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 350 bits per second—not a blazingly 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

![Data waveform image](image-url)

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

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.”

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-
corded data signal from the tape recorder (Fig. 3E) and converts it block to the original magnetic tape. This circuit consists of IC7B, C, and D, and works with an input signal level from 0.75 V to 4 V, although 2.5 V is ideal. The input is amplified up to IC10 and IC11 (Figs. 3F and G) and then rectified by diodes D1 and D2. The combined output (H) is then applied to a Schmidt trigger (IC1D) which produces the output signal (I), an exact reproduction of the original envelope signal.

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 15 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, data tapes can be written using a standard format tape. Even the slower 8085 CPU can write out bits that have times 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 TAPED 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.

The least bit is always written as a "0." The time that it takes this bit to move past the reading head is the time that we can use to process the character and store it in memory.

The tape 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.

In reading the data back, 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 or the reading data bit is a 0, or the counter goes to zero (data is a 0). Note that each bit condition must be sensed two times to ensure that the output is clean.

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 is finished, the leading-edge of the ninth bit has thus always been sensed.

Data Format. Having a standard medium and a standard recording format is 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 a beginning point to add to or subtract from the data to the end, there should be no dead spots. At this time, it should be pointed out that when a recorder has gap or limiter circuits. When the data first appears at the record input of such a recorder, it may cause positive or negative things to the waveforms. By not allowing this to happen, except in the first position, the output waveform is permisable, many problems can be avoided.

This is done as follows: Each data block begins with at least 2 ASCII SYN (synchronizing codes 0001 0110). The four codes repeat long enough to allow the recorder's gap to settle down and the software to go into character "sync." A special character signal at the start of text (ASCII 020 code 0000 0100) appears next, followed by an eight-bit count word. That count specifies how many characters appear in the data 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-byte appears in the data record. At the end of the data bytes (0x7F), 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 data must also be expected to provide a computer program in the first few data blocks for the machine in which the data will be used to correct and utilize them. This program should appear before the front of the tape and be terminated by an end-of-file block. The data to be read in should then follow the tape. This front-end reading routine is called a "bootstrap loader."

Programs for reading and writing standard format data tapes are available in the memory of an 8080 as 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 non-inexpensive input/output devices. However, in your own computer system, it is important to consider the time that the interface takes in the transmission of a hardware calculator. For large data handling, data exchange, etc., a faster cassette interface will probably be necessary. However, many people will probably use two versions of these circuits: one to be used to write standard tapes at standard speeds and the other to write whatever speed your own tape recorder can handle without error.

The playback circuit can be expanded. The count-up/count-down software can be converted into two pairs of timers that control ramps. Similarly, you may want to assemble incoming signals in an eight-bit compatible manner in hardware. With all these hardware installed, it takes about 10 I/Os, the software becomes only a few input and output statements.

What is needed now is a central exchange point. Perhaps someone in the emerging hobbyist group (or even individuals) will agree to create a library of tapes for exchange or further work. This will be available at a nominal charge. A brief listing of program function.

**Note:** The text is in a format that is not easily readable, containing handwritten notes and diagrams. The content appears to be related to computer programming and data handling, possibly involving a magnetic tape system and ASCII code. The diagrams show waveforms, timing charts, and other technical details related to the programming of a computer system.
machine, and source could be highly useful.

For starters, David Yulke, 121 Liberty Ave., Stuaten, 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 8085 system with cassette, CRT terminal, ASR-33 Teletype, 1702A or 82030 programmer. Software includes MOS-8 modified for UART operation and a RAM test fixture, modified cassette routine, oscillator and hex converter (paper taped), all on 3 PROM's with an error routine. He is working on a "black jack" program and a home assembly 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 (July). 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 many reader requests to do so. -- Ed.)