Computer Bits

By Jerry Ogdin

HOBBYIST INTERCHANGE TAPE SYSTEM

COMPUTER hobbyists have an insatiable appetite for new programs. Consequently, they are increasingly using the practice of sharing their programs.

But efficient sharing requires a common communications medium. Short programs can be exchanged easily by correspondence on a typewriter or even longhand. As software becomes more complex, however, the possibility of translation error increases so it is essential that a universally recognized exchange medium be used. Further, price and simplicity are of great importance since many hobbyists can't afford expensive commercial equipment.

With no such common exchange medium available to hobbyists today, we have taken the bull by the horns and developed a standard which we think meets all of the foregoing requirements. We call it the Hobbyists Interchange Tape System or simply HIT. The system uses an ordinary low-cost audio cassette tape recorder as the hardware/software interface; and it can be adapted for use with any computer. In the following discussion, HIT is used with an 8080 CPU-design microcomputer.

HIT is probably not the most efficient nor simplest possible system, but we think it is the best compromise for public interchange of software. At the tape speeds used, data will appear on the tape at rates between 30 and 360 bits per second—not a blindingly fast speed, but reliable! However, by changing some of the circuit and software values and using a high-quality recorder, 2500 bits per second can be achieved.

The technique does not depend on frequency, amplitude, or phase. Indeed, the low-cost cassette recorder does not even have to handle digital pulses directly. In practice, short and long bursts of tone are used, with each zero bit represented by a short burst and each one bit by a long burst. Here is how it works.

Basic Theory. Every digital pulse has a leading and trailing edge; a bit interval extends from the leading edge of one bit to the leading edge of the next. If we synchronously count up during the time from the leading edge to the trailing edge, as shown by the dotted line in Fig. 1, and then count down from the trailing edge to the next leading edge, we can determine whether the pulse is long or short. If, as shown in the upper waveform of Fig. 1, we can count down to zero before the next leading edge, we know that the data bit was a "0". If, however, the counter is stopped by the leading edge of the next pulse (lower waveform), we know that the bit was long and the data was a "1."

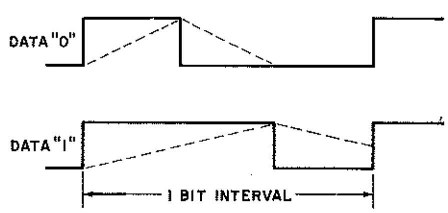

Fig. 1. Pulse waveforms show how zero and one bits differ in length.

Unfortunately, steep-edged pulses are unacceptable to most cassette recorders. So we convert them into audio tones, with a data pulse represented by a burst of approximately 2000 Hz, which is compatible with most low-cost recorders. The schematic for the complete HIT translator is shown in Fig. 2, and the associated waveforms are shown in Fig. 3.

The output of the computer consists of two data lines from an output port latch. One (Fig. 3A) is called the envelope and is true during the tone burst. The other (B) is called modulation and is a software-controlled 2000-Hz square wave. Op amp *IC1A* converts the TTL-level signals into an approximate 2-kHz sine-wave burst (D) which can be recorded easily on any tape machine. The output of *IC1A* is about 2 volts peak-to-peak at the AUX output jack and about 50 mV at the MIC jack. When recording on a stereo cassette, write this data into the *right* channel.

The playback circuit takes the re-

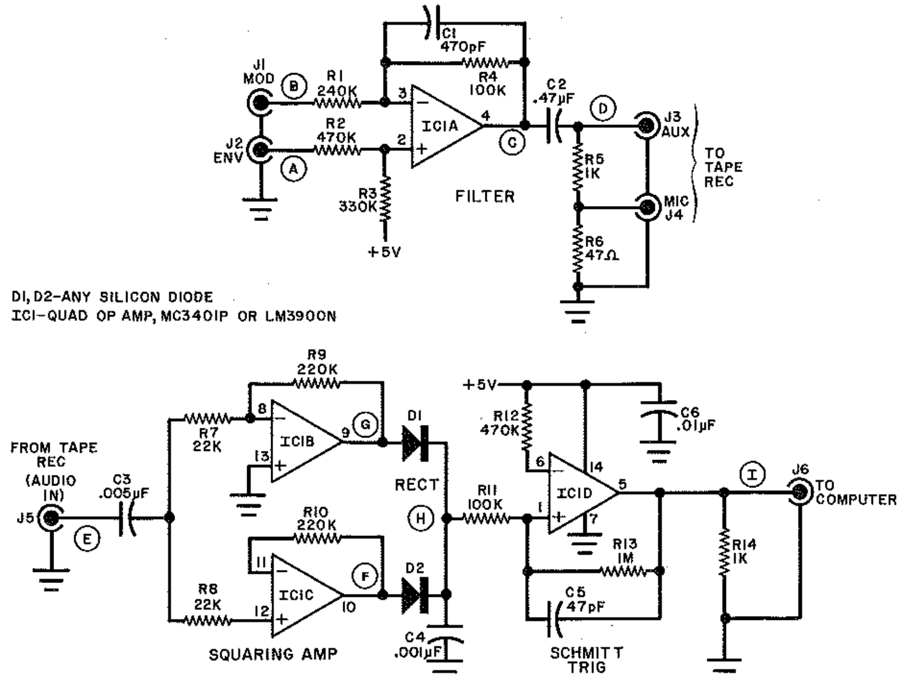

Fig. 2. At top is schematic for recording end of HIT system. Circuit at bottom reads from cassette into computer.

corded data signal from the tape recorder (Fig. 3E) and converts it back to the original digital signal. This circuit, consisting of *IC1B, C,* and *D,* works with an input signal level from 0.75 V to 4 V, although 2 to 2.5 V is ideal. The input is squared up in *IC1B* and *IC1C* (Figs. 3F and G) and then rectified by diodes *D1* and *D2.* The combined output (H) is then applied to a Schmitt trigger *(IC1D)* which produces the output signal (I), an exact reproduction of the original envelope input.

The frequency of the tone burst is not critical. In writing a tape to be mailed to another person, use a frequency near 2 kHz as the modulating input. The reliability of the recorded data depends on how long each pulse is written. With very brief tone bursts, the data rate is high, but the reliability can be adversely affected by poor-quality tape and inexpensive cassette recorders. Each bit may be as short as 1250 microseconds or as long as 35 milliseconds, depending on the writer of the tape. In the programs that follow, 2.75 milliseconds is used as the bit time. The playback circuit and software should be capable of adapting automatically to pulse lengths since it is the ratio of the first half to the second half that determines the data value.

With this wide range of permissible pulse lengths, virtually any computer can be used to write these standard format tapes. Even the slower 8008 CPU can write out bits that have 1-ms durations and still be able to recover them successfully.

Programs. The software we have used with an 8080 is shown in Program 1 (overleaf). The output port (named TAPEO in the program) puts the envelope signal on the most-significant bit and the modulation on the least-significant bit. Since most output ports are TTL-compatible, the simple writing circuit of Fig. 2 can be directly connected. Each data bit is shifted into the CARRY flag of the computer, where the decision to emit a short or long pulse is made. The least-significant bit of the counter is used to determine how long to emit the tone burst (modulation) signal. After all of the tone burst has been sent out, we wait in a counting loop (built into the program) for some tape to move past the recording head before starting the next output bit.

Nine bits are written for each 8-bit byte. Since this new recording scheme uses the leading edge of a burst as the "clock," it is necessary to assure that there is a data bit after the eighth bit of a byte. This ninth bit is always written as a "0". The time that it takes this bit to move past the recording head is the time that we can use to process the character and store it away in memory.

The data rate is 364 bits per second, using all the values in the illustrations. This writing routine, like the reading routine of Program 2, is critically timed. Consequently, do not change the instruction sequences unless you fully understand the timing relationships of the instructions.

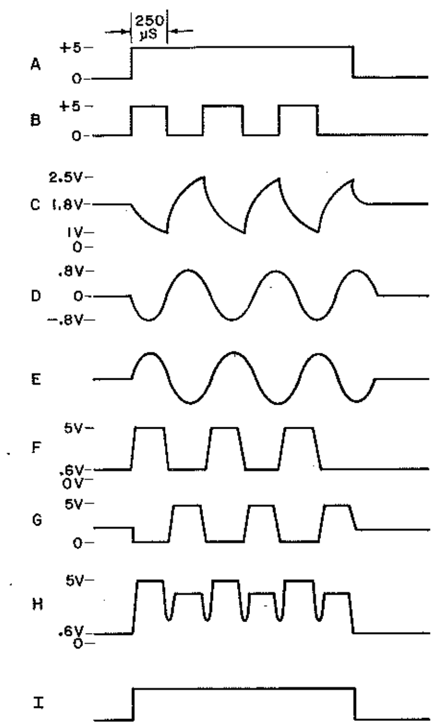

Fig. 3. Waveforms at various points in the writing and reading circuits of the HIT.

In reading the data back in, the input port (the least significant bit is used) is examined until a zero-to-one transition is found; that is the leading edge of the burst. We now count up (in the B register) until the trailing edge is found. After that, we count down until either a new leading edge is detected (making the data bit a 1), or the counter goes to zero (data is a 0). Note that each bit condition must be sensed two times in succession to be considered valid. This provides noise protection.

Each time a bit is found, it is shifted into place. After eight bits are located, the return is taken. When the character reading routine returns, the leading edge of the ninth bit has thus always been sensed.

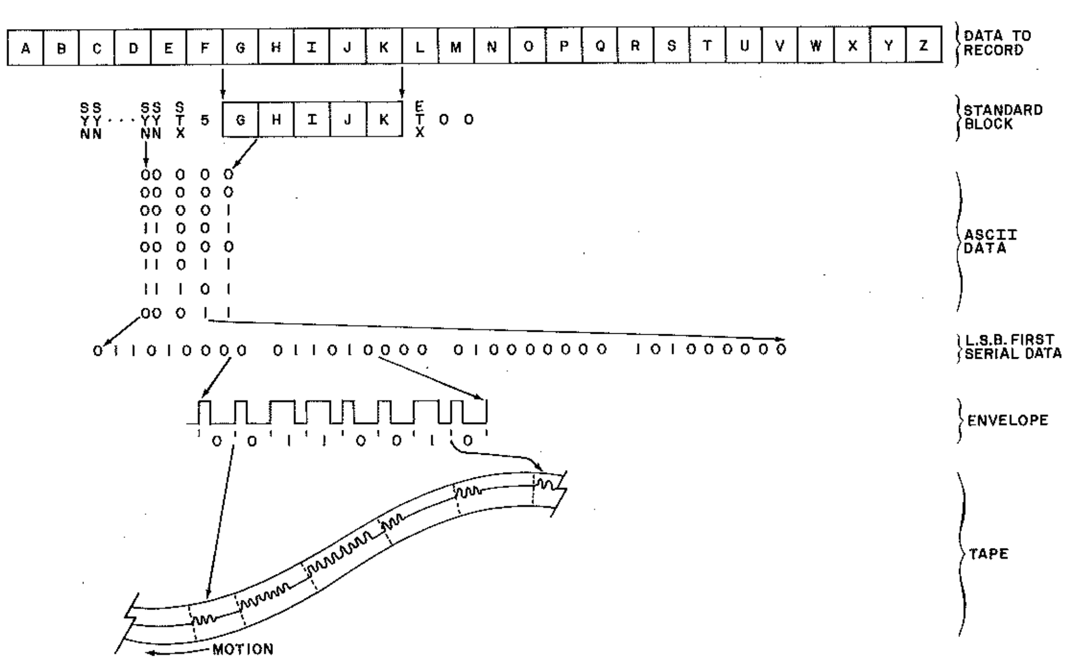

Fig. 4. How data to be recorded is first translated to ASCII code and then put on tape as tone bursts.

Data Format. Having a standard medium and a standard recording form is not enough for successful computer data exchange. We must all agree on the code and format of the data. As far as possible, the method described here uses national and international data communications standards. All data is written in ASCII code unless otherwise agreed upon by writer and reader. It is possible, for example, to agree on the transmission of actual eight-bit object code. All data is recorded with the least-significant bit first.

The record format we use is shown in Fig. 4. This technique is synchronous, and from the beginning of the data to the end, there should be no dead spots. At this time, it should be pointed out that cassette recorders have agc or limiter circuits. When the data first appears at the record input of such a machine, the agc does cruel things to the waveshape. By not allowing this to happen, except in the first part of the data where it is permissable, many problems can be avoided. This is done as follows: Each data block begins with at least 32 ASCII SYN (synchronizing codes 0001 0110). The SYN codes repeat long enough to allow the recorder's agc to settle down and the software to go into character "sync." A special character signal at the start of text (ASCII STX code 0000 0010) appears next, followed by an eight-bit count word. That count specifies how many more characters appear in the date record. If the count is zero, then this is called an end-of-file block. If it is not zero, it specifies how many eight-bit bytes appear in the data record. At the end of the data bytes (if any), is an ASCII ETX (end-of-text 0000 0011) character and two block-check characters. These two characters are normally zero, but can be used to hold the CRC code, or a check-sum, or whatever error protection the writer wishes to employ here.

If the block-check characters are used, the writer of the original tape is expected to provide a computer program in the first few data blocks for the machine of interest that will read and utilize them. This program should appear at the front of the tape and be terminated by an end-of-file block. The data to be read in should then follow on the tape. This front-end program is called a "bootstrap leader."

Programs for reading and writing standard format data tapes from memory of an 8080 are shown in Program 3. We can read or write 1024 bits in about 30 seconds using the standard format.

Higher Speed. This cassette interface can also be used locally for normal input/output needs. However, in your own computer, you may be able to go substantially faster. Our experiments have shown that you can expect to have a data bit rate about one-fourth of the modulation frequency. If your tape recorder will faithfully reproduce a 10-kHz signal, as many better decks do, you can expect to handle 2500 bits (240 characters) per second.

You may also want to add some additional hardware to eliminate some of the software. A simple gated oscillator can be used instead of performing the modulation in software. The envelope signal can drive the gate of an oscillator. You can even go so far as to have an eight-bit parallel output bit port and perform all of the timing and serialization external to the computer. You will probably want to have two versions of these circuits: one to be used to write standard tapes at standard frequencies and rates, and the other to write at whatever speed your own tape recorder can handle without errors.

The playback circuitry can also be expanded. The count-up/count-down software can be converted into a couple of timers that control ramps. Similarly, you may want to assemble incoming bits into eight-bit characters in hardware. With all this hardware installed (it takes about 10 IC's), the software becomes only a few input and output statements.

What is needed now is a central exchange point. Perhaps some of the emerging hobbyist groups (or even individuals) will agree to create a library of tapes for exchange or have them available at a nominal charge. A brief listing of program function,

PROGRAM 1

```
;WRITE THE BYTE IN THE -C- REGISTER OUT TO TAPE,
;   LEAST-SIGNIFICANT BIT FIRST.  AFTER EIGHTH BIT
;   WRITE OUT A DATA '0'.  REGISTERS (A & B) ARE
;   DESTROYED.  OCCUPIES 74 BYTES.
;
;   VARIABLES -WRWAI- AND -WRLEN- CONTROL DATA RATE.
;   -WRWAI- DEFINES PERIOD OF EACH MODULATION HALF-
;   CYCLE; -WRLEN- DEFINES LENGTH OF EACH DATA BIT.
;   DATA RATE IN BPS IS:
;                          1000000
;         BPS = -----------------------------------
;               (15 WRWAI + 64)  (6 WRLEN - 1) C
;         WHERE -C- IS 8080 CYCLE TIME IN MICROSECONDS
WRWAI   EQU   29        ;2004 HZ IF C=500 NANOSECONDS
WRLEN   EQU   2         ;REDUNDANCY = 2
;       GET NEXT DATA BIT TO TRANSMIT
WRCHA:  MOV   A,C
        STC             ;JAM IN STOPPER BIT
WRCHX:  RAR             ;GET LEAST-SIGNIFICANT BIT
        MOV   C,A       ;SAVE ALL OTHER BITS FOR LATER
        LDA   WRLNG     ; (FOR DATA '1')
        JC    WRBST
        LDA   WRSHT     ; (FOR DATA '0')
;       WRITE TONE BURST OUT TO TAPE
WRBST:  CALL  WRTIM     ;WRITE OUT FIRST HALF-CYCLE
        JZ    WRFIN     ;  (GO DO LONG PART NOW)
        CALL  WRTIM     ;WRITE OUT SECOND HALF-CYCLE
        JNZ   WRBST     ;  (KEEP GOING)
        OUT   TAPEO     ;TERMINATE MODULATION
        LDA   WRSHT     ;  (GO DO SHORT PART NOW)
        JMP   WRDLY-1
WRFIN:  OUT   TAPEO     ;TERMINATE MODULATION
        LDA   WRLNG
;       WRITE OUT NO MODULATION FOR REST OF BIT TIME
        MOV   B,A
WRDLY:  CALL  WRTIM+3   ;JUST DELAY
        JMP   $+3       ;  (WASTE MORE TIME)
        MOV   A,A
        JNZ   WRDLY
;       PREPARE NEXT BIT FOR OUTPUT
        MOV   A,C
        ORA   A
        RZ              ;IF ZERO, CHARACTER'S ALL DONE
        CPI   1         ;IF 1, WE'VE FOUND STOPPER BIT
        JNZ   WRCHX     ;(JUST ANOTHER DATA BIT)
        XRA   A         ;EMIT A TERMINAL '0'
        JMP   WRCHX
;       TIMING DELAY LOOP FOR CONTROLLING MODULATION
WRTIM:  MOV   A,B       ;GET COUNTER WORD
        OUT   TAPEO     ;WRITE OUT CARRIER
        MVI   A,WRWAI   ;SET UP WAIT
        DCR   A
        JNZ   $-1
        INR   B         ;UPDATE COUNTER
        RET
; SHORT- AND LONG-BURST CONSTANTS
WRSHT:  DB    255-WRLEN-WRLEN+2 ;(MUST BE ODD)
WRLNG:  DB    255-WRLEN-WRLEN-WRLEN-WRLEN+1 ;(EVEN)
```

PROGRAM 2

```
;READ A BYTE FROM TAPE INTO THE -C- REGISTER.
;   -C- IS LOADED LEAST-SIGNIFICANT BIT FIRST
;   INTO THE MOST-SIGNIFICANT POSITION.  122 BYTES
;
;   SAMPLE PERIOD FOR INCOMING DATA IS SET BY -RDTIM-,
;   WHICH IS COMPUTED AS:
;                T - 100C       (T IS TIME IN USEC,
;       RDTIM = ---------        C IS 8080 CYCLE TIME
;                  15C              IN USEC)
RDTIM   EQU   7         ; TO SAMPLE EACH 100 USEC
;       SET UP NORMAL WORD-SIZE STOPPER
RDCHA:  MVI   C,128
        IN    TAPEI     ;AWAIT DATA '0' CONDITION
        RRC             ;  BEFORE LOOKING FOR
        JC    RDCHA+2 ;  LEADING EDGE
        MVI   A,RDTIM+2
        CALL  RDBIT     ;WAIT FOR SAMPLE PERIOD,
        JC    RDCHA+2 ;  THEN CONFIRM '0'
;       FIND AND CONFIRM LEADING EDGE OF DATA BURST
RDCHC:  IN    TAPEI     ;LOOK FOR LEADING EDGE
        RRC
        JNC   RDCHC
        MVI   B,1       ;INITIALIZE RAMP COUNT
        MVI   A,RDTIM+2
        CALL  RDBIT     ;GO CONFIRM LEADING EDGE
        JNC   RDCHC
        MVI   A,1       ;CONFIRMED.  START COUNTING
;       RAMP UP (-B- REGISTER) UNTIL TRAILING EDGE
RDCH3:  ADD   B         ;INCREMENT RAMP COUNT
        MOV   B,A       ;SAVE RAMP COUNT
        JC    RDCHE     ;  (BAD DATA; PULSE TOO LONG)
        MVI   A,RDTIM+1
RDCHR:  CALL  RDBIT     ;GO READ NEXT SAMPLE
        MVI   A,1
        JC    RDCH3     ;IF SAMPLE = '1', CONTINUE COUNT
        CONFIRM TRAILING EDGE
        NOP
        MVI   A,RDTIM+2
        CALL  RDBIT     ;CONFIRM
        MVI   A,2
        JC    RDCH3     ;EARLIER '0' WAS NOISE
        MVI   A,-2      ;BEGIN TO COUNT DOWN
;       COUNT DOWN AFTER TRAILING EDGE
RDCH5:  ADD   B         ;DECREMENT RAMP COUNT
        MOV   B,A
        JNC   RDCH0     ;DATA BURST WAS SHORT.  DATA='0'
        MVI   A,RDTIM+1
        CALL  RDBIT     ;READ NEXT SAMPLE
        MVI   A,-1
        JNC   RDCH5     ;STILL '0', CONTINUE COUNT DOWN
;       CONFIRM CLOCK (NEXT LEADING EDGE)
        NOP
        MVI   A,RDTIM+2
        CALL  RDBIT     :GET SAMPLE TO CONFIRM
        MVI   A,-2
        JNC   RDCH5     ;EARLIER '1' WAS NOISE
;       FOUND NEW LEADING EDGE; DATUM = '1'
        MOV   A,C
        RAR             ;INSERT '1' INTO BYTE
        MOV   C,A
        RC              ;IF STOPPER BIT IN -CY-, QUIT
        MVI   B,2
        MVI   A,RDTIM
        JMP   RDCHR     ;GO CATCH THIRD SAMPLE
;       COUNTED DOWN TO ZERO; DATUM = '0'
RDCH0:  MOV   A,C       ;INSERT '0' INTO BYTE
        RAR
        MOV   C,A
        JNC   RDCHC     ;GO WAIT FOR LEADING EDGE
        IN    TAPEI     ;AT END OF BYTE, BE SURE
        RRC             ; TO AWAIT LEADING EDGE
        JNC   $-3       ; OF TERMINAL '0' BIT
        RET
;       TIMING DELAY FOR READ SAMPLE PERIOD
RDBIT:  DCR   A         ;DELAY
        JNZ   RDBIT
        IN    TAPEI
        RRC             ;PUT SAMPLE INTO CARRY BIT
        RET
;       ERROR ROUTINE.  CLEAR CARRY TO REPORT ERROR
RDCHE:  XRA   A
        RET
```

PROGRAM 3

```
;READ A BLOCK OF DATA FROM TAPE INTO LOCATIONS
;   NAMED IN (H,L).  REGISTER -E- WILL BE SET
;   TO THE INPUT BLOCK SIZE; (A,B,C,D) ARE ALL
;   USED.  OCCUPIES 60 BYTES.
;   UPON RETURN, FLAGS REPORT CONDITIONS FOUND:
;    ZERO   CARRY   CONDITION
;    ----   -----   ---------
;      1      1     NORMAL DATA BLOCK
;      1      0     END-OF-FILE BLOCK
;      0      1     BAD BLOCK FORMAT READ
XXSTX   EQU   2         ;ASCII START-OF-TEXT (STX)
XXETX   EQU   3         ;ASCII END-OF-TEXT (ETX)
XXSYN   EQU   20        ;ASCII SYNC CODE (SYN)
;       SET WORD-SIZE STOPPER BIT IN -C-
RDBLK:  MVI   C,128
        CALL  RDCHC     ;AT OUTSET, READ ANYTHING
        MOV   A,C
        CPI   XXSYN     ;SEE IF SYN FOUND YET
        JZ    RDSYN
;       GET ONE MORE BIT TO SEE IF SYNC CODE YET
RDNXT:  ORI   1         ;SET TO READ ONLY ONE BIT
        MOV   C,A
        JMP   RDBLK+2
;       CONFIRM THE SYNC CODE FOUND
RDSYN:  CALL  RDCHA     ;READ A SECOND SYNC CODE
        MOV   A,C
        CPI   XXSYN
        JNZ   RDNXT
;       FIND THE STX AND COUNT WORDS
        CALL  RDCHA
        MOV   A,C
        CPI   XXSTX
        JNZ   RDNXT     ;LOST SYNC. TRY AGAIN
        CALL  RDCHA     ;READ IN BLOCK SIZE
        MOV   A,C
        MOV   D,C       ;SAVE FOR OUR COUNTING
        MOV   E,C       ;SAVE FOR THE CALLER
        ORA   A
        RZ              ;IF ZERO, RETURN END-FILE.
;       READ IN DATA BYTES AND STORE THEM AWAY
RDATA:  CALL  RDCHA     ;READ NEXT DATA BYTE
        MOV   M,C       ;  AND PUT INTO STORAGE
        INX   H         ;ADDRESS NEXT BYTE
        DCR   D         ;SEE IF WE'RE DONE YET
        JNZ   RDATA     ; (NO)
        READ AND PROCESS BLOCK EPILOG
        CALL  RDCHA     ;READ IN ETX CODE
        MOV   A,C
        SUI   XXETX     ;SET ERROR FLAG
        STC             ;MARK NOT-EOF
        RET
```


PROGRAM 3 (Continued)

```
;WRITE A BLOCK OF DATA TO TAPE FROM THE ARRAY
;   STARTING AT ADDRESS IN (H,L).  DATA IS
;   ASSUMED TO BE IN ASCII AND THE NUMBER OF
;   CHARACTER TO WRITE IS IN -E-.  IF -E- = 0,
;   WRITE A NULL BLOCK AS END-OF-FILE.
;   (A,B,C,D,E) ARE USED. (H,L) WILL END UP
;   POINTING TO END OF ARRAY + 1.  USES 50 BYTES.
;
;   RECORD FORMAT:
;     SS          SSSN            EBB
;     YY...32...YYTN...DATA...TCC
;     NN          NNXN            XCC
XXBCC   EQU   0         ;DUMMY BLOCK-CHECK WORD
;       WRITE OUT SYNC CODES AT FRONT OF BLOCK
WRBLK:  MVI   D,32
        MVI   C,XXSYN
        CALL  WRCHA     ;WRITE OUT NEXT SYN CODE
        DCR   D
        JNZ   WRBLK+2
;       WRITE OUT STX AND THEN COUNT WORD (NNN)
        MVI   C,XXSTX
        CALL  WRCHA     ;WRITE OUT THE STX CODE
        MOV   C,E
        CALL  WRCHA     ;WRITE OUT COUNT
;       DETERMINE WHETHER DATA NEEDS TO BE WRITTEN
        MOV   A,E
        ORA   A         ;IF COUNT=0,
        JZ    WRBLF     ;  DON'T WRITE ANY DATA
;       WRITE OUT DATA BLOCK
WRBLL:  MOV   C,M       ;GET DATA BYTE
        INX   H
        CALL  WRCHA     ;WRITE BYTE OUT
        DCR   E
        JNZ   WRBLL     ;REPEAT UNTIL DONE
;       WRITE OUT BLOCK EPILOG
WRBLF:  MVI   C,XXETX
        CALL  WRCHA     ;WRITE OUT END-TEXT CODE
        MVI   C,XXBCC
        CALL  WRCHA     ;WRITE OUT BLOCK CHECK BYTES
        MVI   C,XXBCC
        CALL  WRCHA
        RET
```

machine, and source could be highly useful. For starters, reader David Yulke, 121 Liberty Ave., Selden, NY 11784, wants to trade software at no cost and offers PROM programming and assembling service at nominal cost to cover his time and postage. He has a home-designed 8008 sytem with cassette, CRT terminal, ASR-33 Teletype, and 1702A or 5203 programmer. Software includes MON-8 modified for UART operation and a RAM test feature; modified cassette routine, octal loader and hex loader (paper tape), all on 3 PROM's with an error routine. He is working on a "black-jack" program and a home accounting program. So let us hear from any other readers who wish to list such information.

Response. Thanks for the overwhelming response to our first column in June. We're gathering material on hobbyist computer clubs and will alert writers as soon as our input is complete. (POPULAR ELECTRONICS will be increasing the frequency of this column shortly as a result of so many reader requests to do so. —Ed.)

"He wants to call in a few other computers for consultation."