# The Impossible Dream
# Cassette Interface

**Daniel Lomax**
**Community Data Systems**
**114 E Mohave Rd**
**Tucson AZ 85705**

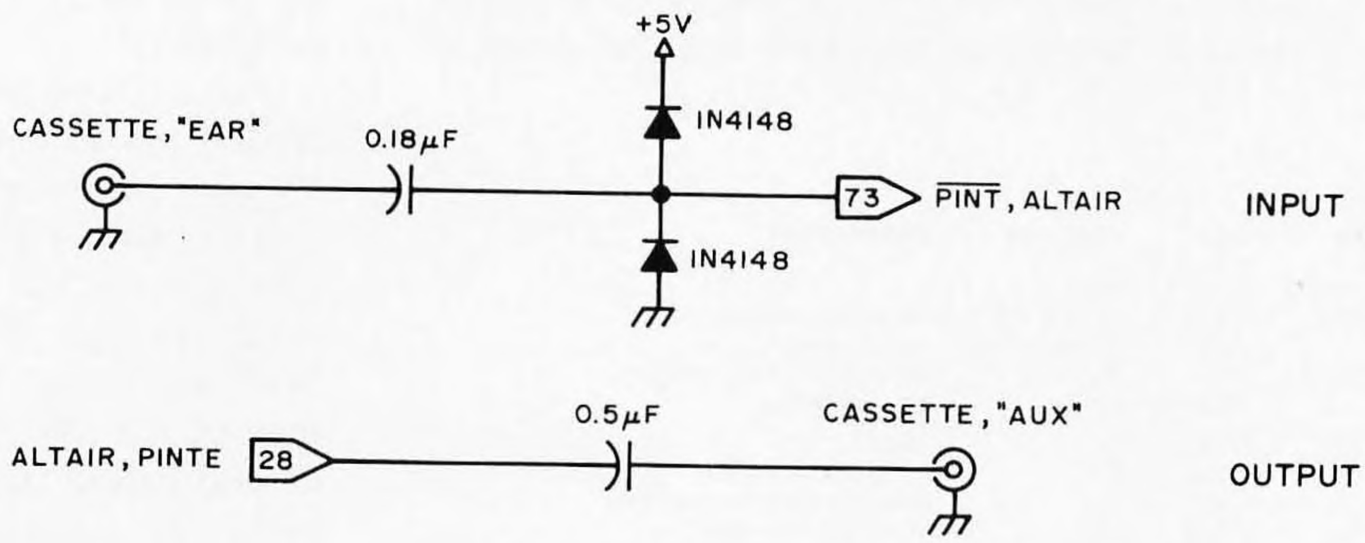
In May 1975, I had a new Altair 8800, from the original *Popular Electronics* offer, with 256 bytes of memory and no more money. What could I do besides blink lights? The first thing I noticed was that there is an addressable latch in the system, the Interrupt Enabled latch on the 8080, which is nicely buffered and displayed on the Altair front panel. After turning it on and off for a few hours, it occurred to me that, with an earphone, the light might make music, and, after several day's mad programming, some incredibly accurate baroque music emerged, including one recorder piece of which a musician friend — who loaded the data for it — said he had never before been able to hear, being too busy playing it.

After making recordings of the music, the question arose: "If I can record music, why not digital data?" I hadn't heard of the various systems being developed at that time, and my tape recorder is a Ward's Airline $30 cheapie. But, anyway, I recorded various tones on cheap tape, played them back, and looked at them on an oscilloscope. I found that a 2000 Hz tone, linked to the tape recorder through a 0.1 uF capacitor, was reliably reproduced — more or less — with the tape recorder volume turned all the way up, as an 8 V peak to peak "square" wave: That is, "reliably" in the sense that the signal never failed to clip, had no visible glitches, and I could see no missed cycles. There was jitter in the frequency, a few percent.

So, I built a breadboard single channel input interface to look at the signal, capacitor-coupled, and diode-limited between ground and +5, with Altair IN instructions. Though this interface was all TTL — no active linear components — it was still unnecessarily complex, as I will show. Anyway, using one cycle of 1100 Hz as 0 and two cycles of 2200 Hz as 1, I found that I could record data and recover it reliably, using the Altair to time the interval between transitions of the playback signal. According to what I have read, this is impossible. 3M Corp is supposed to have spent many millions of dollars working on cassette data recording systems, only to find that audio cassettes were too unreliable. Therefore, established engineers need read no further (except as entertainment), since this might

**About the Author:**

*Daniel Lomax learned electronics in the physics laboratory at Cal Tech in the mid 60s, but never graduated. Recent work in printing and publishing brought him in contact with a burned out Honeywell Controller which was part of a nonworking Photon phototypesetter, repair of which created a business for him (phototypesetter repair) and taught him TTL logic. He is active in the L-5 Society, a group working to encourage the establishment of permanent human colonies at the L-5 Lagrangian point of the Earth-Moon system. Demonstration of his typesetting proficiencies came to us in the form of excellent typeset manuscripts (which we reset for editorial and stylistic reasons).*




be in the same class as perpetual motion and angle trisection with compass and straightedge.

But, if you are an impoverished hobbyist, and would like to store programs and data at more than 1500 baud without spending any *money* – assuming you have a tape recorder, some capacitors, diodes, and connectors – let us dream the impossible dream together. *[The "unreliability" of a device is not necessarily dependent upon the modulation method alone. This method hardly contradicts any principles of information theory . . . .* **CH***]*

After doing the above experiments, the corporation which owned the Altair folded, and with it my source of income and support for my family. I ended up with the Altair, but had no time to play with it until recently. Meanwhile, I have been following the literature, and have observed all kinds of proposed systems, none of them fast enough for the kinds of applications I have been considering and cheap enough for me to afford. Like Dr Suding *[see "Why Wait?" page 46, BYTE, July 1976]*, I cringe at the thought of waiting 15 minutes to find out that noise has destroyed data and I have to start over.

My original bootstrap loader program was 64 bytes long and included a routine which automatically set the appropriate timing value by examining a string of zeros which preceded the data on the tape, and which updated that value using the stop bit between each byte. This article, however, describes a shorter loader, not automatically self-adjusting, and the hardware has been practically eliminated.

It seems I had overlooked the fact that in the Altair there is, in addition to the sense switches, one free input channel – of sorts – $\overline{\text{PINT}}$. If $\overline{\text{PINT}}$ cannot be used for some reason, a program can be written using normal input channels. Also, there is no reason to output two cycles for a single bit,

*Figure 1: Schematic of the "Impossible Dream" Signal Conditioning Logic. The output consists of simply driving the cassette recorder's input with a TTL level signal. The 0.5 uF capacitor is optional, according to the author, and can be replaced by a direct coupling. The input is a simple network to clip the signal coming back from the tape recorder.*

*Listing 1: Minimum Hardware Cassette Output Program. This program is a stand alone method of recording data starting at location BUFFER on to the recorder through the Altair PINTE line. This program terminates when the page address is zero. A more general program could of course be written by changing the initial conditions, and the end of execution test at locations 046 and 047. Note that in the listings of this article, the notation <0> is used to indicate page addresses. The programs shown can be loaded at any arbitrary page boundary by substituting an octal number (such as 003) for <0> every time it appears.*

```
Split
Octal
Address     Octal Code          Label   Op      Operands    Commentary

            377                 SSW     EQU     377
            200                 BUFFER  EQU     200
<0>/000     041 200 <0>     *   START   LXI     H,BUFFER    set initial output pointer;
<0>/003     061 200 <0>         LXI     SP,BUFFER   set the stack;
<0>/006     333 377         LOAD    IN      SSW         input timing value;
<0>/010     117                 MOV     C,A         save it in C;
<0>/011     027                 RAL                 set carry if SSW7 active;
<0>/012     324 055 <0>         CNC     ZERO        if not, output data '0';
<0>/015     322 006 <0>         JNC     LOAD        and if not, look again;
<0>/020     017                 RRC                 recover timing value bit 7;
<0>/021     117                 MOV     C,A         save it in C;
<0>/022     315 066 <0>     NEXT    CALL    ONE         output '1' as start bit;
<0>/025     176                 MOV     A,M         look up data byte;
<0>/026     006 010             MVI     B,010       load bit counter to one byte length;
<0>/030     007             BIT     RLC             set carry if data '1';
<0>/031     334 066 <0>         CC      ONE         if '1', output '1';
<0>/034     324 055 <0>         CNC     ZERO        if not '1'; output '0';
<0>/037     005                 DCR     B           decrement bit counter;
<0>/040     302 030 <0>         JNZ     BIT         if byte incomplete, output next bit;
<0>/043     315 055 <0>         CALL    ZERO        byte complete, output stop bit;
<0>/046     054                 INR     L           advance output pointer;
<0>/047     302 022 <0>         JNZ     NEXT        go output next byte;
<0>/052     166                 HLT                 page done, halt;
<0>/053     000                 NOP                 space for
<0>/054     000                 NOP                     exit jump;

<0>/055     363             ZERO    DI              turn off PINTE;
<0>/056     315 105 <0>         CALL    TIMEA       wait 2C cycles;
<0>/061     373                 EI                  turn on PINTE;
<0>/062     315 105 <0>         CALL    TIMEA       wait 2C cycles;
<0>/065     311                 RET

<0>/066     363             ONE     DI              turn off PINTE;
<0>/067     315 112 <0>         CALL    TIMEB       wait C cycles;
<0>/072     315 105 <0>         CALL    TIMEA       wait 2C cycles;
<0>/075     373                 EI                  turn on PINTE;
<0>/076     315 112 <0>         CALL    TIMEB       wait C cycles;
<0>/101     315 105 <0>         CALL    TIMEA       wait 2C cycles;
<0>/104     311                 RET

<0>/105     121             TIMEA   MOV     D,C         load timing counter;
<0>/106     025             WAITA   DCR     D           count cycles;
<0>/107     302 106 <0>         JNZ     WAITA       count until zero;
<0>/112     121             TIMEB   MOV     D,C         load timing counter;
<0>/113     025             WAITB   DCR     D           count cycles;
<0>/114     302 113 <0>         JNZ     WAITB       count until zero;
<0>/117     311                 RET
```

*Listing 2: Minimum Hardware Cassette Bootstrap Loader. This program is used to read the data recorded on a tape by the output program of listing 1. The program is set up to assume coordination through the Altair interrupt line $\overline{PINT}$, but the method could be applied using timing loops on input as well.*

| Split Octal Address | Octal Code | | Label | Op | Operands | Commentary |
|---|---|---|---|---|---|---|
| | 200 | | BUFFER | EQU | 200 | |
| <0>/000 | 041 200 <0> | * | START | LXI | H,BUFFER | set initial load pointer; |
| <0>/003 | 061 200 <0> | | | LXI | SP,BUFFER | set the stack; |
| <0>/006 | 066 000 | | CLEAR | MVI | M,000 | clear initial load location; |
| <0>/010 | 303 106 <0> | | | JMP | SET | go to work; |
| <0>/070 | 063 | | INT | INX | SP | reset |
| <0>/071 | 063 | | | INX | SP | stack pointer; |
| <0>/072 | 270 | | | CMP | B | was interrupt immediate? |
| <0>/073 | 312 110 <0> | | | JZ | INTE | if so, try, try again; |
| <0>/076 | 326 001 | | | SUI | 001 | set carry if data '1'; |
| <0>/100 | 176 | | | MOV | A,M | look up byte under construction; |
| <0>/101 | 027 | | | RAL | | rotate through carry; |
| <0>/102 | 167 | | | MOV | M,A | put it away; |
| <0>/103 | 332 122 <0> | | | JC | BYTE | if byte complete, go advance pointer; |
| <0>/106 | 333 377 | | SET | IN | SSW | input timing criterion (sense switches); |
| <0>/110 | 107 | | | MOV | B,A | hold for comparison; |
| <0>/111 | 373 | | INTE | EI | | enable interrupt; |
| <0>/112 | 000 | | | NOP | | give it time to act before timing; |
| <0>/113 | 075 | | COUNT | DCR | A | time period until interrupt; |
| <0>/114 | 302 113 <0> | | | JNZ | COUNT | A>0 at interrupt, data '0'; |
| <0>/117 | 303 117 <0> | | LOOP | JMP | LOOP | A=0 at interrupt, data '1'; |
| <0>/122 | 054 | | BYTE | INR | L | advance load pointer; |
| <0>/123 | 302 006 <0> | | | JNZ | CLEAR | if not end of page, go load next byte; |
| <0>/126 | 052 001 <0> | | | LHLD | START | restore initial load pointer; |
| <0>/131 | 351 | | | PCHL | | transfer control to object program; |

*Listing 3: Timing Test Patches to Listing 2. These patches are used to verify the timing for the outputs by testing the actual timing values received for each bit, storing them instead of the data.*

| Split Octal Address | Octal Code | | Name | Op | Operands |
|---|---|---|---|---|---|
| | | | | ORG | 113 |
| <0>/113 | 074 | | COUNT | INR | A |
| | | | | ORG | 076 |
| <0>/076 | 000 | | | NOP | |
| <0>/077 | 000 | | | NOP | |
| <0>/100 | 000 | | | NOP | |
| <0>/101 | 000 | | | NOP | |
| <0>/102 | 167 | | | MOV | M,A |
| <0>/103 | 303 122 <0> | | | JMP | BYTE |
| | | | | ORG | 131 |
| <0>/131 | 166 | | | HLT | |

*Listing 4: Dropout Test Patches to Listing 2: These patches are used to look for spurious binary 1 data in a tape filled with binary 0 data. The Altair will halt on any byte which is not 000 (octal).*

| Split Octal Address | Octal Code | | Name | Op | Operands |
|---|---|---|---|---|---|
| | | | | ORG | 122 |
| <0>/122 | 054 000 | | BYTE | CPI | 000 |
| <0>/124 | 312 006 <0> | | | JZ | CLEAR |
| <0>/127 | 166 | | | HLT | |

so the revised program looks for one cycle of 2020 Hz as 0, and one cycle of 1470 Hz as 1.

To try the system out, you can use a solderless breadboard, or even just a bunch of jumpers with alligator clips. PINTE (for output to tape) can be picked up on the front panel. Both $\overline{PINT}$ and PINTE can be found on the motherboard, at Altair backplane connector pins 73 and 28, respectively. I have found it convenient, for debugging programs using interrupts, to wire $\overline{PINT}$ to one of the extra switches on the Altair front panel, connecting the center terminal of the switch to ground. For the clipping network, I pick off ground from the motherboard support rails, and +5 V from the front panel. Connect it all up as shown in figure 1.

For a system test, clear the memory, then deposit the output program shown in listing 1 into the memory. Replace the HLT at 000,052 with a JMP START,303. The NOPs will serve as the START address. Set the sense switches to 010, and initiate RUN. Start recording. Wait about five seconds, then switch SSW7 to 1. Let the tape run to its end before stopping the Altair. This test begins by outputting continuous zero bits and then, when SSW7 is turned on, it outputs a start bit in the 1 state, then eight data zeros followed by a stop zero. Then it repeats with another start bit, and so forth.

To read back this data, deposit the bootstrap loader into the memory. Change the PCHL at 000,131 to HLT (166). With the connector out of the earphone jack of the recorder, so you can hear the recording, start playing the tape. When the clean, high pitched tone starts (the train of zeros), stop the tape recorder immediately. Put the connector back in, and turn the recorder volume all the way up. Set the sense switches to 050. Start the recorder, wait a second or so for it to settle, then start the Altair with the RUN switch. The Altair should, when the tape runs into the data and begins transmitting bytes, load for about a half second and then halt. To get out of the halt condition, hold the STOP switch up while you RESET. The memory, from 000,200 to 000,377 should be blank, all zeros. Put 377 into 000,377, and try loading the tape again. 000,377 should come out blank again.

If it doesn't work, tape recorder signal polarity may be reversed between recording and playback. Try reversing the signal and ground leads from the tape recorder to the input network. (Disconnect the output connector and any other common grounds.) If the system then works, interchange the EI and DI instructions in the output program to produce correct results with normal connector polarity.

To verify the timing, you can modify the loader as shown in listing 3. Set the sense switches to 000. Start reading the tape while data is being played back, rather than during the leader zeros as usual. The Altair should quickly halt. At address 000,200, and in sequential addresses, you should find the timing values for each bit as it came in. Make a list of these values, and you should see the data pattern. The value 050 was chosen to be in between the timing values for 0 and 1.

To test tape for dropouts, which will read as spurious 1s, use the bootstrap loader with

the patch shown in listing 4. Start the recorder and Altair as usual for data, with the test tape having been filled with data 000 as in the first test. The Altair will halt if it finds any byte that is not 000. It will also probably halt when the tape ends, from shutoff noise.

The data rate for this system, as described, varies with the data: 1470 baud for all binary 1s, 2020 baud for all 0s. I suspect that it would work with higher data rates; but, for my cheap cassette, the signal level won't drive TTL reliably much above 2 kHz. The addition of an amplifier or zero-crossing detector could compensate for that problem, possibly increasing the data rate by a factor of two to four; of course, a better recorder and better tape would also help.

The key feature of this method of recording data is that the recorded signal is symmetrical: It spends as much time high as low. I found that, if I tried to record unsymmetrical signals on the cassette, the narrower pulses tended to be present only as dips and bulges in the distorted attempt at a sine wave that the recorder produces.

Figure 2 shows the waveforms present in the system under various conditions. If the cassette output does not produce a reliable interrupt, try a larger value capacitor or a lower frequency (increase the sense switch setting from 010).

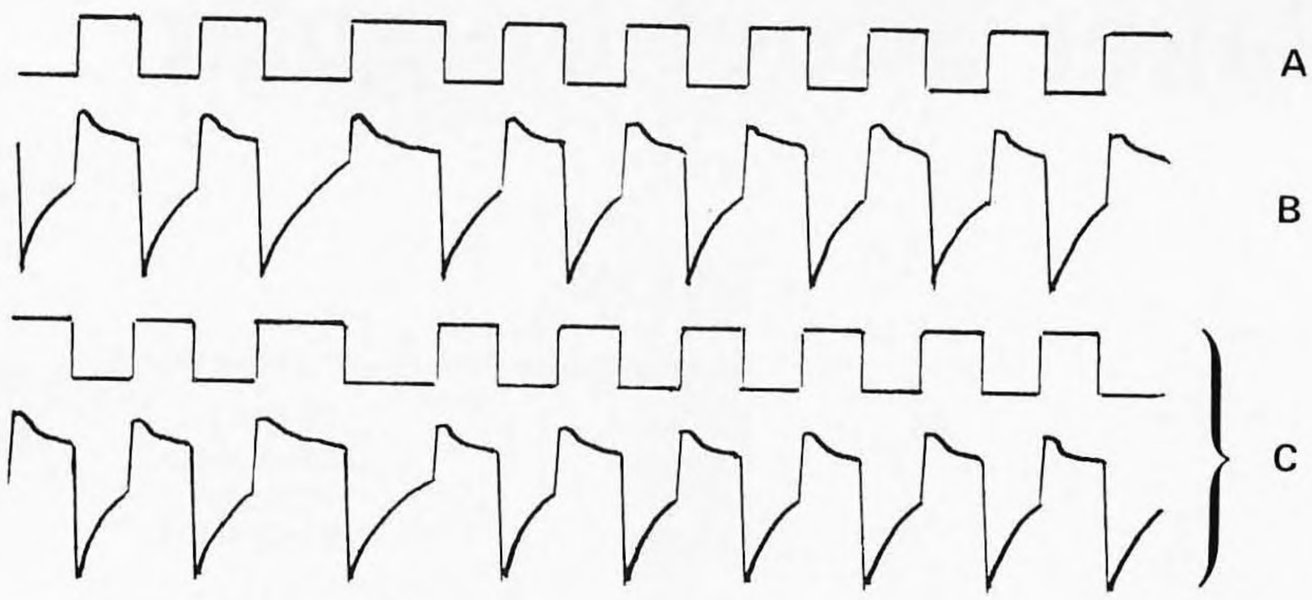



*Figure 2: Tracings of Typical Signals.*

*a. The PINTE output signal from the Altair which is fed to the recorder.*

*b. The input signal clipped and seen by $\overline{PINT}$ when a recording of (a) is fed back into the computer.*

*c. Typical signals, in the case where polarity is reversed. See text for a complete explanation.*

A final note: Timing values (sense switch settings) described in this article are appropriate for an Altair 8800 with memory wait cycles. If the processor is running at 2 MHz with no wait states, try 014 as sense switch setting for the Output Program, and 074 for the bootstrap loader.■