Piotr and the Seven Segments

Introduction

Not too long ago, our dear friend Piotr, in Poland, undertook the restoration and modernization of two CASIO MD-120 floppy disk drives. He also completed the schematic of the SB-5 interface and built one.

One of those drives remained with Piotr in Poland. The other made her way across the Atlantic to Brookhaven, Georgia, where she joined an ever growing family of vintage CASIO devices. Two old floppy drives, lovingly restored to their 1990 glory, separated by thousands of miles, but sharing a common history and, as it turned out, a few more adventures ahead.

Piotr's curiosity did not end with restoring the hardware. He continued exploring the MD-120 firmware in considerable detail, gradually uncovering more of its secrets. Along the way, he discovered a hidden bug in the original firmware, which we helped investigate and correct. His work continued with an increasingly detailed, annotated disassembly of the ROM, revealing capabilities that had remained undocumented for decades.

Among these discoveries was an intriguing program-loading mechanism, possibly connected with the mysterious expansion-card slot on top of the MD-120. That remains a subject for another day, as there is still much to understand.

But another discovery was considerably simpler, and perhaps more delightful.

Piotr realized that using a diagnostic command he could control the MD-120's seven-segment display.

Now, a seven-segment display is hardly an impressive piece of technology. It consists of seven little illuminated segments, arranged to represent numbers and a limited collection of characters. But is very satisfying to see Piotr’s small assembly-language program display five letters, one after another: **C A S I O**

He generously shared his code with us.

Naturally, we wanted to see whether we could reproduce his result on the MD-120 sitting in Brookhaven.

And that is where our little story about Piotr and the seven segments really begins.

CASIO

Piotr's original program (ch14p1.s) performs the following operations:

1. Send command 05h to enter the MD-120 diagnostic mode.
2. Send command 31h to begin sending seven-segment patterns.
3. Send 4E (C) 77 (A) 5B (S) 30 (I) 7E (O) with a little pause after each one.
4. Send a zero terminator, receive the drive's response
5. Send a 21h command to restart the drive firmware, and return to the pocket computer.

HD61 assembler listing — build/ch14p1.lst

```
HD61700 ASSEMBLER Rev 0.45 - ASSEMBLY LIST OF [ch14p1.s]
00001:0000            ; MD-120 diagnostic command 0x31 on the FX-870P
00002:0000
00003:0000                          org     0
00004:0000                          start   0
00005:0000
00006:0000  420005                  ld      $0,&H05             ;FDD command
&H05: enter the diagnostic mode
00007:0003  778A00                  cal     cm1                 ;send the
command $0 to the FDD
00008:0006  420031                  ld      $0,&H31             ;diagnostic
command &H31: write to the LED
00009:0009  778A00                  cal     cm1                 ;send the
command $0 to the FDD
00010:000C  42004E                  ld      $0,&H4E             ;pattern 'C'
00011:000F  777000                  cal     wr1                 ;write byte $0
to the FDD
00012:0012  774E00                  cal     delay
00013:0015  420077                  ld      $0,&H77             ;pattern 'A'
00014:0018  777000                  cal     wr1                 ;write byte $0
to the FDD
00015:001B  774E00                  cal     delay
00016:001E  42005B                  ld      $0,&H5B             ;pattern 'S'
00017:0021  777000                  cal     wr1                 ;write byte $0
to the FDD
00018:0024  774E00                  cal     delay
00019:0027  420030                  ld      $0,&H30             ;pattern 'I'
00020:002A  777000                  cal     wr1                 ;write byte $0
to the FDD
00021:002D  774E00                  cal     delay
00022:0030  42007E                  ld      $0,&H7E             ;pattern 'O'
00023:0033  777000                  cal     wr1                 ;write byte $0
to the FDD
00024:0036  774E00                  cal     delay
00025:0039  0200                    ld      $0,$sx              ;terminate the
command
00026:003B  777000                  cal     wr1                 ;write byte $0
to the FDD
00027:003E  774E00                  cal     delay
00028:0041  775700                  cal     rd1                 ;read a byte
from the FDD to $0
00029:0044  420021                  ld      $0,&H21             ;diagnostic
command &H21: system restart
00030:0047  778A00                  cal     cm1                 ;send the
command $0 to the FDD
00031:004A  566054                  pst     ua,&H54
00032:004D  F7                      rtn
00033:004E
```

```
00034:004E            ; delay 500 ms
00035:004E  D100324D  delay:       ldw     $0,19762
00036:0052  8920      delay1:      sbw     $0,$sy                ;14 cycles
00037:0054  B483                   jr      nz,delay1             ;9 cycles
00038:0056  F7                     rtn
00039:0057
00040:0057            ; read a byte from the FDD to $0
00041:0057  1E61      rd1:         gst     ua,$1
00042:0059  026201                 ld      $2,$1
00043:005C  4E0230                 or      $2,&H30
00044:005F  1662                   pst     ua,$2
00045:0061  D600FFFF               pre     ix,&HFFFF
00046:0065  2800      rd2:         ld      $0,(ix+$sx)           ;FDD status
00047:0067  0420                   anc     $0,$sy
00048:0069  B485                   jr      nz,rd2                ;wait until the
FDD sends the data byte
00049:006B  28A0                   ld      $0,(ix-$sy)           ;FDD data
00050:006D  1661                   pst     ua,$1
00051:006F  F7                     rtn
00052:0070
00053:0070            ; write a byte $0 to the FDD
00054:0070  1E61      wr1:         gst     ua,$1
00055:0072  026201                 ld      $2,$1
00056:0075  4E0230                 or      $2,&H30
00057:0078  1662                   pst     ua,$2
00058:007A  D600FFFF               pre     ix,&HFFFF
00059:007E  20A0                   st      $0,(ix-$sy)           ;FDD data
00060:0080  2800      wr2:         ld      $0,(ix+$sx)           ;FDD status
00061:0082  440002                 anc     $0,&H02
00062:0085  B086                   jr      z,wr2                 ;wait until the
FDD reads the data byte
00063:0087  1661                   pst     ua,$1
00064:0089  F7                     rtn
00065:008A
00066:008A            ; write a command $0 to the FDD
00067:008A  1E61      cm1:         gst     ua,$1
00068:008C  026201                 ld      $2,$1
00069:008F  4E0230                 or      $2,&H30
00070:0092  1662                   pst     ua,$2
00071:0094  D600FFFF               pre     ix,&HFFFF
00072:0098  2000                   st      $0,(ix+$sx)           ;FDD command
00073:009A  2800      cm2:         ld      $0,(ix+$sx)           ;FDD status
00074:009C  440004                 anc     $0,&H04
00075:009F  B086                   jr      z,cm2                 ;wait until the
FDD reads the command byte
00076:00A1  1661                   pst     ua,$1
00077:00A3  F7                     rtn
00078:00A4
00079:00A4

ASSEMBLY COMPLETE, NO ERRORS FOUND

HD61700 ASSEMBLER Rev 0.45 - MAP LIST OF [ch14p1.s]
LABEL           : ADDRESS(hex)  | LABEL           : ADDRESS(hex)
-------------------------------------------------------------------
delay           :   004Eh     A | delay1          :   0052h      A
rd1             :   0057h     A | rd2             :   0065h      A
wr1             :   0070h     A | wr2             :   0080h      A
cm1             :   008Ah     A | cm2             :   009Ah      A
```

```
START ADDRESS   = 0000h
END ADDRESS     = 00A3h
EXECUTE ADDRESS = 0000h
```

The BASIC host — `src/vx-4/ch14p1.bas`

The HD61700 program cannot be loaded directly into the VX-4. The following BASIC program places its machine-code bytes into memory, beginning at address 0000h.

```
10 CLS:PRINT "CH14 R001 - LOAD ONLY"
20 PRINT "64K VX-4 RAM REQUIRED"
30 PRINT "RAM 0000-A3 MUST BE FREE"
60 RESTORE 1000:A=0
70 FOR R=1000 TO 1020
80 READ H$,Q$
90 S=0
100 FOR K=1 TO LEN(H$) STEP 2
110 B=VAL("&H"+MID$(H$,K,2)):S=S+B
120 NEXT K
130 Q=VAL("&H"+Q$)
140 IF S-256*INT(S/256)<>Q THEN PRINT "DATA CHECK ERROR";R:END
150 FOR K=1 TO LEN(H$) STEP 2
160 B=VAL("&H"+MID$(H$,K,2))
170 POKE A,B
180 IF PEEK(A)<>B THEN PRINT "RAM VERIFY ERROR";A:END
190 A=A+1:NEXT K
200 NEXT R
210 IF A<>164 THEN PRINT "SIZE ERROR";A:END
220 PRINT "LOAD OK: 164 BYTES"
230 PRINT "NEXT: MODE110(0)"
240 END
999 DATA CH14P1.EXE,&H0,&HA3,&H0
1000 DATA 420005778A004200,8A
1001 DATA 31778A0042004E77,39
1002 DATA 7000774E00420077,EE
1003 DATA 777000774E004200,EE
1004 DATA 5B777000774E0042,49
1005 DATA 0030777000774E00,DC
1006 DATA 42007E777000774E,6C
1007 DATA 000200777000774E,AE
1008 DATA 0077570042002177,A8
1009 DATA 8A00566054F7D100,5C
1010 DATA 324D8920B483F71E,74
1011 DATA 610262014E023016,5C
1012 DATA 62D600FFFF280004,62
1013 DATA 20B48528A01661F7,8F
1014 DATA 1E610262014E0230,64
1015 DATA 1662D600FFFF20A0,0C
1016 DATA 2800440002B08616,BA
1017 DATA 61F71E610262014E,8A
1018 DATA 02301662D600FFFF,7E
1019 DATA 20002800440004B0,40
1020 DATA 861661F7,F4
```

Running this program places the native program containing **164 bytes** at address 0000h–00A3h.

```
LOAD OK: 164 BYTES
NEXT: MODE110(0)
```

Making sure that the VX-4 is connected to the SB-5 and that the SB-5 is connected to the MD-120 enter

MODE110(0)

And witness CASIO appear on the seven-segment display.

THE SEVEN-SEGMENT LIGHT SHOW

Our second program (ch14p2.s) takes the same MD-120 diagnostic command and uses it for a longer display sequence:

1. Send command 05h to enter the MD-120 diagnostic mode.
2. Send command 31h to control the seven-segment display.
3. Rotate a single illuminated segment around the perimeter three times, followed by a progressive build-up of all seven segments.
4. Hold all segments illuminated briefly, then display C A S I O, one letter at a time.
5. Send a zero terminator, receive the drive's response, send command 21h to restart the firmware, and return to the pocket computer.

HD61 assembler listing — build/ch14p2.lst

```
HD61700 ASSEMBLER Rev 0.45 - ASSEMBLY LIST OF [ch14p2.s]
00001:0000            ; CS617 Chapter 14 R002 - Seven-Segment Light Show
00002:0000            ; Derived from Piotr Piatek cmd31a.s (2026-11-06).
00003:0000            ; R001 is preserved unchanged. This file is a new
experimental derivative.
00004:0000            ; Diagnostic 31h: exactly one session, one 00h terminator,
one ACK/restart.
00005:0000            ; 3 perimeter revolutions (150 ms/frame), cumulative
buildup (150 ms/frame),
00006:0000            ; full-on extra hold (~500 ms), then original CASIO (500
ms/letter).
00007:0000            ; Segment glyphs beyond CASIO: inferred, NOT YET hardware
validated.
00008:0000            ; Requires 64-KB VX-4 usable RAM at 0000h and same SB-5 /
MD-120 setup.
00009:0000            ; Machine code must be built with real HD61; no ROM image
required.
00010:0000
00011:0000                        org     0
00012:0000                        start   0
00013:0000
00014:0000  420005                ld      $0,&H05               ; enter MD-120
diagnostic mode
00015:0003  77BD00                cal     cm1
00016:0006  420031                ld      $0,&H31               ; LED pattern
command
00017:0009  77BD00                cal     cm1
00018:000C
00019:000C            ; IZ points to table; $4 is offset, $5 is remaining frame
count.
00020:000C            ; wr1/delay use only $0-$2 and IX, preserving IZ / $4 /
$5.
00021:000C  D6406C00              pre     iz,animtab
00022:0010  420400                ld      $4,0
00023:0013  420519                ld      $5,25
00024:0016  296004    animlp:     ld      $0,(iz+$4)
00025:0019  77A300                cal     wr1
00026:001C  775A00                cal     dly150
00027:001F  480401                ad      $4,1
00028:0022  490501                sb      $5,1
00029:0025  B490                  jr      nz,animlp
00030:0027
00031:0027            ; Last animation byte is 7Fh; hold all segments before
CASIO.
00032:0027  776300                cal     dly500
```

```
00033:002A  D6408500              pre     iz,casiotb
00034:002E  420400                ld      $4,0
00035:0031  420505                ld      $5,5
00036:0034  296004    caslp:      ld      $0,(iz+$4)
00037:0037  77A300                cal     wr1
00038:003A  776300                cal     dly500
00039:003D  480401                ad      $4,1
00040:0040  490501                sb      $5,1
00041:0043  B490                  jr      nz,caslp
00042:0045
00043:0045            ; Original Piotr command termination / response / restart
sequence.
00044:0045  0200                  ld      $0,$sx
00045:0047  77A300                cal     wr1
00046:004A  776300                cal     dly500
00047:004D  778A00                cal     rd1
00048:0050  420021                ld      $0,&H21
00049:0053  77BD00                cal     cm1
00050:0056  566054                pst     ua,&H54
00051:0059  F7                    rtn
00052:005A
00053:005A            ; Short delay (~150ms) scaled from Piotr's original 500ms
calibration.
00054:005A            ; Physical timing must be measured; this is not a
guaranteed duration.
00055:005A  D1002917  dly150:     ldw     $0,5929
00056:005E  8920      dly15a:     sbw     $0,$sy
00057:0060  B483                  jr      nz,dly15a
00058:0062  F7                    rtn
00059:0063
00060:0063            ; Original Piotr 500ms nominal delay, preserved
arithmetically.
00061:0063  D100324D  dly500:     ldw     $0,19762
00062:0067  8920      dly50a:     sbw     $0,$sy
00063:0069  B483                  jr      nz,dly50a
00064:006B  F7                    rtn
00065:006C
00066:006C            ; Table: clockwise a,b,c,d,e,f (g omitted from perimeter).
00067:006C            ; 18 chase frames + 7 buildup frames; no zero terminator
within table.
00068:006C  40201008  animtab:    db
&H40,&H20,&H10,&H08,&H04,&H02,&H40,&H20,&H10,&H08,&H04,&H02,&H40,&H20,&H10,&H08,
&H04,&H02
            04024020
            10080402
            40201008
            0402
00069:007E  40607078              db      &H40,&H60,&H70,&H78,&H7C,&H7E,&H7F
            7C7E7F
00070:0085
00071:0085            ; Exact five working Piotr glyph bytes, in original order.
00072:0085  4E775B30  casiotb:    db      &H4E,&H77,&H5B,&H30,&H7E
            7E
00073:008A
00074:008A            ; read a byte from the FDD to $0
00075:008A  1E61      rd1:        gst     ua,$1
00076:008C  026201                ld      $2,$1
00077:008F  4E0230                or      $2,&H30
00078:0092  1662                  pst     ua,$2
```

```
00079:0094  D600FFFF              pre     ix,&HFFFF
00080:0098  2800      rd2:        ld      $0,(ix+$sx)           ;FDD status
00081:009A  0420                  anc     $0,$sy
00082:009C  B485                  jr      nz,rd2                ;wait until the
FDD sends the data byte
00083:009E  28A0                  ld      $0,(ix-$sy)           ;FDD data
00084:00A0  1661                  pst     ua,$1
00085:00A2  F7                    rtn
00086:00A3
00087:00A3            ; write a byte $0 to the FDD
00088:00A3  1E61      wr1:        gst     ua,$1
00089:00A5  026201                ld      $2,$1
00090:00A8  4E0230                or      $2,&H30
00091:00AB  1662                  pst     ua,$2
00092:00AD  D600FFFF              pre     ix,&HFFFF
00093:00B1  20A0                  st      $0,(ix-$sy)           ;FDD data
00094:00B3  2800      wr2:        ld      $0,(ix+$sx)           ;FDD status
00095:00B5  440002                anc     $0,&H02
00096:00B8  B086                  jr      z,wr2                 ;wait until the
FDD reads the data byte
00097:00BA  1661                  pst     ua,$1
00098:00BC  F7                    rtn
00099:00BD
00100:00BD            ; write a command $0 to the FDD
00101:00BD  1E61      cm1:        gst     ua,$1
00102:00BF  026201                ld      $2,$1
00103:00C2  4E0230                or      $2,&H30
00104:00C5  1662                  pst     ua,$2
00105:00C7  D600FFFF              pre     ix,&HFFFF
00106:00CB  2000                  st      $0,(ix+$sx)           ;FDD command
00107:00CD  2800      cm2:        ld      $0,(ix+$sx)           ;FDD status
00108:00CF  440004                anc     $0,&H04
00109:00D2  B086                  jr      z,cm2                 ;wait until the
FDD reads the command byte
00110:00D4  1661                  pst     ua,$1
00111:00D6  F7                    rtn
00112:00D7
00113:00D7

ASSEMBLY COMPLETE, NO ERRORS FOUND

HD61700 ASSEMBLER Rev 0.45 - MAP LIST OF [ch14p2.s]
LABEL           : ADDRESS(hex)  | LABEL           : ADDRESS(hex)
-----------------------------------------------------------------
animlp          :   0016h     A | caslp           :   0034h      A
dly150          :   005Ah     A | dly15a          :   005Eh      A
dly500          :   0063h     A | dly50a          :   0067h      A
animtab         :   006Ch     A | casiotb         :   0085h      A
rd1             :   008Ah     A | rd2             :   0098h      A
wr1             :   00A3h     A | wr2             :   00B3h      A
cm1             :   00BDh     A | cm2             :   00CDh      A

START ADDRESS   = 0000h
END ADDRESS     = 00D6h
EXECUTE ADDRESS = 0000h
```

The BASIC host — src/vx-4/ch14p2.bas

As in the CASIO experiment, the BASIC host places the assembled HD61700 bytes into memory at address 0000h. This version contains 215 bytes.

```
10 CLS:PRINT "CH14 R002 - LOAD ONLY"
20 PRINT "64K VX-4 RAM REQUIRED"
30 PRINT "RAM 0000-D6 MUST BE FREE"
60 RESTORE 1000:A=0
70 FOR R=1000 TO 1026
80 READ H$,Q$
90 S=0
100 FOR K=1 TO LEN(H$) STEP 2
110 B=VAL("&H"+MID$(H$,K,2)):S=S+B
120 NEXT K
130 Q=VAL("&H"+Q$)
140 IF S-256*INT(S/256)<>Q THEN PRINT "DATA CHECK ERROR";R:END
150 FOR K=1 TO LEN(H$) STEP 2
160 B=VAL("&H"+MID$(H$,K,2))
170 POKE A,B
180 IF PEEK(A)<>B THEN PRINT "RAM VERIFY ERROR";A:END
190 A=A+1:NEXT K
200 NEXT R
210 IF A<>215 THEN PRINT "SIZE ERROR";A:END
220 PRINT "LOAD OK: 215 BYTES"
230 PRINT "NEXT: MODE110(0)"
240 END
999 DATA CH14P2.EXE,&H0,&HD6,&H0
1000 DATA 42000577BD004200,BD
1001 DATA 3177BD00D6406C00,E7
1002 DATA 4204004205192960,2F
1003 DATA 0477A300775A0048,37
1004 DATA 0401490501B49077,0F
1005 DATA 6300D64085004204,44
1006 DATA 0042050529600477,50
1007 DATA A300776300480401,CA
1008 DATA 490501B490020077,0C
1009 DATA A300776300778A00,7E
1010 DATA 42002177BD005660,4D
1011 DATA 54F7D10029178920,05
1012 DATA B483F7D100324D89,07
1013 DATA 20B483F740201008,C6
1014 DATA 0402402010080402,84
1015 DATA 4020100804024060,1E
1016 DATA 70787C7E7F4E775B,81
1017 DATA 307E1E610262014E,E0
1018 DATA 02301662D600FFFF,7E
1019 DATA 28000420B48528A0,4D
1020 DATA 1661F71E61026201,52
1021 DATA 4E02301662D600FF,CD
1022 DATA FF20A02800440002,2D
1023 DATA B0861661F71E6102,25
1024 DATA 62014E02301662D6,31
1025 DATA 00FFFF2000280044,8A
1026 DATA 0004B0861661F7,A8
```

Running the BASIC loader places 215 bytes at addresses 0000h–00D6h. The program reports:

```
LOAD OK: 215 BYTES
NEXT: MODE110(0)
```

With the VX-4 connected to the SB-5 and the SB-5 connected to the MD-120, and after confirming that the required low RAM is free, enter:

```
MODE110(0)
```

And witness CASIO appear on the seven-segment display, preceded by by an impressive demonstration of segment display.

IT'S TIME TO BUZZ THE TOWER!

Piotr also discovered a piezoelectric buzzer inside the MD-120. He found that diagnostic command 32h could make the drive produce a sound. After two experiments with the seven-segment display, it seemed only fitting to let the little drive make some noise.

Piotr's original program (ch14p3.s, copied from his cmd32a.s) performs the following operations:

1. Send command 05h to enter the MD-120 diagnostic mode.
2. Send command 32h to invoke the buzzer diagnostic.
3. Read the drive's response.
4. Send command 21h to restart the drive firmware, and return to the pocket computer.

HD61 assembler listing — build/ch14p3.lst

```
HD61700 ASSEMBLER Rev 0.45 - ASSEMBLY LIST OF [ch14p3.s]
00001:0000            ; MD-120 diagnostic command 0x32 on the FX-870P
00002:0000
00003:0000                          org     0
00004:0000                          start   0
00005:0000
00006:0000  420005                  ld      $0,&H05             ;FDD command
&H05: enter the diagnostic mode
00007:0003  774C00                  cal     cm1                 ;send the
command $0 to the FDD
00008:0006  420032                  ld      $0,&H32             ;diagnostic
command &H32: beep
00009:0009  774C00                  cal     cm1                 ;send the
command $0 to the FDD
00010:000C  771900                  cal     rd1                 ;read a byte
from the FDD to $0
00011:000F  420021                  ld      $0,&H21             ;diagnostic
command &H21: system restart
00012:0012  774C00                  cal     cm1                 ;send the
command $0 to the FDD
00013:0015  566054                  pst     ua,&H54
00014:0018  F7                      rtn
00015:0019
00016:0019            ; read a byte from the FDD to $0
00017:0019  1E61      rd1:          gst     ua,$1
00018:001B  026201                  ld      $2,$1
00019:001E  4E0230                  or      $2,&H30
00020:0021  1662                    pst     ua,$2
00021:0023  D600FFFF                pre     ix,&HFFFF
00022:0027  2800      rd2:          ld      $0,(ix+$sx)         ;FDD status
00023:0029  0420                    anc     $0,$sy
00024:002B  B485                    jr      nz,rd2              ;wait until the
FDD sends the data byte
00025:002D  28A0                    ld      $0,(ix-$sy)         ;FDD data
00026:002F  1661                    pst     ua,$1
00027:0031  F7                      rtn
00028:0032
00029:0032            ; write a byte $0 to the FDD
00030:0032  1E61      wr1:          gst     ua,$1
00031:0034  026201                  ld      $2,$1
00032:0037  4E0230                  or      $2,&H30
00033:003A  1662                    pst     ua,$2
00034:003C  D600FFFF                pre     ix,&HFFFF
00035:0040  20A0                    st      $0,(ix-$sy)         ;FDD data
```

```
00036:0042  2800      wr2:         ld      $0,(ix+$sx)           ;FDD status
00037:0044  440002                 anc     $0,&H02
00038:0047  B086                   jr      z,wr2                 ;wait until the
FDD reads the data byte
00039:0049  1661                   pst     ua,$1
00040:004B  F7                     rtn
00041:004C
00042:004C            ; write a command $0 to the FDD
00043:004C  1E61      cm1:         gst     ua,$1
00044:004E  026201                 ld      $2,$1
00045:0051  4E0230                 or      $2,&H30
00046:0054  1662                   pst     ua,$2
00047:0056  D600FFFF               pre     ix,&HFFFF
00048:005A  2000                   st      $0,(ix+$sx)           ;FDD command
00049:005C  2800      cm2:         ld      $0,(ix+$sx)           ;FDD status
00050:005E  440004                 anc     $0,&H04
00051:0061  B086                   jr      z,cm2                 ;wait until the
FDD reads the command byte
00052:0063  1661                   pst     ua,$1
00053:0065  F7                     rtn
00054:0066
00055:0066

ASSEMBLY COMPLETE, NO ERRORS FOUND

HD61700 ASSEMBLER Rev 0.45 - MAP LIST OF [ch14p3.s]
LABEL           : ADDRESS(hex)  | LABEL           : ADDRESS(hex)
-----------------------------------------------------------------
rd1             :   0019h     A | rd2             :   0027h      A
wr1             :   0032h     A | wr2             :   0042h      A
cm1             :   004Ch     A | cm2             :   005Ch      A

START ADDRESS   = 0000h
END ADDRESS     = 0065h
EXECUTE ADDRESS = 0000h
```

The BASIC host — src/vx-4/ch14p3.bas

As in the earlier experiments, the BASIC host places the assembled HD61700 bytes into memory at address 0000h. This program contains 102 bytes.

```
10 CLS:PRINT "CH14 R004 - LOAD ONLY"
20 PRINT "64K VX-4 RAM REQUIRED"
30 PRINT "RAM 0000-65 MUST BE FREE"
60 RESTORE 1000:A=0
70 FOR R=1000 TO 1012
80 READ H$,Q$
90 S=0
100 FOR K=1 TO LEN(H$) STEP 2
110 B=VAL("&H"+MID$(H$,K,2)):S=S+B
120 NEXT K
130 Q=VAL("&H"+Q$)
140 IF S-256*INT(S/256)<>Q THEN PRINT "DATA CHECK ERROR";R:END
150 FOR K=1 TO LEN(H$) STEP 2
160 B=VAL("&H"+MID$(H$,K,2))
170 POKE A,B
180 IF PEEK(A)<>B THEN PRINT "RAM VERIFY ERROR";A:END
190 A=A+1:NEXT K
200 NEXT R
210 IF A<>102 THEN PRINT "SIZE ERROR";A:END
220 PRINT "LOAD OK: 102 BYTES"
```

```
 230 PRINT "NEXT: MODE110(0)"
 240 END
 999 DATA CH14P3.EXE,&H0,&H65,&H0
 1000 DATA 420005774C004200,4C
 1001 DATA 32774C0077190042,C7
 1002 DATA 0021774C00566054,EE
 1003 DATA F71E610262014E02,2B
 1004 DATA 301662D600FFFF28,A4
 1005 DATA 000420B48528A016,3B
 1006 DATA 61F71E610262014E,8A
 1007 DATA 02301662D600FFFF,7E
 1008 DATA 20A02800440002B0,DE
 1009 DATA 861661F71E610262,D7
 1010 DATA 014E02301662D600,CF
 1011 DATA FFFF200028004400,8A
 1012 DATA 04B0861661F7,A8
 Running the BASIC loader places 102 bytes at addresses 0000h–0065h. The program
 reports:
```

```
LOAD OK: 102 BYTES

NEXT: MODE110(0)
```

With the VX-4 connected to the SB-5 and the SB-5 connected to the MD-120, and after confirming that the required low RAM is free, enter:

```
MODE110(0)
```

Listen for the buzzer. The program asks the drive to perform its diagnostic and then return to BASIC; the precise sound is best described from the recorded demonstration.

Acknowledgments

This report was produced by Xandrian Labs. 付喪神.

Special thanks to Piotr and to Betka for her continued and loyal support.

This is a collaboration with my AI friend ChatGPT, a.k.a “Sol.” All errors are human.

This work is freely shared with the vintage computing community. Please use it, build upon it, and cite the source.

Final.