Behavior Research Methods & Instrumentation
1982, Vol. 14(6), 539-544

Software synchronizing of video displays and Z-80 processing in the Model III TRS-80

JOSEPH G. DLHOPOLSKY
St. John's University, Staten Island, New York 10301

A machine language technique is described whereby the Z-80 microprocessor of the Model III TRS-80 can be programmed to monitor position of the electron beam during CRT scanning. This technique provides the opportunity to synchronize the appearance of video displays with Z-80 processing. The programmer can therefore be assured of crisp stimulus displays and precisely recorded reaction times. The computer's real-time clock operates on video circuitry as part of a routine that is initiated by a maskable interrupt. The real-time clock interrupt can be vectored from its normal use to a routine that signals the Z-80 when the electron beam is at a known screen location. A machine language program and a TRSDOS BASIC program that demonstrate the technique are described.

Observers have noted a lack of precision in microcomputers in synchronizing the appearance of a stimulus on the video screen with processing taking place in the central processing unit (CPU) (Lincoln & Lane, 1980; Merikle, Cheesman, & Bray, 1982; Reed, 1979). Reed (1979) reports a hardware modification that overcomes the problem in the Apple II microcomputer. The modification makes the video circuitry's vertical synch pulse available to the computer's CPU. Since this pulse occurs between video frames, it signals the CPU when the electron beam is at a known location. Grice (1981) reports a similar hardware modification for the Model I TRS-80.

Software techniques have been more limited, in that the design of the internal wiring of many microcomputers precludes software access to the vertical synch pulse. However, Merikle et al. (1982) report a machine language routine that allows the PET/CBM microcomputer's CPU to detect the vertical synch pulse. This technique is preferable to hardware modifications, since mistakes cause no costly damage (they are undone by pressing the reset key) and no expertise in electronic rewiring or circuit design is needed.

The Model III TRS-80 is an improvement over the Model I because video synchronization can be carried out using a software technique similar to that described for the PET computer.[1] This article reports such a technique.

The video screen on the TRS-80 has a scanning frequency of 60 Hz. Effectively, this means that every 1/60 sec a new "page" is drawn on the video screen. The TRS-80's video display is memory mapped. There are 1,024 locations in random-access memory (RAM) for each character location on the video screen. The so-called video RAM addresses are 3C00 to 3FFF hexadecimal (hex) or 15360 to 16383 decimal (dec). Most TRS-80 programmers are aware that whenever the TRS-80's CPU, the Z-80, is instructed to load a value into a video RAM location, the corresponding ASCII character appears on the video screen. However, the video circuitry and the Z-80 in fact operate independently. The video RAM serves as an interface into which the Z-80 loads data and from which the video circuitry reads data when it is prepared to display each succeeding line on the screen.

Once one recognizes that the Z-80 does not actively place characters on the screen, a number of problems become apparent. For one, the researcher may wish to display an alphabetic character for 4 msec. Accordingly, the Z-80 can be programmed to store the appropriate ASCII code in video RAM for exactly 4 msec. The video circuitry, however, is operating at its constant pace of 60 Hz. If the electron beam happens to be at the correct location when the stimulus appears for 4 msec in video RAM, the stimulus will appear on the screen. But it will stay on the screen until the electron beam returns to that location to erase it 17 msec later, not the intended 4 msec. If the electron beam happens to be just past the correct location when the ASCII code appears in video RAM, it will take 17 msec to return to the point at which the stimulus is intended to appear. In that time, the 4-msec exposure will have elapsed and the ASCII code will no longer be in video RAM. So, the stimulus will not appear at all.

One can partially surmount the lack of synchronization between the Z-80 and the video circuitry by limiting stimulus exposures to multiples of 16.667 msec. Although they do not offer a great range, the resultant values include 17, 33, 50, 67, 83, 100, 117, 133, 167, 183, and 200 msec, sufficient for many tachistoscopic purposes. By choosing multiples of 16.667 msec, one is assured that the stimulus will appear for the desired exposure. The only drawback is that the larger the vertical dimension of the stimulus, the more likely it is that the stimulus will be drawn in parts. For example, the tail end of a vertical bar may be drawn on the screen

0005-7878/82/060539-06$00.85/0

first, followed by the top portion, which will be completed 17 msec later. This may not be desirable if short stimulus exposures are used.

When measuring response latencies, one has an additional problem. If the subject is to detect a stimulus and then make an appropriate response, the Z-80 loads the stimulus ASCII values into video RAM. It then waits for the exposure interval to pass, erases or masks the stimulus by loading blanks or masking characters into the appropriate video RAM, and then starts to count milliseconds until the subject's response is detected. The millisecond value is then adjusted for the execution time for all commands after the appearance of the stimulus on the screen. The resultant latency ignores the fact that a single alphanumeric character might have appeared at any time within the first 17 msec of the exposure interval or that a multiline stimulus was completed only near the end of the first 17 msec of the exposure interval.[2] Lincoln and Lane (1980) indicate that the resultant error variance is small compared to subject-related error variance. However, there are occasions when the magnitude of this error variance may be close to a hypothesized treatment effect.

The problem could be solved if one were able to have the video circuitry inform the Z-80 when the electron beam is aimed at a known position. The Z-80 could then wait for the beam to advance to the correct line and load the stimulus data into video RAM within microseconds of the beam's approach to the critical position. Alternatively, the Z-80 could load the stimulus data while the electron beam was in a noncritical location, for example, during the beam's vertical retrace. The precise time that the stimulus appears would be known, and the millisecond timer would be adjusted accordingly. In either case, the stimulus would appear complete and unbroken on the screen for the precise interval. The Z-80 could then erase the stimulus and wait for the subject's response. Knowing exactly when the stimulus appeared, the Z-80 could then report a response latency with microsecond precision. All this sounds quite involved and unbelievably precise, but each of the Z-80 instructions that carries out the necessary processes has a known processing duration of from 1.98 to 10.4 microsec at the 2.02752-MHz frequency of the TRS-80 Model III's system clock (Zaks, 1980).

The TRS-80 was not designed to serve as a research device for experimental psychologists. A means of synchronizing the Z-80 with the video circuitry was not consciously designed into the computer. However, there is a conceptually simple way to solve the problem.[3] The procedure makes use of the Model III real-time clock (RTC) and the routines must be written in Z-80 machine language.

The Model III's RTC is software driven. The Z-80 increments the clock at precise intervals by virtue of an RTC interrupt that operates at 30 Hz. When an RTC interrupt occurs, the Z-80 processing is vectored to a machine language routine at an interrupt handling address. This routine increments a 33.3-msec clock at address 4216 hex (16918 dec) and adjusts seconds, minutes, and hours accordingly. After the RTC is updated, the Z-80 returns to whatever process it was carrying out before the interrupt occurred. By means of the interrupt, the Z-80 spends most of its time executing the current computer program, suspending this processing for a small fraction of 1 msec every 33.3 msec to update the clock.

The 30-Hz operating frequency of the interrupt is derived from the 60-Hz video scan frequency. This provides the means by which video synchronization can be achieved, because the interrupt occurs when the electron beam is at a known position every other screen page.

In order to alter the RTC interrupt for use in video synchronization routines, the programmer needs to "steal" the interrupt from the RTC handling routine. The RTC interrupt normally vectors the Z-80 to RAM address 4046 hex (16454 dec). This address and the two following it contain the codes for a JP 2935H instruction. This instruction tells the Z-80 where in RAM to jump in order to update the RTC. Normally, RAM address 2935 hex (13609 dec) is the beginning of this routine. To use the RTC interupt for video synchronization, the programmer must change the 2-byte address in 4047 and 4048 hex from the RTC interrupt handling address to the address of the machine language routine that carries out the experimental objective.

17 MSEC VERTICAL LINE (Figure 1) is an assembly language routine that demonstrates the techniques for accomplishing this task. It is designed to be called from a BASIC program, written in TRSDOS 1.3 BASIC. The object code for this routine should be loaded beginning at memory location FE00 hex (65024 dec) in a 48-KB Model III TRS-80. For computers with 16-KB or 32-KB RAM, the program will operate if all of the relevant addresses are changed to lower ones. The object code can be prepared on disk or cassette from the source listing in Figure 1 by using the appropriate editor/assembler software. Alternatively, the user may write a BASIC program that POKEs the decimal values of the object code into the correct memory addresses. The RAM addresses appear in the first column of Figure 1; the decimal object codes may be derived from the hexidecimal codes in the second column.[4]

17 MSEC VERTICAL LINE carries out the following processes. Line 150 disables all maskable interrupts, including the RTC interrupt. This is necessary to assure that an interrupt does not occur while the RTC interrupt is being revectored. Lines 160 and 170 change the RTC interrupt handling routine address to START (at FE0F hex, 65039 dec), which is the beginning of the stimulus display routine of 17 MSEC VERTICAL LINE. Line 180 enables the interrupts. Lines 190 and 210 execute a pause that essentially causes the Z-80 to wait for the

```
                00100 ;17 MSEC VERTICAL LINE
                00110 ;By Joseph G. Dlhopolsky, Ph.D.
                00120
FE00            00130          ORG     0FE00H  ;RAM of routine
0060            00140 DELAY    EQU     0060H   ;Delay RAM
FE00 F3         00150          DI              ;Disable interrupts
FE01 210FFE     00160          LD      HL,START;New RTC int add
FE04 224740     00170          LD      (4047H),HL;New RTC int vect
FE07 FB         00180          EI              ;Enable interrupts
FE08 01320D     00190          LD      BC,3378 ;50 msec delay
FE0B CD6000     00200          CALL    DELAY
FE0E C9         00210          RET             ;Return to BASIC
FE0F F3         00220 START    DI              ;DRAW BAR ROUTINE
FE10 ED4BE0FF   00230          LD      BC,(0FFE0H);Pause
FE14 CD6000     00240          CALL    DELAY
FE17 114000     00250          LD      DE,40H
FE1A 0610       00260          LD      B,10H   ;Counts lines
FE1C FD21203C   00270          LD      IY,3C20H;Video RAM
FE20 3EBF       00280          LD      A,191   ;Print bar CHR$
FE22 FD7700     00290 LOOPA    LD      (IY),A
FE25 FD19       00300          ADD     IY,DE   ;Skip to next line
FE27 10F9       00310          DJNZ    LOOPA
FE29 ED4BDBFF   00320          LD      BC,(0FFDBH);Finish 16.7 msec
FE2D CD6000     00330          CALL    DELAY
FE30 0610       00340          LD      B,10H   ;Sets counter
FE32 3E80       00350          LD      A,128   ;Blank character
FE34 FD21203C   00360          LD      IY,3C20H;Points video RAM
FE38 FD7700     00370 LOOPB    LD      (IY),A  ;Draws blank
FE3B FD19       00380          ADD     IY,DE   ;Jump to next line
FE3D 10F9       00390          DJNZ    LOOPB
FE3F 212935     00400          LD      HL,3529H;Normal RTC int
FE42 224740     00410          LD      (4047H),HL
FE45 FB         00420          EI
FE46 ED4D       00430          RETI
FE00            00440          END     0FE00H
00000 Total Errors

LOOPB   FE38
LOOPA   FE22
START   FE0F
DELAY   0060
```

Figure 1. Source code for 17 MSEC VERTICAL LINE. The first column contains the hexadecimal addresses for the object code that appears in the second column. The program lines, to which the text refers, are in the third column. The fourth column contains the assembly language instructions. Comments appear on the far right.

RTC interrupt. When the interrupt occurs, the Z-80 processing is vectored to START at Line 220.

Line 220 disables interrupts again. This is necessary for a number of reasons. For one, if response latency is to be measured by a software timer, any interrupt processing causes an underestimation of the subject's latency. Most software-based millisecond timers are calibrated in reference to the known execution times of Z-80 instructions. When an interrupt is encountered in the course of measuring response latency, the millisecond timer will effectively stop until the interrupt is serviced, but real-time will continue.

A second reason for disabling interrupts is that most routines take longer than the 33.3 msec before the next RTC interrupt occurs. Failing to disable interrupts in this case leads to an endless loop.

Lines 230 and 240 execute a delay before the stimulus data are loaded into video RAM. The duration of the delay is determined by a constant that is stored in two addresses: the least significant byte (LSB), in address FFE0 hex (65504 dec), and the most significant byte (MSB), in FFE1 hex (65505 dec). For most experimental purposes, this delay is unnecessary. It is included in this demonstration routine in order to allow the programmer to observe the location of the electron beam at different times within the 17-msec duration of each page. The BASIC program described later demonstrates the implementation of this delay.

Without going into elaborate detail about the remaining instructions, the START routine begins with the occurrence of the RTC interrupt; the CPU pauses for the designated interval and then loads Code 191 into the video RAM addresses for a full-length vertical bar in the center of the screen. Code 191 in the TRS-80 is interpreted as a solid block composed of 2 by 3 pixels. It takes 380 microsec to complete this loading. So, in effect, the stimulus can be loaded into video RAM right in front of the electron beam. Once the bar is drawn, the program pauses (Lines 320 and 330) for the remainder of the 17-msec existence of the screen page and then loads

blanks (Code 128) into the video RAM locations of the to-be-erased bar (Lines 340-390). With the next pass of the electron beam 17 msec from the first RTC interrupt, the bar will be erased.

Line 420 enables interrupts, and Line 430 returns from the current interrupt, bringing the Z-80 back to the instruction in Line 210. Line 210 is designed to return control to the BASIC program from which the machine language routine was called.

In order to understand what is being seen on the screen during the operation of 17 MSEC VERTICAL LINE, the characteristics of the video screen need to be described. The video circuitry acts as if there are actually 22 lines on the screen, even though only the first 16 are actually displayed. The remaining six lines (Lines 17-22) are devoted to vertical retrace and can be considered invisible. Data loaded into video RAM while the electron beam is aimed at these lines will not appear on the screen until the beam reaches the designated screen location. Each visible and invisible line takes .758 msec (16.667/22) to draw.

The RTC interrupt occurs at the beginning of Line 17. The following six invisible lines take a total of 4.545 msec (.758 X 6) to complete before the electron beam reaches the top of the screen for the first visible line. It takes about 100 microsec from the RTC interrupt before the Z-80 begins the interrupt handling routine. This leaves about 4.4 msec for 17 MSEC VERTICAL LINE to load the bar into video RAM before the beam reaches the first visible line. This leaves plenty of time.

If the programmer chooses a longer delay, the electron beam will have started its trip down the screen when the bar is loaded into video RAM. Therefore, the top portions will be missing, the length of missing material being related to the duration of the initial pause.

Some experiments might make use of physically larger stimuli. In the present example, it took 380 microsec to load 16 bytes into video RAM: about 24 microsec/byte. A stimulus made up of more than 702 bytes could not be stored in video RAM fast enough to beat the beam to the end of the screen. While 702 bytes is a sizable stimulus, some research may demand a program that changes the contents of all 1,024 video RAM addresses.

Other subroutines for loading data into selected video RAM locations may be faster than the one described here. One simple solution for large screen changes is to make all desired video changes on an invisible page of memory. For example, addresses FC00 hex through FFFF hex (64512 through 65535 dec) comprise 1,024 bytes. All intended video changes can be made in analogous areas of this invisible page. When the changes are completed, the video synch routine can be engaged to transfer the block of data to video RAM, using the assembly language LDIR command. To use this instruction, the Z-80's BC, DE, and HL registers need to be initialized as follows. The BC register counts out the number of bytes and should be set at 1,024 dec. The DE register is set to the first video RAM address, 3C00 hex (15360 dec). Finally, the HL register is set to the first address of the invisible page (FC00 hex, 64512 dec in the present example). Execution of the LDIR instruction then transfers all the data in the invisible page to video RAM. This is carried out at 10.4 microsec/byte. A single video line's data take 665 microsec to transfer; all 16 lines take 10.6 msec. If the video synch routine has the typical 4.4-msec jump on the electron beam, the entire page can be transferred before the beam gets to the ninth visible line. This means that the video RAM loads will always be ahead of the electron beam.

For more complicated stimulus displays, additional invisible pages of memory can be used and rapidly presented. Twenty such pages would not be unreasonable in a computer with 48 KB of RAM, leaving room for the program, data, and disk operating system (if there is one). This means that the TRS-80 could act as a 20-field tachistoscope, in which all 20 pages could be sequentially displayed in .33 sec.

Figure 2 shows the listing of a BASIC program, Z-80 VIDEO SYNC DEMONSTRATION, that can be used to demonstrate the operation of 17 MSEC VERTICAL LINE. The program is written in TRSDOS 1.3 BASIC, which differs in some respects from Level II BASIC. It may also differ from the BASIC enhancements of other disk operating systems. If they exist, the differences are likely to be in the following areas: (1) the representation of hexadecimal values, (2) the means for calling a Z-80 routine, and (3) the adopted conventions for string and integer variables.

All the hexadecimal values may be changed to their decimal equivalents to solve the first problem. To solve the second problem, the manual for the disk operating system in use will contain information for calling a Z-80 routine. Lines 120 and 130 load the machine language disk file, and Line 410 calls the routine. These lines may need to be changed to conform to any non-TRSDOS operating system being used. The last problem is a convention adopted by the author, which could present problems for users unfamiliar with it. Model III TRS-80 BASIC allows the programmer to define ranges of variables as strings, integers, single precision, and double precision. Once such variables are so defined, they need not have identifying codes in the remainder of the program (e.g., AA$, IA%, XA# can appear as AA, IA, and XA and still be interpreted as string, integer, and double-precision variables, respectively). This saves memory space and programming time. The BASIC program described here defines all variables beginning with the letters A through H as string variables (see Line 110) and all variables from I through N as integers. The remaining variables are single precision.

To execute the BASIC program, the object code for 17 MSEC VERTICAL LINE must be in a floppy disk

```
10 '     Z-80 VIDEO SYNCH DEMONSTRATION
         48 K DISK MODEL III TRS-80
         Written for TRSDOS 1.3
20 '     Revised 8208.21

100 CLEAR500:CLS
110 DEFSTR A-H:DEFINT I-J
120 CMD"L","BEAMCAL/CMD":'Loads 17 MSEC VERTICAL BAR
130 DEFUSR0=&HFE00:'Defines origin of machine code
200 CLS:INPUT"How many flashes";JA
210 INPUT"Length of pause before drawing line (usec)";JB
240 P0=JB-40:'Corrects for machine language routine overhead
242 OA=(P0-2.46)/14.8:'Converts to BC register value
244 GOSUB9000:'Rounding
246 JC=INT(OA)
250 J0=JCAND255:'LSB
252 J1=(-256ANDJC)/256:IFJ1<0THENJ1=256+J1:'MSB
260 POKE&HFFE0,J0:POKE&HFFE1,J1:'LSB & MSB for DELAY call
270 P0=50000/3-P0-403.5:'Gets rest of 16.7 msec
272 OA=(P0-2.46)/14.8:'Converts to BC register value
274 GOSUB9000:'Rounding
276 JD=INT(OA)
280 J0=JDAND255:'LSB
282 J1=(-256ANDJD)/256:IFJ1<0THENJ1=256+J1:'MSB
290 POKE&HFFDB,J0:POKE&HFFDC,J1:'LSB & MSB for rest of 17 msec
300 GOSUB9200:CLS
310 CLS:'Lines 310-340 draw screen display
320 FORJ=29TO989STEP64
322 PRINT@J,CHR$(170);
324 PRINT@J+6,CHR$(149);
326 NEXT
330 PRINT@16,"VIDEO LINE";
331 J0=1:FORJ=26TO538STEP64
332 PRINT@J,J0;:PRINT@J+4,STRING$(5,95);
333 J0=J0+1:NEXT
334 FORJ=601TO985STEP64
335 PRINT@J,J0;:PRINT@J+5,STRING$(5,95);
336 J0=J0+1:NEXT
340 PRINT@361,'Pause from Real Time";
342 PRINT@425,"Clock interrupt:";
344 PRINT@488,JB;"+ 100 usec";
400 FORJ=1TOJA:'Start flash sequence loop
410 J0=USR0(0):'Calls 17 MSEC VERTICAL LINE
430 NEXT:'Next flash
440 GOSUB9200:CLS:GOTO210:'Wait for response then start over
9000 IFOA-INT(OA)<.49999THENOA=INT(OA)ELSEOA=INT(OA+1)
9010 RETURN
9200 A=INKEY$
9210 PRINT@962,"( P R E S S   A N Y   K E Y   T O   C O N T
I N U E )";
9220 A=INKEY$
9230 IFA=""THEN9220ELSERETURN
```

Figure 2. Source code for Z-80 VIDEO SYNC DEMONSTRATION, a BASIC program that calls 17 MSEC VERTICAL LINE and demonstrates video synchronization based on the real-time clock interrupt.

file named BEAMCAL/CMD. When starting up the computer under TRSDOS, the user must reserve high memory for the machine language routine by answering 65000 to the MEMORY SIZE query. Z-80 VIDEO SYNC DEMONSTRATION loads the object code from floppy disk and then asks the user two questions. The first question, "How many flashes?", allows the user to call the machine language routine a number of times in succession. 17 MSEC VERTICAL LINE displays the bar for 17 msec and then erases it. So multiple flashes are helpful.

The second question, "Length of pause before drawing line?", allows the user to select the initial microsecond pause that will be taken by the machine language routine. By selecting values in excess of about 4,400 microsec, the user can demonstrate the location of the electron beam at the end of the pause: The portion of the bar above the electron beam will not appear on the screen, even though it existed in video RAM for a time.

Upon answering the computer's queries, the user can implement the demonstration by pressing any key. The program POKEs the correct values in RAM locations for use by 17 MSEC VERTICAL LINE and draws a screen display that helps to locate and interpret the length of the 17-msec bar (Figure 3). For calibration purposes,

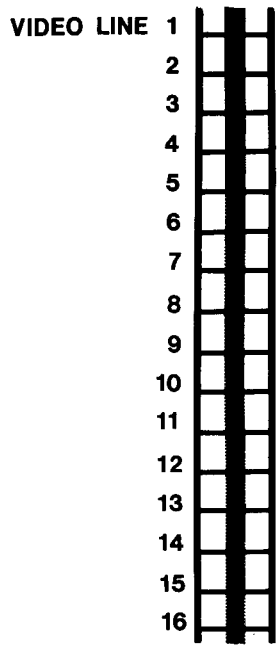

Figure 3. A negative-image representation of the video screen during the execution of Z-80 VIDEO SYNC DEMONSTRATION. Each horizontal line divides a video character line into an upper two-thirds and a lower one-third. The thick central vertical bar depicts the location of the 17-msec display that is carried out by the machine language routine, 17 MSEC VERTICAL LINE.

the short horizontal lines indicate the general location of each video line. About two-thirds of the video line is located above the horizontal line, and one-third below it. All alphanumeric characters appear in the space above the horizontal line; some graphics characters span both above and below the horizontal line.

In research applications, an experimental trial will be set up before the video synchronization routine replaces the RTC interrupt. Often, this can be done within the BASIC portion of the program. All stimulus data can be placed in invisible memory locations. Then, control may be passed to the machine language routine that loads the stimulus data into video RAM in synchronization with the electron beam.

During program development, the programmer should determine the time it will take to carry out all the machine language instructions up to those that begin the stimulus transfer to video RAM. This time should be added to 100 microsec, the approximate time it takes before the interrupt servicing routine begins. The resultant value can be divided by .758 microsec to determine at which line the electron beam is expected to be aimed when the stimulus appears in video RAM. It is then a simple step to calculate the delay between the stimulus's appearance in video RAM and its appearance on the screen. Consequently, the software video synchronization described here provides the programmer with the capability to control stimulus exposure and record response latency with precision that approaches a fraction of 1 msec.

REFERENCES

GRICE, G. R. Accurate reaction time research with the TRS-80 microcomputer. *Behavior Research Methods & Instrumentation*, 1981, **13**, 674-676.

LINCOLN, C. E., & LANE, D. M. Reaction time measurement errors resulting from the use of CRT displays. *Behavior Research Methods & Instrumentation*, 1980, **12**, 27-39.

MERIKLE, P. M., CHEESMAN, J., & BRAY, J. PET Flasher: A machine language subroutine for timing visual displays and response latencies. *Behavior Research Methods & Instrumentation*, 1982, **14**, 26-28.

REED, A. V. Microcomputer display timing: Problems and solutions. *Behavior Research Methods & Instrumentation*, 1979, **11**, 572-575.

ZAKS, R. *How to program the Z-80.* Berkeley: Sybex, 1980.

NOTES

1. Some readers may wonder if the real-time clock-based video synchronization described in this article might be possible for the Model I TRS-80. Unfortunately, it is not. The real-time clock in the older model does not operate from the vertical synch pulse but from a 4-MHz clock in the disk controller circuits. Also, the computer's internal circuits do not allow maskable interrupts of the type used in the Model III's real-time clock interrupt. The hardware modification described by Grice (1981) remains the only viable method for the Model I.

2. The TRS-80 video screen actually performs as if it has 22 lines, even though only 16 lines appear on the screen. The remaining six lines are devoted to vertical retrace of the electron beam. As a result, a full screen can be drawn in about 12.1 msec, not 17 msec. This is followed by a 4.5-msec pause, during which there is no change in the screen display.

3. The author is indebted to Mike Berger, Model III design engineer, who, in a personal communication, provided much of the information about the operation of the TRS-80's real-time clock and interrupt.

4. The author will provide a 5.25-in. double-density floppy disk containing the object and source codes for 17 MSEC VERTICAL LINE and the sample BASIC (TRSDOS 1.3) program that implements it, Z-80 VIDEO SYNC DEMONSTRATION. The cost is $10.

(Received for publication August 27, 1982;
revision accepted October 15, 1982.)