A Modem Interface for the 80s

(Z-80s and 8080s That Is)

by Neil M. Harrington

If you are thinking about a modem for your personal computer but haven't settled on the best approach, here is one to consider: a software driven acoustic coupler. This is applicable to just about any computer, even one without a serial I/O port.

The software will show you how to: interface an acoustic coupler through a parallel I/O port, produce any desired baud rate, and make your personal computer act like a remote terminal.

A simple hardware interface for connecting an acoustic coupler to your computer is one bit in and one bit out from a parallel I/O port. Of course, if you have a serial I/O port, it will considerably simplify the interface.

The complete modem interface program is presented in listing 1. It is written in Z-80 assembly language using Zilog mnemonics. However, only 8080-compatible instructions are used. A quick scan shows that the program is highly modular. This costs a few extra bytes of instructions, but should make it easier to adapt the program to your system.

A flow diagram showing the main program functions is presented in figure 1. Upon entry, the program initializes the video display, then drops into the main routine at lines 400-590 in listing 1. The main routine is an endless loop that alternately polls the modem and keyboard looking for input. If modem input is detected, subroutine 'rmodem' is called to fetch the character; 'dchar' is called to display it. If valid keyboard input is detected, the character is sent to the modem via subroutine 'wmodem.' This process continues until ESC is pressed to exit to the monitor.

Scheme of actual data byte

The modem I/O drivers transfer data between the modem and the computer, translating between serial and parallel data forms in the process. Before examining the software techniques for doing this, let's look at the serial representation of a typical data byte. Figure 2 shows the serial transmission of E which is 1000101 in seven bit Ascii code.

Reading figure 2 from left to right, we first see that a logical one is transmitted between characters. The beginning of the character is signalled by a logical zero start bit, followed by the seven Ascii data bits in reverse order and then a parity bit. Finally, the character is terminated by one or two stop bits at logical one.

The parity bit is provided as an optional data check. It is set either to one or to zero by the transmitter to

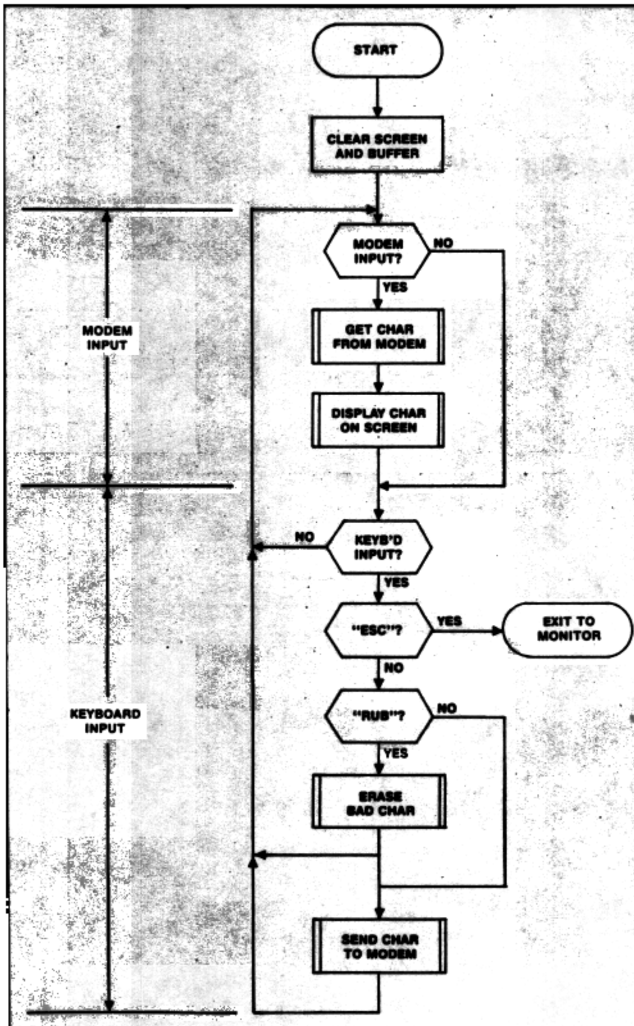

Figure 1. Program flow diagram.

make the sum of the eight data bits an odd number (for odd parity) or an even number (for even parity). The receiver checks the sum of the eight bits to determine if a transmission has occurred. The parity bit is usually ignored by time-shared computers, so it is not implemented in the modem drivers. However, only a few additional instructions are required to include parity if desired.

Now let's look at the modem driver software. Subroutine 'rmodem' (lines 820-940 in listing 1) accepts serial bits from the modem and returns with the input character in the accumulator. 'Rmodem' is called as soon as the start bit is detected (i.e., the signal drops from one to zero). 'Rmodem' calls 'delay1' at line 830 to wait for the center of the first data bit to arrive. Each of the eight data bits is read in turn from the least significant bit (LSB) of input port 'inmodm.' As each bit is read into the accumulator, it is added to the bits read previously and saved temporarily in the D register. The precise time between each bit is obtained by the call to 'delay' at line 900. On exit from 'rmodem,' the completed data byte is in the accumulator. At line 930, the most significant bit (MSB) is set for proper handling in the display routines.

An Ascii character input from the keyboard goes into the accumulator. From there, subroutine 'wmodem' (lines 1150-1310) transmits it to the modem a bit at a time from the LSB of output port 'otmodm.' First a zero start bit is sent (lines 1170-1190). The eight data bits are sent out one at a time starting with the LSB (lines 1200-1260). The precise time required for each bit is provided by the call to 'delay' at line 1240. Finally, two stop bits are transmitted at lines 1270-1300.

Software baud rate generator

The precise time between bits is provided by a short subroutine 'delay' (lines 1350-1410). This is the heart of the modem interface, so let's take a closer look. 'Delay' consists of two tightly nested loops. The number of times the loops are executed depends on the integer timing constants TIM1 and TIM2. Since the values for TIM1 and TIM2 will differ for each baud rate and CPU clock frequency, we will need a convenient way to compute them.

The time per bit produced by the timing loops can be expressed by the following equation:

TL = (C*(14*B + 21) + 73)/F where
C = TIM1
B = TIM2
F = CPU clock frequency

The constants 14, 21 and 73 were obtained by adding the number of CPU cycles required to execute the instructions in each loop. (The value 73 includes the average number of cycles for lines 850-920 of 'rmodem' and lines 1200-1260 of 'wmodem.) If F is entered in megahertz, TL will be in microseconds.

The required time per bit for a given baud rate R is given by TR = 1.E6/R, where the constant 1.E6 causes the value for TR to be in microseconds. Equating TL with TR and solving for B to the nearest integer, we obtain

B = INT(((1.E6*F/R-73)/C-21)/14 + 0.5).

The resultant percentage error in the time per bit is E = 100*(TL-TR)/TR. Substituting for TL and TR, we obtain

E = 100*((C*(14*B + 21) + 73)*1.E-6*R/F-1).

The optimum values for the timing constants B and C are those that produce minimum error, E. These values can best be found by systematically selecting values for C and then computing values for B and E from the equations above. It should be noted that C must be an even number because a third timing constant, TIM0, is C*3/2 to time a bit and a half.

To make it easy to determine timing constants, the whole procedure outlined above has been put into a Basic program. The program is given in listing 2, and a sample run for 300 baud at a clock frequency of 2.5 MHz is shown in the table, indicating that minimum timing error is achieved with C = 4 and B = 146 or with C = 20 and B = 28. The timing constants are entered in the assembly program at lines 2600-2620.

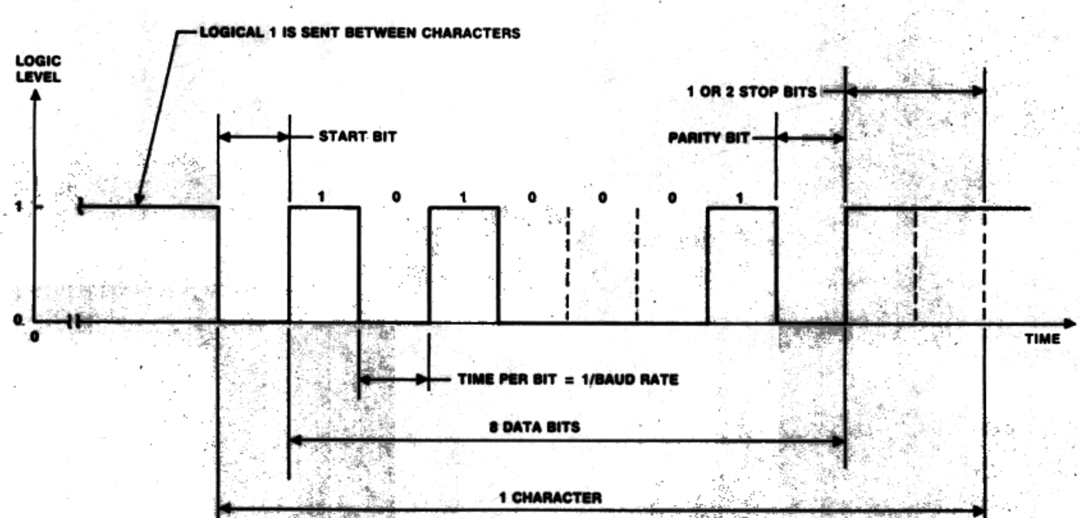

Figure 2. Transmission of Ascii character, E = 1000101. (Note: bits are sent in reverse order)

The I/O functions performed by the modem interface program are now described. Since computers differ widely in the way they interface with the keyboard and video display, it is unlikely the listed I/O routine can be used as is. Therefore it is important to understand how they work so you can modify them for your computer.

A personal example

The keyboard on my system works as follows: while a key is depressed, its Ascii code is present at input port 'inkeyb.' When the key is pressed, the MSB is strobed (i.e., goes briefly to logical one and then returns to logical zero). To avoid sending a character to the modem more than once per keystroke, the strobe bit is tested at line 540. If the strobe bit is off, the character is ignored unless it is a control character. The character displayed after a keystroke is the one echoed by the host computer to show that it was received properly.

The video monitor on my system displays 1024 characters as 16 lines of 64 characters each. Characters are sent to the screen via output port 'otscrn.' All characters appearing on the screen are contained in a 1024 byte circular buffer starting at address 'buffer' and ending at 'endbuf.' It is not necessary to start the buffer on a page boundary; it may reside in memory wherever you choose. Characters in the buffer are never moved. Instead, pointers to the current first and last lines (top and inline) are maintained. When it is time to scroll the display, these pointers are simply moved down to the next line in the buffer. If this action moves a pointer beyond the physical end of the buffer, the pointer is reset to the beginning.

Here are brief descriptions of the three main I/O routines:

RUBOUT (lines 630-780): used to erase an incorrect keyboard input. The bad character is removed from the screen and the buffer. A delete is sent to the modem, and the next three characters are intercepted to suppress their display. (These characters are "backslash-badchar-backslash".) Special routines 'erase,' 'bspace' and 'space' are used.

DCHAR (lines 980-1110): if the byte in the accumulator is a printable Ascii character, this stores it in the buffer, displays it on the screen and advances the cursor using special subroutines 'tvout' and 'dscrsr.'

LNFEED (lines 1450-1560): if the character in the accumulator is a carriage return with MSB set, this routine will scroll the display using special subroutines 'movelp,' 'filmem' and 'dsplay.'

Hardware requirements

Only three electrical connections are required: signal in, signal out and ground. If a serial port is available, hook up the three wires and you are ready to go.

However, if you must connect to a parallel port, translate voltage levels between the computer and the acoustic coupler. The computer's parallel port uses TTL levels (i.e., anything from 0.0 to 0.8 volt is a logical zero; anything from 2.4 to 5.0 volts is a logical one). The acoustic coupler uses RS-232C levels: +5 to +15 volts is a logical zero; and −5 to −15 volts is a logical one.

There are a number of simple circuits for effecting the required voltage conversions. I chose MC1488/1489 integrated circuits specifically designed for this purpose. The complete hardware interface is shown in figure 3.

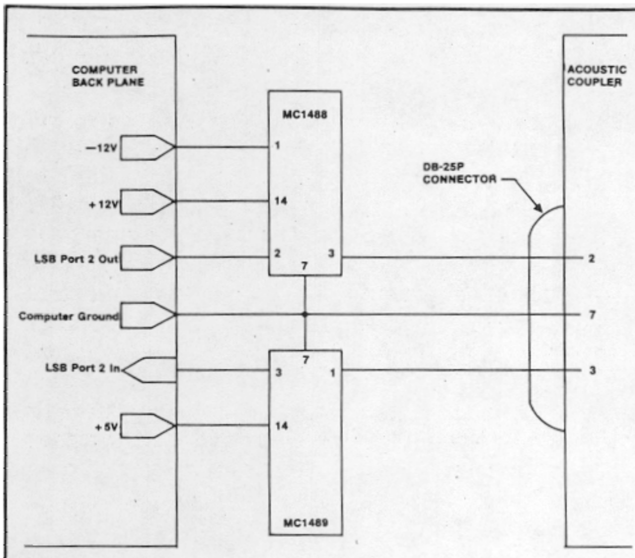

Figure 3. Hardware interface for parallel I/O port.

Here are brief summaries of some program changes that may be needed to customize the software:

Timing constants (lines 2600-2620 in listing 1) will probably have to be changed for your computer. The Basic program in listing 2 determines the best timing constant values for your computer's clock frequency. In the program output, C is TIM1 and B is TIM2.

Keyboard and video routines of the modem interface software will require the greatest amount of change. To design them effectively, it will be necessary to learn how I/O is handled in your system. One note of caution: if I/O functions are interrupt driven, disable interrupts on entry to subroutines 'rmodem' and 'wmodem,' and reenable them on exit. This is essential to avoid interruption during the timing loops.

Special control characters can be added to instructions near lines 500-530 of listing 1. On some host computers, control-S suspends program operation and control-Q restarts it. If control-S is keyed in while a stream of characters is being received from the modem, the ensuing control-Q restart will usually produce gibberish. This is overcome by adding code to cause the modem program to wait until a carriage return is received before transmitting a control-S character. Restart will then begin cleanly at the start of the next line.

On some systems, control-O toggles the program output on/off without suspending program execution. Here it is essential to add instructions to ensure that only one control-O is transmitted per key-in.

I have used the modem interface program to successfully connect into four different computers (VAX/11-780, Univac 1108, IBM 360 and PDP-10). Communication has been accomplished at 110, 300 and 600 baud. It works! If you decide to implement a software modem interface, you will learn a lot about your personal computer. And you might even have fun doing it. □

Program on Page 134

A Modem Interface

Continued from Page 98

LISTING 1

```
MODEM INTERFACE ASSEMBLY LISTING

6200                0100 ***********************************************************
6200                0110 *   Z80/8080 MODEM INTERFACE   VERSION 1.3                 *
6200                0120 *     N. M. HARRINGTON   NOVEMBER, 1979                    *
6200                0130 ***********************************************************
6200                0140 *
6200                0150 *    INITIALIZE SCREEN BUFFER
6200                0160 *-----------------------------------------------------------
6200 CD 55 63       0170        CALL CLEARS        :CLEAR SCREEN.
6203 21 72 63       0180        LD   HL,BUFFER     :INITIALIZE TOP OF DISPLAY
6206 22 72 67       0190        LD   (TOP),HL      :   POINTER.
6209 3E A0          0200        LD   A,BLANK       :FILL BUFFER WITH BLANKS.
620B 0E 10          0210        LD   C,NLINES
620D 06 40          0220 CLRBUF LD   B,NCHARS
620F CD 4E 63       0230        CALL FILMEM
6212 0D             0240        DEC  C
6213 C2 0D 62       0250        JP   NZ,CLRBUF
6216 0E 0F          0260        LD   C,NLINES-1    :MOVE TO BOTTOM LINE ON SCREEN.
6218 06 40          0270 CLOOP  LD   B,NCHARS
621A CD 6A 63       0280 BLOOP  CALL SPACE
621D 05             0290        DEC  B
621E C2 1A 62       0300        JP   NZ,BLOOP
6221 0D             0310        DEC  C
6222 C2 18 62       0320        JP   NZ,CLOOP
6225 21 32 67       0330        LD   HL,LASLIN     :SET INPUT LINE POINTER TO LAST
6228 22 74 67       0340        LD   (INLINE),HL   :   LINE ON SCREEN.
622B CD 45 63       0350        CALL DSCRSR        :DISPLAY CURSOR.
622E                0360 *===========================================================
622E                0370 *   *****  MAIN ROUTINE  *****
622E                0380 *   ALTERNATELY POLL MODEM, THEN KEYBOARD.
622E                0390 *===========================================================
622E DB 02          0400 CYCLE  IN   INMODM        :CHECK MODEM.
6230 E6 01          0410        AND  1             :ANYTHING THERE?
6232 C2 3B 62       0420        JP   NZ,KEYIN      :NO.  GO CHECK KEYBOARD.
6235 CD 88 62       0430        CALL RMODEM        :GET CHARACTER FROM MODEM.
6238 CD A2 62       0440        CALL DCHAR         :DISPLAY IT.
623B DB 00          0450 KEYIN  IN   INKEYB        :NOW CHECK KEYBOARD.
623D FE 00          0460        CP   0             :ANYTHING THERE?
623F CA 2E 62       0470        JP   Z,CYCLE       :NO.  GO AROUND AGAIN.
6242 FE 20          0480        CP   CCHAR         :IS IT A CONTROL CHARACTER?
6244 D2 51 62       0490        JP   NC,STROBE     :NO.  GO CHECK STROBE BIT.
6247 FE 1B          0500        CP   ESCAPE          :IS IT "ESCAPE" KEY?
6249 CA 00 05       0510        JP   Z,MONITR      :YES.  EXIT TO OP SYSTEM.
624C FE 0D          0520        CP   CR            :IS IT "RETURN" KEY?
624E C2 5B 62       0530        JP   NZ,WRITEM     :NO.  SEND CHARACTER TO MODEM.
6251 FE 80          0540 STROBE CP   MSBIT         :HAS KEY JUST BEEN PRESSED?
6253 DA 2E 62       0550        JP   C,CYCLE       :NO.  IGNORE IT.

62BD 57             1150 WMODEM LD   D,A           :PUT CHARACTER IN D.
62BE 1E 08          1160        LD   E,8D          :SET E FOR 8 BITS.
62C0 AF             1170        XOR  A             :CLEAR A.
62C1 D3 02          1180        OUT  OTMODM        :SEND START BIT.
62C3 CD DD 62       1190        CALL DELAY         :TIME THE BIT.
62C6 7A             1200 OUTIT  LD   A,D           :GET THE CHARACTER.
62C7 D3 02          1210        OUT  OTMODM        :SEND THE LSB TO MODEM.
62C9 0F             1220        RRCA               :SHIFT THE NEXT BIT TO LSB.
62CA 57             1230        LD   D,A           :SAVE IN D.
62CB CD DD 62       1240        CALL DELAY         :TIME THE BIT.
62CE 1D             1250        DEC  E             :DONE WITH 8 BITS?
62CF C2 C6 62       1260        JP   NZ,OUTIT      :NO.  GO SEND NEXT BIT.
62D2 3E 01          1270        LD   A,1           :DONE.  SEND 2 STOP BITS.
62D4 D3 02          1280        OUT  OTMODM
62D6 CD DD 62       1290        CALL DELAY
62D9 CD DD 62       1300        CALL DELAY
62DC C9             1310        RET
62DD                1320 *-----------------------------------
62DD                1330 *    BAUD RATE TIMING LOOP
62DD                1340 *-----------------------------------
62DD 0E 04          1350 DELAY  LD   C,TIM1
62DF 06 92          1360 DELAY1 LD   B,TIM2
62E1 05             1370 STALL  DEC  B
62E2 C2 E1 62       1380        JP   NZ,STALL
62E5 0D             1390        DEC  C
62E6 C2 DF 62       1400        JP   NZ,DELAY1
62E9 C9             1410        RET
62EA                1420 *-----------------------------------
62EA                1430 *    SUBROUTINE TO PROCESS A LINEFEED
62EA                1440 *-----------------------------------
62EA 2A 72 67       1450 LNFEED LD   HL,(TOP)      :MOVE TOP OF DISPLAY POINTER
62ED CD 25 63       1460        CALL MOVELP        :   DOWN A LINE.
62F0 22 72 67       1470        LD   (TOP),HL
62F3 2A 74 67       1480        LD   HL,(INLINE)   :MOVE INPUT LINE POINTER DOWN
62F6 CD 25 63       1490        CALL MOVELP        :   A LINE.
62F9 22 74 67       1500        LD   (INLINE),HL
62FC 3E A0          1510        LD   A,BLANK       :BLANK OUT BOTTOM DISPLAY LINE
62FE 06 40          1520        LD   B,NCHARS      :   IN BUFFER.
6300 CD 4E 63       1530        CALL FILMEM
6303 CD 0A 63       1540        CALL DSPLAY        :SCROLL THE DISPLAY.
6306 CD 45 63       1550        CALL DSCRSR        :DISPLAY CURSOR AT BOTTOM LINE.
6309 C9             1560        RET
630A                1570 *-----------------------------------
630A                1580 *    SUBROUTINE TO PUT ENTIRE DISPLAY BUFFER ON SCREEN
630A                1590 *-----------------------------------
630A CD 55 63       1600 DSPLAY CALL CLEARS        :FIRST CLEAR SCREEN.
630D 2A 72 67       1610        LD   HL,(TOP)      :POINT TO TOP OF DISPLAY.
6310 0E 0F          1620        LD   C,NLINES-1    :SET C AND B FOR NLINES -1 OF
6312 06 40          1630 LINE   LD   B,NCHARS      :   NCHARS EACH.
6314 7E             1640 CHAR   LD   A,M           :GET CHARACTER FROM BUFFER.
6315 CD 6C 63       1650        CALL TVOUT         :DISPLAY IT.
6318 23             1660        INC  HL            :ADVANCE POINTER TO NEXT CHAR.
6319 05             1670        DEC  B             :DO UNTIL END OF LINE.
631A C2 14 63       1680        JP   NZ,CHAR
631D CD 29 63       1690        CALL TESTLP        :POINT TO NEXT LINE IN BUFFER.
6320 0D             1700        DEC  C             :DONE WITH NLINES -1?
6321 C2 12 63       1710        JP   NZ,LINE       :NO.  GO DO ANOTHER LINE.
6324 C9             1720        RET
```

```
6256 FE FF           CP   DEL          :IS IT "DELETE" KEY?
6258 CA 61 62        JP   Z,RUBOUT     :YES.  GO ERASE CHARACTER.
625B CD BD 62 WRITEM CALL WMODEM       :SEND CHARACTER TO MODEM.
625E C3 2E 62        JP   CYCLE        :GO AROUND AGAIN.
6261          *-----------------------------------------------
6261          *   ROUTINE TO ERASE A BAD INPUT CHARACTER
6261          *-----------------------------------------------
6261 CD 39 63 RUBOUT CALL ERASE        :FIRST ERASE CURSOR FROM SCREEN.
6264 CD 39 63        CALL ERASE        :THEN ERASE PRECEDING CHARACTER.
6267 2A 76 67        LD   HL,(CURSOR)  :BACK UP THE CURSOR.
626A 2B              DEC  HL
626B 36 A0           LD   M,BLANK      :BLANK OUT CHARACTER IN BUFFER.
626D CD 45 63        CALL DSCRSR
6270 3E FF           LD   A,DEL        :SEND "DELETE" CHARACTER TO
6272 CD BD 62        CALL WMODEM       :   MODEM.
6275 2E 03           LD   L,3          :NOW INTERCEPT NEXT 3 CHARACTERS:
6277 DB 02    INCEPT IN   INMODM       :   BACKSLASH, DELETED CHARACTER,
6279 E6 01           AND  1            :   BACKSLASH.
627B C2 77 62        JP   NZ,INCEPT
627E CD 88 62        CALL RMODEM
6281 2D              DEC  L
6282 C2 77 62        JP   NZ,INCEPT
6285 C3 3B 62        JP   KEYIN        :GO GET CORRECT KEYIN.
6288          *-----------------------------------------------
6288          *   SUBROUTINE TO READ A CHARACTER FROM MODEM
6288          *-----------------------------------------------
6288 0E 06    RMODEM LD   C,TIMO       :WAIT FOR CENTER OF FIRST DATA
628A CD DF 62        CALL DELAY1       :   BIT TO ARRIVE.
628D 11 08 00        LD   DE,8D        :CLEAR D.  SET E FOR 8 BITS.
6290 DB 02    NEXT   IN   INMODM       :GET INPUT FROM MODEM.
6292 E6 01           AND  1            :MASK LSB.
6295 82              ADD  D            :ADD IN PREVIOUS BITS READ.
6296 0F              RRCA              :SHIFT FOR NEXT INPUT.
6297 57              LD   D,A          :SAVE PARTIAL BYTE IN D.
6298 CD DD 62        CALL DELAY        :WAIT FOR CENTER OF NEXT BIT.
629B 1D              DEC  E            :DONE WITH 8 BITS?
629C C2 90 62        JP   NZ,NEXT      :NO.  GO GET NEXT BIT.
629F F6 80           OR   80H          :DONE.  SET MSB.
62A1 C9              RET
62A2          *-----------------------------------------------
62A2          *   SUBROUTINE TO DISPLAY CHARACTER ON SCREEN
62A2          *-----------------------------------------------
62A2 FE 8D    DCHAR  CP   LF           :IS IT "LINEFEED"?
62A4 CC EA 62        CALL Z,LNFEED     :YES.  GO DO IT.
62A7 FE A0           CP   BLANK        :IS IT A CONTROL CHARACTER?
62A9 D8              RET  C            :YES.  IGNORE IT.
62AA FE FF           CP   DEL          :IS IT "DELETE"?
62AC C8              RET  Z            :YES.  IGNORE IT.
62AD 2A 76 67        LD   HL,(CURSOR)  :STORE CHARACTER IN SCREEN BUFFER
62B0 77              LD   M,A          :   AND DISPLAY IT ON SCREEN IN
62B1 CD 3F 63        CALL BSPACE       :   PLACE OF CURSOR.
62B4 7E              LD   A,M
62B5 CD 6C 63        CALL TVOUT
62B8 23              INC  HL           :ADVANCE CURSOR AND DISPLAY IT.
62B9 CD 45 63        CALL DSCRSR
62BC C9              RET
62BD          *-----------------------------------------------
62BD          *   SUBROUTINE TO WRITE A CHARACTER TO MODEM
62BD          *-----------------------------------------------
```

```
6325          *-----------------------------------------------
6325          *   SUBROUTINE TO ADVANCE BUFFER POINTER TO NEXT LINE
6325          *-----------------------------------------------
6325 11 40 00 MOVELP LD   DE,NCHARS    :ADD NCHARS TO CURRENT POINTER
6328 19              ADD  HL,DE        :   LOCATION.
6329 11 71 67 TESTLP LD   DE,ENDBUF    :NOW CHECK IF THIS PUTS POINTER
632C 7C              LD   A,H          :   PAST THE END OF THE ACTUAL
632D BA              CP   D            :   BUFFER REGION.
632E D8              RET  C
632F C2 35 63        JP   NZ,WRAP
6332 7D              LD   A,L
6333 BB              CP   E
6334 D8              RET  C            :OK.  RETURN.
6335 21 72 63 WRAP   LD   HL,BUFFER    :WRAP AROUND TO START OF ACTUAL
6338 C9              RET               :   BUFFER REGION.
6339          *-----------------------------------------------
6339          *   SUBROUTINE TO ERASE A CHARACTER
6339          *-----------------------------------------------
6339 CD 3F 63 ERASE  CALL BSPACE       :BACK DISPLAY UP OVER LAST CHAR.
633C CD 6A 63        CALL SPACE        :BLANK IT OUT.
633F          *                        :DROP THRU TO BACK OVER BLANK.
633F          *-----------------------------------------------
633F          *   SUBROUTINE TO BACKSPACE SCREEN DISPLAY
633F          *-----------------------------------------------
633F 3E 06    BSPACE LD   A,BAKSPC     :LOAD BACKSPACE CHARACTER.
6341 CD 6C 63        CALL TVOUT        :SEND IT TO SCREEN.
6344 C9              RET
6345          *-----------------------------------------------
6345          *   SUBROUTINE TO DISPLAY CURSOR
6345          *-----------------------------------------------
6345 22 76 67 DSCRSR LD   (CURSOR),HL  :SAVE PRESENT CURSOR LOCATION.
6348 3E DF           LD   A,USCORE     :LOAD UNDERSCORE CHARACTER.
634A CD 6C 63        CALL TVOUT        :DISPLAY IT.
634D C9              RET
634E          *-----------------------------------------------
634E          *   SUBROUTINE TO FILL A BLOCK OF MEMORY
634E          *-----------------------------------------------
634E 77       FILMEM LD   M,A          :PUT CHARACTER IN MEMORY.
634F 23              INC  HL           :ADVANCE MEMORY POINTER.
6350 05              DEC  B            :DO IT B TIMES.
6351 C2 4E 63        JP   NZ,FILMEM
6354 C9              RET
6355          *-----------------------------------------------
6355          *   SUBROUTINE TO CLEAR SCREEN AND HOME TO UPPER LEFT
6355          *-----------------------------------------------
6355 3E 7F    CLEARS LD   A,RUB        :LOAD DELETE CHARACTER.
6357 CD 6C 63        CALL TVOUT        :HOME TO UPPER LEFT CORNER.
635A 06 10           LD   B,NLINES     :OUTPUT NULL CHARACTERS TO
635C 0E 40    CLOAD  LD   C,NCHARS     :   ERASE SCREEN.
635E CD 6A 63 BLANKM CALL SPACE
6361 0D              DEC  C
6362 C2 5E 63        JP   NZ,BLANKM
6365 05              DEC  B
6366 C2 5C 63        JP   NZ,CLOAD
6369 C9              RET
636A          *-----------------------------------------------
636A          *   SUBROUTINE TO DISPLAY A BLANK OR A CHARACTER
636A          *-----------------------------------------------
636A 3E A0    SPACE  LD   A,BLANK      :LOAD A BLANK.
```

```
636C D3 00      TVOUT   OUT  OTSCRN            :DISPLAY CHARACTER IN A.
636E AF                 XOR  A                 :CLEAR A.
636F D3 00              OUT  OTSCRN            :OUTPUT A NULL CHARACTER.
6371 C9                 RET
6372            *==========================================================
6372            *   SYMBOL DEFINITIONS
6372            *----------------------------------------------------------
6372            BLANK   EQU  0A0H              :BLANK CHARACTER
6372            CR      EQU  0DH               :CARRIAGE RETURN
6372            LF      EQU  8DH               :LINE FEED (CR W MSB ON)
6372            RUB     EQU  7FH               :RUBOUT CHARACTER
6372            DEL     EQU  0FFH              :DELETE CHARACTER
6372            USCORE  EQU  0DFH              :UNDERSCORE CHARACTER
6372            ESCAPE  EQU  1BH               :ESCAPE CHARACTER
6372            BAKSPC  EQU  6                 :BACKSPACE CONTROL CHARACTER
6372            MSBIT   EQU  80H               :MOST SIGNIFICANT BIT
6372            MONITR  EQU  0500H             :ADDRESS OF MONITOR
6372            CCHAR   EQU  20H               :1ST CHAR BEYOND CONTROL CHARS.
6372            *----------------------------------------------------------
6372            *   I/O PORT DEFINITIONS
6372            *----------------------------------------------------------
6372            INKEYB  EQU  0                 :KEYBOARD INPUT PORT
6372            OTSCRN  EQU  0                 :VIDEO OUTPUT PORT
6372            INMODM  EQU  2                 :MODEM INPUT PORT
6372            OTMODM  EQU  2                 :MODEM OUTPUT PORT
6372            *----------------------------------------------------------
6372            *   TIMING CONSTANTS
6372            *----------------------------------------------------------
6372            TIM1    EQU  4                 :TIMING LOOP CONSTANTS FOR 300
6372            TIM2    EQU  146D              :    BAUD WITH 2.5 MHZ CLOCK
6372            TIM0    EQU  6                 :TIM1*3/2
6372            *----------------------------------------------------------
6372            *   VIDEO DISPLAY BUFFER STORAGE
6372            *----------------------------------------------------------
6372            NLINES  EQU  16D               :NUMBER OF LINES ON SCREEN
6372            NCHARS  EQU  64D               :NUMBER OF CHARACTERS PER LINE
6372            NBUF    EQU  1024D             :NLINES*NCHARS
6372            BUFFER  DS   NBUF              :STORAGE FOR NBUF CHARACTERS
6772            ENDBUF  EQU  BUFFER+NBUF-1     :ADDRESS OF LAST LOC IN BUFFER
6772            LASLIN  EQU  BUFFER+NBUF-NCHARS
6772            TOP     DS   2                 :POINTER TO 1ST LINE OF DISPLAY
6774            INLINE  DS   2                 :POINTER TO INPUT (BOTTOM) LINE
6776            CURSOR  DS   2                 :POINTER TO CURSOR LOCATION

SYMB
CLRBUF 620D   CLOOP  6218   BLOOP  621A   CYCLE  622E
KEYIN  623B   STROBE 6251   WRITEM 625B   RUBOUT 6261
INCEPT 6277   RMODEM 6288   NEXT   6290   DCHAR  62A2
WMODEM 62BD   OUTIT  62C6   DELAY  62DD   DELAY1 62DF
STALL  62E1   LNFEED 62EA   DSPLAY 630A   LINE   6312
CHAR   6314   MOVELP 6325   TESTLP 6329   WRAP   6335
ERASE  6339   BSPACE 633F   DSCRSR 6345   FILMEM 634E
CLEARS 6355   CLOAD  635C   BLANKM 635E   SPACE  636A
TVOUT  636C   BLANK  00A0   CR     000D   LF     008D
RUB    007F   DEL    00FF   USCORE 00DF   ESCAPE 001B
BAKSPC 0006   MSBIT  0080   MONITR 0500   CCHAR  0020
INKEYB 0000   OTSCRN 0000   INMODM 0002   OTMODM 0002
TIM1   0004   TIM2   0092   TIM0   0006   NLINES 0010
NCHARS 0040   NBUF   0400   BUFFER 6372   ENDBUF 6771
LASLIN 6732   TOP    6772   INLINE 6774   CURSOR 6776
```

LISTING 2

BASIC PROGRAM TO COMPUTE TIMING LOOP CONSTANTS

```
10 INPUT "ENTER CLOCK FREQUENCY IN MEGAHERTZ: ", F
20 INPUT "ENTER BAUD RATE: ", R
30 C = 0
40 PRINT
50 PRINT "  C      B       %ERROR"
60 PRINT "______ ______ _________"
70 FOR I=1 TO 10
80 C = C + 2
90 IF C>255 THEN 170
100 B = INT(((1.E6*F/R - 73)/C - 21)/14 + .5)
110 IF B>255 THEN 80
120 E = 100*((C*(14*B + 21) + 73)*1.E-6*R/F - 1)
130 PRINT C, B, E
140 NEXT I
150 INPUT "DO YOU WANT MORE (Y OR N)", A$
160 IF A$="Y" THEN 40
170 END
```

SAMPLE RUN

SAMPLE RUN OF BASIC TIMING CONSTANT PROGRAM

```
READY
RUN

ENTER CLOCK FREQUENCY IN MEGAHERTZ: 2.5
ENTER BAUD RATE: 300

  C       B       %ERROR
______  ______  _________
  4      146     -.004
  6       97     .164
  8       72     -.34
 10       58     .836
 12       48     .668
 14       41     .836
 16       35     -1.012
 18       31     -.844
 20       28     -.004
 22       25     -1.18
DO YOU WANT MORE (Y OR N)N
READY
```