Build the BIT

Don Lancaster
Synergetics

The format of the BIT BOFFER is compatible with the provisional audio cassette standard described on page 72 of BYTE's February 1976 issue.

The BIT BOFFER is a low cost way to store and retrieve digital data onto and off an ordinary cassette tape, using stock audio tape recorders. You can also use it to exchange data, recording on one machine and playing back on a second. The BIT BOFFER is totally software independent, meaning that it can be used with or without a microprocessor system. All that's needed to control it is a simple "data ready" command to send something on to the recorder; you also get a "data available" command when the recorder produces an output. The BIT BOFFER can be connected to any serial data port of a microprocessor or other system.

The BIT BOFFER is highly speed tolerant, which lets you record on one machine and play back on a second. Better yet, it lets user groups or software sources provide multiple copies of programs and data simply and at very low cost.

Table 1: Advantages of the BIT BOFFER system.

***SPEED TOLERANT**	Up to ±25% speed variation without adjustment.
***SOFTWARE INDEPENDENT**	CPU or microprocessor not needed for operation. Runs with simple "send" and "ready" commands.
***IO COMPATIBLE**	The BIT BOFFER modem works directly with existing UART or ACIA. Coding compatible with TTY, 103 modems, TVTs and RS232C.
***NON CRITICAL**	One adjustment in system. No preamble or block limits. Adapts to wide range of baud rates.
***CHEAP AND SIMPLE**	Single sided compact PC card circuit costs less than $6 in quantity. Can be redesigned even lower.

The BIT BOFFER is very much compatible with standard serial interface using UARTs or microprocessor serial interface chips. It works with any serial interface that uses 16X clocks and has separate clocks for receive and transmit.

While the error rate of the BIT BOFFER system is more than adequate for hobbyist quality use, software techniques can optionally be added to the basic system to "harden" the error rate for fully professional, quality data storage and retrieval. The format of the BIT BOFFER is compatible with the provisional audio cassette standard described on page 72 of BYTE's February issue. Some characteristics of the BIT BOFFER system are summarized in table 1.

You can use the BIT BOFFER with almost any cassette recorder, although a medium quality unit with an auxiliary input and an automatic level control is recommended *for recording*. Just about any machine will play back a properly recorded tape. You do have to use a reasonable quality tape, free of dropouts and other defects. You also should "certify" the tape for dropouts, but this is only a commom sense precaution.

Cost of the system? Several vendors are now offering versions of this interface; the version we'll show you here uses six (optionally seven) stock CMOS integrated circuits with a typical cost of 50¢ each or so. The circuit fits a small single sided PC board and uses very little current from a single 5 volt supply.

BOFFER

How It Works

Most audio recorders are very fussy about what they are willing to accept as input signals. The cheaper the recorders, the fussier they get.

If the automatic level control of the recorder is going to help us rather than fight us, we want to record a constant amplitude signal of some sort.

What the recording head likes to work with is even more restrictive. For instance, figure 1 shows us the response of an *ideal* audio recording head. Over the useful recording range, the sensitivity doubles with each doubling of frequency. Go too low in frequency and the system noise level interferes with recording. Go too high and a gap resonance cancellation is caused when the physical length of the signal recorded on the tape exactly equals the gap width.

Doubling response as you double frequency is the same as mathematical differentiation. You respond to the *changes* or the *slope* of what you are trying to record, and not to the value of the signal. Note that this slope taking process happens twice, once while recording and once while playing back.

So, let's take some derivatives. The first derivative of a sine function is a cosine function, and the second derivative is a sine function, again of the same frequency. Put in a continuous sine wave and after the second derivative you get back a continuous sine wave, along with a phase reversal or two. But sine waves often require expensive hardware.

Can we use square waves? Put in a square wave and you record an impulse on the leading and trailing edges. Play this back and you get a *double* impulse, one pair at the leading edge and one at the trailing edge of the original square wave. There is no way to get a square wave back for a square wave in.

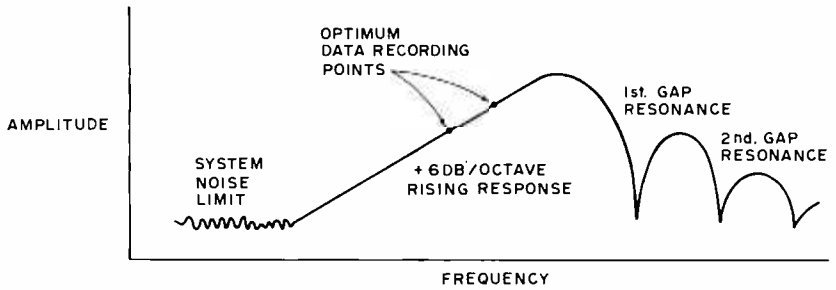


Figure 1: The response curve of an ideal audio recording head.

Now, the computer data recording people pull some tricks to simulate the recording of square waves. They use special heads and levels to saturate the tape in one direction or another. On playback, they sense the flux reversals with sense amplifiers and then use the amplified impulses to set and reset a flip flop, thus regenerating a square wave. Thus, saturating the tape eliminates one of the differentiations, and the set reset flip flop gives us an integration that cancels the second differentiation.

But this is vastly different from an audio cassette head working with linear, non-saturating tape signals. The audio cassette people attempt to compensate for the recording slopes, both on record and playback, to make it more or less flat over the audio spectrum. This compensation doesn't affect sine waves much, but with a square wave, it spreads out the harmonic energy and generally makes a mess of it. This is called the group delay distortion problem, and is caused by different response to the

The BIT BOFFER's data format is a "self clocking" one in which the receiving circuitry is locked to the actual data.

The absolute optimum signal we can record on an unmodified low cost audio cassette recorder is a constant amplitude zero offset sine wave whose frequency is changed only when the sine wave goes through zero.

harmonics of the square wave than to the fundamental frequency. Also, much of the compensation is done by the least expensive method and simply rings badly when driven by a square wave. While it is possible to record square waves on a quality machine, if you want reasonable performance for any machine, the *only* signal you should consider using is a sine wave.

Further, if we turn our sine wave on or off at the wrong time, we introduce a transient that's the same thing as the leading edge of a square wave. This causes considerable distortion for us. We can beat this by using a transient of zero amplitude, which means that the only time you can change the sine wave is as it goes through zero.

From all this (plus a mountain of lost time and painful testing), we can conclude that the absolute optimum signal we can record on an unmodified low cost audio cassette recorder is a *constant amplitude, zero offset sine wave whose frequency is changed only when the sine wave goes through zero*. Better yet, as a refinement anticipating the "coasting" effect of a recorder's group delay distortion, we should change frequency coherently but somewhat *before* each zero crossing.

So much for making the recorder happy. We also have to make our serial interface or UART happy. A serial interface using a single clock gets very unhappy if the transmitted and received speeds differ by more than a percent or two. On reception it starts outputting garbage since the bits at the end of the word get ahead of or behind the "ideal" positions. A high quality, line operated recorder can usually hold its speed variation to within a percent or two, but inexpensive battery operated units cannot. This speed variation is enhanced if we are recording on one machine and playing back on another.

There's a simple way out of this bind. Put a signal on the tape that not only tells you ones and zeros but also *how fast* the tape is going. Use the recovered speed information to speed up or slow down the UART receiver to match the data rate. The result is called a "self clocking" recording method.

Table 2: Tone standards for BIT BOFFER system.

110 baud:
1 = 16 half cycles of 880 Hz
0 = 8 half cycles of 440 Hz

300 baud: *
1 = 16 half cycles of 2400 Hz
0 = 8 half cycles of 1200 Hz

600 baud:
1 = 16 half cycles of 4800 Hz
0 = 8 half cycles of 2400 Hz

***This is the provisional standard resulting from the BYTE symposium.**

Table 2 shows us some standards for a set of signals that meet the specification of constant amplitude sine waves or *half sine* waves that are zero crossing switchable and still give us the ability to extract self clocking digital information off a single audio cassette track.

To record a one, we record 16 half sine waves of a frequency that is eight times the data rate. For a zero, we record eight half sine waves of a frequency that is four times the data rate. The half sine waves are switched just before their zero crossings and always are phased for a continuous waveform through zero. Timing is controlled by a reference clock at 64 times the data rate. These are often already available in minicomputer systems. The reference clock (64 times data rate) is divided by four to run the clock input of the UART's transmitter at 16 times the data rate. It is selectively divided as needed to synthesize sine waves for recording.

During playback, these signals are amplitude limited to minimize tape variations and interference from bias, hum, and other noise. One clock pulse is reconstructed from each zero crossing. The distance between zero crossings is also measured by a retriggerable monostable circuit. If the distance is too great to be a one, a zero output is provided on the data line, and a new clock pulse is thrown in for the UART receiver circuitry. With this design, you automatically get 16 clock pulses out for each one and 16 clock pulses for each zero. In the zero case, the clock pulse spacing varies somewhat as the BIT BOFFER tracks the tape speed. But this is what makes the scheme work, since the UART receiver doesn't care and will still operate properly.

Figure 2 shows the word and timing formats for the BIT BOFFER system. A typical system might use ASCII characters sent in the Universal Asynchronous Data format that consists of a start bit, the ASCII code starting with the least significant bit, a parity bit, and two stop bits. The width of each bit is set by the data rate.

If we're working with 8 bit binary data instead of ASCII characters, we send a start bit, eight bits of data beginning with the least significant bit, an optional parity bit, and then two stop bits.

Using the asynchronous format, words or characters can go on the tape with any spacing between them. For the most efficient use of tape storage, large blocks are recommended. For example, cycling once

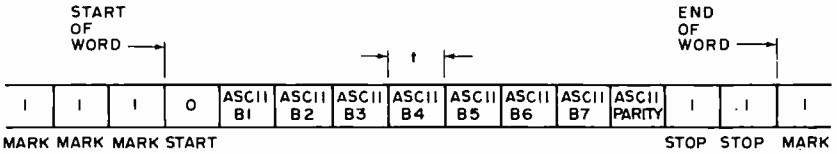
through a typical television typewriter system's memory while driving a UART output (or reading input) would provide 512 character blocks corresponding to the 32 x 16 character display. With a microcomputer system, block sizes can be chosen arbitrarily as a software convention. However, the overhead of starting and stopping the tape can be minimized with larger block sizes. The only preamble or ending needed on the character blocks is some "mark time" or a string of digital ones, similar to the initial and final rubouts on a paper tape system.

Figure 2c also shows the timing standards for recording and playback for the various data rates.

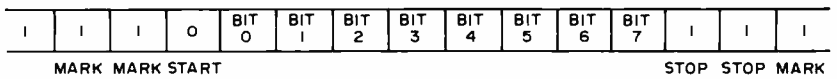
One problem you may have to allow for in your system is *overspeeding*. Overspeeding is caused by recording on a slow machine and playing back on a fast one. The BIT BOFFER can easily take care of this, but the system which is ultimately doing the receiving and using the data has to handle the faster rate. For instance, on a TVT with a single frame update, 60 characters per second is the upper limit on entry. With a Teletype, 10 characters per second is usually tops. To beat this problem when interfacing fixed speed systems, you simply hold the character rate down a *little when* recording, say to 75 or 80 percent of capacity. This is done by having the data source wait several tens of milliseconds between characters. If the BIT BOFFER happens to speed things up a little, the maximum rate is still under what your system can accept.

Note that the overspeeding problem only exists for systems which assume a fixed input data rate from the interface. In the typical microcomputer system, use of the interrupt structure or status bits from the interface can synchronize a program to the actual received data rate. Since the program is then synchronized to the received data bits (which are in turn synchronized to the actual tape speed), the combined microcomputer and BIT BOFFER interface becomes speed independent within the BIT BOFFER's range of ±25% variation from nominal.

A related problem that's also easy for you to fix is that some of your peripherals may need absolutely constant data rates. This is particularly true of a Teletype, and the playback rate of the BIT BOFFER may be out of tolerance. To get around this, you simply connect your UART's parallel receiver outputs back to its own parallel transmitter inputs and retransmit at the correct rate. This is particularly easy to do with ACIA type microprocessor interface devices with bi-directional parallel IO lines.



(A) Universal asynchronous data format.



(B) 8-bit binary data format.

BASE RATE	110 BAUD	300 BAUD	600 BAUD
Transmit time "t"	9.09 msec ±1%	3.33 msec ±1%	1.67 msec ±1%
Receive time (without adjustment)	9.09 msec ±20%	3.33 msec ±20%	1.67 msec ±20%
Word time	100 msec	36.67 msec	18.33 msec
Recommended max word rate (allows for overplay)	7.5 char/ second	22 char/ second	44 char/ second

(C) Timing standards.

Figure 2: Data format and time standards for several data rates. The BYTE provisional audio cassette standard utilizes the 300 baud data rate; communication with a Teletype requires a 110 baud data rate.

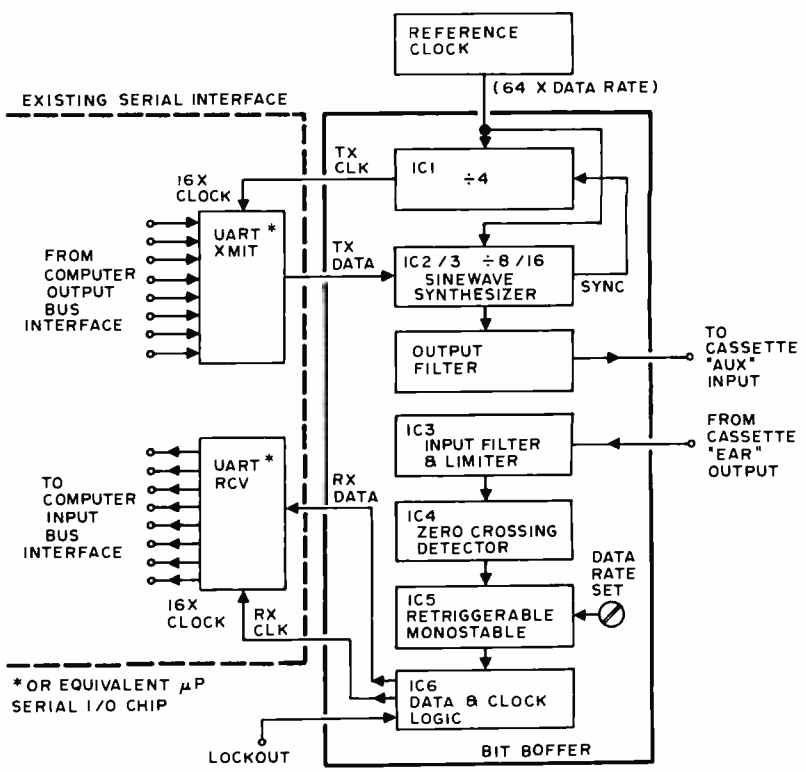
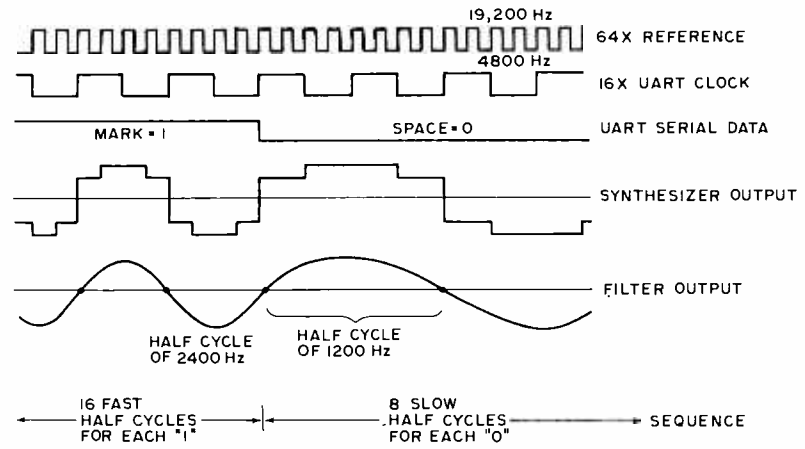
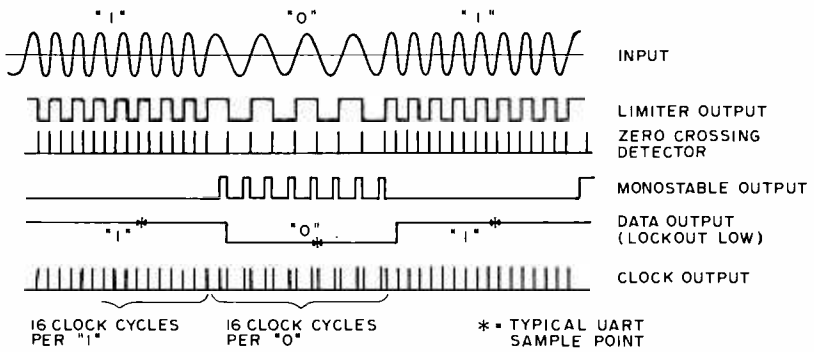


Figure 3: Block diagram of the BIT BOFFER. The integration of BIT BOFFER into a microcomputer system requires a UART or other existing serial IO interface. This is shown at the left in the block diagram. The connections to the recorder are shown at right. The BIT BOFFER proper is enclosed in the box in the middle.



(A) TRANSMITTER



(B) RECEIVER

Figure 4: Key timing diagrams of the interface. These diagrams show the relationship of several waveforms in the BIT BOFFER: (a) The transmitter section and (b) the receiver.

The complete tape cassette interface for a microcomputer consists of the BIT BOFFER hardware plus software to drive it. Turn to Jack Hemenway's article for an example of some tape control software for this interface.

An earlier version of the BIT BOFFER system was described in BYTE, September 1975. Note that this improved version eliminates any dependence on the clock duty cycle, works directly at higher baud rates, and has no need to worry about phase reversals inside the recorder. The 300 baud version of BIT BOFFER is compatible with the provisional BYTE cassette data interchange standard.

About the Circuit

The block diagram of the BIT BOFFER appears in figure 3 with key wave shapes shown in figure 4. Transmitter and receiver schematics follow in figures 5 and 6. The PC and component layouts are shown in figures 7 and 8.

Our transmitter (figure 5) starts with a 64X frequency reference, or 19,200 Hertz in the case of a 300 baud system. This is divided by four with two type D flip flops set up as binary dividers. The clock is divided by one or by two, as selected by the pair of NOR GATES (IC3a and b), using the TXDATA command out of the serial interface. This selectively divided clock is routed to an eight level sine wave synthesizer consisting of a four stage walking ring counter and a three resistor summing network. The synthesized sine wave at this point has no harmonics till the seventh and ninth, both of which are over 16 decibels down. The synthesizer is followed by a third order Bessell active lowpass filter. This provides a low impedance, one volt peak to peak sine wave at the recorder AUX input.

Note that the UART serial output is inherently synchronized to the UART clock input so that we always switch sine waves just before the leading zero. Feedback from this switching edge is also used to synchronize the divide by four via a pulsed reset generated in the C3-R2 network. This makes sure all signals are properly phased with respect to each other.

On playback (see figure 6), an input RC filter minimizes the hum and bias interference. It then routes the EAR output signal to a high gain limiter that outputs a 5 volt square wave whose zero crossings correspond to the zero crossings received from the recorder. The actual crossings are detected with the Exclusive OR circuit (IC4c and IC4d) that follows. This Exclusive OR circuit outputs one narrow positive going pulse per zero crossing. These pulses are used to continuously discharge a timing capacitor set up as a retriggerable monostable.

Between zero crossings, the capacitor is allowed to charge at a rate set by the BAUD ADJUST control. The CMOS inverter stages that follow (IC3c to IC4b, IC3d to IC4a) act as comparators. Circuit timing is adjusted so that the monostable produces an output at 3/4th the low frequency period, and nothing for the high frequency period. These output pulses are present only during digital zeros and are routed to an output flip flop that regenerates the one and zero data. A lockout on this flip flop, via the RESET input, prevents zeros from being output during leader time, and recorder start and stop intervals. This prevents the UART receiver from reacting to random garbage.

Meanwhile, the recovered clock pulses are added to the new clock pulses that you get from the leading edge of the monostable during zeros-as-data times. These clock and data outputs are routed to the UART receiver serial data (RXDATA) and clock inputs (RXCLK) for recovery.

The .01 μF timing capacitor is set up for 300 baud. This can be doubled for 150 baud or halved for 600 baud operation. Alternately, a larger value pot can be used, perhaps a multiple turn one. Sine wave filter

capacitors in the transmitter portion of the circuit must also be changed for different baud rates.

Checkout and Operating Hints

For your first time preliminary checkout and setup, it's handy to have a scope available; but once you're set up properly with a known good recorder, the circuit essentially takes care of itself with only one non-critical adjustment.

To make an initial test, connect up +5 and ground and connect the reference clock to a source of 19,200 Hertz. This can come from a 555 timer; or, if you're using an existing data rate generator, as a "300 baud, 64X clock" or a "1200 baud, 16X clock." Temporarily break the serial data input to the BIT BOFFER so you can connect it to +5 for a one, to ground for a zero, or to the UART serial output for data.

Set the data input to a one and check for a clean one volt peak to peak 2400 Hertz sine wave at test point A in figure 5. Now switch to a grounded (zero) input and check for an identical sine wave of half frequency.

Next, jumper the BIT BOFFER transmitter output to its own receiver input and check for a clean square wave at test point "C" (figure 6). Now, monitor test point "E". With the data input grounded, adjust the BAUD ADJUST control to get a waveform that is positive between 25 and 30 percent of the time. If you have no scope, use a voltmeter to set the voltage at test point D to a 1.2 to 1.6 volt range on a 5.0 volt supply. Now, switch to an input one, and verify that this voltage and its waveform goes to zero.

The next thing to do is to write a program for your computer which tries sending data through the BIT BOFFER-UART combination, displaying the transmitted data on a Teletype, a TVT, or other output device. Note that if you can't get the BIT BOFFER to talk to itself, there is no way in which it will work when you add a recorder to the middle of the loop. If you have no scope, use the optional but recommended tuning circuit shown in figure 9.

If the BIT BOFFER passes its self transmission test, select a recorder and connect it. If your recorder allows it, monitor the EAR output on a scope while recording. If the sine waves look wrong, try adjusting the input signal level going into the AUX input, or select a different recorder. If the signals going onto the tape look bad, what you get back will be even worse. The BIT BOFFER should properly "echo" the message during recording, provided your recorder has output to the EAR jack for record monitoring.

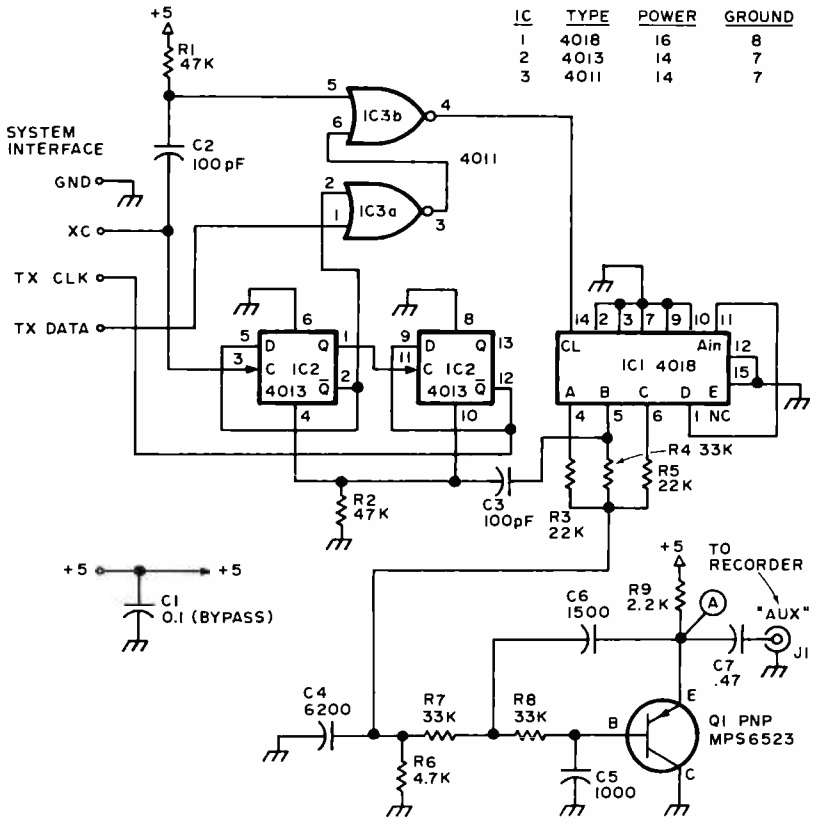


Figure 5: Schematic diagram of the BIT BOFFER transmitter section.

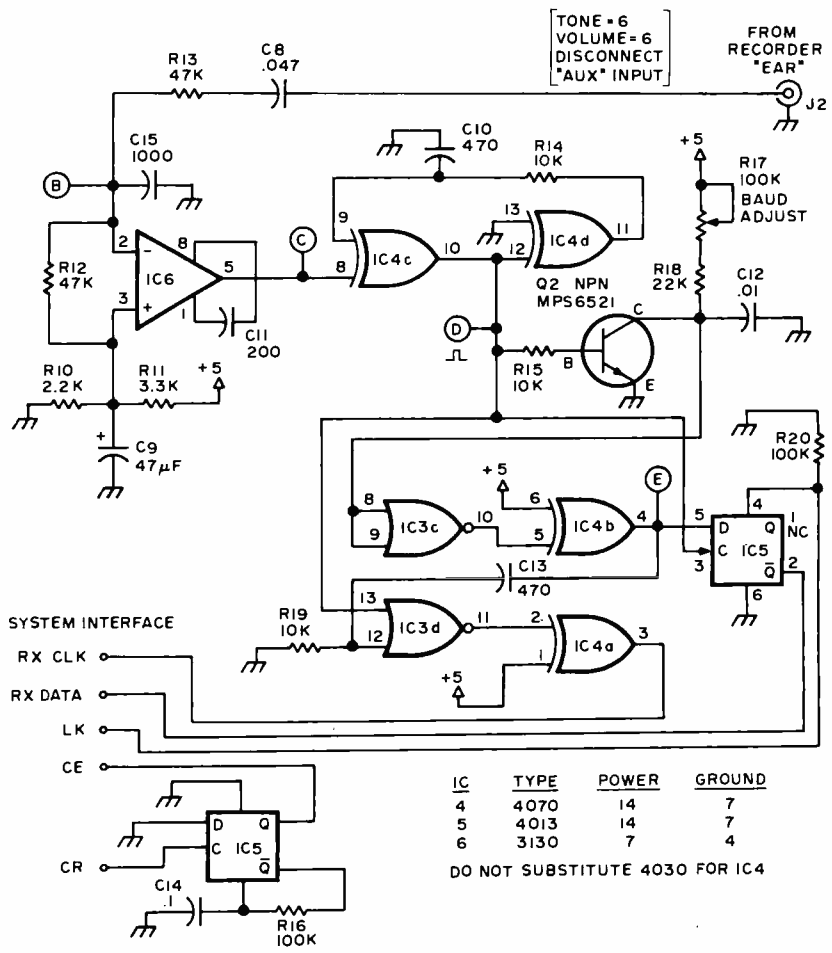


Figure 6: Schematic diagram of the BIT BOFFER receiver section.

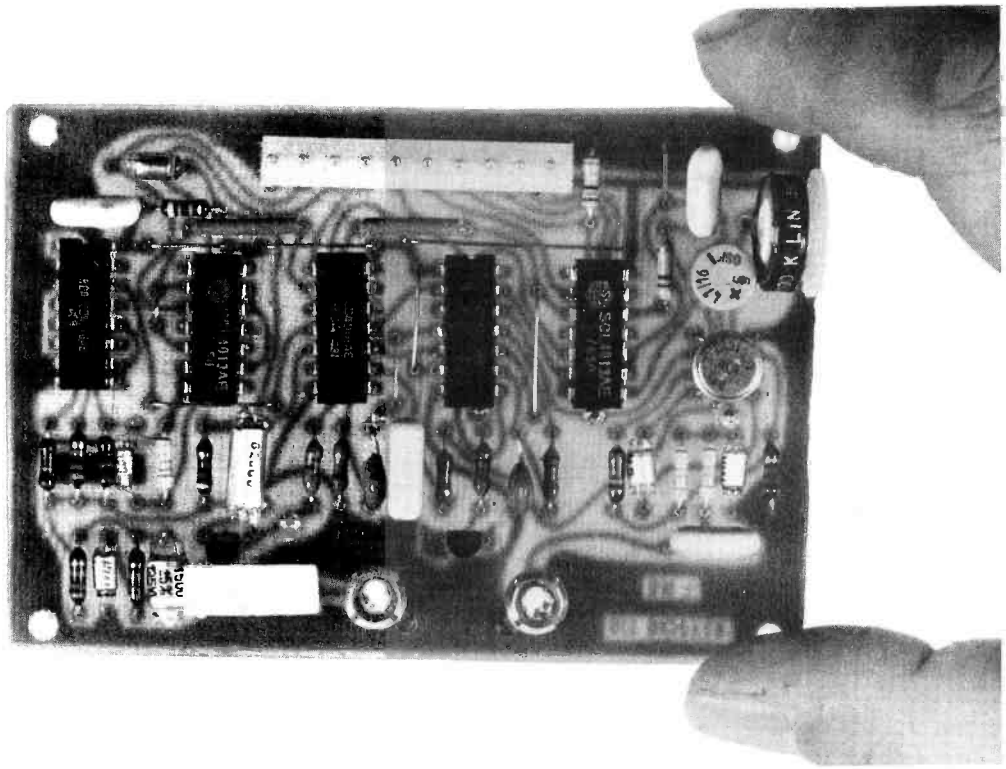


Photo 1: The assembled BIT BOFFER built from the board layout of figure 7. The parts can be identified using the placement diagram of figure 8.

Now we're ready to record some data. Use a high quality, audio cassette tape, and record a character sequence generated by a TVT or a microprocessor program. [See Jack Hemenway's article in this issue for an example. *Ed.*]

During playback, monitor test point "B". Some recorders get unhappy if you connect both the AUX input and the EAR output at the same time during playback, particularly on line operation. If yours does, *unplug the AUX input during playback*. Check for reasonable appearing sine waves at test point "B". Try various settings of the volume control and tone controls. Usually settings of "5" for volume and "7" for tone are optimum.

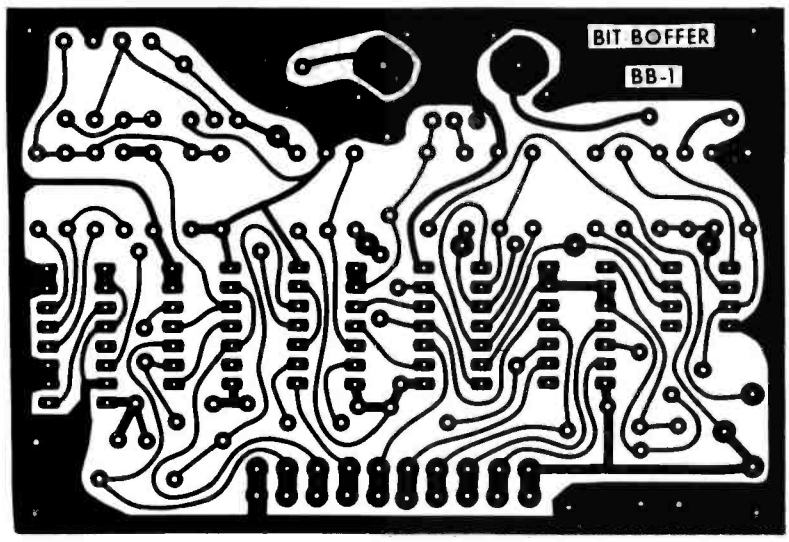


Figure 7: Printed circuit layout of the BIT BOFFER modulator. A board with this pattern is available from Southwest Technical Products Corporation, 219 W Rhapsody, San Antonio TX 78216. Also available is a kit of parts for the BIT BOFFER, a kit which contains a UART interface to mate with the SWTPC 6800 system, and a kit of parts for the tuning indicator.

Now, set up an oscilloscope and look at the *eye diagram* as shown in figure 10. You should get this same pattern at test point "C", with only a small amount of jitter. Set your BAUD ADJUST control to sample in the center of the right eye. If you can't get the jitter down far enough that this sampling is perfectly clean, stop and find out why. Use the optional tuning circuit shown in figure 9 if you have no oscilloscope.

If all this works, you should now be able to reliably record and playback data of your choice. Above all, get to know your recorder and know what its limits are in the way of input level, tone, jitter, and volume settings. Note particularly that any second harmonic distortion of the recorder must be held down to something reasonable or alternate zero crossings will jitter. *Your eye diagram tells all about the quality, jitter, and error rate of your system.* Learn to use it. (By the way, don't make the mistake of using a dual channel scope on alternate vertical inputs, as this can mask any second harmonic jitter. Use a single trace or else chop the display.) Note that you can get continuous data for debugging by connecting the receiver RDY output to the transmitter KP input. You

Table 3: Error rate hardening techniques for premium systems.

***CLOCK PLL**
A phaselock loop on clock output can fill in occasional missed clock pulses. Present system only allows six misses per word.

***DATA INTEGRATION**
Analog integration or speed-independent majority logic voting takes average of several sequential half-cycles. Present system depends only on UART sampled mid-bit word.

***ERROR FLAGS**
Parity, overrun, and framing error flags already in UART circuit can be used to stop loading if error occurs.

***FULL CR DECODE**
Some TVTs interpret any machine command as a carriage return. Limiting decoding to proper command minimizes severity of display errors.

***LOCKOUT**
Existing lockout circuit can be used to disable reception during tape start, leader, and stop times.

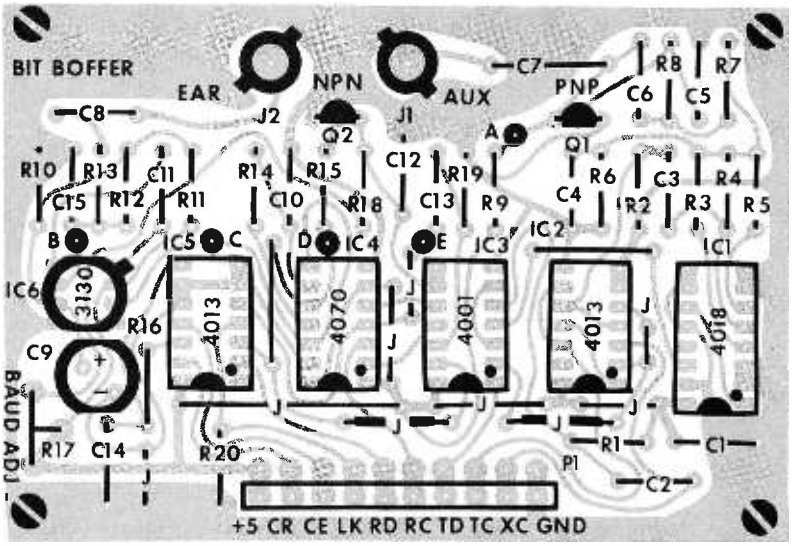
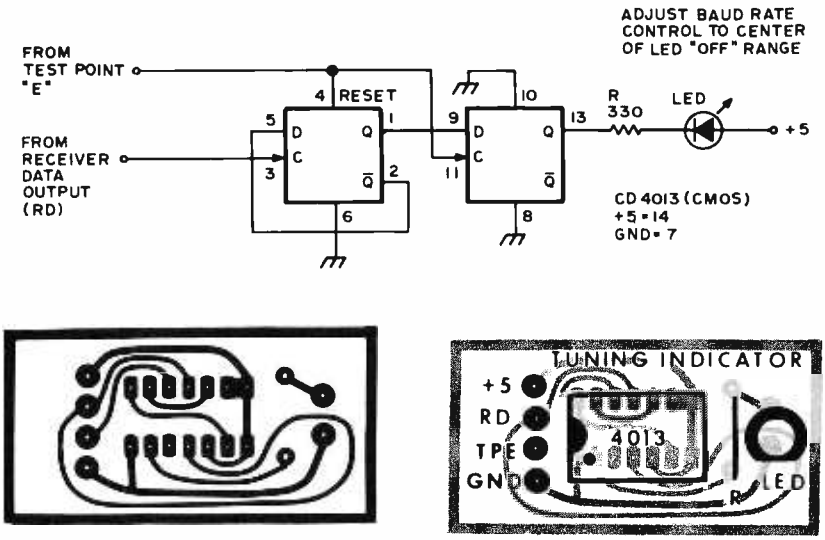


Figure 8: Parts placement overlay for the BIT BOFFER circuit board.

Figure 9: Tuning circuit board, overlay and schematic.



may have to *momentarily* short this line to ground to start the action.

Once you're confident of your BIT BOFFER, you're ready to exchange data. If you get an outside cassette, it should work without adjustment, if it was recorded properly. If you get errors, simply use your tuning indicator and rotate the BAUD ADJUST control to note the two limits where serious errors start, and then set the pot halfway between these limits. Better yet, use your scope and the eye diagram to properly set this control. In actual day to day use, this pot is quite uncritical. Don't forget to change capacitors or use wider range control if you make wide changes in the data rate.

Some Modifications

"Make it cheaper" and "make it better" are usually the name of the game in any new electronic system. What can we do to further improve the BIT BOFFER?

The error rate seems acceptable for most hobbyist interchange uses as is, but there are many things you can do to "harden" the error rate for a fully professional and reliable system.

Some hardening techniques are listed in table 3. Careful adjustment of the baud rate control, using the tuning indicator on a scope eye diagram, is a first and foremost line of defense against errors. You can add an output phaselock divider loop to reconstruct and more evenly space the UART clock pulses. This fills in for an occasional miss. The present system allows up to six or

A TUNING INDICATOR

If you don't have an oscilloscope for your initial testing or don't want to tie one up for eye diagram testing of incoming tapes, you can use this simple tuning indicator instead. When the light emitting diode is out, your data rate control is properly set. Adjust the data rate control to the CENTER of the off portion of the display. The light emitting diode should stay off while good data is being received.

If your data rate control is set slightly too high or too low, you'll get one too many or one too few clock pulses for each string of zeros. The tuning indicator works by checking for an odd or even number of monostable pulses during zero times. It then stores this odd-even information between zero strings. Very nicely, the tuning indicator will often indicate trouble BEFORE it causes errors, thanks to the UART being able to withstand a few extra or a few missing clock pulses per word without error.

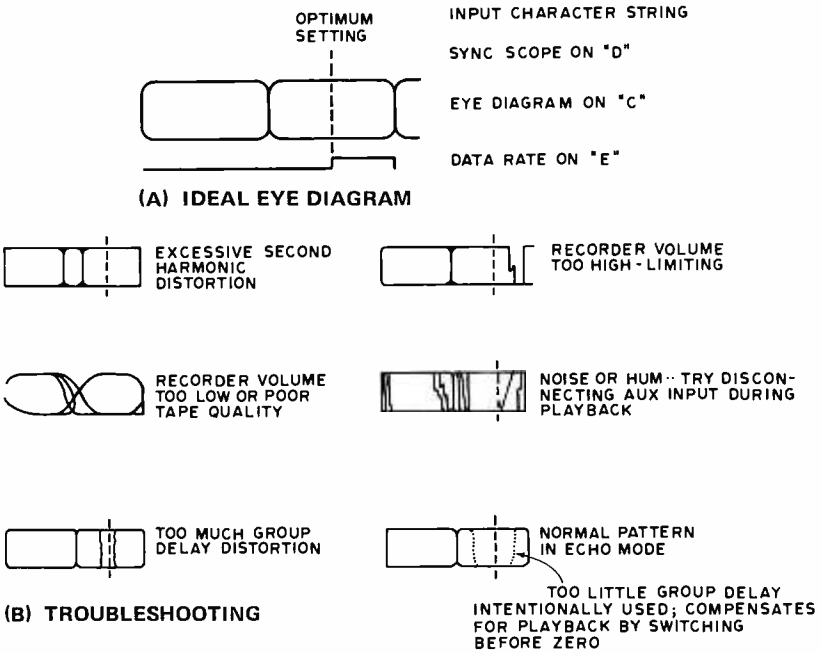


Figure 10: The "eye" diagram is useful for initial set up and testing of the receiver of the BIT BOFFER interface. This diagram can be displayed on a two-channel oscilloscope set up to trigger on test point "D" in the circuit. The scope should be set for a "chopped" display, and the sampling edge can be observed at test point "E" along with the data eye found on test point "C".

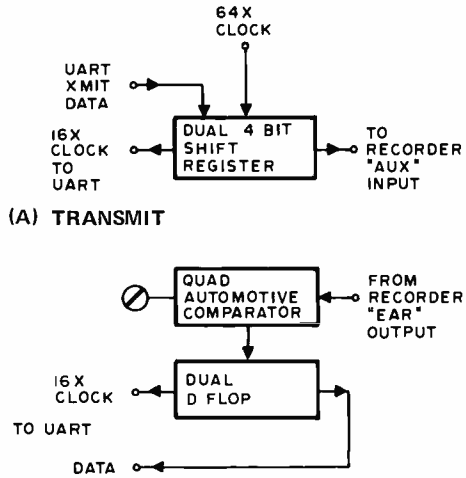


Figure 11: A possible future development of the system is a simplification of the circuit using only three integrated circuits.

so misses per word received. You can add data integration to "vote" on what constitutes a one or a zero. In the present system, only the middle bit is all the UART samples for valid data. If this bit happens to be wrong, you lose. Voting can be done with a RC network, or a shift register and a majority logic gate. The analog method is cheaper, but the shift register scheme is data rate independent.

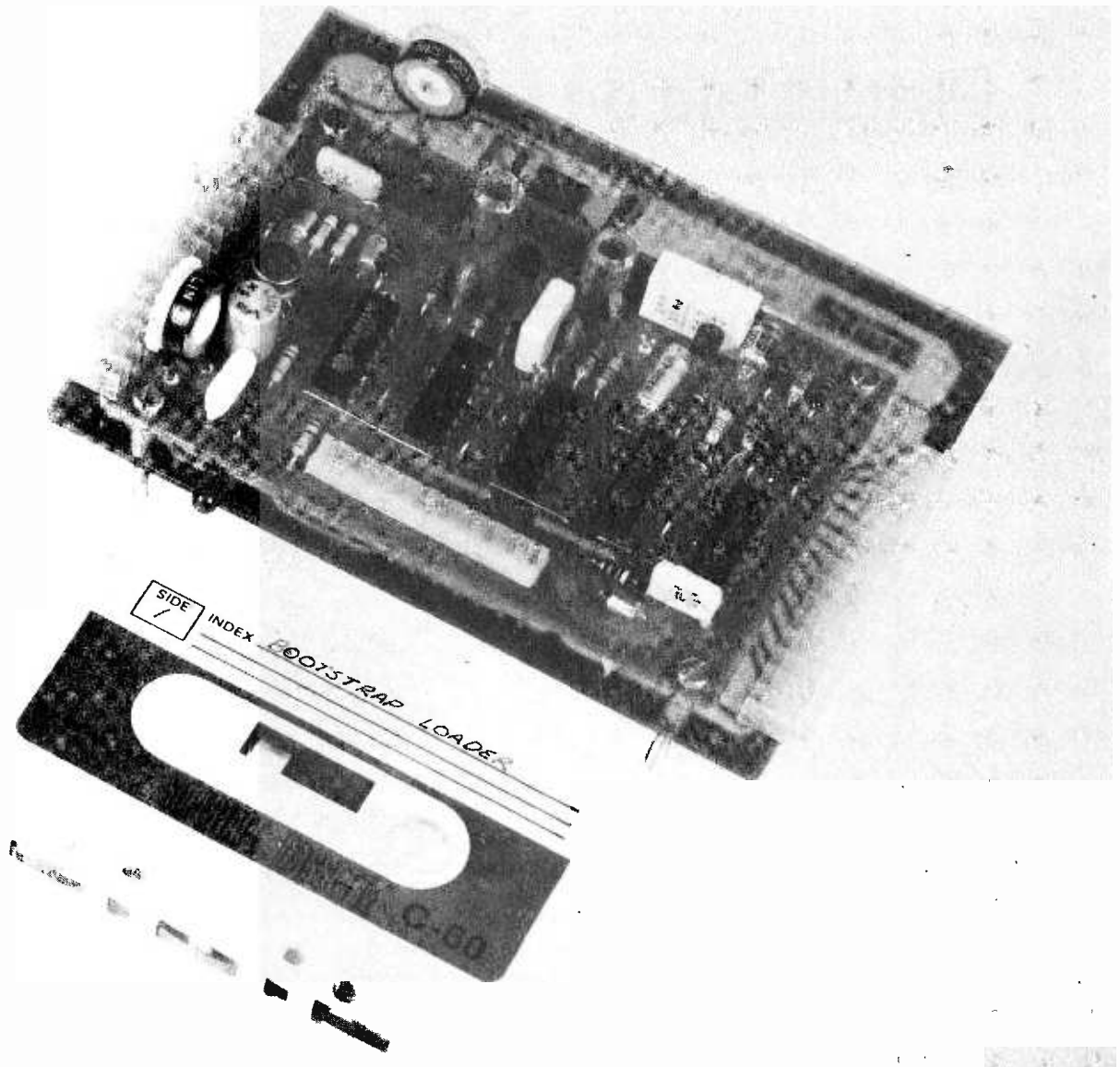
You can use your lockout to prevent data entry during recorder leader, start, and stop times. You can use your existing error flags on the UART output to automatically stop TVT data entry or playback if a parity, framing, or overrun error happens. As noted earlier, these flags can be used with your computer software.

If you have a TVT that uses any control signal as a carriage return, add a decoder to detect only the valid CR signal as a return and ignore all other machine commands. This eliminates errors that tear up the whole display. For instance, a single long tape dropout can be read as an ASCII null command. It gets ignored with a full decoding but defaults to a carriage return if you don't do your decoding properly.

If you're running your BIT BOFFER above 600 baud, you can try making the high frequency sine waves bigger by pre-emphasizing them. This is simply done using a CMOS 4066 analog switch to change the load resistor on the sine wave synthesizer. Pre-emphasis can eliminate much of the recorder's differential gain to higher frequency signals. Since most automatic level controls are a fast attack, slow release type of device, they won't excessively interfere with this particular type of amplitude modulation, so long as the normal or mark state corresponds to the higher frequency and the higher amplitude.

You can probably come up with some other schemes for further error rate hardening on your own.

Can we make it cheaper and simpler? Figure 11 shows a three IC system that's still under evaluation. The receiver uses a quad automotive comparator to perform zero crossing detection (1 and 2), reset the timing capacitor (3), and sense the capacitor voltage



(4). This is routed to a dual D flop that detects ones and zeros with one side and combines the clocks with the other. A dual four bit shift register is one possible route to a simpler transmitter. The actual circuits are still under test.

Let us know your experiences with the BIT BOFFER system so that further improvements can be made, particularly to handle your particular uses for this simple, speed tolerant, software independent cassette interchange system.■

Photo 2: This is how the BIT BOFFER mates with a Southwest Technical Products Corporation UART adapter board in a "piggy back" fashion. See the caption of figure 7 for information on where to get the interface parts.