super-simple cassette interface

low-cost non-critical tape memory system takes advantage of your RS-232C terminal port

By:
Don Kinzer
Technical Systems Consultants
P.O. Box 2574
West Lafayette, IN 47906

Anyone who has spent hours loading a program into a computer from the keyboard or front panel realizes the need for permanent or *non-volatile storage*.

The solutions to the storage problem are many: core memory, magnetic tape, magnetic disks and drums, paper tape and cards, and newly developed techniques using magnetic bubbles, lasers and even acoustic waves. Unfortunately for the hobbyist, all of these means of storage are extremely expensive except one form of magnetic storage — the audio cassette. Depending on the data format and recording method the audio cassette data recording and recovery system circuitry can be built for less than $15 and a few hours work. Keep in mind that this will *not* be the most efficient system in terms of transfer rate or bit density but the hardware and software necessary to realize the system are extremely simple.

The system about to be described can be likened to an ASR TTY paper tape storage system. If your system has software for low speed paper tape loading and punching then you need no more! The cassette system will appear to the computer to be the paper tape reader (or keyboard) when playing back and will appear to be the paper tape punch (or a display) during recording.

This is accomplished by attaching the cassette system to the same I/O line that operates the terminal device. This will be easy to accomplish if the interface to the terminal is RS-232C compatible. With a current loop interface, it is slightly more difficult but can be done.

compatible recording standard

The recording method is compatible with the so-called "Kansas City" standard, first to keep it simple and secondly to promote standardization. Under this standard, when operating at 300 baud, a mark or logic '1' will be recorded as 8 cycles of 2400 Hz while a space or logic '0' will be recorded as 4 cycles of 1200 Hz. The data will be stored as frequency encoded ASCII code. Other provisions of the standard need not concern us at this point so they will be ignored for the time being.

recording circuitry

Before describing the operation of the circuit, we'll assume that we have TTL-level data and baud rate clock signals available. Suggestions for obtaining these signals will be given later. The circuitry for recording is shown in **fig. 1**. It supplies two signals at the output, one at about 500 mV p-p for recorders having an auxiliary input and one at about 5 mV p-p for recorders having only a microphone input. Using an

auxiliary input is desirable because the larger signal allows a larger signal-to-noise ratio. The circuit operates as follows: the data is latched into the D flip-flop synchronously with the 4800 Hz clock signal. The latched data is used to control the J-K flip-flop pair to divide the clock signal by either 2 or 4 giving either 2400 Hz or 1200 Hz at the output of the second J-K flip-flop IC2b. This output is attenuated by 10 for the AUX output, by 1000 for the MIKE output and then low-pass filtered by capacitor C1. The signal is then capacitively coupled into the recorder through C2.

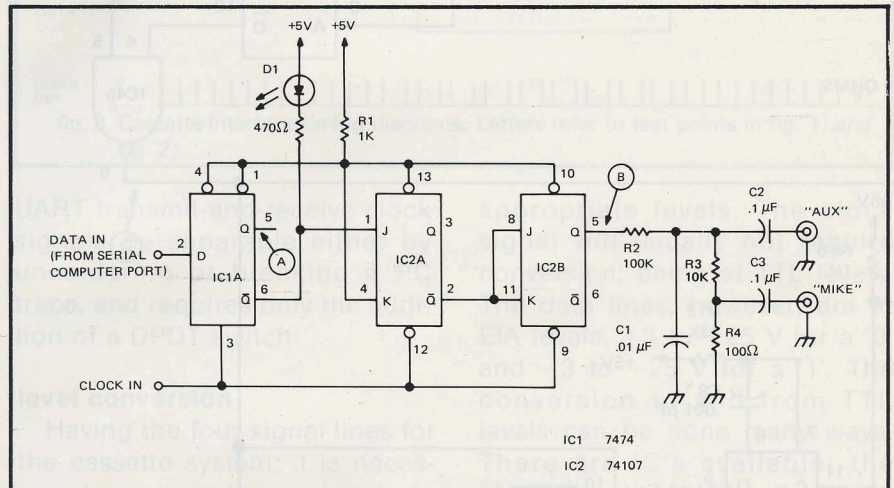

fig. 1. Cassette interface recording circuitry. Test point waveforms appear in fig. 3.

recovery circuitry

The data recovery circuitry is slightly more complicated, shown here in **fig. 2**. When studying the operation of this circuit, it will be helpful to refer to the timing diagram shown in **fig. 3**. The input signal to the recovery circuit is taken from the earphone or monitor output of the tape recorder. This signal is capacitively coupled into the op amp IC3 whose inputs are clamped by diodes D2 and D3. The signal will be large enough to drive the op amp into saturation in either direction, the result being a square wave at the output of the op amp. This signal is then clipped and translated to TTL levels by the transistor switch Q1 and then buffered by NAND gate IC4a. Next, the signal is fed to two one-shots, IC5a and IC5b, each set for a pulse width of about 25 usec. One one-shot fires on the rising edge of the signal while the other fires on the falling edge. The outputs of the one-shots are logically ORed producing a waveform at NAND gate IC4b which is twice the frequency of the recorder output, that is, 4800 Hz for a '1' and 2400 Hz for a '0'. This signal is fed to two places. First, it is applied to the one-shot IC6a which is set for a pulse width of ⅔ of the period of a '0' signal, or ⅔ of 416 usec, being about 277 usec. With this time constant the one-shot will be continually triggered (every 208 usec) when a '1' signal is being received and the output at IC6a will never drop low. However, when a '0' is being received the one-shot will time out and the output will drop low before the next trigger pulse is received. These events are illustrated in **fig. 3**.

The signal at NAND gate IC4b is also fed to the clock input of the D flip-flop IC1b in order to latch up the recovered data which is the output of the one-shot IC6a delayed by the low-pass filter R13-C9 to reduce noise problems. **Fig. 3** again illustrates this timing sequence.

We now have the data recovered but it is also desirable to derive a clock signal from the recorded signal to reduce the chance of errors due to variations in tape speed between record and playback. Recall that a 4800 Hz signal was generated at NAND gate IC4b when a '1' was being received but a 2400 Hz signal was generated when a '0' was received. Since the output of the one-shot IC6a only drops when a '0' is being received this signal is used to trigger another one-shot, IC6b, whose output is logically ORed at IC4d with the signal from NAND gate IC4b to supply a 4800 Hz signal at CLOCK OUT for both a '0' and '1' being received. The clock signal is slightly asymmetrical when a '0' is being received but is still periodic so the UART, to which this signal is being fed, doesn't mind at all. A handy feature has been added to enable the user to monitor the data flow. A LED connected to each of the D flip-flops will indicate the logic state of the flip-flop and will flash when data is being processed, but will remain illuminated during 'mark' intervals on the line. Thus the user can tell when data flow has started and stopped.

Now that we can store and retrieve data we must connect the system to the computer. Notice that a UART has not appeared in the diagrams yet. This is because this simple system utilizes the UART or UART-like device already available on your computer's terminal I/O port. Assuming that the computer system in question has an RS-232C type interface, the arrangement in **fig. 4** will enable the cassette system to communicate with the computer. This system requires only that the

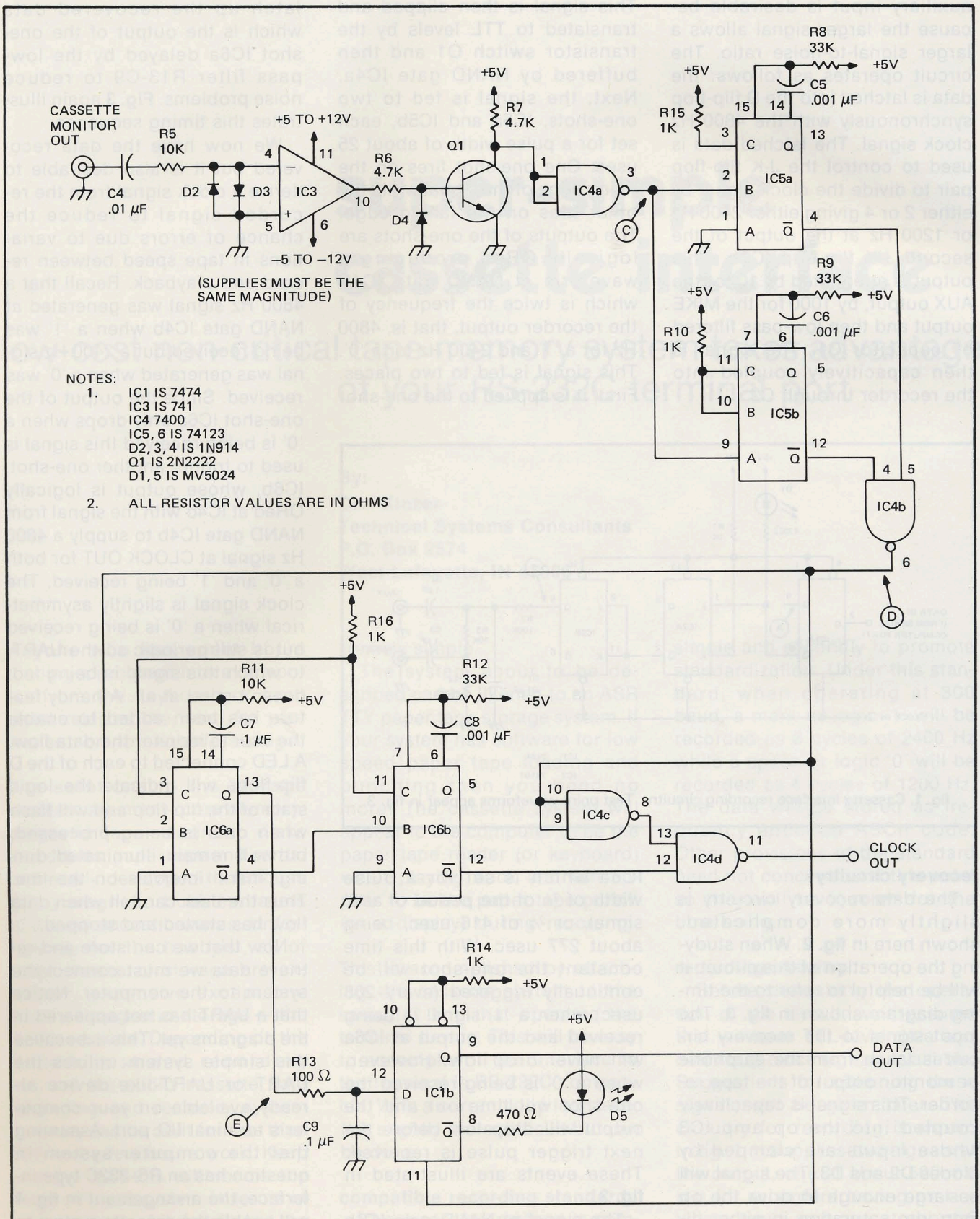

fig. 2. Cassette interface data recovery circuitry. Test point waveforms appear in fig. 3.

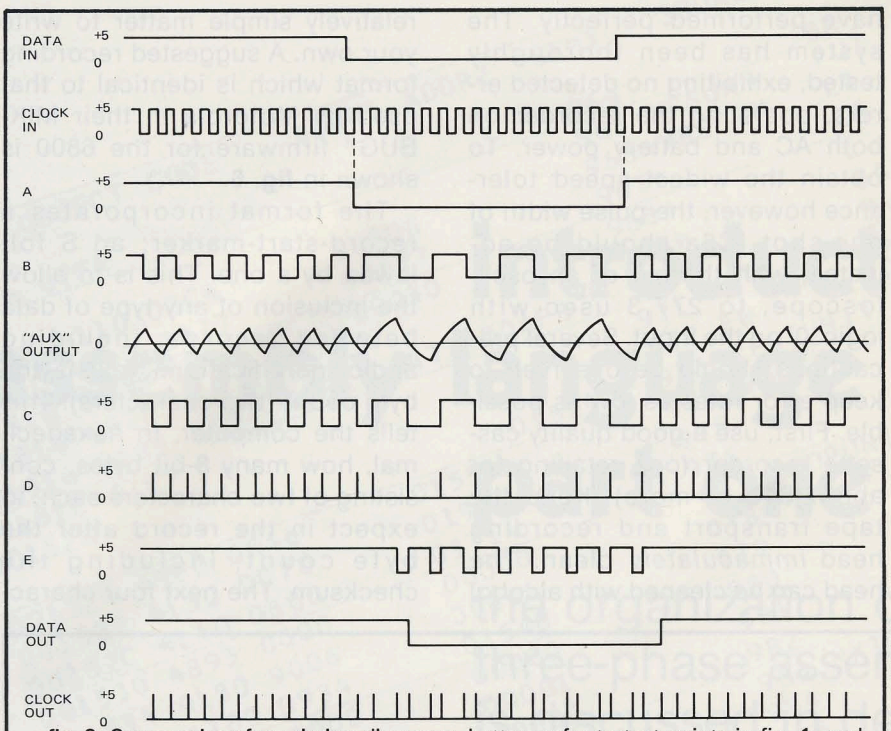

fig. 3. Cassette interface timing diagrams. Letters refer to test points in fig. 1. and fig. 2.

UART transmit-and-receive clock signals be separable either by unsoldering or breaking a PC trace, and requires only the addition of a DPDT switch.

level conversion

Having the four signal lines for the cassette system, it is necessary to convert these signals to appropriate levels. The clock signal will usually not require conversion, being at TTL levels. The data lines, however, are at EIA levels, +3 to +25 V for a '0' and −3 to −25 V for a '1'. The conversion to and from TTL levels can be done many ways. There are IC's available; the LM1488 and LM1489 or the 75150 and 75154, but they are quite expensive ($2 to $4 each). Alternate approaches are shown in **fig. 5** using commonly available parts. The transistors and diode are not critical. Any silicon switching or general purpose type transistor should work. The diode is of the silicon small signal variety.

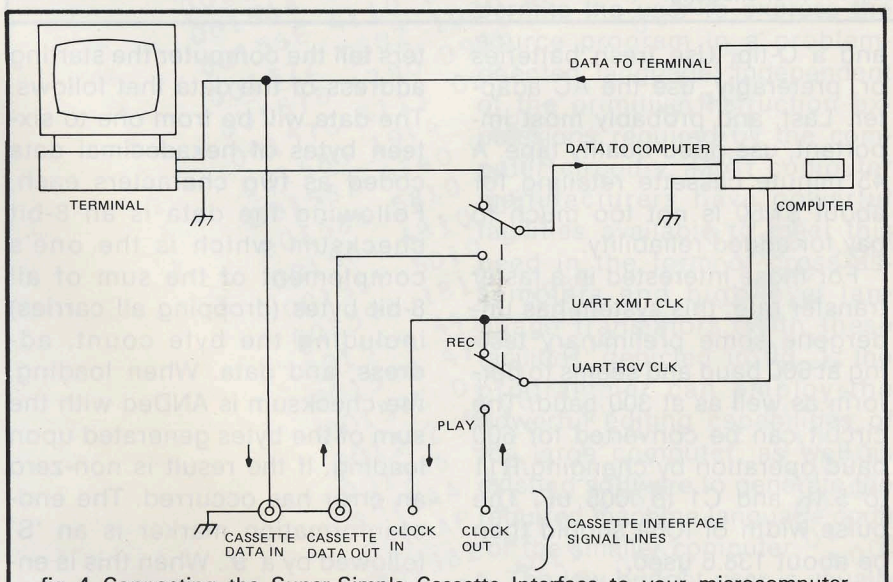

fig. 4. Connecting the Super-Simple Cassette Interface to your microcomputer. Mode select switch is shown in the RECORD position.

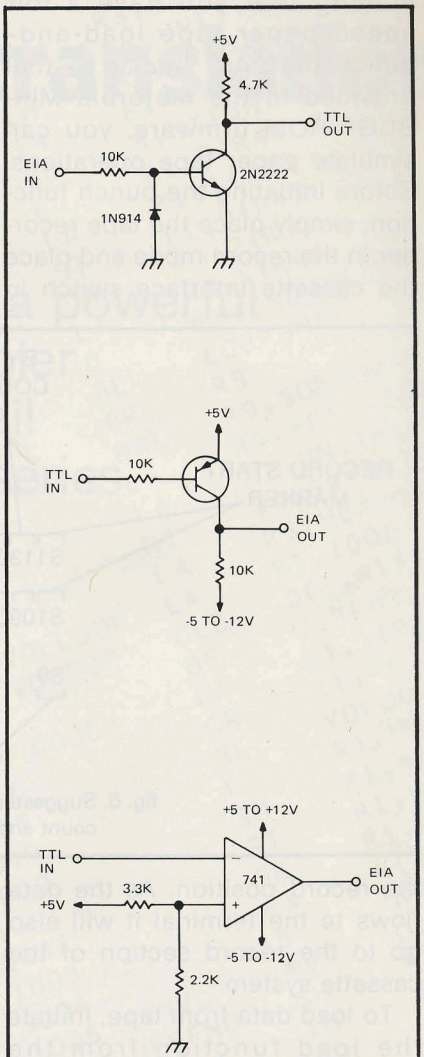

fig. 5. TTL to EIA and EIA to TTL level conversion circuits.

system check out

The system can be tested by connecting the 'AUX' output to the input of the recovery section, eliminating the recorder from the loop. Connect a 4800 Hz clock to the clock line and then alternately connect the 'data in' line to a logic '0' and a logic '1'. The

data out should reflect the state of the data in. If not, something has gone astray. If you have access to an oscilloscope, the 'clock out' line can also be checked and matched against the diagrams in **fig. 3**.

The use of the system is as simple as its construction. Assuming that you have a low speed paper tape load-and-punch program similar to that provided in the Motorola MIKBUG® ROM firmware, you can simulate paper tape operations. Before initiating the punch function, simply place the tape recorder in the record mode and place the cassette interface switch in the record position. As the data flows to the terminal it will also go to the record section of the cassette system.

To load data from tape, initiate the load function from the keyboard, place the cassette interface switch in the playback position and switch the recorder to the playback mode. Data flows to the computer as if the source were the paper tape reader.

reliability

The entire cassette system is relatively non-critical. Several different units have been built using wide tolerance components with absolutely no adjustments and have performed perfectly. The system has been thoroughly tested, exhibiting no detected errors, operating the recorder on both AC and battery power. To obtain the widest speed tolerance however, the pulse width of one-shot IC6a should be adjusted, with the aid of an oscilloscope, to 277.3 usec with logic '0' on the input. Several precautions should be observed to keep error rates as low as possible. First, use a good quality cassette recorder (one retailing for about $40 or more). Keep the tape transport and recording head *immaculately* clean. The head can be cleaned with alcohol and a Q-tip. Use fresh batteries or, preferably, use the AC adapter. Last, and probably most important, use good quality tape. A 45 minute cassette retailing for about $1.50 is not too much to pay for added reliability.

For those interested in a faster transfer rate, this system has undergone some preliminary testing at 600 baud and seems to perform as well as at 300 baud. The circuit can be converted for 600 baud operation by changing R11 to 5.1K and C1 to .005 uF. The pulse width of IC6a should then be about 138.6 usec.

If you don't have a paper tape load-and-punch program, it is a relatively simple matter to write your own. A suggested recording format which is identical to that used by Motorola in their MIKBUG® firmware for the 6800 is shown in **fig. 6**.

The format incorporates a record-start-marker; an S followed by a one. This is to allow the inclusion of any type of data between records, including audio identification. Next is the byte count (two characters). This tells the computer, in hexadecimal, how many 8-bit bytes, consisting of two characters each, to expect in the record after the byte count, including the checksum. The next four characters tell the computer the starting address of the data that follows. The data will be from one to sixteen bytes of hexadecimal data coded as two characters each. Following the data is an 8-bit checksum which is the one's complement of the sum of all 8-bit bytes (dropping all carries) including the byte count, address, and data. When loading, the checksum is ANDed with the sum of the bytes generated upon loading. If the result is non-zero an error has occurred. The end-of-information marker is an 'S' followed by a '9'. When this is encountered the loader should terminate loading. END

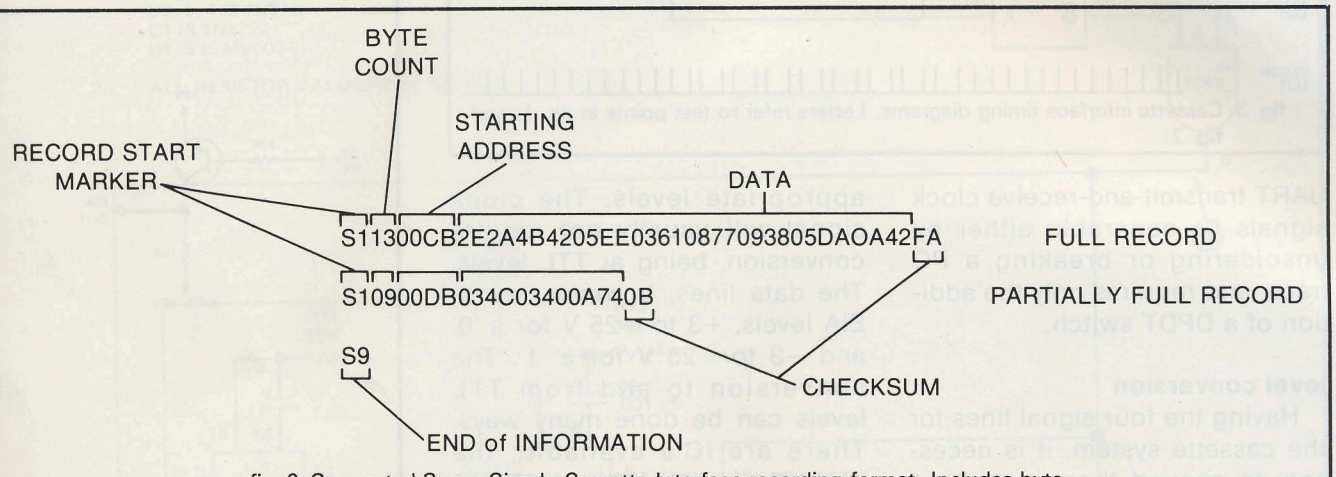

fig. 6. Suggested Super-Simple Cassette Interface recording format. Includes byte count and checksum error checking.