4142    00550 LBL002  JSR     POLCAT
701D 27   FB      00560         BEQ     LBL002  NO KEYPRESS
701F 35   03      00570         PULS    A,CC
                  00580
                  00590 * SET PMODE 4 PAGE 5
7021 BD   4608    00600         JSR     SVG450
                  00610
                  00620 * CLEAR THE SCREEN TO GREEN
7024 86   01      00630         LDA     #$01
7026 BD   4642    00640         JSR     PCLS4
                  00650
                  00660 * WAIT FOR A KEYPRESS
7029 34   03      00670         PSHS    A,CC
702B BD   4142    00680 LBL003  JSR     POLCAT
702E 27   FB      00690         BEQ     LBL003  NO KEYPRESS
7030 35   03      00700         PULS    A,CC
                  00710
                  00720 * CLEAR THE SCREEN TO BLACK
7032 86   00      00730         LDA     #$00
7034 BD   4642    00740         JSR     PCLS4
                  00750
                  00760 * WAIT FOR A KEYPRESS
7037 34   03      00770         PSHS    A,CC
7039 BD   4142    00780 LBL004  JSR     POLCAT
703C 27   FB      00790         BEQ     LBL004  NO KEYPRESS
703E 35   03      00800         PULS    A,CC
                  00810
                  00820 * RETURN TO TEXT MODE
7040 BD   458F    00830         JSR     SVT
                  00840
                  00850 * WAIT FOR A KEYPRESS
7043 34   03      00860         PSHS    A,CC
7045 BD   4142    00870 LBL005  JSR     POLCAT
7048 27   FB      00880         BEQ     LBL005  NO KEYPRESS
704A 35   03      00890         PULS    A,CC
```

```
                    00900
704C 35   02      00910          PULS    A
704E 39           00920          RTS
                  00930
          0000    00940          END
```

Making the Preliminary MLFT File: This program combines the Graphics Control Routines into the single MLFT.BIN file so that the machine language Graphics Control System can be loaded as a single entity. (See the Final PSET41 Chapter below for making the Final MLFT File.)

Note that the FLSYS.BIN file specified on Line 2000 is provided by the MAKEFALS.BAS program in ([MDJ01], p.104). It is not used in these Graphics Control Routines, but is included as a check that the MLCORE, False Disk Routines and Graphics Control Routines all fit together properly without any erroneous overlaps.

```
1000 '*****
1010 '*
1020 '* MAKEMLFT.BAS
1030 '* MDJ 2023/03/16
1040 '*
1050 '* MAKES THE
1060 '* MLFT.BIN FILE
1070 '*
1080 '*****
1090 '

1500 CLEAR 200, &H4000
1510 '

2000 LOADM "FLSYS.BIN"
2010 LOADM "STSAMG.BIN"
2020 LOADM "SVT.BIN"
2030 LOADM "SVG410.BIN"
2030 LOADM "SVG450.BIN"
2040 LOADM "PCLS4.BIN"
2050 '

3000 SAVEM "MLFT.BIN", &H4416, &H4656, &H4416
3010 '

32767 END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0201.BAS
1030 '* MDJ 2023/03/16
1040 '*
1050 '* INITIAL PMODE 4
1060 '* GRAPHICS TEST
1070 '*
1080 '*****
1090 '

1100 'SETUP MEMORY
1110 CLEAR 0, &H4000
1120 PCLEAR 8
1120 '

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

1300 'LOAD THE MLFT FILE
1310 LOADM "MLFT.BIN"
1320 '

1400 'LOAD THE
1410 'ML TEST ROUTINE
1420 LOADM "TEST0201.BIN"
1430 '

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

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

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

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

32767 END
```

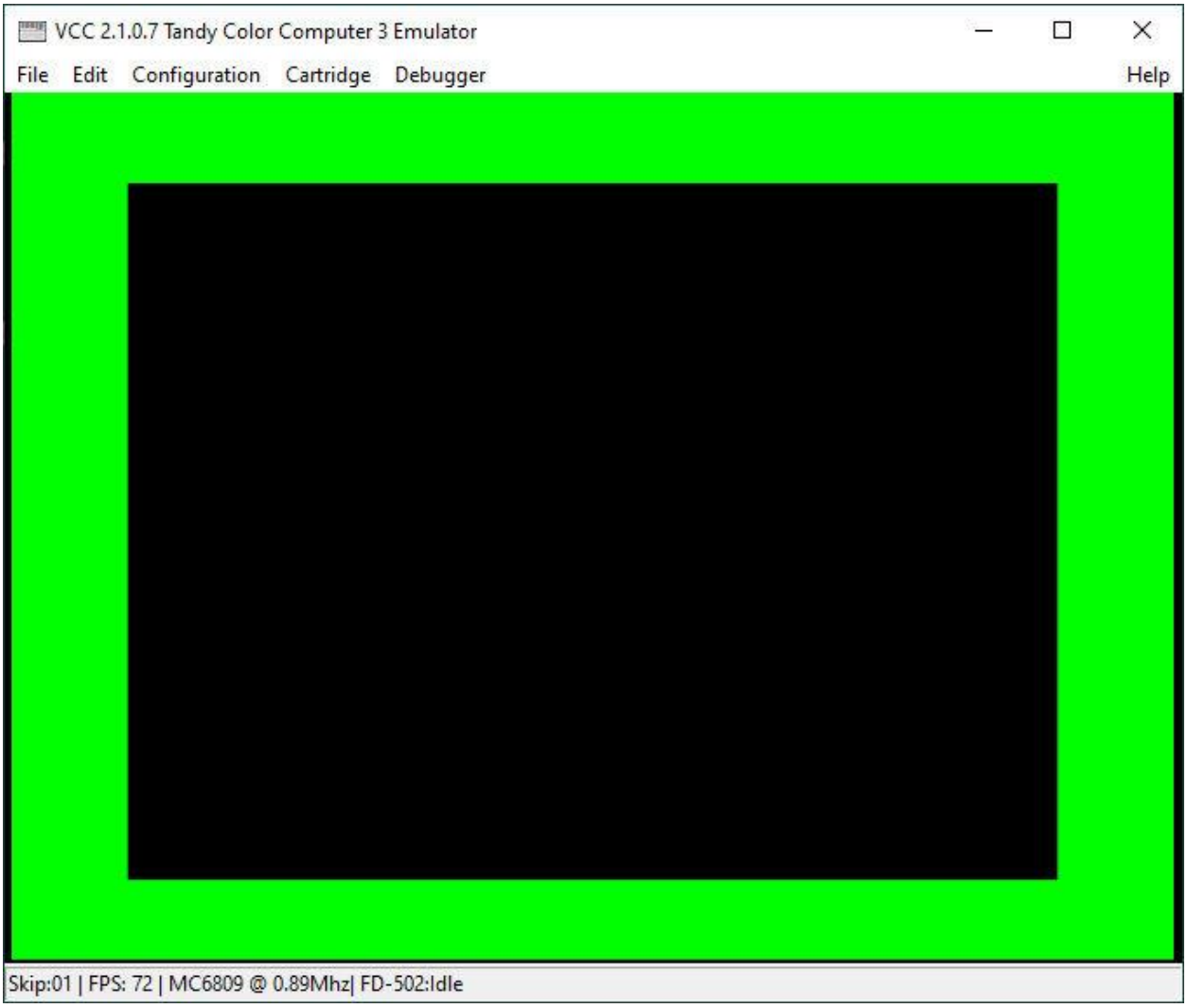
The BASIC Language Control Program, Abbreviated: The full program, as written above, bombs with a reported OM ERROR. This is intentional - The entire ML Foundation System is intended to be initialized using the absolute minimum of BASIC code; just enough to get the Machine Language system loaded and started. It is the following abbreviated BASIC code which is on disk as TEST0201.BAS.

```
1110 CLEAR0,&H4000
1120 PCLEAR8
1220 LOADM"MLCORE.BIN"
1310 LOADM"MLFT.BIN"
1420 LOADM"TEST0201.BIN"
2920 RA=&H400A
2930 RB=&H400B
3020 C=&H7000
3030 C1=INT(C/256)
3040 C2=INT(C-(C1*256))
3050 POKERA,C1
3060 POKERB,C2
6020 EXEC&H4403
9020 PRINT
9030 PRINT" MEM = ";MEM
9040 PRINT"FREE = ";FREE(0)
32767 END
```

Results:

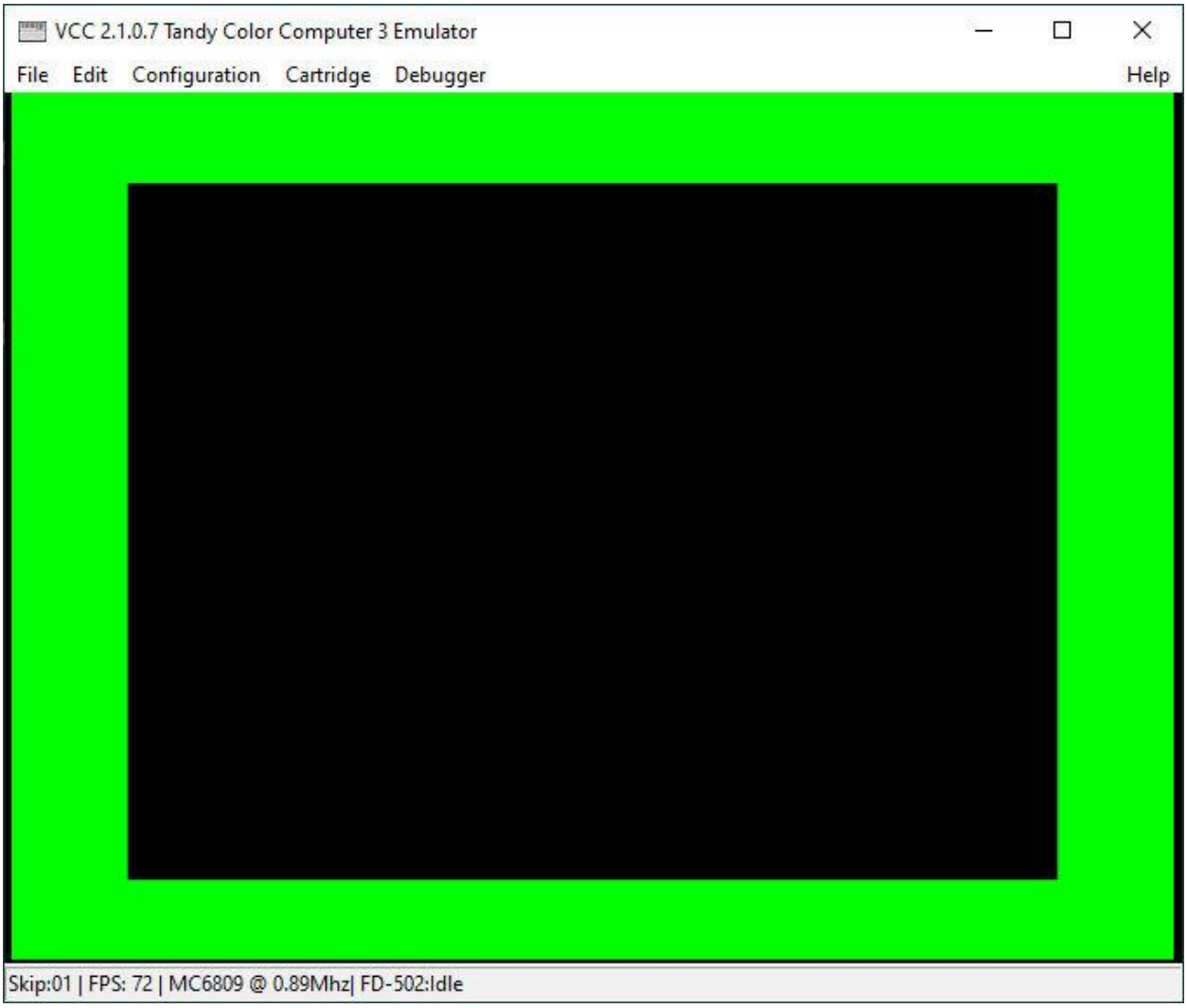
On Startup of TEST0201.BAS: PMODE 4. Start Page 1; cleared to Green.

On First Keypress: PMODE 4. Start Page 1; cleared to Black.

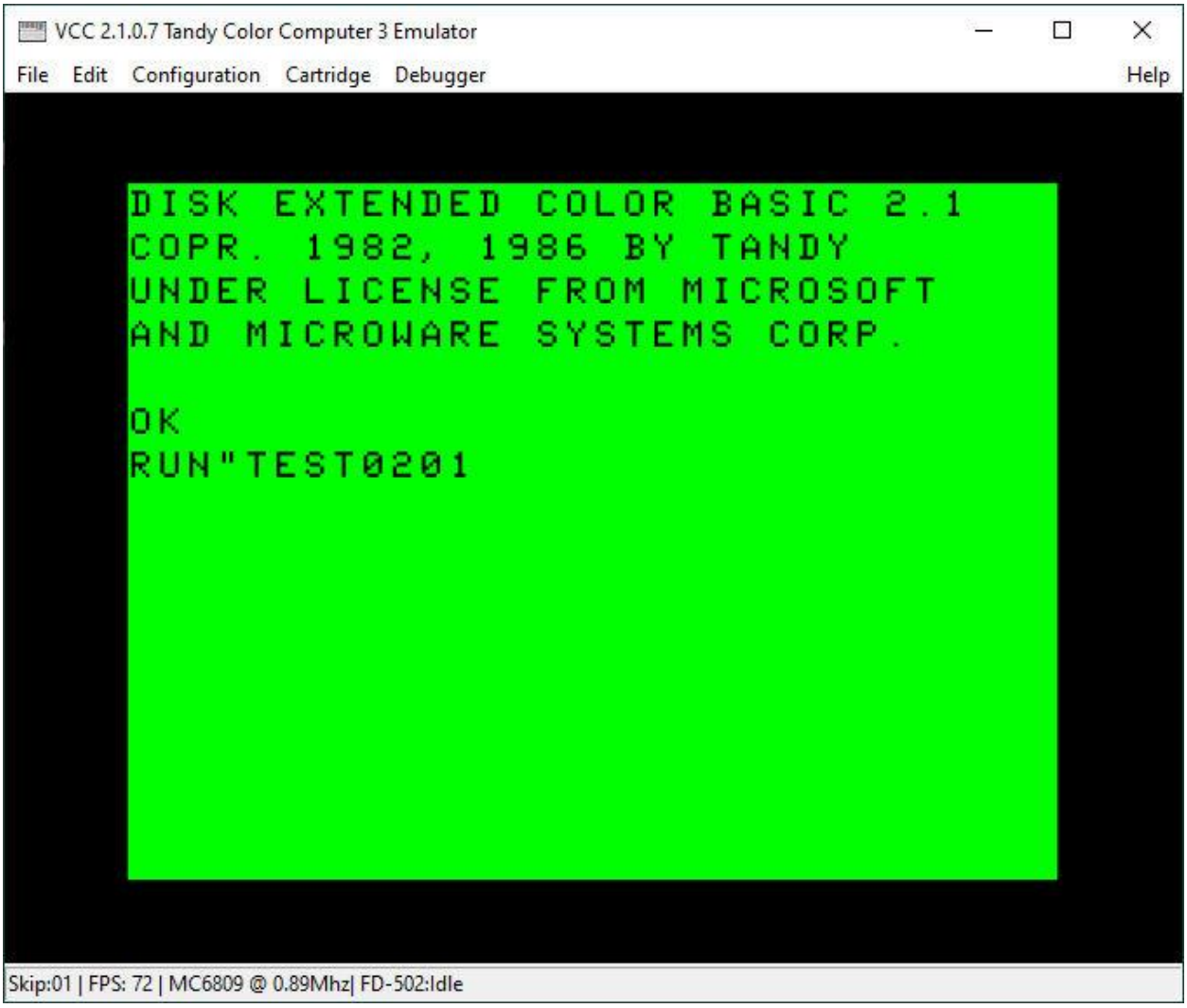


On Second Keypress: PMODE 4. Start Page 5; cleared to Green.

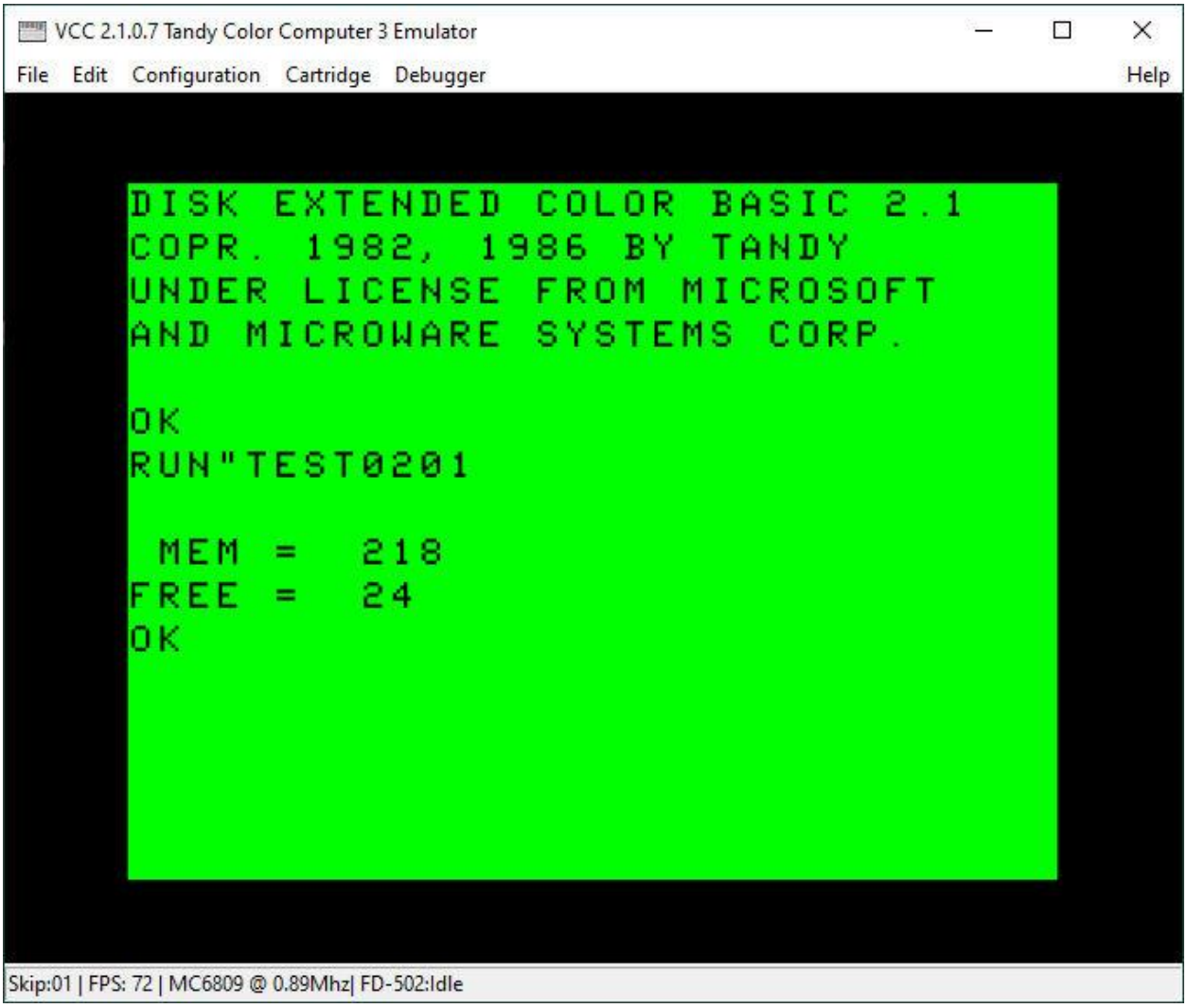
On Third Keypress: PMODE 4. Start Page 1; cleared to Black.



On Fourth Keypress: The return to Text Mode.



On Fiifth Keypress: After exiting the TEST0201.BAS program.



All as expected.

=====

The ROM PSET Revisited

Since the PSET routines will eventually be the foundation of many other Graphics routines (LINE, BOX, FILL, etc.) I wanted to make sure that my PSET implementations were highly efficient; both in terms of bytes of memory occupied, and also in terms of CPU cycles consumed.

So it naturally occurred to me that the standard of comparison should be the CoCo's Extended BASIC ROM PSET routine. Regardless of its actual size and speed, the ROM PSET would be something I could measure improvements against; e.g. "This implementation is X% smaller and Y% faster than the ROM PSET."

So, I initially began to step through (Zydhek)'s ROM assembly listing, unwinding the PSET and recording the number of bytes used. I intended to go back later and fill-in the CPU cycles for each instruction.

1,147 lines (of the most outrageous spaghetti code I've ever seen up close) later, I finally reached the end and had "PSET Unwound" in my hands. It was so huge, I didn't even bother to go back and record the cycles. I didn't even total up the number of bytes.

Nor did I think it worthy of adding 49 pages to this paper to include it here. If you're interested, you can find it at:

https://www.bds-soft.com/files/coco/NewFiles/MyPapers/2023/MLGC/ROMPSETUnwound.pdf .

PSTGB4: Get PMODE 4 Byte and Bit Position from (X,Y) Coordinates

PSTGB4 translates the selected screen's X-Coordinate and Y-Coordinate into the screen's byte address, byte contents, and the bit position within that byte. PSTGB4 is a subroutine which is called by both PSET40 and PSET41..

```
                    00100 *****
                    00110 *
                    00120 * PSTGB4.ASM
                    00130 * MDJ 2023/03/15
                    00140 *
                    00150 * GET PMODE 4 BYTE
                    00160 * AND BIT POSITION FROM
                    00170 * (X,Y) COORDINATES
                    00180 *
                    00190 * ENTRY CONDITIONS:
                    00200 * A = Y-COORDINATE (0-191)
                    00210 * B = X-COORDINATE (0-255)
                    00220 *
                    00230 * EXIT CONDITIONS
                    00240 * A = BIT POSITION (0-7)
                    00250 * B = BYTE CONTENTS
                    00260 * X = BYTE ADDRESS
                    00270 *
                    00280 *****
                    00290
                    00300 * LOW MEMORY
                    00310 * GRAPHICS
                    00320 * VARIABLE
          00BA      00330 BEGGRP  EQU     $00BA   START OF
GRAPHIC PAGE
                    00340
4657                00350         ORG     $4657
                    00360
                    00370 * AT THIS POINT THE
                    00380 * STACK CONTAINS
                    00390 *   1,S RTS ADDRESS LOW BYTE
                    00400 *    ,S RTS ADDRESS HIGH BYTE
                    00410
                    00420 * MAKE ROOM FOR
```

```
                    00430 *   7,S RTS ADDRESS LOW BYTE
                    00440 *   6,S RTS ADDRESS HIGH BYTE
                    00450 *   5,S PIXEL NUMBER LOW BYTE
                    00460 *   4,S PIXEL NUMBER HIGH BYTE
                    00470 *   3,S BYTE NUMBER LOW BYTE
                    00480 *   2,S BYTE NUMBER HIGH BYTE
                    00490 *   1,S TEMP VALUE LOW BYTE
                    00500 *    ,S TEMP VALUE HIGH BYTE
4657 32   7A        00510 PSTGB4  LEAS    -6,S
                    00520
                    00530 * WITH A = Y-CO0RDINATE
                    00540 *  AND B = X-COORDINATE,
                    00550 * D = A:B = PIXEL NUMBER = PN
                    00560 * AND BYTE NUMBER = BN = PN/8
4659 ED   64        00570         STD     4,S     SAVE PN
465B 44             00580         LSRA
465C 56             00590         RORB
465D 44             00600         LSRA
465E 56             00610         RORB
465F 44             00620         LSRA
4660 56             00630         RORB            D = D/8
4661 ED   62        00640         STD     2,S     SAVE BN
                    00650
                    00660 * GET BYTE ADDRESS
4663 DC   BA        00670         LDD     BEGGRP  START OF
GRAPHIC PAGE
4665 E3   62        00680         ADDD    2,S     BYTE NUMBER
4667 1F   01        00690         TFR     D,X     BYTE ADDRESS
                    00700
                    00710 * BIT POSITION IN BYTE = BP
                    00720 * = 7 - (PN - (BN*8))
4669 EC   62        00730         LDD     2,S     BN = BYTE
NUMBER
466B 58             00740         LSLB
466C 49             00750         ROLA
466D 58             00760         LSLB
466E 49             00770         ROLA
466F 58             00780         LSLB
4670 49             00790         ROLA            BN*8
4671 ED   E4        00800         STD     ,S      SAVE TEMP
4673 EC   64        00810         LDD     4,S     PN
4675 A3   E4        00820         SUBD    ,S      PN - (BN*8)
4677 ED   E4        00830         STD     ,S      SAVE TEMP
4679 86   07        00840         LDA     #7
467B A0   61        00850         SUBA    1,S     LSB OF TEMP -->
BP
                    00860
```

```
                    00870 * GET BYTE CONTENTS
467D E6   84        00880         LDB     ,X
                    00890
                    00900 * CLEAN THE STACK AND RETURN
467F 32   66        00910         LEAS    6,S
4681 39             00920         RTS
                    00930
          0000      00940         END
```

See the Final PSET41 Chapter below for testing.

=====

First Attempt
PSET40: Set a PMODE 4 Point to Black = 0

This was my first attempt at writing PSET40. It calls the subroutine PSTGB4 to translate the coordinates into the screen's byte data and then puts the black pixel to the screen. As can be seen below, this first attempt was already quite efficient (certainly so as compared to the ROM PSET), consuming 80 bytes of memory and 132 CPU cycles.

```
                    00100 *****
                    00110 *
                    00120 * PSET40F.ASM
                    00130 * MDJ 2023/03/15
                    00140 * FIRST ATTEMPT
                    00150 *
                    00160 * SET A PMODE 4
                    00170 * POINT TO
                    00180 * BLACK = 0
                    00190 *
                    00200 * ENTRY CONDITIONS:
                    00210 * A = Y-COORDINATE (0-191)
                    00220 * B = X-COORDINATE (0-255)
                    00230 *
                    00240 * EXIT CONDITIONS:
                    00250 * NONE
                    00260 *
                    00270 *****
                    00280
                    00290 * BYTE DATA
                    00300 * ROUTINE ADDRESS
          4657      00310 PSTGB4  EQU     $4657
                    00320
4682                00330         ORG     $4682
                    00340
4682 34   10        00350 PSET40  PSHS    X
                    00360
                    00370 * AT THIS POINT THE
                    00380 * STACK CONTAINS
                    00390 *   3,S RTS ADDRESS LOW BYTE
                    00400 *   2,S RTS ADDRESS HIGH BYTE
                    00410 *   1,S REGISTER X LOW BYTE
                    00420 *    ,S REGISTER X HIGH BYTE
```

```
                    00430
                    00440 * MAKE ROOM FOR
                    00450 *   5,S RTS ADDRESS LOW BYTE
                    00460 *   4,S RTS ADDRESS HIGH BYTE
                    00470 *   3,S REGISTER X LOW BYTE
                    00480 *   2,S REGISTER X HIGH BYTE
                    00490 *   1,S BYTE CONTENTS
                    00500 *    ,S PIXEL MASK
4684 32   7E        00510         LEAS    -2,S
                    00520
                    00530 * GO GET BYTE DATA
4686 BD   4657      00540         JSR     PSTGB4
                    00550
                    00560 * NOW:
                    00570 * A = BIT POSITION (0-7)
                    00580 * B = BYTE CONTENTS
                    00590 * X = BYTE ADDRESS
                    00600
                    00610 * SAVE THE BYTE CONTENTS
4689 E7   61        00620         STB     1,S
                    00630
                    00640 * ALL ONES MASK TO
                    00650 * SET BIT = 0 = BLACK
                    00660 * USING "AND"
468B 81   00        00670         CMPA    #0
468D 26   04        00680         BNE     LBL001
468F C6   FE        00690         LDB     #$FE
4691 20   32        00700         BRA     LBL008
4693 81   01        00710 LBL001  CMPA    #1
4695 26   04        00720         BNE     LBL002
4697 C6   FD        00730         LDB     #$FD
4699 20   2A        00740         BRA     LBL008
469B 81   02        00750 LBL002  CMPA    #2
469D 26   04        00760         BNE     LBL003
469F C6   FB        00770         LDB     #$FB
46A1 20   22        00780         BRA     LBL008
46A3 81   03        00790 LBL003  CMPA    #3
46A5 26   04        00800         BNE     LBL004
46A7 C6   F7        00810         LDB     #$F7
46A9 20   1A        00820         BRA     LBL008
46AB 81   04        00830 LBL004  CMPA    #4
46AD 26   04        00840         BNE     LBL005
46AF C6   EF        00850         LDB     #$EF
46B1 20   12        00860         BRA     LBL008
46B3 81   05        00870 LBL005  CMPA    #5
46B5 26   04        00880         BNE     LBL006
46B7 C6   DF        00890         LDB     #$DF
```

```
46B9 20   0A       00900         BRA     LBL008
46BB 81   06       00910 LBL006  CMPA    #6
46BD 26   04       00920         BNE     LBL007
46BF C6   BF       00930         LDB     #$BF
46C1 20   02       00940         BRA     LBL008
46C3 C6   7F       00950 LBL007  LDB     #$7F
46C5 E7   E4       00960 LBL008  STB     ,S      PIXEL MASK
                   00970
                   00980 * SET THE PIXEL TO 0 = BLACK
46C7 A6   61       00990         LDA     1,S     BYTE CONTENTS
46C9 A4   E4       01000         ANDA    ,S      PIXEL MASK
46CB A7   84       01010         STA     ,X      PUT TO SCREEN
                   01020
                   01030 * CLEAN THE STACK AND RETURN
46CD 32   62       01040         LEAS    2,S
46CF 35   10       01050         PULS    X
46D1 39            01060         RTS
                   01070
          0000     01080         END
```

———

Cycles and Bytes:

```
                   00100 *****
                   00110 *
                   00120 * PSET40FD.ASM
                   00130 * MDJ 2023/03/15
                   00135 * FIRST ATTEMPT
                   00136 * CYCLES AND BYTES
                   00250 *
                   00260 *****
          4657     00300 PSTGB4  EQU     $4657       CYCLES   BYTES
4682               00320         ORG     $4682       ------   -----
4682 34   10       00340 PSET40  PSHS    X                7       2
4684 32   7E       00500         LEAS    -2,S             5       2
4686 BD   4657     00530         JSR     PSTGB4           8       3
4689 E7   61       00610         STB     1,S              5       2
468B 81   00       00660         CMPA    #0               2       2
468D 26   04       00670         BNE     LBL001           3       2
468F C6   FE       00680         LDB     #$FE             2       2
4691 20   32       00690         BRA     LBL008           3       2
4693 81   01       00700 LBL001  CMPA    #1               2       2
4695 26   04       00710         BNE     LBL002           3       2
4697 C6   FD       00720         LDB     #$FD             2       2
4699 20   2A       00730         BRA     LBL008           3       2
469B 81   02       00740 LBL002  CMPA    #2               2       2
```

```
469D 26   04       00750          BNE     LBL003            3       2
469F C6   FB       00760          LDB     #$FB              2       2
46A1 20   22       00770          BRA     LBL008            3       2
46A3 81   03       00780 LBL003   CMPA    #3                2       2
46A5 26   04       00790          BNE     LBL004            3       2
46A7 C6   F7       00800          LDB     #$F7              2       2
46A9 20   1A       00810          BRA     LBL008            3       2
46AB 81   04       00820 LBL004   CMPA    #4                2       2
46AD 26   04       00830          BNE     LBL005            3       2
46AF C6   EF       00840          LDB     #$EF              2       2
46B1 20   12       00850          BRA     LBL008            3       2
46B3 81   05       00860 LBL005   CMPA    #5                2       2
46B5 26   04       00870          BNE     LBL006            3       2
46B7 C6   DF       00880          LDB     #$DF              2       2
46B9 20   0A       00890          BRA     LBL008            3       2
46BB 81   06       00900 LBL006   CMPA    #6                2       2
46BD 26   04       00910          BNE     LBL007            3       2
46BF C6   BF       00920          LDB     #$BF              2       2
46C1 20   02       00930          BRA     LBL008            3       2
46C3 C6   7F       00940 LBL007   LDB     #$7F              2       2
46C5 E7   E4       00950 LBL008   STB     ,S                5       2
46C7 A6   61       00980          LDA     1,S               5       2
46C9 A4   E4       00990          ANDA    ,S                4       2
46CB A7   84       01000          STA     ,X                4       2
46CD 32   62       01030          LEAS    2,S               5       2
46CF 35   10       01040          PULS    X                 7       2
46D1 39            01050          RTS                       5       1
          0000     01070          END                   -----   -----
                                                TOTALS      132      80
```

See the Final PSET41 Chapter below for testing.

=====

First Attempt
PSET41: Set a PMODE 4 Point to Green = 1

This was my first attempt at writing PSET41. It calls the subroutine PSTGB4 to translate the coordinates into the screen's byte data and then puts the green pixel to the screen. Except for using a zeroes mask instead of a ones mask, this routine is identical to the first attempt at PSET40. Therefore, I did not do a separate cycles and bytes analysis for this routine.

```
                    00100 *****
                    00110 *
                    00120 * PSET41F.ASM
                    00130 * MDJ 2023/03/15
                    00140 * FIRST ATTEMPT
                    00150 *
                    00160 * SET A PMODE 4
                    00170 * POINT TO
                    00180 * GREEN = 1
                    00190 *
                    00200 * ENTRY CONDITIONS:
                    00210 * A = Y-COORDINATE (0-191)
                    00220 * B = X-COORDINATE (0-255)
                    00230 *
                    00240 * EXIT CONDITIONS:
                    00250 * NONE
                    00260 *
                    00270 *****
                    00280
                    00290 * BYTE DATA
                    00300 * ROUTINE ADDRESS
          4657      00310 PSTGB4  EQU     $4657
                    00320
46D2                00330         ORG     $46D2
                    00340
46D2 34   10        00350 PSET41  PSHS    X
                    00360
                    00370 * AT THIS POINT THE
                    00380 * STACK CONTAINS
                    00390 *   3,S RTS ADDRESS LOW BYTE
                    00400 *   2,S RTS ADDRESS HIGH BYTE
                    00410 *   1,S REGISTER X LOW BYTE
                    00420 *    ,S REGISTER X HIGH BYTE
```

```
                    00430
                    00440 * MAKE ROOM FOR
                    00450 *   5,S RTS ADDRESS LOW BYTE
                    00460 *   4,S RTS ADDRESS HIGH BYTE
                    00470 *   3,S REGISTER X LOW BYTE
                    00480 *   2,S REGISTER X HIGH BYTE
                    00490 *   1,S BYTE CONTENTS
                    00500 *    ,S PIXEL MASK
46D4 32   7E        00510         LEAS    -2,S
                    00520
                    00530 * GO GET BYTE DATA
46D6 BD   4657      00540         JSR     PSTGB4
                    00550
                    00560 * NOW:
                    00570 * A = BIT POSITION (0-7)
                    00580 * B = BYTE CONTENTS
                    00590 * X = BYTE ADDRESS
                    00600
                    00610 * SAVE THE BYTE CONTENTS
46D9 E7   61        00620         STB     1,S
                    00630
                    00640 * ALL ZEROES MASK TO
                    00650 * SET BIT = 1 = GREEN
                    00660 * USING "OR"
46DB 81   00        00670 LBL000  CMPA    #0
46DD 26   04        00680         BNE     LBL001
46DF C6   01        00690         LDB     #$01
46E1 20   32        00700         BRA     LBL008
46E3 81   01        00710 LBL001  CMPA    #1
46E5 26   04        00720         BNE     LBL002
46E7 C6   02        00730         LDB     #$02
46E9 20   2A        00740         BRA     LBL008
46EB 81   02        00750 LBL002  CMPA    #2
46ED 26   04        00760         BNE     LBL003
46EF C6   04        00770         LDB     #$04
46F1 20   22        00780         BRA     LBL008
46F3 81   03        00790 LBL003  CMPA    #3
46F5 26   04        00800         BNE     LBL004
46F7 C6   08        00810         LDB     #$08
46F9 20   1A        00820         BRA     LBL008
46FB 81   04        00830 LBL004  CMPA    #4
46FD 26   04        00840         BNE     LBL005
46FF C6   10        00850         LDB     #$10
4701 20   12        00860         BRA     LBL008
4703 81   05        00870 LBL005  CMPA    #5
4705 26   04        00880         BNE     LBL006
4707 C6   20        00890         LDB     #$20
```

```
4709 20   0A       00900          BRA     LBL008
470B 81   06       00910 LBL006   CMPA    #6
470D 26   04       00920          BNE     LBL007
470F C6   40       00930          LDB     #$40
4711 20   02       00940          BRA     LBL008
4713 C6   80       00950 LBL007   LDB     #$80
4715 E7   E4       00960 LBL008   STB     ,S      PIXEL MASK
                   00970
                   00980 * SET THE PIXEL TO 0 = BLACK
4717 A6   61       00990          LDA     1,S     BYTE CONTENTS
4719 AA   E4       01000          ORA     ,S      PIXEL MASK
471B A7   84       01010          STA     ,X      PUT TO SCREEN
                   01020
                   01030 * CLEAN THE STACK AND RETURN
471D 32   62       01040          LEAS    2,S
471F 35   10       01050          PULS    X
4721 39            01060          RTS
                   01070
          0000     01080          END
```

See the Final PSET41 Chapter below for testing.

=====

Final PSET40: Set a PMODE 4 Point to Black = 0

This is the final PSET40. It calls the subroutine PSTGB4 to translate the coordinates into the screen's byte data and then puts the black pixel to the screen. As can be seen below, this final code is somewhat more efficient than the first attempt; consuming 65 bytes of memory and 126 CPU cycles, as compared to the first attempt's 80 and 132 respectively.

```
                    00100 *****
                    00110 *
                    00120 * PSET40.ASM
                    00130 * MDJ 2023/03/25
                    00140 * FINAL
                    00150 *
                    00160 * SET A PMODE 4
                    00170 * POINT TO
                    00180 * BLACK = 0
                    00190 *
                    00200 * ENTRY CONDITIONS:
                    00210 * A = Y-COORDINATE (0-191)
                    00220 * B = X-COORDINATE (0-255)
                    00230 *
                    00240 * EXIT CONDITIONS:
                    00250 * NONE
                    00260 *
                    00270 *****
                    00280
                    00290 * BYTE DATA
                    00300 * ROUTINE ADDRESS
          4657      00310 PSTGB4  EQU     $4657
                    00320
4682                00330         ORG     $4682
                    00340
4682 34   30        00350 PSET40  PSHS    X,Y
                    00360
                    00370 * AT THIS POINT THE
                    00380 * STACK CONTAINS
                    00390 *   3,S RTS ADDRESS LOW BYTE
                    00400 *   2,S RTS ADDRESS HIGH BYTE
                    00410 *   1,S REGISTER X LOW BYTE
                    00420 *    ,S REGISTER X HIGH BYTE
                    00430
                    00440 * MAKE ROOM FOR
```

```
                    00450 *   5,S RTS ADDRESS LOW BYTE
                    00460 *   4,S RTS ADDRESS HIGH BYTE
                    00470 *   3,S REGISTER X LOW BYTE
                    00480 *   2,S REGISTER X HIGH BYTE
                    00490 *   1,S BYTE CONTENTS
                    00500 *    ,S PIXEL MASK
4684 32   7E        00510         LEAS    -2,S
                    00520
                    00530 * GO GET BYTE DATA
4686 BD   4657      00540         JSR     PSTGB4
                    00550
                    00560 * NOW:
                    00570 * A = BIT POSITION (0-7)
                    00580 * B = BYTE CONTENTS
                    00590 * X = BYTE ADDRESS
                    00600
                    00610 * SAVE THE BYTE CONTENTS
4689 E7   61        00620         STB     1,S
                    00630
                    00640 * SAVE THE BYTE ADDRESS
468B 1F   12        00650         TFR     X,Y
                    00660
                    00670 * ALL ONES MASK TO
                    00680 * SET BIT = 0 = BLACK
                    00690 * USING "AND"
468D 48             00700         LSLA            A = A * 4
468E 48             00710         LSLA             (4 BYTES PER
LBL)
468F 8E   4694      00720         LDX     #LBL000
4692 6E   86        00730         JMP     A,X
4694 C6   FE        00740 LBL000  LDB     #$FE
4696 20   1A        00750         BRA     LBL008
4698 C6   FD        00760 LBL001  LDB     #$FD
469A 20   16        00770         BRA     LBL008
469C C6   FB        00780 LBL002  LDB     #$FB
469E 20   12        00790         BRA     LBL008
46A0 C6   F7        00800 LBL003  LDB     #$F7
46A2 20   0E        00810         BRA     LBL008
46A4 C6   EF        00820 LBL004  LDB     #$EF
46A6 20   0A        00830         BRA     LBL008
46A8 C6   DF        00840 LBL005  LDB     #$DF
46AA 20   06        00850         BRA     LBL008
46AC C6   BF        00860 LBL006  LDB     #$BF
46AE 20   02        00870         BRA     LBL008
46B0 C6   7F        00880 LBL007  LDB     #$7F
46B2 E7   E4        00890 LBL008  STB     ,S      PIXEL MASK
                    00900
```

```
                    00910 * SET THE PIXEL TO 0 = BLACK
46B4 1F   21        00920         TFR     Y,X     BYTE ADDRESS
46B6 A6   61        00930         LDA     1,S     BYTE CONTENTS
46B8 A4   E4        00940         ANDA    ,S      PIXEL MASK
46BA A7   84        00950         STA     ,X      PUT TO SCREEN
                    00960
                    00970 * CLEAN THE STACK AND RETURN
46BC 32   62        00980         LEAS    2,S
46BE 35   30        00990         PULS    X,Y
46C0 39             01000         RTS
                    01010
          0000      01020         END
```

Cycles and Bytes:

```
                    00100 *****
                    00110 *
                    00120 * PSET40D.ASM
                    00130 * MDJ 2023/03/25
                    00135 * FINAL
                    00136 * CYCLES AND BYTES
                    00140 *
                    00270 *****
                    00280
          4657      00310 PSTGB4  EQU     $4657       CYCLES   BYTES
4682                00330         ORG     $4682       ------   -----
4682 34   30        00350 PSET40  PSHS    X,Y              9       2
4684 32   7E        00510         LEAS    -2,S             5       2
4686 BD   4657      00540         JSR     PSTGB4           8       3
4689 E7   61        00620         STB     1,S              5       2
468B 1F   12        00650         TFR     X,Y              7       2
468D 48             00700         LSLA                     2       1
468E 48             00710         LSLA                     2       1
468F 8E   4694      00720         LDX     #LBL000          3       3
4692 6E   86        00730         JMP     A,X              4       2
4694 C6   FE        00740 LBL000  LDB     #$FE             2       2
4696 20   1A        00750         BRA     LBL008           3       2
4698 C6   FD        00760 LBL001  LDB     #$FD             2       2
469A 20   16        00770         BRA     LBL008           3       2
469C C6   FB        00780 LBL002  LDB     #$FB             2       2
469E 20   12        00790         BRA     LBL008           3       2
46A0 C6   F7        00800 LBL003  LDB     #$F7             2       2
46A2 20   0E        00810         BRA     LBL008           3       2
46A4 C6   EF        00820 LBL004  LDB     #$EF             2       2
46A6 20   0A        00830         BRA     LBL008           3       2
```

```
46A8 C6   DF       00840 LBL005  LDB     #$DF              2       2
46AA 20   06       00850         BRA     LBL008            3       2
46AC C6   BF       00860 LBL006  LDB     #$BF              2       2
46AE 20   02       00870         BRA     LBL008            3       2
46B0 C6   7F       00880 LBL007  LDB     #$7F              2       2
46B2 E7   E4       00890 LBL008  STB     ,S                5       2
46B4 1F   21       00920         TFR     Y,X               7       2
46B6 A6   61       00930         LDA     1,S               5       2
46B8 A4   E4       00940         ANDA    ,S                4       2
46BA A7   84       00950         STA     ,X                4       2
46BC 32   62       00980         LEAS    2,S               5       2
46BE 35   30       00990         PULS    X,Y               9       2
46C0 39            01000         RTS                       5       1
          0000     01020         END                   -----   -----
                                                TOTALS      126      65
```

_____

See the Final PSET41 Chapter below for testing.

=====

Final PSET41: Set a PMODE 4 Point to Green = 1

This is the final PSET41. It calls the subroutine PSTGB4 to translate the coordinates into the screen's byte data and then puts the green pixel to the screen. Except for using a zeroes mask instead of a ones mask, this routine is identical to the final PSET40. Therefore, I did not do a separate cycles and bytes analysis for this routine.

```
                        00100 *****
                        00110 *
                        00120 * PSET41.ASM
                        00130 * MDJ 2023/03/26
                        00140 * FINAL
                        00150 *
                        00160 * SET A PMODE 4
                        00170 * POINT TO
                        00180 * GREEN = 1
                        00190 *
                        00200 * ENTRY CONDITIONS:
                        00210 * A = Y-COORDINATE (0-191)
                        00220 * B = X-COORDINATE (0-255)
                        00230 *
                        00240 * EXIT CONDITIONS:
                        00250 * NONE
                        00260 *
                        00270 *****
                        00280
                        00290 * BYTE DATA
                        00300 * ROUTINE ADDRESS
           4657         00310 PSTGB4  EQU     $4657
                        00320
46C1                    00330         ORG     $46C1
                        00340
46C1 34   30            00350 PSET41  PSHS    X,Y
                        00360
                        00370 * AT THIS POINT THE
                        00380 * STACK CONTAINS
                        00390 *   3,S RTS ADDRESS LOW BYTE
                        00400 *   2,S RTS ADDRESS HIGH BYTE
                        00410 *   1,S REGISTER X LOW BYTE
                        00420 *    ,S REGISTER X HIGH BYTE
                        00430
                        00440 * MAKE ROOM FOR
```

```
                        00450 *   5,S RTS ADDRESS LOW BYTE
                        00460 *   4,S RTS ADDRESS HIGH BYTE
                        00470 *   3,S REGISTER X LOW BYTE
                        00480 *   2,S REGISTER X HIGH BYTE
                        00490 *   1,S BYTE CONTENTS
                        00500 *    ,S PIXEL MASK
     46C3 32   7E       00510         LEAS    -2,S
                        00520
                        00530 * GO GET BYTE DATA
     46C5 BD   4657     00540         JSR     PSTGB4
                        00550
                        00560 * NOW:
                        00570 * A = BIT POSITION (0-7)
                        00580 * B = BYTE CONTENTS
                        00590 * X = BYTE ADDRESS
                        00600
                        00610 * SAVE THE BYTE CONTENTS
     46C8 E7   61       00620         STB     1,S
                        00630
                        00640 * SAVE THE BYTE ADDRESS
     46CA 1F   12       00650         TFR     X,Y
                        00660
                        00670 * ALL ZEROES MASK TO
                        00680 * SET BIT = 1 = GREEN
                        00690 * USING "OR"
     46CC 48            00700         LSLA            A = A * 4
     46CD 48            00710         LSLA            (4 BYTES
PER LBL)
     46CE 8E   46D3     00720         LDX     #LBL000
     46D1 6E   86       00730         JMP     A,X
     46D3 C6   01       00740 LBL000  LDB     #$01
     46D5 20   1A       00750         BRA     LBL008
     46D7 C6   02       00760 LBL001  LDB     #$02
     46D9 20   16       00770         BRA     LBL008
     46DB C6   04       00780 LBL002  LDB     #$04
     46DD 20   12       00790         BRA     LBL008
     46DF C6   08       00800 LBL003  LDB     #$08
     46E1 20   0E       00810         BRA     LBL008
     46E3 C6   10       00820 LBL004  LDB     #$10
     46E5 20   0A       00830         BRA     LBL008
     46E7 C6   20       00840 LBL005  LDB     #$20
     46E9 20   06       00850         BRA     LBL008
     46EB C6   40       00860 LBL006  LDB     #$40
     46ED 20   02       00870         BRA     LBL008
     46EF C6   80       00880 LBL007  LDB     #$80
     46F1 E7   E4       00890 LBL008  STB     ,S      PIXEL MASK
                        00900
```

```
                        00910 * SET THE PIXEL TO 1 = GREEN
     46F3 1F    21      00920         TFR     Y,X     BYTE
ADDRESS
     46F5 A6    61      00930         LDA     1,S     BYTE
CONTENTS
     46F7 AA    E4      00940         ORA     ,S      PIXEL MASK
     46F9 A7    84      00950         STA     ,X      PUT TO
SCREEN
                        00960
                        00970 * CLEAN THE STACK AND RETURN
     46FB 32    62      00980         LEAS    2,S
     46FD 35    30      00990         PULS    X,Y
     46FF 39            01000         RTS
                        01010
                0000    01020         END
```

_____

The Assembly Language Test Routine:

```
                    00100 *****
                    00110 *
                    00120 * TEST0202.ASM
                    00130 * MDJ 2023/03/26
                    00140 *
                    00150 * SECOND PMODE 4
                    00160 * GRAPHICS TEST
                    00170 *
                    00180 *****
                    00190
                    00200 * MLCORE ADDRESS
          4142      00210 POLCAT  EQU     $4142
                    00220
                    00230 * GRAPHICS ROUTINES
                    00240 * ADDRESSES
          4574      00250 STSAMG  EQU     $4574
          4588      00260 STSAM2  EQU     $4588
          458F      00270 SVT     EQU     $458F
          45D4      00280 SVG410  EQU     $45D4
          4608      00290 SVG450  EQU     $4608
          4642      00300 PCLS4   EQU     $4642
          4657      00310 PSTGB4  EQU     $4657
          4682      00320 PSET40  EQU     $4682
          46C1      00330 PSET41  EQU     $46C1
                    00340
7000                00350         ORG     $7000
                    00360
```

```
7000 34   06       00370         PSHS    A,B
                   00380
                   00390 * SET PMODE 4 PAGE 1
7002 BD   45D4     00400         JSR     SVG410
                   00410
                   00420 * CLEAR THE SCREEN TO GREEN
7005 86   01       00430         LDA     #$01
7007 BD   4642     00440         JSR     PCLS4
                   00450
                   00460 * PUT A BLACK DOT CTR OF GREEN SCREEN
700A 86   5C       00470         LDA     #92     Y
700C C6   80       00480         LDB     #128    X
700E BD   4682     00490         JSR     PSET40  BLACK DOT
                   00500
                   00510 * WAIT FOR A KEYPRESS
7011 34   03       00520         PSHS    A,CC
7013 BD   4142     00530 LBL001  JSR     POLCAT
7016 27   FB       00540         BEQ     LBL001  NO KEYPRESS
7018 35   03       00550         PULS    A,CC
                   00560
                   00570 * CLEAR THE SCREEN TO BLACK
701A 86   00       00580         LDA     #$00
701C BD   4642     00590         JSR     PCLS4
                   00600
                   00610 * PUT A GREEN DOT CTR OF BLACK SCREEN
701F 86   5C       00620         LDA     #92     Y
7021 C6   80       00630         LDB     #128    X
7023 BD   46C1     00640         JSR     PSET41  GREEN DOT
                   00650
                   00660 * WAIT FOR A KEYPRESS
7026 34   03       00670         PSHS    A,CC
7028 BD   4142     00680 LBL002  JSR     POLCAT
702B 27   FB       00690         BEQ     LBL002  NO KEYPRESS
702D 35   03       00700         PULS    A,CC
                   00710
                   00720 * SET PMODE 4 PAGE 5
702F BD   4608     00730         JSR     SVG450
                   00740
                   00750 * CLEAR THE SCREEN TO GREEN
7032 86   01       00760         LDA     #$01
7034 BD   4642     00770         JSR     PCLS4
                   00780
                   00790 * PUT A BLACK DOT CTR OF GREEN SCREEN
7037 86   5C       00800         LDA     #92     Y
7039 C6   80       00810         LDB     #128    X
703B BD   4682     00820         JSR     PSET40  BLACK DOT
                   00830
```

```
                    00840 * WAIT FOR A KEYPRESS
703E 34   03      00850         PSHS    A,CC
7040 BD   4142    00860 LBL003  JSR     POLCAT
7043 27   FB      00870         BEQ     LBL003  NO KEYPRESS
7045 35   03      00880         PULS    A,CC
                    00890
                    00900 * CLEAR THE SCREEN TO BLACK
7047 86   00      00910         LDA     #$00
7049 BD   4642    00920         JSR     PCLS4
                    00930
                    00940 * PUT A GREEN DOT CTR OF BLACK SCREEN
704C 86   5C      00950         LDA     #92     Y
704E C6   80      00960         LDB     #128    X
7050 BD   46C1    00970         JSR     PSET41  GREEN DOT
                    00980
                    00990 * WAIT FOR A KEYPRESS
7053 34   03      01000         PSHS    A,CC
7055 BD   4142    01010 LBL004  JSR     POLCAT
7058 27   FB      01020         BEQ     LBL004  NO KEYPRESS
705A 35   03      01030         PULS    A,CC
                    01040
                    01050 * RETURN TO TEXT MODE
705C BD   458F    01060         JSR     SVT
                    01070
                    01080 * WAIT FOR A KEYPRESS
705F 34   03      01090         PSHS    A,CC
7061 BD   4142    01100 LBL005  JSR     POLCAT
7064 27   FB      01110         BEQ     LBL005  NO KEYPRESS
7066 35   03      01120         PULS    A,CC
                    01130
7068 35   06      01140         PULS    A,B
706A 39           01150         RTS
                    01160
          0000    01170         END
```

Making the Final MLFT File: This program combines all the Graphics Control Routines into the single MLFT.BIN file so that the entire machine language Graphics Control System can be loaded as a single entity.

Note that the FLSYS.BIN file specified on Line 2000 is provided by the MAKEFALS.BAS program in ([MDJ01], p.104). It is not used in these Graphics Control Routines, but is included as a check that the MLCORE, False Disk Routines and Graphics Control Routines all fit together properly without any erroneous overlaps.

```
1000 '*****
1010 '*
1020 '* MAKEMLFT.BAS
1030 '* MDJ 2023/03/26
1040 '*
1050 '* MAKES THE
1060 '* MLFT.BIN FILE
1070 '*
1080 '*****
1090 '

1500 CLEAR 200, &H4000
1510 '

2000 LOADM "FLSYS.BIN"
2010 LOADM "STSAMG.BIN"
2020 LOADM "SVT.BIN"
2030 LOADM "SVG410.BIN"
2030 LOADM "SVG450.BIN"
2040 LOADM "PCLS4.BIN"
2050 LOADM "PSTGB4.BIN"
2060 LOADM "PSET40.BIN"
2070 LOADM "PSET41.BIN"
2080 '

3000 SAVEM "MLFT.BIN", &H4416, &H46FF, &H4416
3010 '

32767 END
```

The BASIC Language Control Program:

```
1000 '*****
1010 '*
1020 '* TEST0202.BAS
1030 '* MDJ 2023/03/26
1040 '*
1050 '* SECOND PMODE 4
1060 '* GRAPHICS TEST
1070 '*
1080 '*****
1090 '

1100 'SETUP MEMORY
1110 CLEAR 0, &H4000
```

```
1120 PCLEAR 8
1120 '

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

1300 'LOAD THE MLFT FILE
1310 LOADM "MLFT.BIN"
1320 '

1400 'LOAD THE
1410 'ML TEST ROUTINE
1420 LOADM "TEST0202.BIN"
1430 '

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

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

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

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

32767 END
```

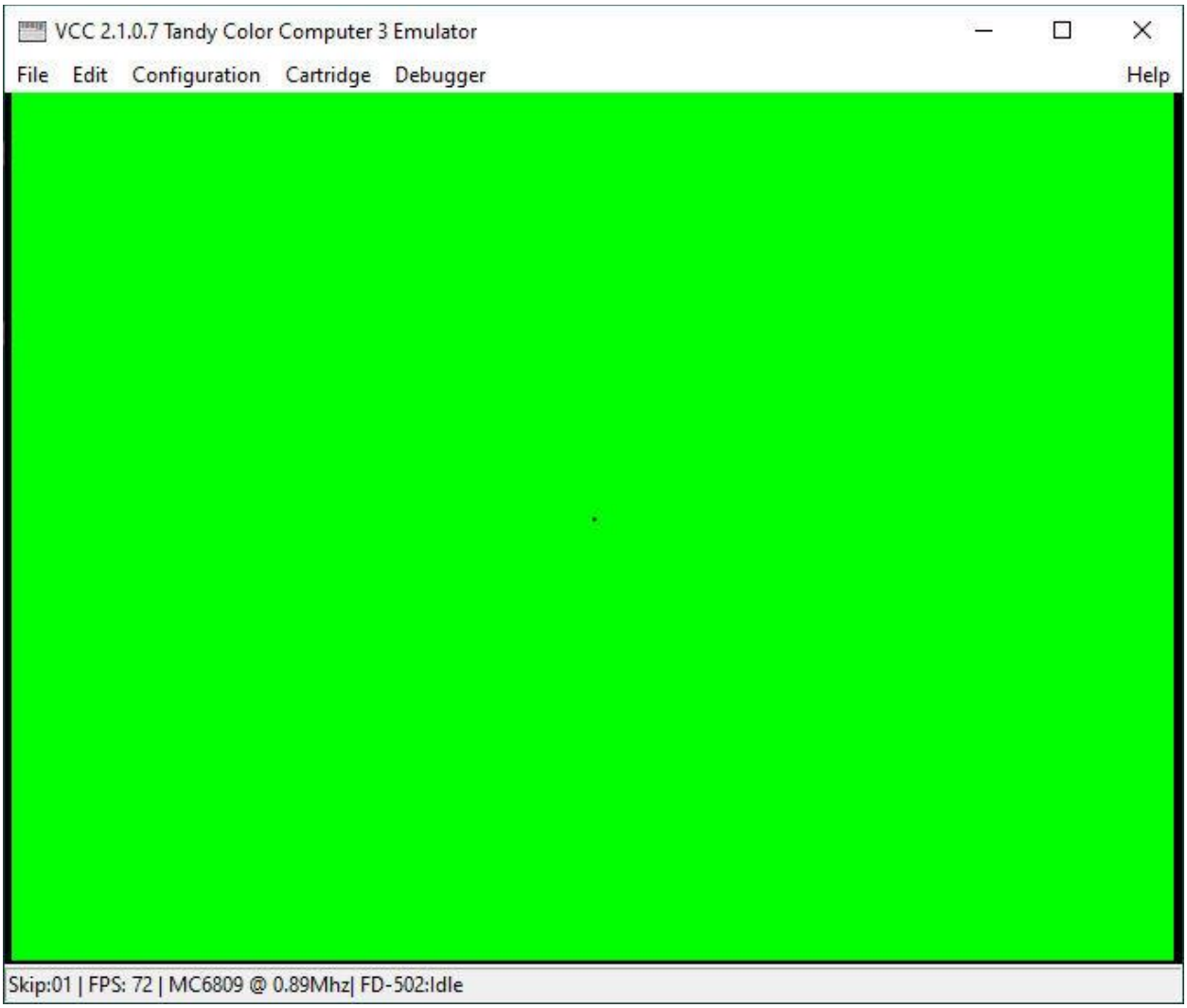
The BASIC Language Control Program, Abbreviated: The full program, as written above, bombs with a reported OM ERROR. This is intentional - The entire ML Foundation System is intended to be initialized using the absolute minimum of BASIC code; just enough to get the Machine Language system loaded and started. It is the following abbreviated BASIC code which is on disk as TEST0202.BAS.

```
1110 CLEAR0,&H4000
1120 PCLEAR8
1220 LOADM"MLCORE.BIN"
1310 LOADM"MLFT.BIN"
1420 LOADM"TEST0202.BIN"
2920 RA=&H400A
2930 RB=&H400B
3020 C=&H7000
3030 C1=INT(C/256)
3040 C2=INT(C-(C1*256))
3050 POKERA,C1
3060 POKERB,C2
6020 EXEC&H4403
9020 PRINT
9030 PRINT" MEM = ";MEM
9040 PRINT"FREE = ";FREE(0)
32767 END
```

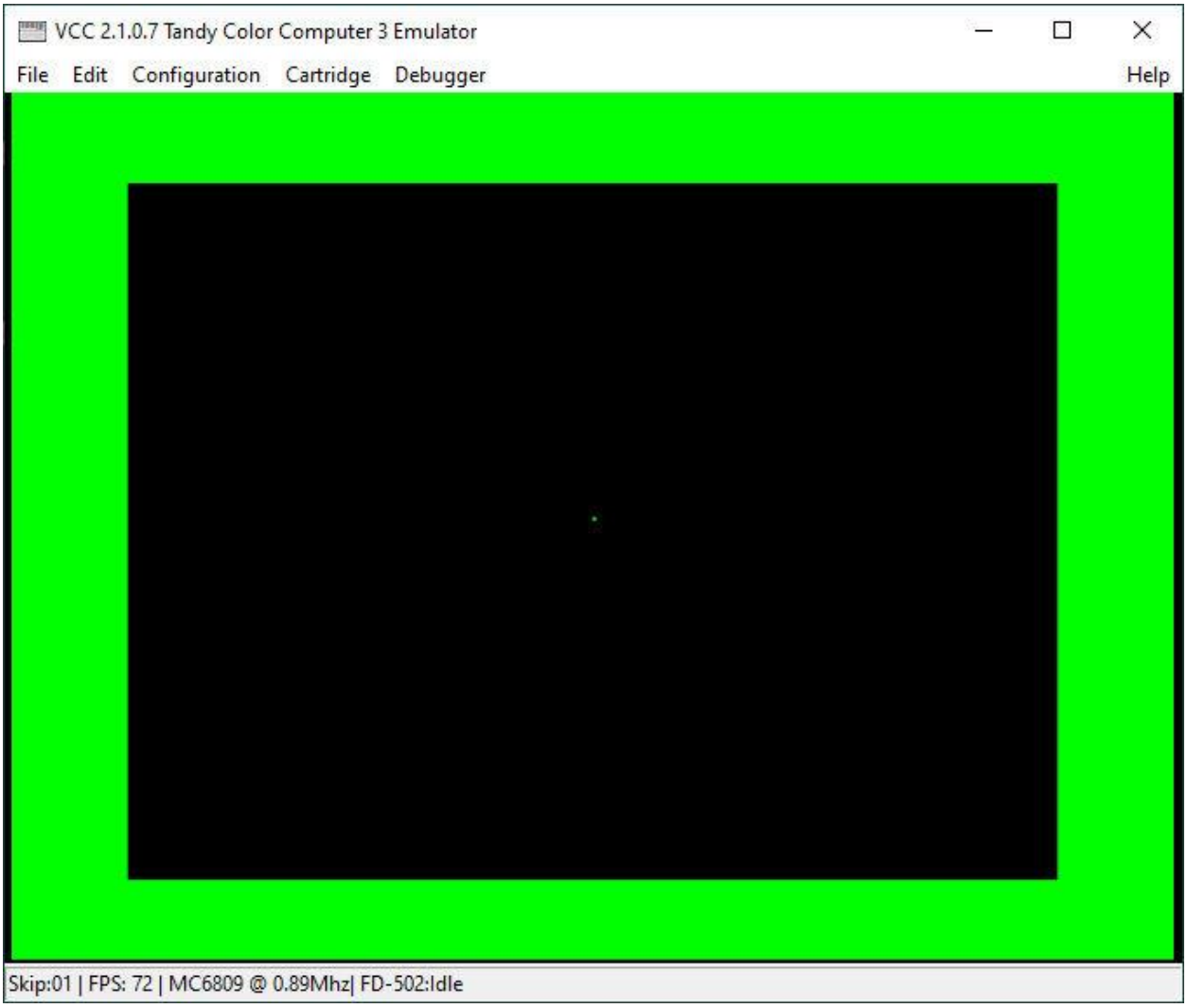
_____

Results:

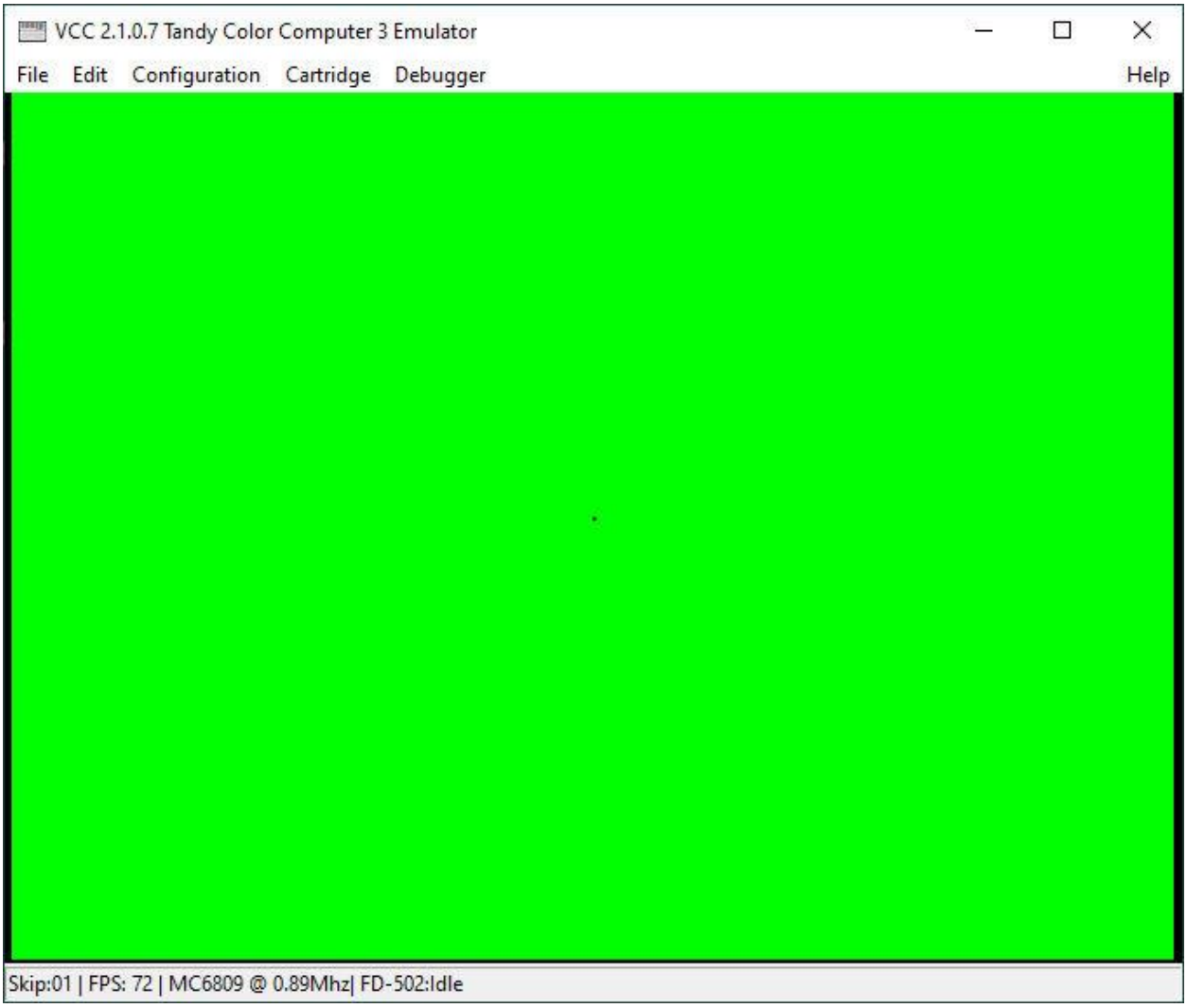
On Startup of TEST0202.BAS: PMODE 4. Start Page 1; cleared to Green, with central Black pixel.



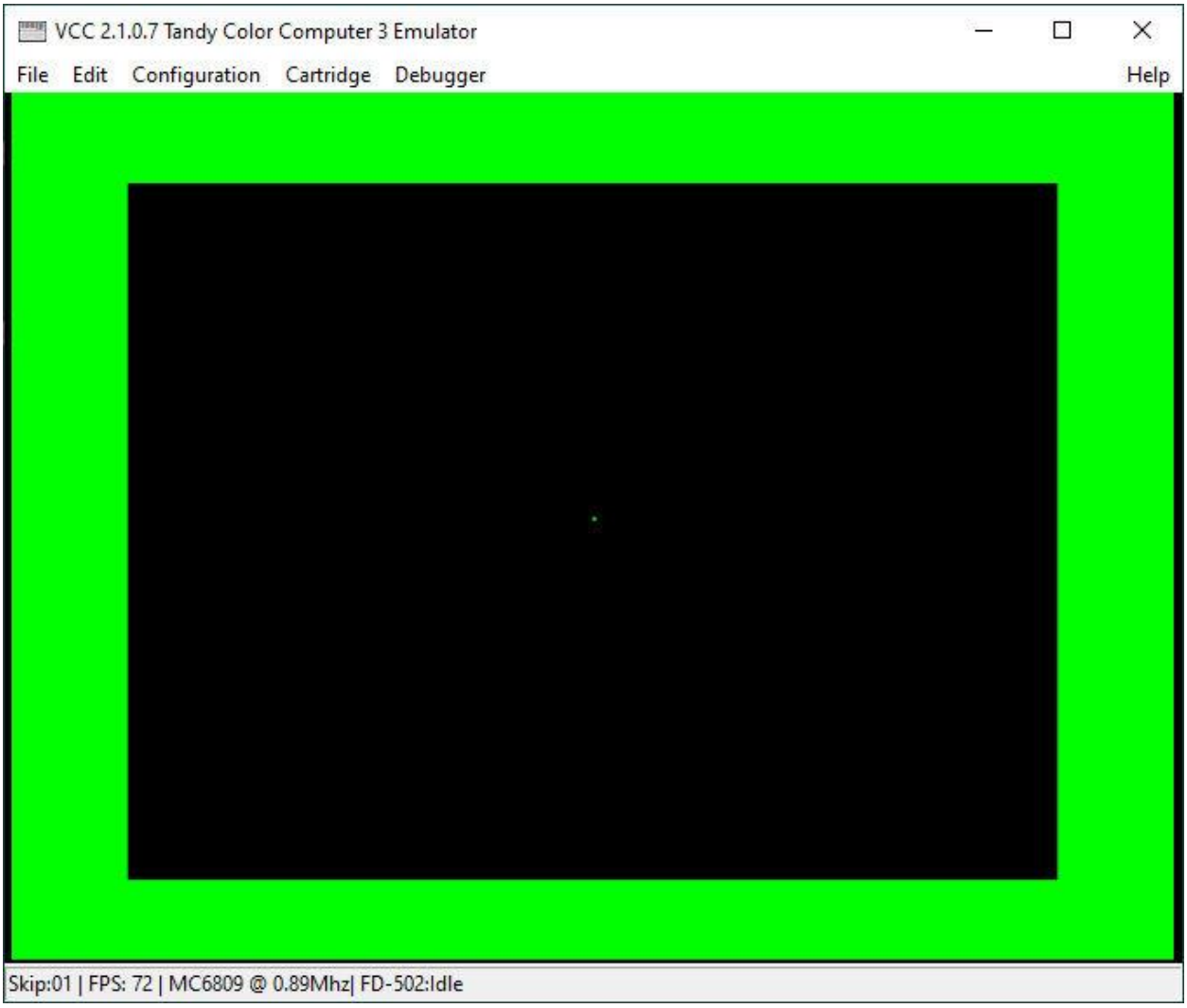
On First Keypress: PMODE 4. Start Page 1; cleared to Black, with central Green pixel.



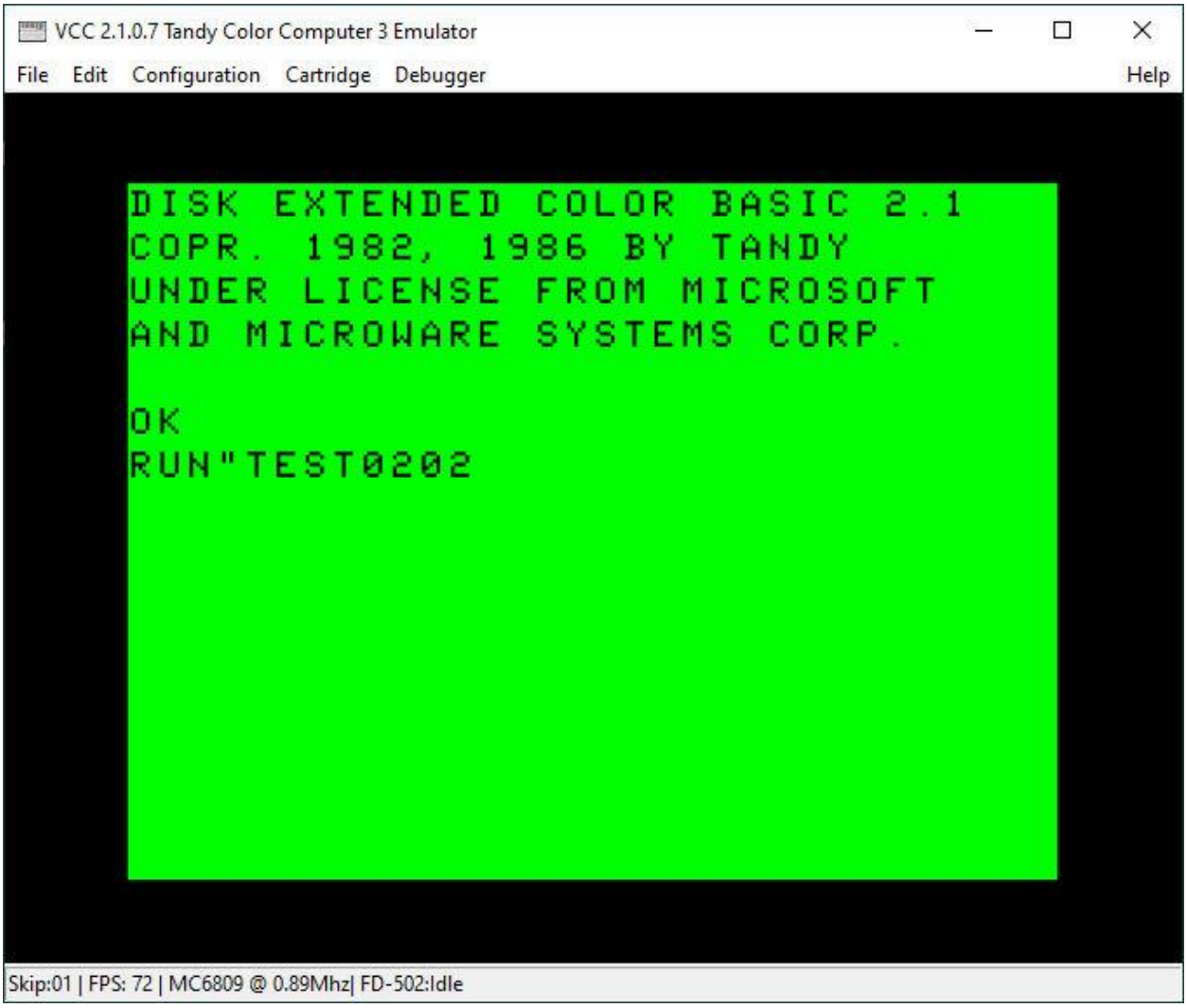
On Second Keypress: PMODE 4. Start Page 5; cleared to Green, with central Black pixel.



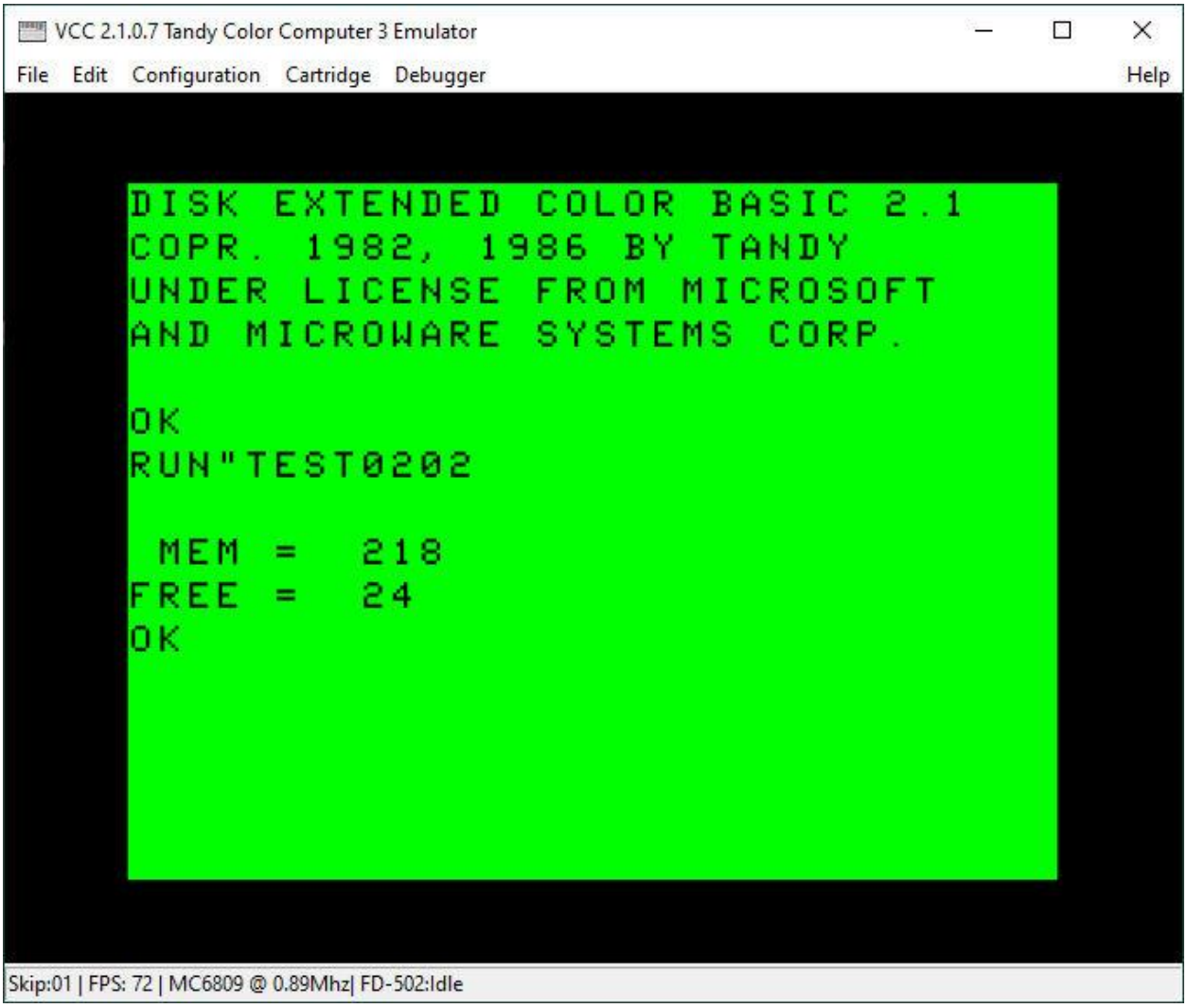
On Third Keypress: PMODE 4. Start Page 5; cleared to Black, with central Green pixel.



On Fourth Keypress: The return to Text Mode.



On Fiifth Keypress: After exiting the TEST0202.BAS program.



All as expected.

=====

Results

These Graphics Control Routines, although Preliminary and for PMODE 4 only, will nevertheless serve to enable the subsequent development of the projected Fake Text PMODE 4 (maze-like) game as originally envisioned.

Finally, I present MAKEMLGC,BAS, which is identical to MAKEMLFT.BAS, except that it excludes the FLSYS.BIN file from [MDJ01]. This will allow the False Disk Routines and the Preliminary Graphics Control Routines to stand separately and on their own within the ML Foundation System.

```
1000 '*****
1010 '*
1020 '* MAKEMLGC.BAS
1030 '* MDJ 2023/04/06
1040 '*
1050 '* MAKES THE
1060 '* MLGC.BIN FILE
1070 '*
1080 '*****
1090 '

1500 CLEAR 200, &H4000
1510 '

2010 LOADM "STSAMG.BIN"
2020 LOADM "SVT.BIN"
2030 LOADM "SVG410.BIN"
2030 LOADM "SVG450.BIN"
2040 LOADM "PCLS4.BIN"
2050 LOADM "PSTGB4.BIN"
2060 LOADM "PSET40.BIN"
2070 LOADM "PSET41.BIN"
2080 '

3000 SAVEM "MLGC.BIN", &H4574, &H46FF, &H4574
3010 '

32767 END
```

=====

Conclusions and Future Work

The Preliminary Graphics Control Routines for the ML Foundation System are complete and functional (subject, of course, to the correction of any as-yet-undiscovered bugs).

They cover the setup of PMODE 4 and the PSET routines for PMODE 4.

Future work should include:

1. Setup and PSET routines for PMODES 0-3.

2, More complex graphics routines for PMODES 0-4, including, but not necessarily limited to:

DRAW
GET
LINE
PAINT
PCOPY
PPOINT
PUT

=====

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: MLGC Tests

Machine Language Graphics Control Parameters Lists Reporting Program:

```
1000 '*****
1010 '*
1020 '* MLGCTST2.BAS
1030 '* MDJ 2023/03/14
1040 '*
1070 '* REPORTS
1080 '* GRAPHICS
1090 '* PARAMETERS
1100 '*
1110 '* CF. EXTENDED BASIC
1120 '*     UNRAVELLED,
1130 '*     PAGE B-31
1140 '*
1150 '* RUN IMMEDIATELY AFTER
1160 '* STARTUP TO AVOID ANY
1170 '* PRECONTAMINATION OF
1180 '* THE PARAMETERS
1190 '*
1200 '* NOTE: ALL OUTPUT IS
1210 '*       TO THE PRINTER
1220 '*
1230 '*****
1240 '
1250 L = 0  'LINE COUNTER
1260 'TEMPORARY VARIABLES
1270 ' = A, A8, A9, B, I, T, X8, X9

1300 CLEAR &H200
1310 PCLEAR 8
1320 PRINT #-2, "*****"
1330 PRINT #-2, "*"
1340 PRINT #-2, "* MLGCTTST2.BAS"
1350 PRINT #-2, "* GRAPHICS PARAMETERS"
1360 PRINT #-2, "*"
1370 PRINT #-2, "*****"
1380 PRINT #-2, " "
1390 '

1400 'BASE RUN
1410 PRINT #-2, "*****"
1420 PRINT #-2, "*"
1430 PRINT #-2, "* BASE: TEXT MODE"
```

```
1440 PRINT #-2, "*"
1450 PRINT #-2, "*****"
1460 PRINT #-2, " "
1470 L = 13
1480 GOSUB 10000  'GET & REPORT GRAPHIC VARIABLES
1490 GOSUB 16000  'NEXT PAGE
1500 '

2500 'PMODE 0 RUNS
2510 P9 = 0  'PMODE
2520 FOR G9 = 1 TO 8  'PAGE
2530   FOR C9 = 0 TO 1  'COLOR SET
2540     PRINT #-2, "PMODE=";P9;", PAGE=";G9;", COLORSET=";C9
2550     L = L + 1
2560     PMODE P9, G9
2570     SCREEN 0, C9
2580     GOSUB 10000  'GET & REPORT GRAPHIC VARIABLES
2590     GOSUB 11000  'DO AND REPORT PRECALC
2600     GOSUB 12000  'DO AND REPORT OR WITH CSSVAL
2610     GOSUB 13000  'DO AND REPORT MIDCALC
2620     GOSUB 14000  'DO AND REPORT POSTCALC
2630     GOSUB 15000  'DO AND REPORT ENDCALC
2640     GOSUB 16000  'NEXT PAGE
2650     SCREEN 1, 0
2660   NEXT C9
2670 NEXT G9
2680 '

3500 'PMODE 1 RUNS
3510 P9 = 1  'PMODE
3520 FOR G9 = 1 TO 7  'PAGE
3530   FOR C9 = 0 TO 1  'COLOR SET
3540     PRINT #-2, "PMODE=";P9;", PAGE=";G9;", COLORSET=";C9
3550     L = L + 1
3560     PMODE P9, G9
3570     SCREEN 0, C9
3580     GOSUB 10000  'GET & REPORT GRAPHIC VARIABLES
3590     GOSUB 11000  'DO AND REPORT PRECALC
3600     GOSUB 12000  'DO AND REPORT OR WITH CSSVAL
3610     GOSUB 13000  'DO AND REPORT MIDCALC
3620     GOSUB 14000  'DO AND REPORT POSTCALC
3630     GOSUB 15000  'DO AND REPORT ENDCALC
3640     GOSUB 16000  'NEXT PAGE
3650     SCREEN 1, 0
3660   NEXT C9
3670 NEXT G9
3680 '
```

```
4500 'PMODE 2 RUNS
4510 P9 = 2  'PMODE
4520 FOR G9 = 1 TO 7  'PAGE
4530   FOR C9 = 0 TO 1  'COLOR SET
4540     PRINT #-2, "PMODE=";P9;", PAGE=";G9;", COLORSET=";C9
4550     L = L + 1
4560     PMODE P9, G9
4570     SCREEN 0, C9
4580     GOSUB 10000  'GET & REPORT GRAPHIC VARIABLES
4590     GOSUB 11000  'DO AND REPORT PRECALC
4600     GOSUB 12000  'DO AND REPORT OR WITH CSSVAL
4610     GOSUB 13000  'DO AND REPORT MIDCALC
4620     GOSUB 14000  'DO AND REPORT POSTCALC
4630     GOSUB 15000  'DO AND REPORT ENDCALC
4640     GOSUB 16000  'NEXT PAGE
4650     SCREEN 1, 0
4660   NEXT C9
4670 NEXT G9
4680 '

5500 'PMODE 3 RUNS
5510 P9 = 3  'PMODE
5520 FOR G9 = 1 TO 5  'PAGE
5530   FOR C9 = 0 TO 1  'COLOR SET
5540     PRINT #-2, "PMODE=";P9;", PAGE=";G9;", COLORSET=";C9
5550     L = L + 1
5560     PMODE P9, G9
5570     SCREEN 0, C9
5580     GOSUB 10000  'GET & REPORT GRAPHIC VARIABLES
5590     GOSUB 11000  'DO AND REPORT PRECALC
5600     GOSUB 12000  'DO AND REPORT OR WITH CSSVAL
5610     GOSUB 13000  'DO AND REPORT MIDCALC
5620     GOSUB 14000  'DO AND REPORT POSTCALC
5630     GOSUB 15000  'DO AND REPORT ENDCALC
5640     GOSUB 16000  'NEXT PAGE
5650     SCREEN 1, 0
5660   NEXT C9
5670 NEXT G9
5680 '

6500 'PMODE 4 RUNS
6510 P9 = 4  'PMODE
6520 FOR G9 = 1 TO 5  'PAGE
6530   FOR C9 = 0 TO 1  'COLOR SET
6540     PRINT #-2, "PMODE=";P9;", PAGE=";G9;", COLORSET=";C9
6550     L = L + 1
```

```
6560     PMODE P9, G9
6570     SCREEN 0, C9
6580     GOSUB 10000  'GET & REPORT GRAPHIC VARIABLES
6590     GOSUB 11000  'DO AND REPORT PRECALC
6600     GOSUB 12000  'DO AND REPORT OR WITH CSSVAL
6610     GOSUB 13000  'DO AND REPORT MIDCALC
6620     GOSUB 14000  'DO AND REPORT POSTCALC
6630     GOSUB 15000  'DO AND REPORT ENDCALC
6640     GOSUB 16000  'NEXT PAGE
6650     SCREEN 1, 0
6660   NEXT C9
6670 NEXT G9
6680 '

7500 PRINT #-2, " "
7510 PRINT #-2, " "
7520 PRINT #-2, "*****"
7530 PRINT #-2, "*"
7540 PRINT #-2, "* END OF RUN"
7550 PRINT #-2, "*"
7560 PRINT #-2, "*****"
7570 GOTO 32767

10000 'GET & REPORT GRAPHIC VARIABLES
10030 B2 = PEEK(&H00B2)
10040 B3 = PEEK(&H00B3)
10050 B5 = PEEK(&H00B5)
10060 B6 = PEEK(&H00B6)
10070 B7 = PEEK(&H00B7)*256+PEEK(&H00B8)
10080 B9 = PEEK(&H00B9)
10090 BA = PEEK(&H00BA)*256+PEEK(&H00BB)
10100 BC = PEEK(&H00BC)
10110 C1 = PEEK(&H00C1)
10120 DB = PEEK(&H00DB)
10130 SCREEN 1, 0
10140 PRINT #-2, "B2 FORCOL = "; HEX$(B2)
10150 PRINT #-2, "B3 BAKCOL = "; HEX$(B3)
10160 PRINT #-2, "B5 ALLCOL = "; HEX$(B5)
10170 PRINT #-2, "B6 PMODE  = "; HEX$(B6)
10180 PRINT #-2, "B7 ENDGRP = "; HEX$(B7)
10190 PRINT #-2, "B9 HORBYT = "; HEX$(B9)
10200 PRINT #-2, "BA BEGGRP = "; HEX$(BA)
10210 PRINT #-2, "BC GRPRAM = "; HEX$(BC)
10220 PRINT #-2, "C1 CSSVAL = "; HEX$(C1)
10230 PRINT #-2, "DB CHGFLG = "; HEX$(DB)
10240 PRINT #-2, " "
10250 L = L + 11
```

```
10260 RETURN
10270 '

11000 'DO AND REPORT PRECALC
11005 'FOR REFERENCE ONLY
11010 A = B6 + 3     'LDA PMODE, ADDA #3
11020 A = A * 16     'LDB #$10, MUL
11030 B = &H80       'DECIMAL 128
11040 GOSUB 23500    'GO DO A = (A OR B)
11050 'PRINT #-2, "PRECALC   = "; HEX$(A)
11060 'L = L + 1
11070 RETURN
11080 '

12000 'DO AND REPORT OR WITH CSSVAL
12005 'FOR REFERENCE ONLY
12010 B = C1         'CSSVAL
12020 GOSUB 23500    'GO DO A = (A OR B)
12030 'PRINT #-2, "OR CSSVAL = "; HEX$(A)
12040 'L = L + 1
12050 RETURN
12060 '

13000 'DO AND REPORT MIDCALC
13005 'TO PIA1+2
13010 T = A
13020 A = PEEK(&HFF22)  'PIA1+2 VALUE
13030 B = &H07       'DECIMAL 7
13040 GOSUB 23000    'GO DO A = (A AND B)
13050 B = T
13060 GOSUB 23500    'GO DO A = (A OR B)
13070 PRINT #-2, "MIDCALC   = "; HEX$(A)
13080 L = L + 1
13090 RETURN
13100 '

14000 'DO AND REPORT POSTCALC
14005 'LDA POSTCALC, THEN JSR STSAM2
14010 A = INT(BA/256)  'MSB OF BEGGRP
14020 A = A/2          'LSRA
14030 PRINT #-2, "POSTCALC  = "; HEX$(A)
14040 L = L + 1
14050 RETURN
14060 '

15000 'DO AND REPORT ENDCALC
15005 'LDA ENDCALC, THEN JSR STSAMG
```

```
15010 A = B6 + 3    'LDA PMODE, ADDA #3
15020 'DECREMENT IF PMODE = 4
15030 IF (A = 7) THEN A = 6
15040 PRINT #-2, "ENDCALC   = "; HEX$(A)
15050 L = L + 1
15060 RETURN
15070 '

16000 'NEXT PAGE
16010 T = 57 - L
16020 FOR I = 0 TO (T-1)
16030   PRINT #-2, " "
16040 NEXT I
16050 L = 0  'RESET LINE COUNT
16060 RETURN
16070 '

20000 'CONVERT AN UNSIGNED INTEGER
20010 'IN REGISTER "A" TO INDIVIDUAL BITS.
20020 'ENTER WITH A = THE UNSIGNED INTEGER.
20030 'EXIT WITH A0-A7 = THE BITS.
20040 A8 = INT(A)
20050 A9 = INT(A8/2)
20060 A0 = A8 - (A9*2)
20070 A8 = A9
20080 A9 = INT(A8/2)
20090 A1 = A8 - (A9*2)
20100 A8 = A9
20110 A9 = INT(A8/2)
20120 A2 = A8 - (A9*2)
20130 A8 = A9
20140 A9 = INT(A8/2)
20150 A3 = A8 - (A9*2)
20160 A8 = A9
20170 A9 = INT(A8/2)
20180 A4 = A8 - (A9*2)
20190 A8 = A9
20200 A9 = INT(A8/2)
20210 A5 = A8 - (A9*2)
20220 A8 = A9
20230 A9 = INT(A8/2)
20240 A6 = A8 - (A9*2)
20250 A8 = A9
20260 A9 = INT(A8/2)
20270 A7 = A8 - (A9*2)
20280 RETURN
20290 '
```

```
20500 'CONVERT INDIVIDUAL BITS IN
20510 'REGISTER "A" TO AN UNSIGNED INTEGER.
20520 'ENTER WITH A0-A7 = THE BITS.
20530 'EXIT WITH A = THE UNSIGNED INTEGER.
20540 A = A7*128 + A6*64 + A5*32 + A4*16
20550 A = A + A3*8 + A2*4 + A1*2 + A0*1
20560 RETURN
20570 '

21000 'CONVERT AN UNSIGNED INTEGER
21010 'IN REGISTER "B" TO INDIVIDUAL BITS.
21020 'ENTER WITH B = THE UNSIGNED INTEGER.
21030 'EXIT WITH B0-B7 = THE BITS.
21040 X8 = INT(B)
21050 X9 = INT(X8/2)
21060 X0 = X8 - (X9*2)
21070 X8 = X9
21080 X9 = INT(X8/2)
21090 X1 = X8 - (X9*2)
21100 X8 = X9
21110 X9 = INT(X8/2)
21120 X2 = X8 - (X9*2)
21130 X8 = X9
21140 X9 = INT(X8/2)
21150 X3 = X8 - (X9*2)
21160 X8 = X9
21170 X9 = INT(X8/2)
21180 X4 = X8 - (X9*2)
21190 X8 = X9
21200 X9 = INT(X8/2)
21210 X5 = X8 - (X9*2)
21220 X8 = X9
21230 X9 = INT(X8/2)
21240 X6 = X8 - (X9*2)
21250 X8 = X9
21260 X9 = INT(X8/2)
21270 X7 = X8 - (X9*2)
21280 RETURN
21290 '

23000 'BITWISE (A AND B) --> A
23010 GOSUB 20000               'A: UINT-->BITS
23020 GOSUB 21000               'B: UINT-->BITS
23030 IF ((A0 = 1) AND (X0 = 1)) THEN A0 = 1 ELSE A0 = 0
23040 IF ((A1 = 1) AND (X1 = 1)) THEN A1 = 1 ELSE A1 = 0
23050 IF ((A2 = 1) AND (X2 = 1)) THEN A2 = 1 ELSE A2 = 0
```

```
23060 IF ((A3 = 1) AND (X3 = 1)) THEN A3 = 1 ELSE A3 = 0
23070 IF ((A4 = 1) AND (X4 = 1)) THEN A4 = 1 ELSE A4 = 0
23080 IF ((A5 = 1) AND (X5 = 1)) THEN A5 = 1 ELSE A5 = 0
23090 IF ((A6 = 1) AND (X6 = 1)) THEN A6 = 1 ELSE A6 = 0
23100 IF ((A7 = 1) AND (X7 = 1)) THEN A7 = 1 ELSE A7 = 0
23110 GOSUB 20500                'A: BITS-->UINT
23120 RETURN
23130 '

23500 'BITWISE (A OR B) --> A
23510 GOSUB 20000                'A: UINT-->BITS
23520 GOSUB 21000                'B: UINT-->BITS
23530 IF ((A0 = 1) OR (X0 = 1)) THEN A0 = 1 ELSE A0 = 0
23540 IF ((A1 = 1) OR (X1 = 1)) THEN A1 = 1 ELSE A1 = 0
23550 IF ((A2 = 1) OR (X2 = 1)) THEN A2 = 1 ELSE A2 = 0
23560 IF ((A3 = 1) OR (X3 = 1)) THEN A3 = 1 ELSE A3 = 0
23570 IF ((A4 = 1) OR (X4 = 1)) THEN A4 = 1 ELSE A4 = 0
23580 IF ((A5 = 1) OR (X5 = 1)) THEN A5 = 1 ELSE A5 = 0
23590 IF ((A6 = 1) OR (X6 = 1)) THEN A6 = 1 ELSE A6 = 0
23600 IF ((A7 = 1) OR (X7 = 1)) THEN A7 = 1 ELSE A7 = 0
23610 GOSUB 20500                'A: BITS-->UINT
23620 RETURN
23630 '

32767 END
```

=====

Appendix C:
Graphics PMODE Parameters

Machine Language Graphics Control Parameters Lists:

```
*****
*
* MLGCTTST2A.BAS ABBREVIATED
* GRAPHICS PARAMETERS
*
*****

*****
*
* BASE: TEXT MODE
*
*****

B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 1400
B9 HORBYT = 10
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0


*****

PMODE= 0 , PAGE= 1 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 1400
B9 HORBYT = 10
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0
```

```
MIDCALC   = B0
POSTCALC  = 7
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 1 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 1400
B9 HORBYT = 10
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC  = 7
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 2 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 1A00
B9 HORBYT = 10
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = B0
POSTCALC  = A
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 2 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
```

B6 PMODE = 0
B7 ENDGRP = 1A00
B9 HORBYT = 10
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC = A
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 3 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 2000
B9 HORBYT = 10
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = B0
POSTCALC = D
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 3 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 2000
B9 HORBYT = 10
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC = D

```
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 4 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 2600
B9 HORBYT = 10
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = B0
POSTCALC  = 10
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 4 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 2600
B9 HORBYT = 10
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC  = 10
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 5 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 2C00
```

B9 HORBYT = 10
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = B0
POSTCALC = 13
ENDCALC = 3

*****

PMODE= 0 , PAGE= 5 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 2C00
B9 HORBYT = 10
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = B8
POSTCALC = 13
ENDCALC = 3

*****

PMODE= 0 , PAGE= 6 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 3200
B9 HORBYT = 10
BA BEGGRP = 2C00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = B0
POSTCALC = 16
ENDCALC = 3

*****

PMODE= 0 , PAGE= 6 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 3200
B9 HORBYT = 10
BA BEGGRP = 2C00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC = 16
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 7 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 3800
B9 HORBYT = 10
BA BEGGRP = 3200
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = B0
POSTCALC = 19
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 7 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 0
B7 ENDGRP = 3800
B9 HORBYT = 10
BA BEGGRP = 3200

```
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC  = 19
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 8 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 3E00
B9 HORBYT = 10
BA BEGGRP = 3800
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = B0
POSTCALC  = 1C
ENDCALC   = 3

*****

PMODE= 0 , PAGE= 8 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 0
B7 ENDGRP = 3E00
B9 HORBYT = 10
BA BEGGRP = 3800
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = B8
POSTCALC  = 1C
ENDCALC   = 3

*****
```

PMODE= 1 , PAGE= 1 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 1A00
B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = C0
POSTCALC = 7
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 1 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 1A00
B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = C8
POSTCALC = 7
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 2 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 2000
B9 HORBYT = 20
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 0

```
DB CHGFLG = 0

MIDCALC   = C0
POSTCALC  = A
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 2 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 1
B7 ENDGRP = 2000
B9 HORBYT = 20
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = C8
POSTCALC  = A
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 3 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 1
B7 ENDGRP = 2600
B9 HORBYT = 20
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = C0
POSTCALC  = D
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 3 , COLORSET= 1
B2 FORCOL = 3
```

B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 2600
B9 HORBYT = 20
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = C8
POSTCALC = D
ENDCALC = 4

*****

PMODE= 1 , PAGE= 4 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 2C00
B9 HORBYT = 20
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = C0
POSTCALC = 10
ENDCALC = 4

*****

PMODE= 1 , PAGE= 4 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 2C00
B9 HORBYT = 20
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = C8
POSTCALC = 10
ENDCALC = 4

*****

PMODE= 1 , PAGE= 5 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 3200
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = C0
POSTCALC = 13
ENDCALC = 4

*****

PMODE= 1 , PAGE= 5 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 3200
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = C8
POSTCALC = 13
ENDCALC = 4

*****

PMODE= 1 , PAGE= 6 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0

B6 PMODE = 1
B7 ENDGRP = 3800
B9 HORBYT = 20
BA BEGGRP = 2C00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = C0
POSTCALC = 16
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 6 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 3800
B9 HORBYT = 20
BA BEGGRP = 2C00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = C8
POSTCALC = 16
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 7 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 1
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 3200
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = C0
POSTCALC = 19

```
ENDCALC   = 4

*****

PMODE= 1 , PAGE= 7 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 1
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 3200
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = C8
POSTCALC  = 19
ENDCALC   = 4

*****

PMODE= 2 , PAGE= 1 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 2
B7 ENDGRP = 1A00
B9 HORBYT = 10
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = D0
POSTCALC  = 7
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 1 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 2
B7 ENDGRP = 1A00
```

B9 HORBYT = 10
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = D8
POSTCALC = 7
ENDCALC = 5

*****

PMODE= 2 , PAGE= 2 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 2000
B9 HORBYT = 10
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = D0
POSTCALC = A
ENDCALC = 5

*****

PMODE= 2 , PAGE= 2 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 2000
B9 HORBYT = 10
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = D8
POSTCALC = A
ENDCALC = 5

*****

PMODE= 2 , PAGE= 3 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 2600
B9 HORBYT = 10
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = D0
POSTCALC = D
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 3 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 2600
B9 HORBYT = 10
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = D8
POSTCALC = D
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 4 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 2C00
B9 HORBYT = 10
BA BEGGRP = 2000

```
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = D0
POSTCALC  = 10
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 4 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 2
B7 ENDGRP = 2C00
B9 HORBYT = 10
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = D8
POSTCALC  = 10
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 5 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 2
B7 ENDGRP = 3200
B9 HORBYT = 10
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = D0
POSTCALC  = 13
ENDCALC   = 5

*****
```

PMODE= 2 , PAGE= 5 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 3200
B9 HORBYT = 10
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = D8
POSTCALC = 13
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 6 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 3800
B9 HORBYT = 10
BA BEGGRP = 2C00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = D0
POSTCALC = 16
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 6 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 2
B7 ENDGRP = 3800
B9 HORBYT = 10
BA BEGGRP = 2C00
BC GRPRAM = E
C1 CSSVAL = 8

```
DB CHGFLG = 0

MIDCALC   = D8
POSTCALC  = 16
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 7 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 2
B7 ENDGRP = 3E00
B9 HORBYT = 10
BA BEGGRP = 3200
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = D0
POSTCALC  = 19
ENDCALC   = 5

*****

PMODE= 2 , PAGE= 7 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 2
B7 ENDGRP = 3E00
B9 HORBYT = 10
BA BEGGRP = 3200
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = D8
POSTCALC  = 19
ENDCALC   = 5

*****

PMODE= 3 , PAGE= 1 , COLORSET= 0
B2 FORCOL = 3
```

B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 3
B7 ENDGRP = 2600
B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = E0
POSTCALC = 7
ENDCALC = 6

*****

PMODE= 3 , PAGE= 1 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 3
B7 ENDGRP = 2600
B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = E8
POSTCALC = 7
ENDCALC = 6

*****

PMODE= 3 , PAGE= 2 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 3
B7 ENDGRP = 2C00
B9 HORBYT = 20
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

```
MIDCALC   = E0
POSTCALC  = A
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 2 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 3
B7 ENDGRP = 2C00
B9 HORBYT = 20
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = E8
POSTCALC  = A
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 3 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 3
B7 ENDGRP = 3200
B9 HORBYT = 20
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = E0
POSTCALC  = D
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 3 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
```

B6 PMODE = 3
B7 ENDGRP = 3200
B9 HORBYT = 20
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = E8
POSTCALC = D
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 4 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 3
B7 ENDGRP = 3800
B9 HORBYT = 20
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = E0
POSTCALC = 10
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 4 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 3
B7 ENDGRP = 3800
B9 HORBYT = 20
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = E8
POSTCALC = 10

```
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 5 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 3
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = E0
POSTCALC  = 13
ENDCALC   = 6

*****

PMODE= 3 , PAGE= 5 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 3
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = E8
POSTCALC  = 13
ENDCALC   = 6

*****

PMODE= 4 , PAGE= 1 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 4
B7 ENDGRP = 2600
```

B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = F0
POSTCALC = 7
ENDCALC = 6

*****

PMODE= 4 , PAGE= 1 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 4
B7 ENDGRP = 2600
B9 HORBYT = 20
BA BEGGRP = E00
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC = F8
POSTCALC = 7
ENDCALC = 6

*****

PMODE= 4 , PAGE= 2 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 4
B7 ENDGRP = 2C00
B9 HORBYT = 20
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC = F0
POSTCALC = A
ENDCALC = 6

*****

PMODE= 4 , PAGE= 2 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 4
B7 ENDGRP = 2C00
B9 HORBYT = 20
BA BEGGRP = 1400
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = F8
POSTCALC = A
ENDCALC   = 6

*****

PMODE= 4 , PAGE= 3 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 4
B7 ENDGRP = 3200
B9 HORBYT = 20
BA BEGGRP = 1A00
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = F0
POSTCALC = D
ENDCALC   = 6

*****

PMODE= 4 , PAGE= 3 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE = 4
B7 ENDGRP = 3200
B9 HORBYT = 20
BA BEGGRP = 1A00

```
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = F8
POSTCALC  = D
ENDCALC   = 6

*****

PMODE= 4 , PAGE= 4 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 4
B7 ENDGRP = 3800
B9 HORBYT = 20
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = F0
POSTCALC  = 10
ENDCALC   = 6

*****

PMODE= 4 , PAGE= 4 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 4
B7 ENDGRP = 3800
B9 HORBYT = 20
BA BEGGRP = 2000
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = F8
POSTCALC  = 10
ENDCALC   = 6

*****
```

```
PMODE= 4 , PAGE= 5 , COLORSET= 0
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 4
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 0
DB CHGFLG = 0

MIDCALC   = F0
POSTCALC  = 13
ENDCALC   = 6

*****

PMODE= 4 , PAGE= 5 , COLORSET= 1
B2 FORCOL = 3
B3 BAKCOL = 0
B5 ALLCOL = 0
B6 PMODE  = 4
B7 ENDGRP = 3E00
B9 HORBYT = 20
BA BEGGRP = 2600
BC GRPRAM = E
C1 CSSVAL = 8
DB CHGFLG = 0

MIDCALC   = F8
POSTCALC  = 13
ENDCALC   = 6

*****



*****
*
* END OF RUN
*
*****

        =====
```

Appendix D: 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 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

[MDJ01] Johnson, M. David. *False Disk Routines For The ML Foundation System*. https://www.bds-soft.com/cocoPapers.php, 2023. Online.

Zydhek, Walter K. *Extended BASIC Unravelled II*. 1999. Print.

=====