Byte February 1977 article included after Fast Cassette article!

FAST CASSETTE INTERFACE DESCRIPTION

The February 1977 issue of BYTE magazine (page 82) contained an interesting article on a minimum hardware cassette interface. I have used this technique to develope a cassette I/O arrangement which records and loads via tape at over 1600 baud. Because I do not unpack bytes for recording, the effective data rate is over 160 bytes/second. The accompanying software listing for 6502 systems provides a record start sequence which requires at least ten 16 bytes followed with an 0F byte to be inputted in succession before loading can commence. At end of loading, a two byte checksum is used for detection of errors. The hardware consists of a direct connection from a one-bit output port to the microphone input and a non-inverting hysteris circuit incorporating an LM339 comparator as the playback electronics. Actually, I've used a direct connection for the playback with success but some cassette decks won't work unless the comparator is used. My General Electric and two Sankyo tape decks work very well without the comparator but the Realistic deck will not operate at all without the comparator.

An interesting note is that some tape decks put the signal on the barrel of the record and play jacks instead of on the inner tip. Also, some tape decks invert the signal on playback. This inversion can be compensated by inserting an inverter (7400 or equiv.) between the LM339 and the input port.

To use this software, enter data in memory locations 0123-0127 as follows:

0123 = LOAD/NO
0124-0125 = START ADDRESS
0126-0127 = END ADDRESS

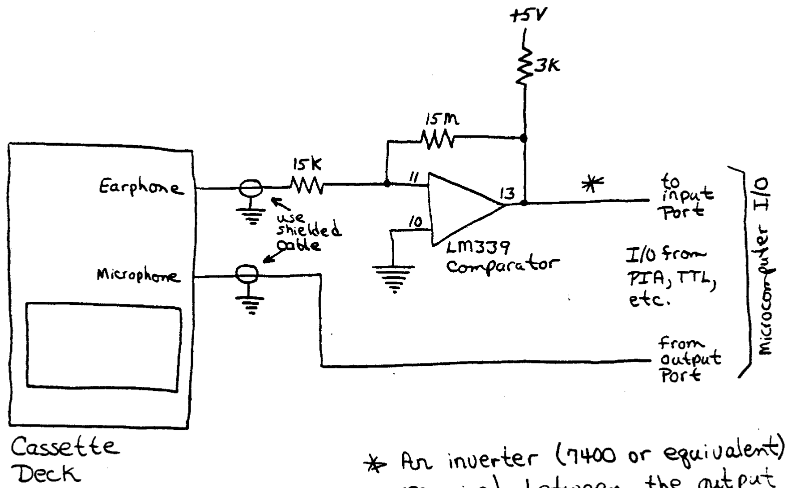
The record routine will record from START ADDRESS to END ADDRESS. LOAD/NO has no meaning to the record software.

The load routine will load from START ADDRESS to END ADDRESS but store data in memory only if LOAD/NO does not equal 0. When equal 0, LOAD/NO can be used for verifying and conditionally selecting modules on tape.

The load and record routines have callable entry points at C/WRITE (4000) and C/READ (40A5), and non-callable entry points at LOAD.ENTRY (4141) and RECORD.ENT (4152). If C/READ is called, the Z-bit in the PSR will be true on return if no error was detected and false if errors occurred. If execution is at the non-callable entry LOAD.ENTRY, a break (via BRK instruction) will be executed at end of loading and register A will indicate if the data was loaded correctly: R(A)=00 for good load, and EE for error.

To sum up, this has been a very reliable scheme and works error-free with the cheapest tapes (even Concert tapes which can be bought at many department stores at 3 for $1.00).

FAST CASSETTE INTERFACE CIRCUITRY



LM339 Power

	Pin
+5V	3
GND	12

* An inverter (7400 or equivalent) may be required between the output of the LM339 and the input port if your tape deck inverts the signal on playback.

If you ordered the KIM version of ASSM/TED, the cassette I/O is preconfigured for the following connections:

		Function	Pin number on Application Connector
tape deck Ø	REMOTE	Motor Control Ø	9
tape deck Ø	Microphone	Cassette record	12
tape deck 1	REMOTE	Motor Control 1	10
tape deck 1	EARPHONE	Cassette playback	11

>ASSEMBLE LIST   *Change Underlined Portion Per Your System Requirements*

```
                0010              .BA $4000
                0020              .OS
                0030 ;
                0040 ;***** FAST CASSETTE INTERFACE *****
                0045 ;       (CONFIGURED FOR KIM)
                0050 ;
                0060 ;
                0070 ;== INPUT/OUTPUT ==
                0080 C/PORT       .DE $1702     ;CASSETTE I/O
                0090 C/PORTD      .DE $1703     ;PORT DIRECTION REG.
                0100 ;BIT 3 = WRITE TO CASSETTE;  BIT 2 = READ FROM CASSETTE
                0120 ;
                0130 ;
                0140 ;== VARIABLES ==
                0150 CHECKSUM     .DE $B2 TWO BYTE CHECKSUM
                0160 COUNT        .DE ADDRS
                0170 FORM+BYTE    .DE $B4
                0180 SYNC+COUNT   .DE FORM+BYTE
                0190 BIT.COUNT    .DE $B5
                0200 ADDRS        .DE $B6
                0210 ;
                0220 ;INPUT PARMS
                0230 LOAD/NO      .DE $0123     ;00=NO LOAD;  01=LOAD
                0240 START        .DE $0124     ;START ADDRESS
                0250 END          .DE $0126     ;END ADDRESS +1
                0260 ;
                0270 ;
                0280 ;
                0290 ;C/WRITE:  WRITE TO TAPE FROM (START) TO (END)
                0300 ;
4000- AD 03 17  0310 C/WRITE      LDA C/PORTD
4003- 09 08     0320              ORA #%00001000       ;BIT 3 = CASSETTE OUT
4005- 8D 03 17  0330              STA C/PORTD
                0340 ;THE ABOVE INITIALIZES BIT 3 FOR OUTPUT ON PIA
                0350 ;
4008- A9 20     0360              LDA #$20 32 TIMES
400A- 85 B6     0370              STA *COUNT
400C- A9 16     0380 LOOP/RECST   LDA #$16 SYNC CHAR.
400E- 20 41 40  0390              JSR WRITE/BYTE
                0400 ;
4011- A9 10     0410              LDA #$10
4013- 85 B4     0420              STA *SYNC+COUNT
4015- 20 5D 40  0430 LOOP/DELSY   JSR OUT:ZERO
4018- C6 B4     0440              DEC *SYNC+COUNT
401A- D0 F9     0450              BNE LOOP/DELSY
                0460 ;DELAY TIME FOR SYNC
                0470 ;
401C- C6 B6     0480              DEC *COUNT
401E- D0 EC     0490              BNE LOOP/RECST
                0500 ;
4020- 20 84 40  0510              JSR MOVE+ST/AD START > ADDRS (2)
                0520 ;
4023- A9 0F     0530              LDA #$0F RECORD START CHAR.
4025- 20 41 40  0540              JSR WRITE/BYTE
                0550 ;
4028- A2 00     0560              LDX #$00
```

```
402A- 86 B2      0570              STX *CHECKSUM CLEAR CHECKSUM
402C- 86 B3      0571              STX *CHECKSUM+$01
                 0580 ;
402E- A1 B6      0590 LOOP/DATA    LDA (ADDRS,X) LOAD DATA
4030- 20 41 40   0600              JSR WRITE/BYTE
4033- 20 8F 40   0610              JSR INC/COMP
4036- 90 F6      0615              BCC LOOP/DATA
                 0619 ;
                 0620 ;
4038- A5 B3      0621              LDA *CHECKSUM+$01
403A- 48         0622              PHA SAVE HI CKSUM
403B- A5 B2      0630              LDA *CHECKSUM
403D- 20 41 40   0631              JSR WRITE/BYTE WRITE LO CKSUM FIRST
4040- 68         0632              PLA HI CKSUM NEXT
                 0640 ;THE ABOVE WRITES BOTH CHECKSUM BYTES
                 0650 ;
                 0660 ;
                 0670 ;ROUTINE TO WRITE A BYTE TO TAPE
                 0680 ;
4041- 85 B4      0690 WRITE/BYTE STA *FORM+BYTE
4043- 20 34 41   0691              JSR CKSUM+ADD UPDATE CHECKSUM COUNTER
4046- 20 7C 40   0700              JSR OUT:ONE START BIT
4049- A9 08      0710              LDA #$08 8 BITS
404B- 85 B5      0720              STA *BIT.COUNT
404D- 06 B4      0730 DATA/LOOP    ASL *FORM+BYTE SHIFT LEFT INTO CARRY
404F- 90 05      0740              BCC ZERO.BIT
4051- 20 7C 40   0760 ONE.BIT      JSR OUT:ONE
4054- F0 03      0770              BEQ CK+END+BY
4056- 20 5D 40   0790 ZERO.BIT     JSR OUT:ZERO
4059- C6 B5      0800 CK+END+BY    DEC *BIT.COUNT
405B- D0 F0      0810              BNE DATA/LOOP
                 0820 ;NOW OUTPUT 1 STOP BIT
                 0830 ;
                 0840 ;ROUTINE OUTPUT A ZERO TO TAPE
                 0850 ;
405D- A9 20      0860 OUT:ZERO     LDA #$20 '0' DELAY CONSTANT
                 0870 ;
                 0880 ;
                 0890 ;ROUTINE WRITE TO TAPE
                 0900 ;
405F- 48         0910 WRITE        PHA SAVE DELAY CONSTANT
4060- AD 02 17   0920              LDA C/PORT
4063- 09 08      0930              ORA #%00001000        ;OUT A '1' ON BIT 3
4065- 8D 02 17   0940              STA C/PORT
4068- 68         0950              PLA
4069- 48         0960              PHA
406A- AA         0970              TAX DELAY CONSTANT
406B- 20 78 40   0980              JSR LOOPD
406E- AD 02 17   0990              LDA C/PORT
4071- 29 F7      1000              AND #%11110111        ;OUT A '0' ON BIT 3
4073- 8D 02 17   1010              STA C/PORT
4076- 68         1020 X            PLA
4077- AA         1030              TAX DELAY CONSTANT
4078- CA         1040 LOOPD        DEX
4079- D0 FD      1050              BNE LOOPD
407B- 60         1060              RTS
                 1070 ;
                 1080 ;
```

```
                 1090 ;
                 1100 ;ROUTINE OUTPUT A ONE TO TAPE
                 1110 ;
407C- A9 50      1120 OUT:ONE     LDA #$50 '1' DELAY CONSTANT
407E- D0 DF      1130             BNE WRITE
                 1140 ;
                 1150 ;
                 1160 ;DELAY FOR '0' TIME FOR READ
                 1170 ;
4080- A2 30      1180 READ.DELAY LDX #$30
4082- D0 F4      1190             BNE LOOPD
                 1200 ;
                 1210 ;
                 1220 ;
                 1230 ;ROUTINE MOVE FROM START TO ADDRS
                 1240 ;
4084- AD 24 01   1250 MOVE+ST/AD LDA START
4087- 85 B6      1260             STA *ADDRS
4089- AD 25 01   1270             LDA START+$01
408C- 85 B7      1280             STA *ADDRS+$01
408E- 60         1290             RTS
                 1300 ;
                 1310 ;
                 1320 ;
                 1330 ;ROUTINE INCREMENT AND COMPARE
                 1340 ;
408F- E6 B6      1350 INC/COMP    INC *ADDRS
4091- D0 02      1360             BNE SKIP/INC
4093- E6 B7      1370             INC *ADDRS+$01
4095- A5 B7      1380 SKIP/INC    LDA *ADDRS+$01
4097- CD 27 01   1390             CMP END+$01
409A- 90 08      1400             BCC NOT/END
409C- A5 B6      1410             LDA *ADDRS
409E- CD 26 01   1420             CMP END
40A1- 90 01      1430             BCC NOT/END
40A3- 38         1440             SEC
40A4- 60         1450 NOT/END     RTS
                 1460 ;ON RETURN, C=CLEAR: NOT END; C=SET: END REACHED
                 1470 ;
                 1480 ;
                 1490 ;
                 1500 ;
                 1510 ;C/READ:  READ FROM TAPE TO (START) TO (END)
                 1520 ;
40A5- A2 00      1530 C/READ      LDX #$00
40A7- 86 B6      1540             STX *COUNT
40A9- 20 EF 40   1550 LOOP/LOAD   JSR READ/BYTE
40AC- C9 16      1560             CMP #$16 SYNC
40AE- D0 04      1570             BNE SKIP/1
40B0- E6 B6      1580             INC *COUNT
40B2- D0 F5      1590             BNE LOOP/LOAD
                 1600 ;
40B4- A4 B6      1610 SKIP/1      LDY *COUNT
40B6- C0 0A      1620             CPY #$0A MUST BE > = 10 SYNC'S
40B8- 90 EB      1630             BCC C/READ
40BA- C9 0F      1640             CMP #$0F RECORD START
40BC- D0 E7      1650             BNE C/READ
                 1660 ;
```

```
40BE- A0 00       1670             LDY #$00
40C0- 84 B2       1680             STY *CHECKSUM
40C2- 84 B3       1681             STY *CHECKSUM+$01 CLEAR CHECKSUM LOCATIONS
40C4- 20 84 40    1690             JSR MOVE+ST/AD START > ADDRS (2)
                  1700 ;
                  1710 ;NOW LOAD DATA
40C7- 20 EF 40    1720 LOOP/69     JSR READ/BYTE
40CA- AC 23 01    1730             LDY LOAD/NO CKG. IF TO STORE
40CD- F0 02       1740             BEQ SKIP/STORE
40CF- 81 B6       1750             STA (ADDRS,X)
40D1- 20 8F 40    1760 SKIP/STORE JSR INC/COMP
40D4- 90 F1       1770             BCC LOOP/69
40D6- A5 B3       1771             LDA *CHECKSUM+$01
40D8- 48          1772             PHA SAVE CHSUM HI
40D9- A5 B2       1780             LDA *CHECKSUM
40DB- 48          1790             PHA SAVE CHECKSUM LO
40DC- 20 EF 40    1800             JSR READ/BYTE
40DF- 68          1810             PLA
40E0- C5 B4       1820             CMP *FORM+BYTE CHECK CHECKSUM LO
40E2- D0 07       1821             BNE RETURN
40E4- 20 EF 40    1822             JSR READ/BYTE
40E7- 68          1823             PLA
40E8- C5 B4       1824             CMP *FORM+BYTE CHECK CHECKSUM HI
40EA- 60          1830             RTS
40EB- 68          1831 RETURN      PLA
40EC- A9 FF       1832             LDA #$FF CLEAR Z-BIT
40EE- 60          1833             RTS
                  1840 ;ON RETURN Z-BIT=TRUE:GOOD LOAD;  Z-BIT==FALSE:ERROR
                  1850 ;
                  1860 ;
                  1870 ;ROUTINE READ A BYTE FROM TAPE
                  1880 ;
40EF- 20 2E 41    1890 READ/BYTE   JSR IN/PORT
40F2- D0 FB       1900             BNE READ/BYTE LOOP UNTIL 0
                  1910 ;
40F4- 20 2E 41    1920 WAIT+FOR+1  JSR IN/PORT
40F7- F0 FB       1930             BEQ WAIT+FOR+1 LOOP UNTIL 1
                  1940 ;
40F9- 20 80 40    1950             JSR READ.DELAY
40FC- 20 2E 41    1960             JSR IN/PORT
40FF- F0 F3       1970             BEQ WAIT+FOR+1 IF ZERO
                  1980 ;
4101- 20 2E 41    1990 WAIT+FOR+0  JSR IN/PORT
4104- D0 FB       2000             BNE WAIT+FOR+0 WAIT TIL END OF START BIT
                  2010 ;
4106- A9 08       2020             LDA #$08
4108- 85 B5       2030             STA *BIT.COUNT
                  2040 ;
410A- 20 2E 41    2050 WAIT+TO+CN  JSR IN/PORT
410D- F0 FB       2060             BEQ WAIT+TO+CN LOOP UNTIL '1'
410F- 20 80 40    2070             JSR READ.DELAY
4112- 20 2E 41    2080             JSR IN/PORT
4115- F0 06       2090             BEQ PROCESS+0 IF '0' THEN ZERO, ELSE ONE
4117- 20 2E 41    2110 PROCESS+1   JSR IN/PORT
411A- D0 FB       2120             BNE PROCESS+1 LOOP UNTIL '0'
411C- 38          2130             SEC
411D- B0 01       2140             BCS ROTATE+IN
411F- 18          2160 PROCESS+0   CLC
```

```
4120- 26 B4     2170 ROTATE+IN   ROL *FORM+BYTE ROTATE CARRY
4122- C6 B5     2180             DEC *BIT.COUNT
4124- D0 E4     2190             BNE WAIT+TO+CH
4126- A5 B4     2200             LDA *FORM+BYTE
4128- 20 34 41  2201             JSR CKSUM+ADD UPDATE CHECKSUM
412B- A5 B4     2202             LDA *FORM+BYTE
412D- 60        2210             RTS
                2220 ;
                2230 ;
                2240 ;INPUT FROM TAPE
                2250 ;
412E- AD 02 17  2260 IN/PORT     LDA C/PORT
4131- 29 04     2270             AND #%00000100       ;MASK OUT ALL BUT BIT 2
4133- 60        2280             RTS
                2281 ;
                2282 ;
                2283 ;
                2284 ;UPDATE CHECKSUM COUNTERS
                2285 ;
4134- 18        2286 CKSUM+ADD   CLC
4135- D8        2287             CLD
4136- 65 B2     2288             ADC *CHECKSUM+$00 ADD R(A) TO CKSUM LO
4138- 85 B2     2289             STA *CHECKSUM+$00
413A- A9 00     2290             LDA #$00
413C- 65 B3     2291             ADC *CHECKSUM+$01 ADD 00 TO CKSUM HI
413E- 85 B3     2292             STA *CHECKSUM+$01
4140- 60        2293             RTS
                2294 ;
                2300 ;
                2310 ;
4141- 20 A5 40  2320 LOAD.ENTRY  JSR C/READ
4144- D0 08     2330             BNE BAD
4146- A9 00     2340             LDA #$00 INDICATE GOOD LOAD BY R(A)=00
4148- 00        2350 B           BRK
4149- EA        2360             NOP
414A- EA        2370             NOP
414B- 4C 41 41  2380             JMP LOAD.ENTRY
414E- A9 EE     2390 BAD         LDA #$EE INDICATE BAD LOAD BY R(A)=EE
4150- D0 F6     2400             BNE B
                2410 ;
4152- 20 00 40  2420 RECORD.ENT  JSR C/WRITE
4155- 00        2430             BRK
4156- EA        2440             NOP
4157- EA        2450             NOP
4158- 4C 52 41  2460             JMP RECORD.ENT
                2470 ;
                2480 END+OF+PGM  .EN
```

LABEL FILE: [/ = EXTERNAL]

```
/C/PORT=1702          /C/PORTD=1703         /CHECKSUM=00B2
/COUNT=00B6           /FORM+BYTE=00B4       /SYNC+COUNT=00B4
/BIT.COUNT=00B5       /ADDRS=00B6           /LOAD/NO=0123
/ ART=0124            /END=0126             C/WRITE=4000
LOOP/RECST=400C       LOOP/DELSY=4015       LOOP/DATA=402E
WRITE/BYTE=4041       DATA/LOOP=404D        ONE.BIT=4051
```

```
ZERO.BIT=4056          CK←END←BY=4059        OUT:ZERO=405D
WRITE=405F             X=4076                LOOPD=4078
OUT:ONE=407C           READ.DELAY=4080       MOVE←ST/AD=4064
INC/COMP=408F          SKIP/INC=4095         NOT/END=40A4
C/READ=40A5            LOOP/LOAD=40A9        SKIP/1=40B4
LOOP/69=40C7           SKIP/STORE=40D1       RETURN=40EB
READ/BYTE=40EF         WAIT←FOR←1=40F4       WAIT←FOR←0=4101
WAIT←TO←CN=410A        PROCESS←1=4117        PROCESS←0=411F
ROTATE←IN=4120         IN/PORT=412E          CKSUM←ADD=4134
LOAD.ENTRY=4141        B=4148                BAD=414E
RECORD.ENT=4152        END←OF←PGM=415B
//0000,415B,415B
>
```

The Impossible Dream Cassette Interface

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

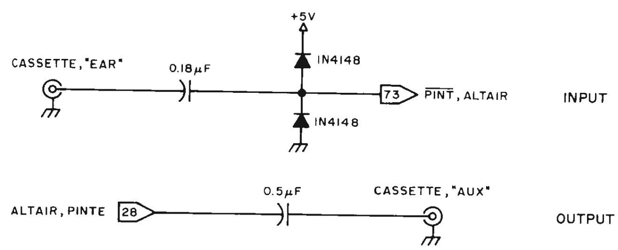
In May 1975, I had a new Altair 8800, from the original *Popular Electronics* offer, with 256 bytes of memory and no more money. What could I do besides blink lights? The first thing I noticed was that there is an addressable latch in the system, the Interrupt Enabled latch on the 8080, which is nicely 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 – PINT. If 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).

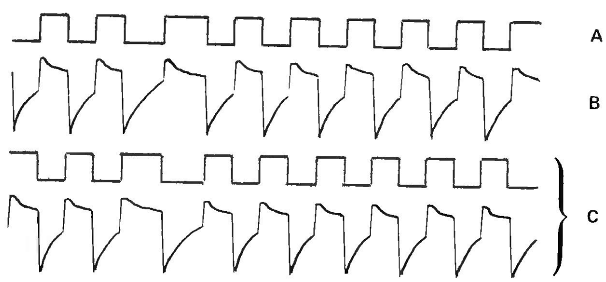


Figure 2: Tracings of Typical Signals.
- *a. The PINTE output signal from the Altair which is fed to the recorder.*
- *b. The input signal clipped and seen by $\overline{PINT}$ when a recording of (a) is fed back into the computer.*
- *c. Typical signals, in the case where polarity is reversed. See text 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.■