magnetic tape data recovery in software

a very nerdy talk for the Vintage Computer Festival West

Len Shustek
CHM Chairman emeritus

the motivation

in my attic
for 50 years!

```
*****************************************************************
*   *************************************************************   *
*   *                                                           *   *
*   *                        PLAGO/360                          *   *
*   *                                                           *   *
*   *    THE POLYTECHNIC LOAD-AND-GO PL/I TRANSLATOR            *   *
*   *                                                           *   *
*   *                        VERSION I                          *   *
*   *************************************************************   *
*****************************************************************

Project Adviser:  Professor Harold W. Lawson, Jr.
Project Staff:    Mr. David Doucette
                  Mr. Jeffrey Goldfarb
                  Mr. Stanley Habib
                  Mr. Aaron Kershenbaum
                  Mr. Leonard Shustek

  The Polytechnic Institute of Brooklyn
  September 25, 1968
```

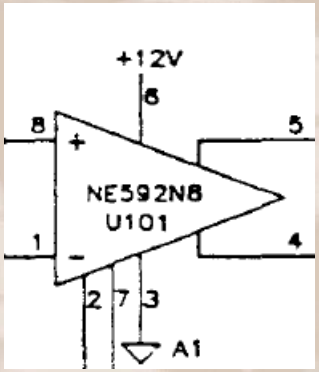
the failure

Qty	Description	Unit Price	Line Total
1	Restore 9-track 800 BPI tape and land data to new media for Client delivery. This is a sole source of media being sent to eMag for processing; no other copies exist.	$1,000 (flat fee)	$1,000

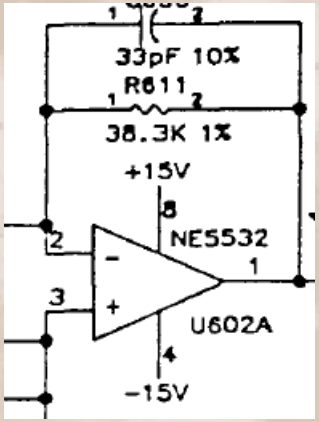
```
*      ?<?PLAGO SYSTEM DUMP ROUTINE
*
*      ???THIS CSECT IS ENTERED BY THE E-MACHINE FOR THE INTERPRETATION
*         OF THE BUILT-IN-FUNCTION 'DEBUG'.
*
*      ???THIS FUNCTION IS INTENDED FOR USE ??LY BY THOSE KNOWLEDGEABLE
*         IN THE INTERNALS OF PLAGO.
*
*INPUT-   R4   OUT?UT BUFFER ADDRESS
*         R5   OUTPUT ROUTINE ADDRESS
*      ?   R15   DUMP ROUTINE BASE ADDRESS
*       ? R6-R12   AS USED IN E-MACHINE
*
*      ?? IN ADDITION, THE TOP ELEMENT OF THE DSTK IS ASSUMED TO BE
*      ????OINTING AT A FIXED BINARY NUMBER WHOSE BITS, STARTING FROM
*       ??THE RIGHT, HAVE THE FOLLOWING SIGNIFICANCE:
*      ???1. REGISTER DISPLAY REQUESTED
*       ??2. D-STK DUMP REQUESTED
```

? marks errors

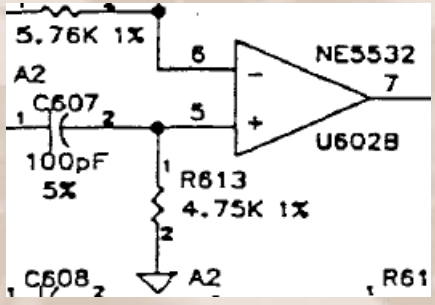
traditional hardware



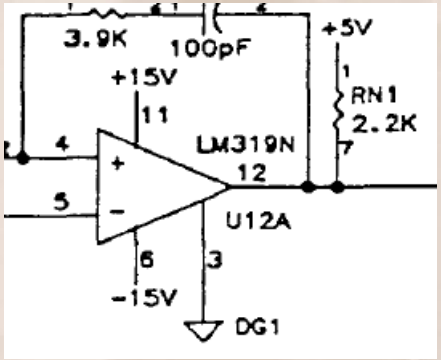
pre-amp



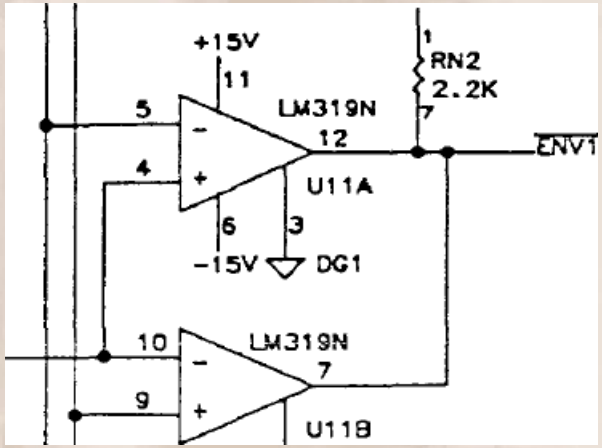
differential amp



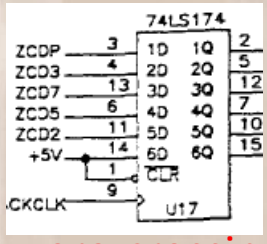
differentiator



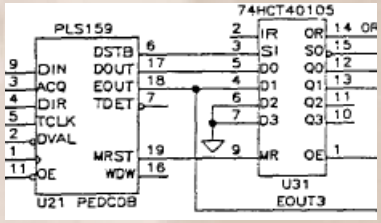
comparator



threshold detector



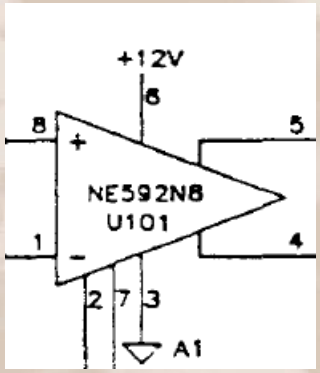
zero-crossing detector



data bit decoder

…and <u>then</u> block and error handling!

software-based*



pre-amp

analog to digital conversion

detect flux transitions

detect bits

assemble into words, then blocks

OK ?

yes

next block

no

in software, so we can retry without rereading the tape!

* Inspired by Paul Pierce's Java-based project in 2004

my first attempt

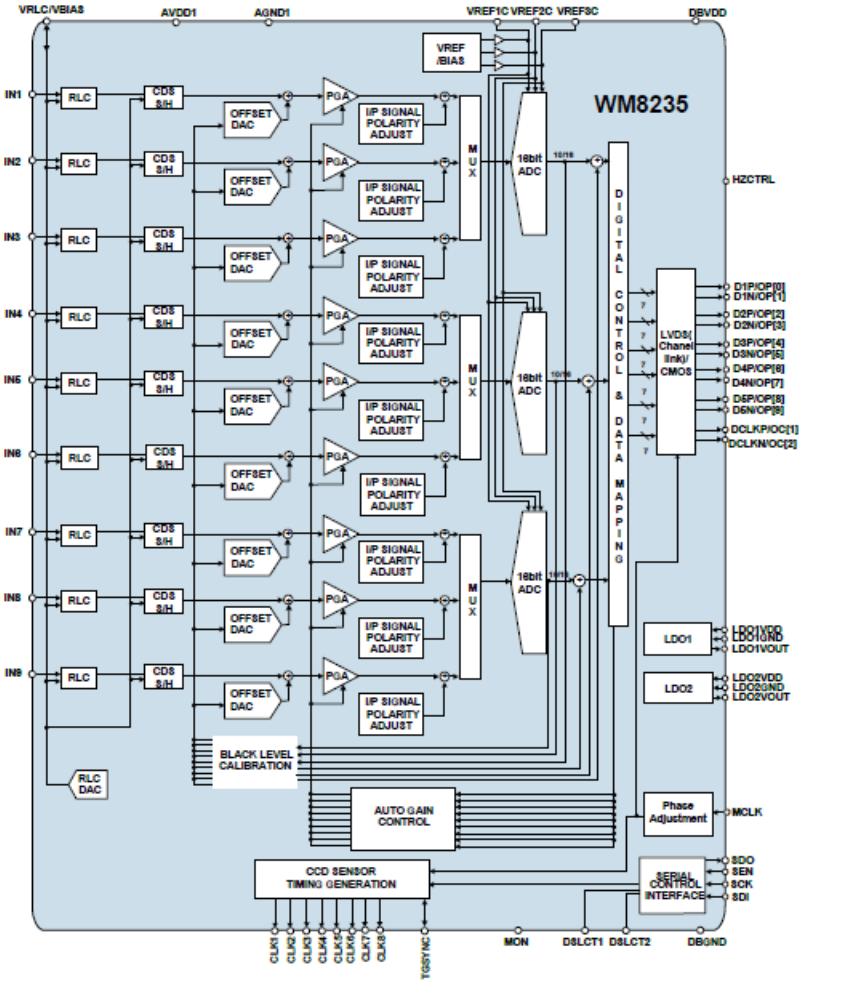
wolfson microelectronics

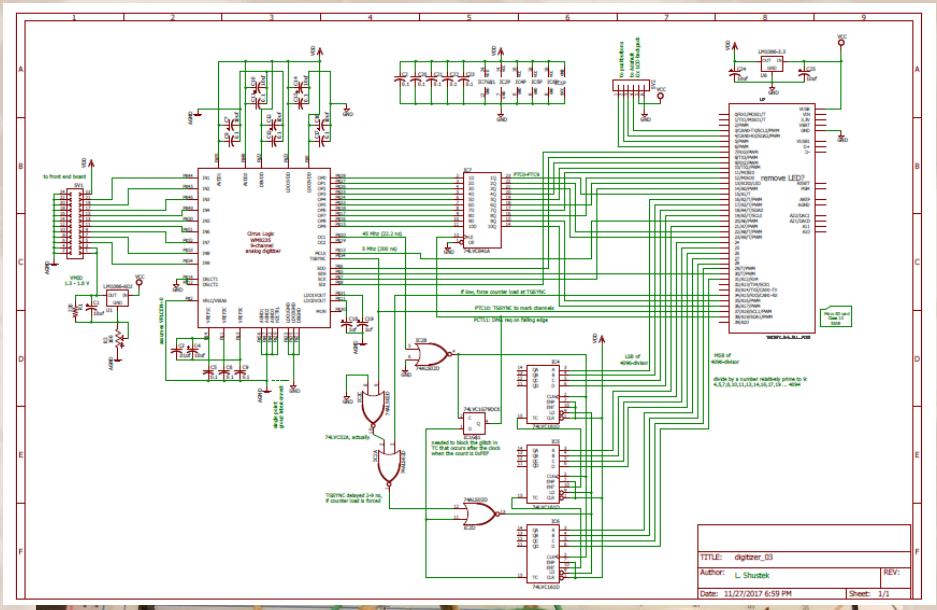
210MSPS 9-Channel AFE

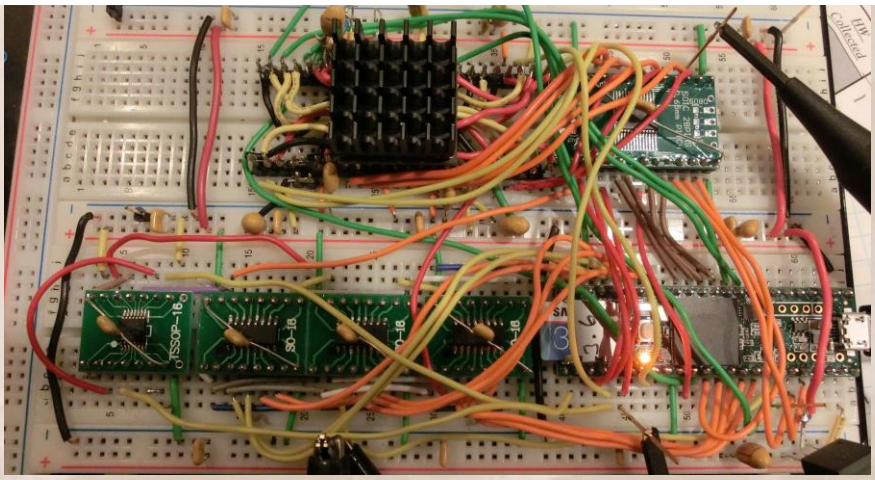
CIRRUS LOGIC

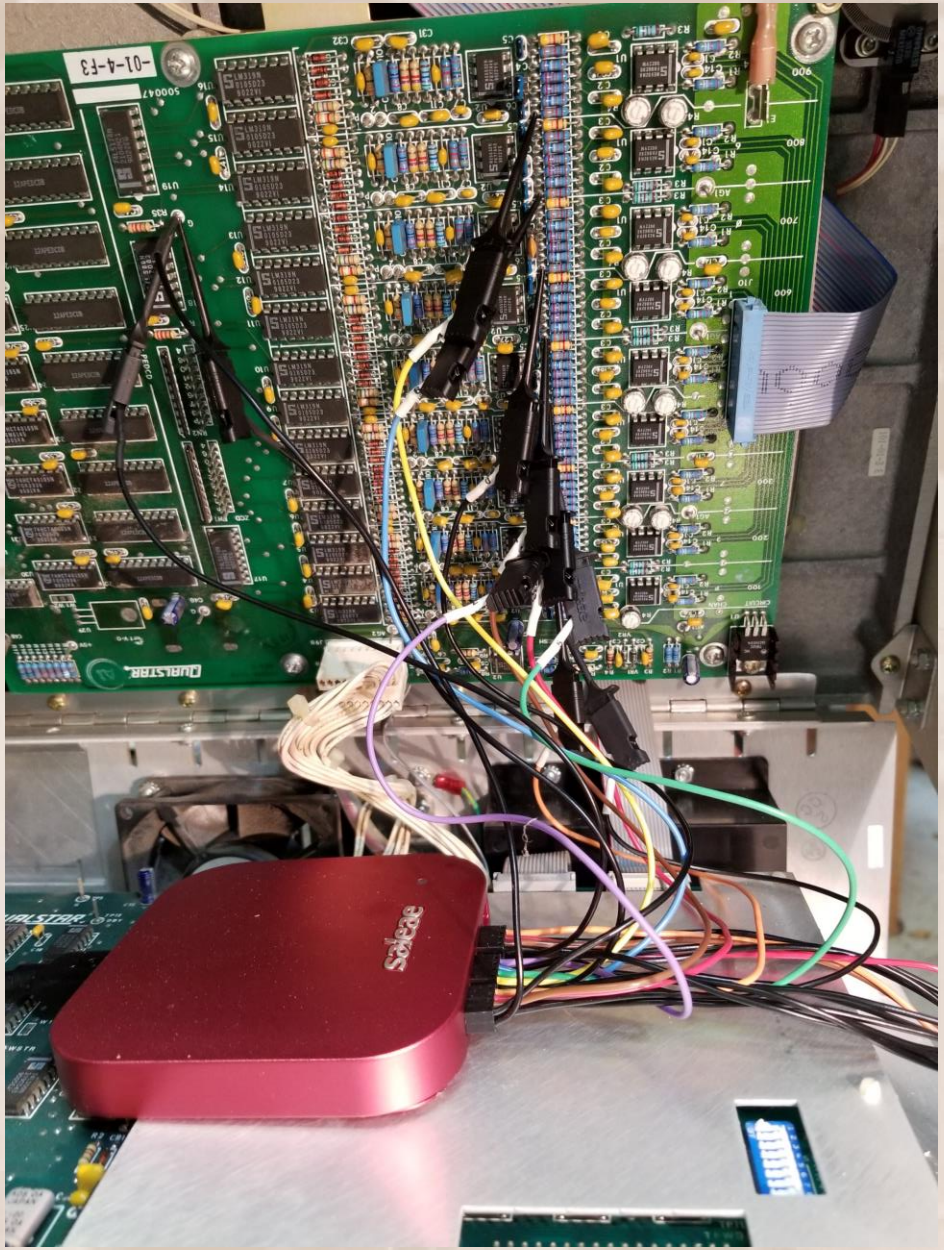
WM8235

BLOCK DIAGRAM







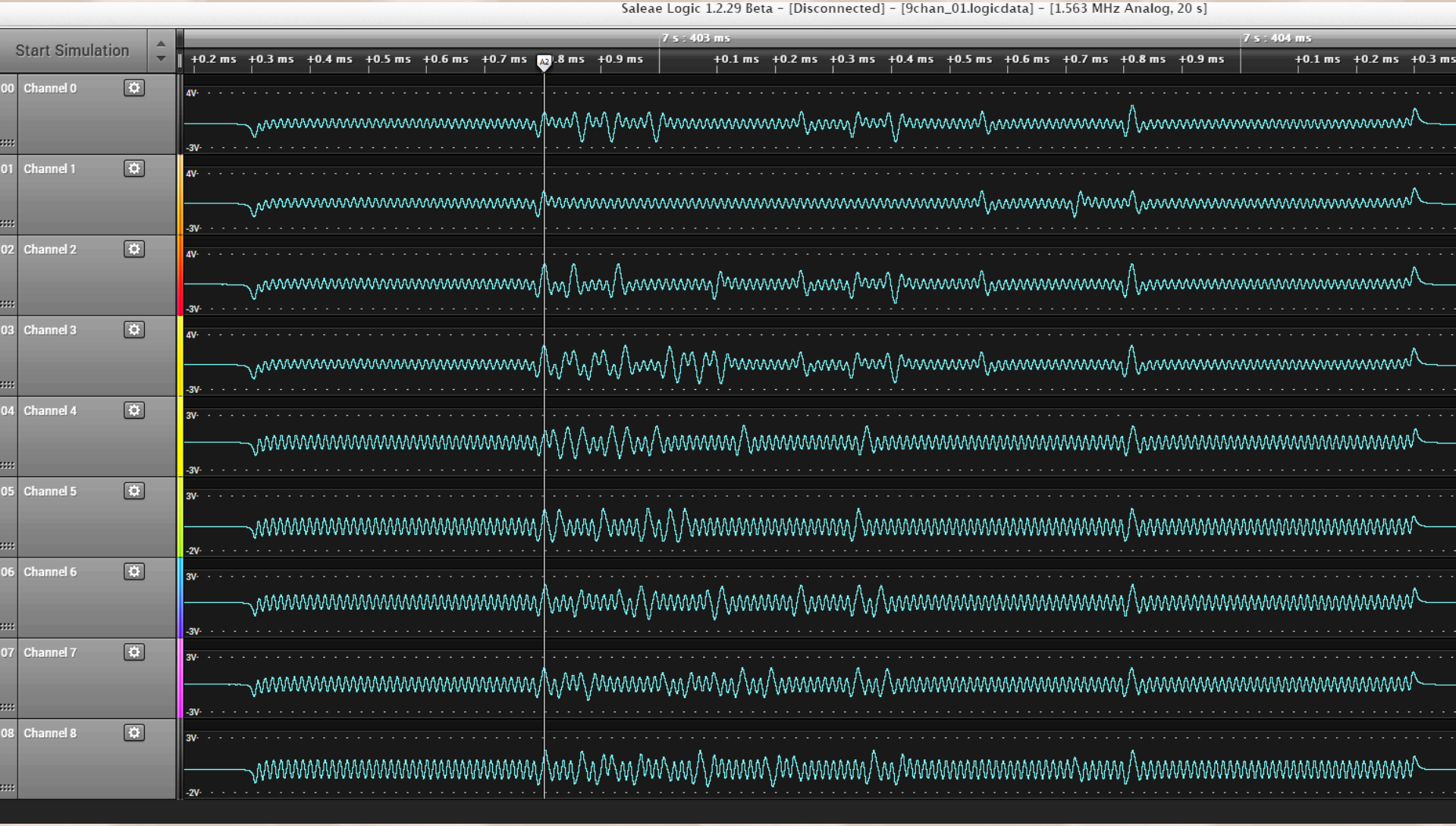


easier:

Saleae Logic Pro 16 logic analyzer

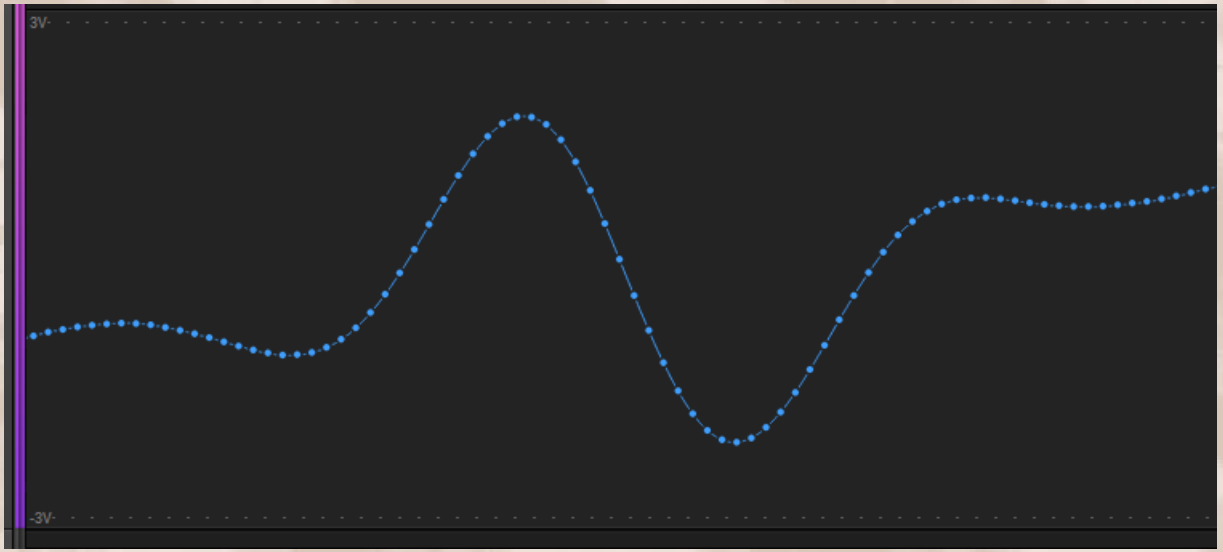
- *16 analog channels*
- *max 50M samples/sec*
- *“unlimited” storage*

connected to a Qualstar 1052 tape deck



6 to 16 tracks, 100 to 10,000 bits/inch

+3 volts



-3 volts

want 20 to 30 samples/bit

50 IPS x 800 BPI x 20 samples/bit =
800K samples/sec

the raw data

```
Time[s], 0 MSB, 1, 2, 3, 4, 5, 6, 7 LSB, 8P
Time [s],0 MSB-Analog,1-Analog,2-Analog,3-Analog,4-Analog,5-Analog,6-Analog,7 LSB-Analog,8P-Analog
4.507594559999999, 0.10085, -0.03400, -0.16784, -1.09570, 0.13550, 0.90018, 0.22597, 1.13374, -0.21236
4.507594880000000, -0.02258, 0.09941, -0.30091, -1.22020, 0.08454, 0.85893, 0.10245, 1.25193, -0.33846
4.507595200000000, -0.14601, 0.23281, -0.43398, -1.34989, 0.02849, 0.80220, -0.02622, 1.34956, -0.45448
4.507595520000000, -0.26944, 0.36109, -0.56193, -1.47439, -0.02247, 0.73515, -0.14974, 1.43179, -0.56545
4.507595840000000, -0.38773, 0.48937, -0.68477, -1.59370, -0.07342, 0.66295, -0.27326, 1.49859, -0.66633
4.507596159999999, -0.50087, 0.61252, -0.80248, -1.70264, -0.12438, 0.58044, -0.39677, 1.53970, -0.75713
4.507596479999999, -0.61401, 0.72540, -0.91508, -1.80120, -0.17534, 0.49793, -0.51000, 1.56539, -0.83279
4.507596800000000, -0.71687, 0.82802, -1.01232, -1.88420, -0.22629, 0.41541, -0.62323, 1.57567, -0.89332
4.507597120000000, -0.81459, 0.92038, -1.09933, -1.95163, -0.27215, 0.33290, -0.72616, 1.56025, -0.93872
4.507597440000000, -0.89687, 0.99735, -1.17098, -2.00351, -0.31292, 0.26070, -0.82395, 1.52942, -0.96898
4.507597759999999, -0.97402, 1.05379, -1.22728, -2.02945, -0.35368, 0.19881, -0.91144, 1.48831, -0.97907
4.507598079999999, -1.03573, 1.09997, -1.26823, -2.03982, -0.38935, 0.14208, -0.98349, 1.43179, -0.97403
4.507598400000000, -1.07688, 1.13076, -1.28870, -2.02426, -0.41483, 0.10598, -1.04011, 1.35984, -0.94881
4.507598720000000, -1.10259, 1.14102, -1.28870, -1.99313, -0.44031, 0.07504, -1.08643, 1.28790, -0.91350
4.507599040000000, -1.11288, 1.13076, -1.26823, -1.94126, -0.45050, 0.06473, -1.11216, 1.20568, -0.85801
4.507599359999999, -1.10259, 1.11023, -1.22216, -1.87382, -0.46069, 0.05957, -1.11730, 1.12346, -0.78739
4.507599679999999, -1.07173, 1.06405, -1.16075, -1.79601, -0.46069, 0.06988, -1.10701, 1.03096, -0.70669
4.507600000000000, -1.02030, 1.00761, -1.08398, -1.70264, -0.45050, 0.09051, -1.08128, 0.94360, -0.61085
4.507600320000000, -0.95345, 0.93577, -0.98673, -1.60407, -0.43012, 0.11630, -1.02981, 0.85110, -0.51501
4.507600640000000, -0.87116, 0.84855, -0.87925, -1.49514, -0.40464, 0.14724, -0.96291, 0.75861, -0.41917
4.507600960000000, -0.77344, 0.75619, -0.76666, -1.38101, -0.36897, 0.18334, -0.88571, 0.66611, -0.32333
4.507601279999999, -0.67059, 0.65356, -0.64382, -1.26689, -0.33330, 0.21428, -0.78792, 0.57361, -0.23253
```

CSV text file; can be over 50 GB

compressed format

- binary “.tbin” file*
- 16-bit scaled integers
- self-describing header
- utility program to convert

10 to 1 space compression

2x faster processing

*defined with CHM software curator Al Kossow

7000 lines of C

```
void ww_assemble_data(void) { // assemble the array of 2-bit characters into bytes
   struct results_t *result = &block.results[block.parmset]; // where we put the results of this decoding
   uint16_t temp_data[MAXBLOCK + 1];
   int outndx = 0, nibble_counter = 0;
   uint16_t accum;
   // special hack: if there is one more clock than a multiple of 8, then assume the first clock is noise
   // and discard the first two bits we derived from it.
   if (ww.datacount % 8 == 1 && ww.datacount >= 9) {
      for (int ndx = 0; ndx < ww.datacount - 1; ++ndx)
         data[ndx] = data[ndx + 1];
      --ww.datacount;
      result->ww_leading_clock = 1; }
   if (reverse_tape) { // assemble going backwards, into a temp array
      for (int inndx = ww.datacount - 1; inndx >= 0; --inndx) {
         //accum = (accum >> 2) | ((data[inndx] & 0x03) << 6); // shift in 2 more bits, least significant first
         accum = (accum << 2) | (data[inndx] & 0x03);   // shift in 2 more bits, most significant first
         if (++nibble_counter % 4 == 0) { // dump a full byte
            temp_data[outndx++] = (accum & 0xff) << 1; } } // create dummy parity bit on the right side
      for (int inndx = 0; inndx < outndx; ++inndx) // copy the temp array into the final data result
         data[inndx] = temp_data[inndx]; }
   else // assemble in place going forward, since we're making it smaller by 4x
      for (int inndx = 0; inndx < ww.datacount; ++inndx) {
         accum = (accum << 2) | (data[inndx] & 0x03);   // shift in 2 more bits, most significant first
         if (++nibble_counter % 4 == 0) { // dump a full byte
            data[outndx++] = (accum & 0xff) << 1; } } // create dummy parity bit on the right side
   result->minbits = result->maxbits = outndx;
   if (ww.datacount % 8 != 0) { // should be a multiple of 16 bits, or 8 2-bit characters
      ++result->ww_bad_length;
      if (!doing_deskew && ww.datacount > 8 ) rlog("  *** the datacount for the next block is %d 2-bit characters, "
               "which is %d more than a multiple of 8\n",
               ww.datacount, ww.datacount % 8); }
   float target_bitspace = 1 / (bpi*ips);
   if (fabs(ww.clkavg.t_bitspaceavg - target_bitspace) / target_bitspace > WW_MAX_CLK_VARIATION) ++result->ww_speed_err; }
```

github.com/LenShustek/readtape

No GUI, but lots of options

```
-ntrks=n        set the number of tracks
-order=         set input data order for tracks 0..ntrks-2,P, where 0=MSB
                default: 01234567P for 9 trk, 012345P for 7 trk
                (for Whirlwind: a combination of C L M c l m and x's)
-pe             PE (phase encoding)
-nrzi           NRZI (non return to zero inverted)
-gcr            GCR (group coded recording)
-whirlwind      Whirlwind I 6-track 2-bit-per-character
-ips=n          speed in inches/sec (default: 50, except 25 for GCR)
-bpi=n          density in bits/inch (default: autodetect)
-zeros          base decoding on zero crossings instead of peaks
-even           expect even parity instead of odd (for 7-track NRZI BCD tapes)
-invert         invert the data so positive peaks are negative and vice versa
-fluxdir=d      flux direction is 'pos', 'neg', or 'auto' for each block
-reverse        reverse bits in a word and words in a block (Whirlwind only)
-skip=n         skip the first n samples
-blklimit=n     stop after n blocks
-subsample=n    use only every nth data sample
-showibg=n      report on interblock gaps greater than n milliseconds
-tap            create one SIMH .tap file from all the data
-deskew         do NRZI track deskewing based on the beginning data
-skew=n,n       use this skew, in #samples for each track, rather than deducing it
-correct        do error correction, where feasible
-addparity      include the parity bit as the highest bit in the data (for ntrks<9)
-tbin           only look for a .tbin input file, not .csv first
-nolog          don't create a log file
-nolabels       don't try to decode IBM standard tape labels
-textfile       create an interpreted .<options>.txt file from the data
                  numeric options: -hex -octal (bytes) -octal2 (16-bit words)
                  character options: -ASCII -EBCDIC -BCD -sixbit -B5500 -SDS -SDSM -flexo
                  characters per line: -linesize=nn
                  space every n bytes of data: -dataspace=n
                  make LF or CR start a new line: -linefeed
-outf=bbb       use bbb as the <basefilename> for output files
-outp=ppp       otherwise use ppp as an optional prepended path for output files
-sumt=sss       append a text summary of results to text file sss
-sumc=ccc       append a CSV summary of results to text file ccc
-m              try multiple ways to decode a block
-nm             don't try multiple ways to decode a block
-v[n]           verbose mode [level n, default is 1]
-q              quiet mode (only say "ok" or "bad")
-f              take a file list from <basefilename>.txt
```

decode each block with different parameters until error-free

```
reading parmsets from "PE.parms"

 clk_window,  clk_alpha, agc_window, agc_alpha, min_peak, clk_factor,  pulse_adj, pkww_bitfrac
{      0,      0.200,         5,       0.000,      0.000,      1.500,      0.400,       0.700, },
{      0,      0.200,         5,       0.000,      0.100,      1.500,      0.400,       0.700, },
{      3,      0.000,         5,       0.000,      0.000,      1.400,      0.000,       0.700, },
{      3,      0.000,         5,       0.000,      0.000,      1.400,      0.200,       0.700, },
{      5,      0.000,         5,       0.000,      0.000,      1.400,      0.000,       0.700, },
{      5,      0.000,         5,       0.000,      0.000,      1.500,      0.200,       0.700, },
{      5,      0.000,         5,       0.000,      0.000,      1.400,      0.400,       0.700, },
{      3,      0.000,         5,       0.000,      0.000,      1.400,      0.200,       0.700, }
```

slow!

.csv file

.logicdata file

csvtbin

.tbin

readtape

speed on a laptop: about real time

.bin

.tap

Supnik SIMH format

.log

.txt

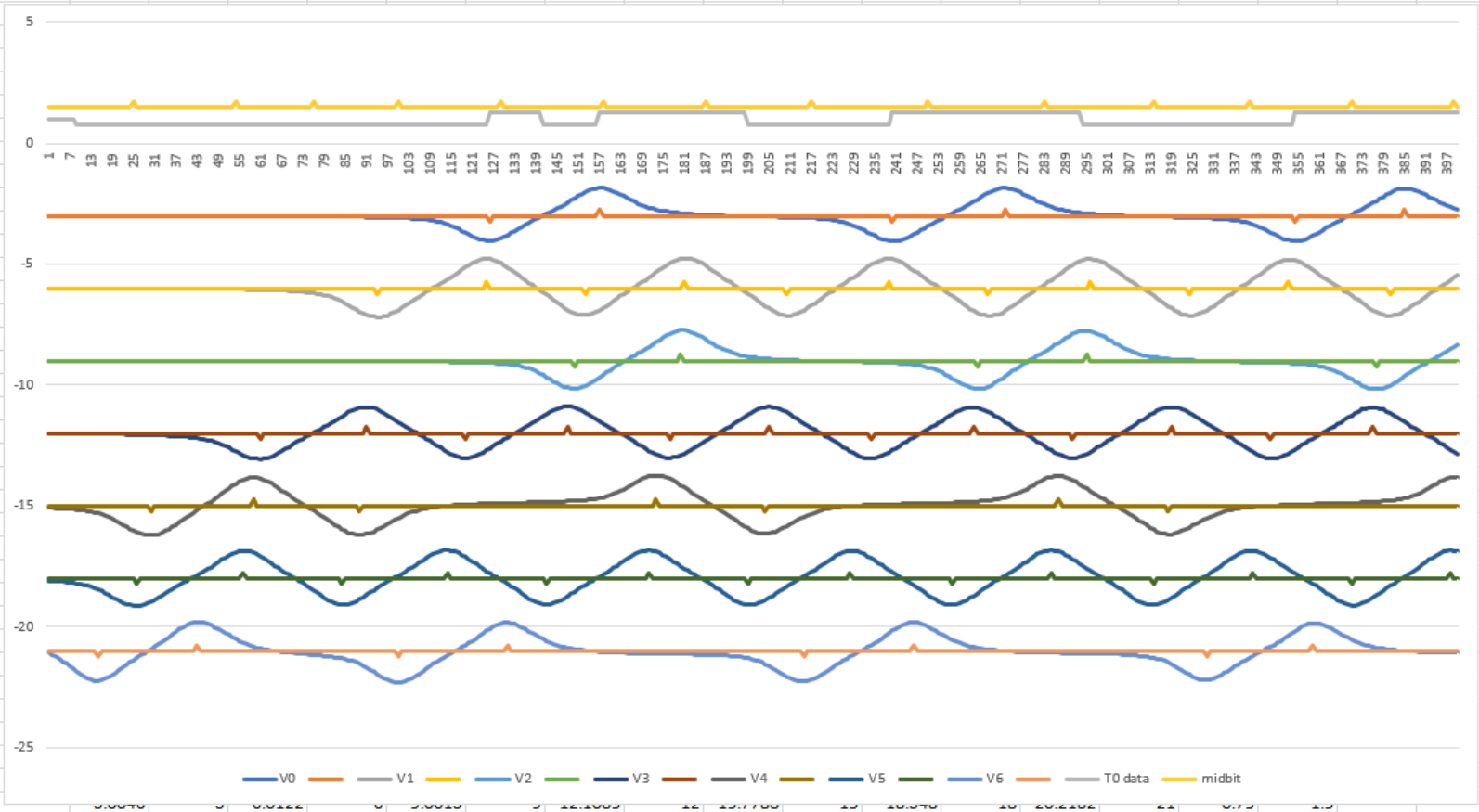
.csv

for debugging

dataflow

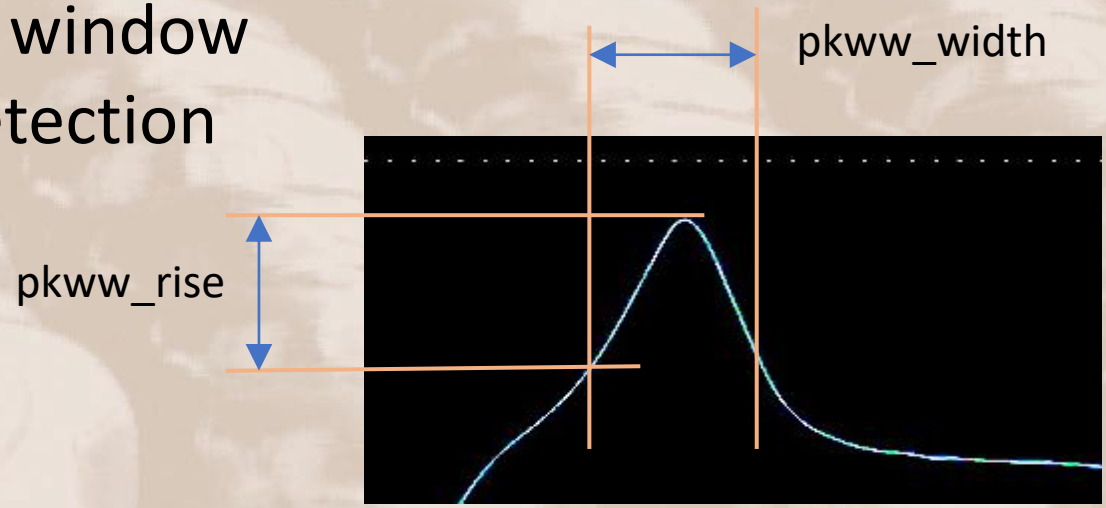
debugging

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	time		V0		V1		V2		V3		V4		V5		V6		T0 data	midbit		T0 datacc	T0 AGC ga	T0 bitspace
2	313.732		-3.0087	-3	-6.0163	-6	-9.0064	-9	-12.0085	-12	-15.0757	-15	-18.095	-18	-21.072	-21	1	1.5		1	1	31.25
3	313.732		-3.0087	-3	-6.0174	-6	-9.0075	-9	-12.0085	-12	-15.0829	-15	-18.1042	-18	-21.1531	-21	1	1.5		1	1	31.25
4	313.732																			1	1	31.25
5	313.732																			1	1	31.25
6	313.732																			1	1	31.25
7	313.732																			1	1	31.25
8	313.732																			1	1	31.25
9	313.732																			1	1	31.25
10	313.732																			1	1	31.25
11	313.732																			1	1	31.25
12	313.732																			1	1	31.25
13	313.732																			1	1	31.25
14	313.732																			1	1	31.25
15	313.732																			1	1	31.25
16	313.732																			1	1	31.25
17	313.732																			1	1	31.25
18	313.732																			1	1	31.25
19	313.732																			1	1	31.25
20	313.732																			1	1	31.25
21	313.732																			1	1	31.25
22	313.732																			1	1	31.25
23	313.732																			1	1	31.25
24	313.732																			1	1	31.25
25	313.732																			1	1	31.25
26	313.732																			1	1	31.25
27	313.732																			2	1	31.25
28	313.732																			2	1	31.25
29	313.732																			2	1	31.25
30	313.732																			2	1	31.25
31	313.732																			2	1	31.25
32	313.732																			2	1	31.25
33	313.732																			2	1	31.25
34	313.732																			2	1	31.25
35	313.732																			2	1	31.25
36	313.732																			2	1	31.25
37	313.732																			2	1	31.25
38	313.732																			2	1	31.25
39	313.732		-3.0036	-3	-6.0132	-6	-9.0013	-9	-12.1176	-12	-15.6783	-15	-18.2627	-18	-20.1124	-21	0.75	1.5		2	1	31.25
40	313.732		-3.0025	-3	-6.0143	-6	-9.0013	-9	-12.1278	-12	-15.5769	-15	-18.1814	-18	-20.0159	-21	0.75	1.5		2	1	31.25
41	313.732		-3.0025	-3	-6.0143	-6	-9.0023	-9	-12.1401	-12	-15.4754	-15	-18.1011	-18	-19.9316	-21	0.75	1.5		2	1	31.25
42	313.732		-3.0015	-3	-6.0143	-6	-9.0033	-9	-12.1533	-12	-15.3749	-15	-18.023	-18	-19.8659	-21	0.75	1.5		2	1	31.25
43	313.732		-3.0025	-3	-6.0153	-6	-9.0033	-9	-12.1687	-12	-15.2766	-15	-17.9458	-18	-19.8217	-21	0.75	1.5		2	1	31.25

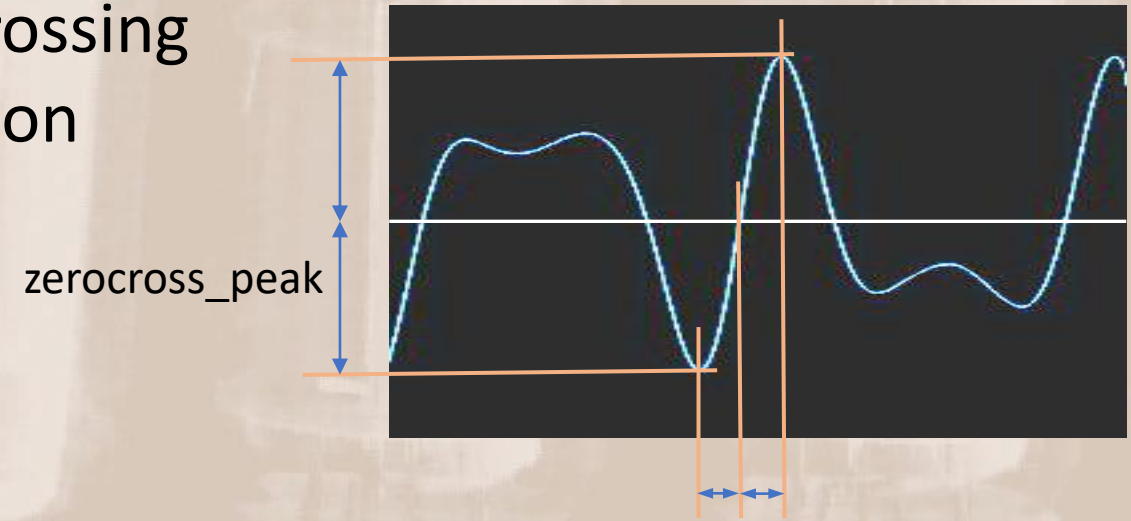


detecting transitions

option 1: moving window peak detection

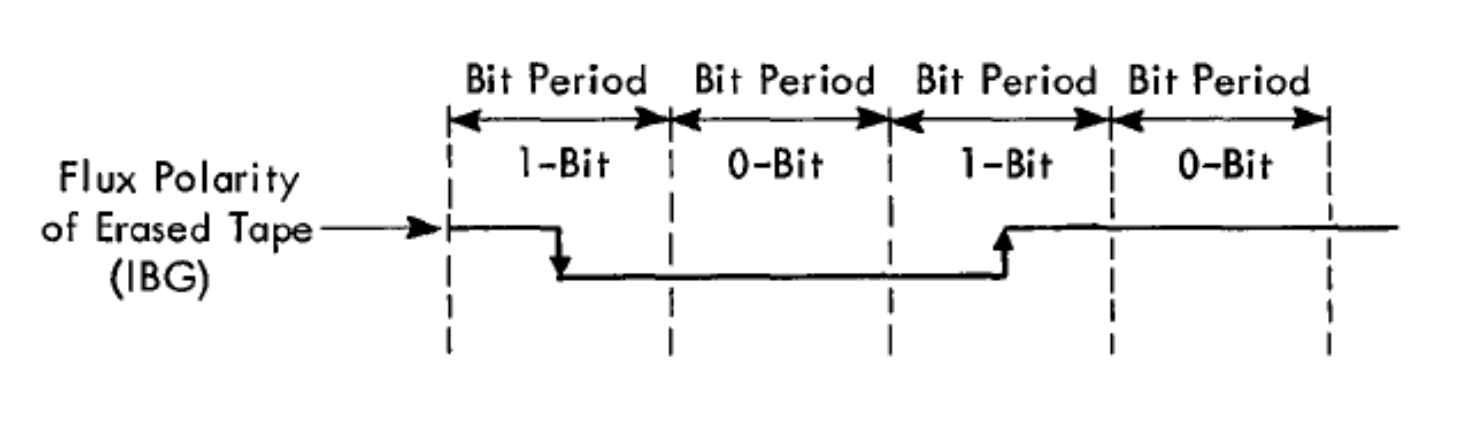


option 2: zero-crossing detection



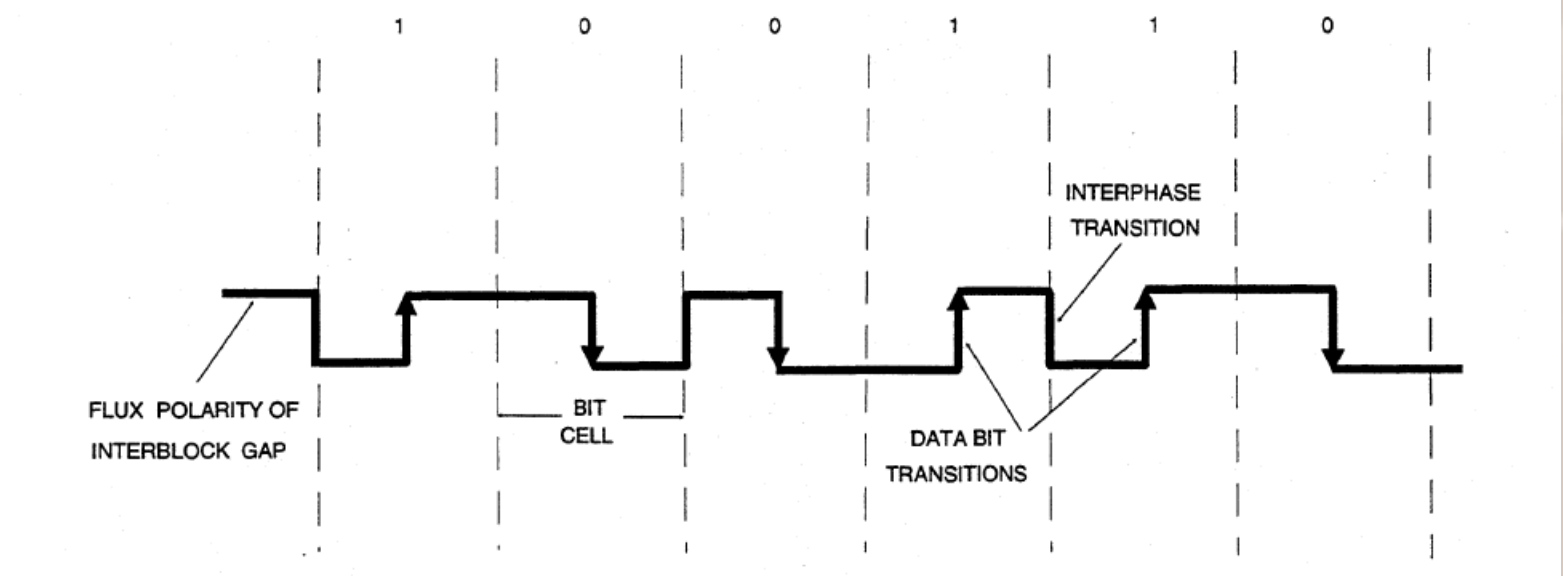
decoding the bits

NRZI: Non return to Zero Inverted



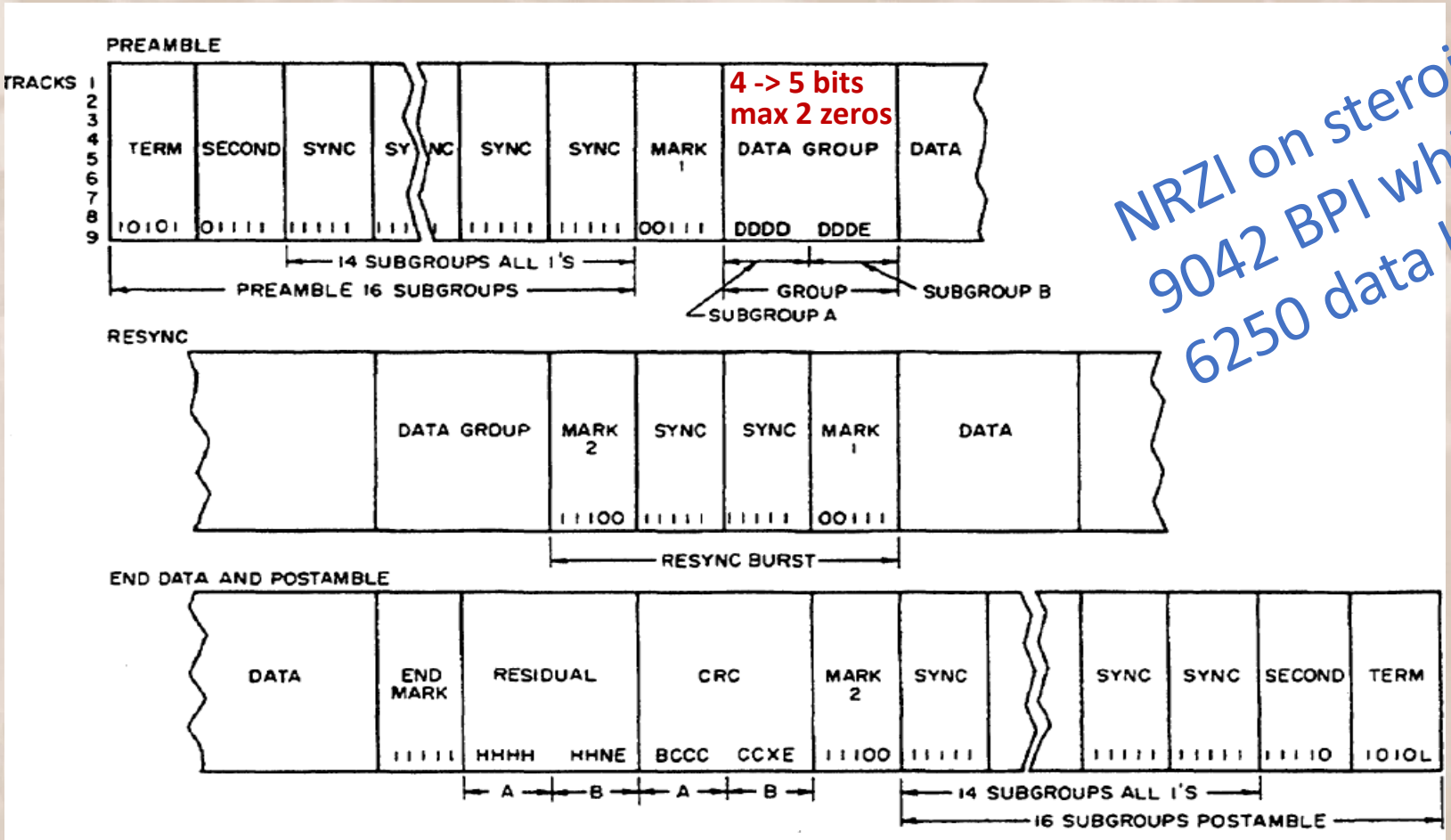
- used for 7-track, 9-track 800 BPI (and 6250 BPI GCR), and Whirlwind tapes
- NOT self-clocking – requires odd parity or group recoding
- often: odd parity, LRC and CRC block check characters
- sometimes: preamble of 40 zeroes followed by 1
- sometimes: filemark is 64-256 flux reversals on tracks 2, 5, 8 only

decoding the bits
PE: Phase Encoding



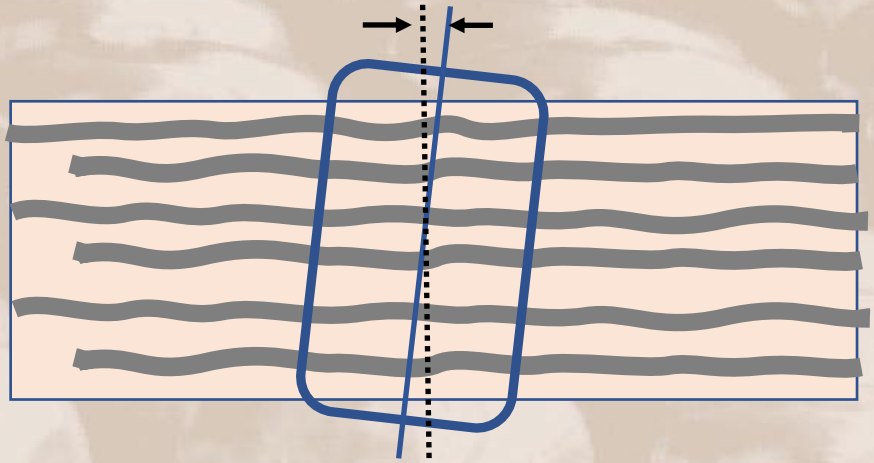
- used for 9-track 1600 BPI drives
- self-clocking
- preamble of 40 zeroes followed by 1
- odd parity, no longitudinal block checks
- filemark: 64-256 flux reversals on tracks 2, 5, 8 only

decoding the bits
GCR: Group Coded Recording

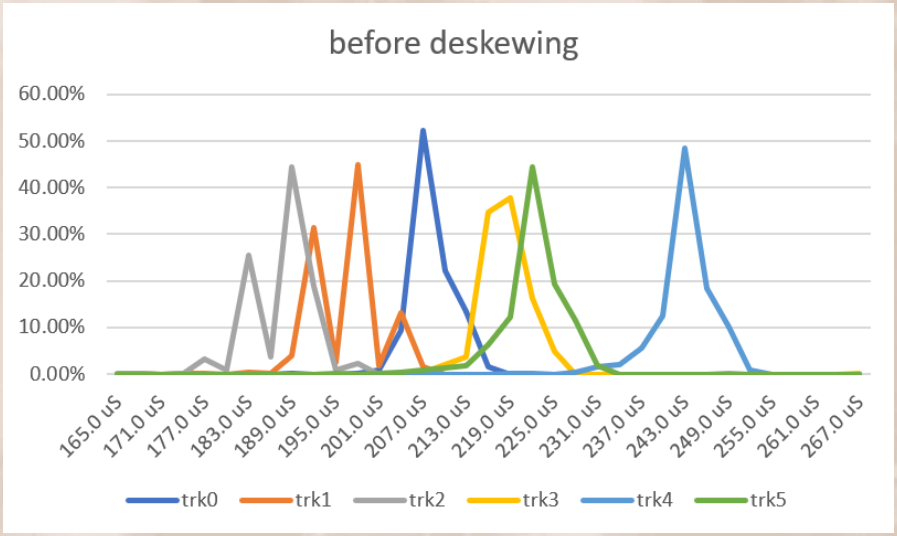


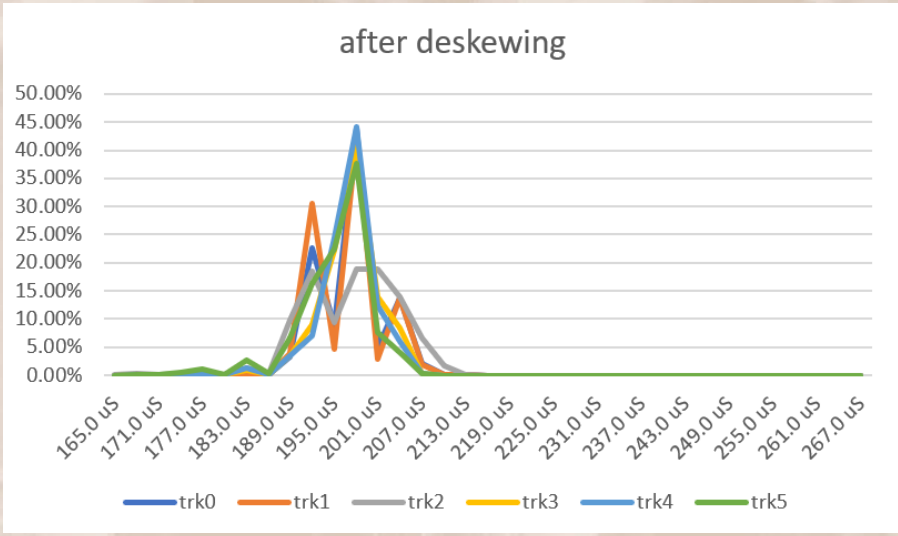
NRZI on steroids!
9042 BPI which yield
6250 data bits/inch

fixing head skew



pre-read 1000 transitions to determine head skew, then delay track data



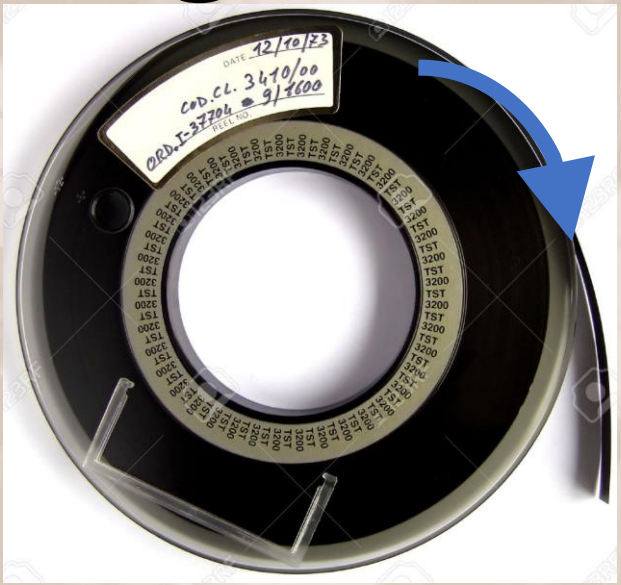


fixing amplitude changes

AGC (Automatic Gain Control) based on:

- average height of last N peaks, or
- exponential weighted average of previous peaks
  gain = α x (height/avgheight) + (1 - α) x gain

fixing speed changes



Initial bit clock estimate is:

1 / ((bits/inch) * inches/sec))

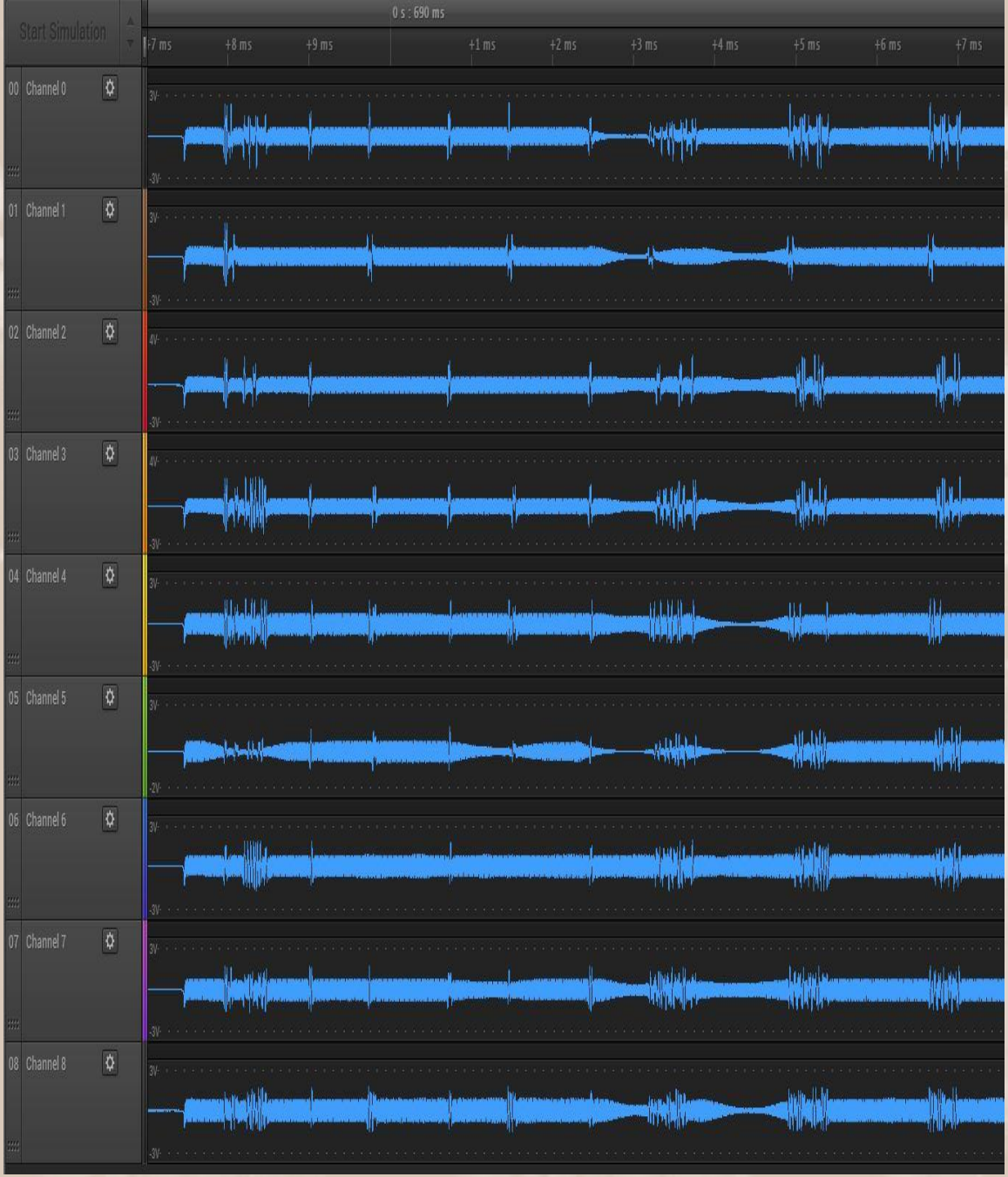
Update it after each bit to:

- same
- average spacing of last N bits, or
- exponential weighted average of previous bit spacings

clock = α x last_bit_spacing + (1 - α) x clock

fixing errors

- deduce the bits from parity, or
- correct the bits from ECC codes, or
- during dropout, fake “same as before” bits



this block decodes correctly!

result: PLAGO 100% recovered

```
reading file "plago_whole_02.csv" on Wed Mar 28 21:22:57 2018
*** tape label VOL1: PLAGOS, owner  P.I.B.
*** tape label HDR1: DOCUMENT         , serno PLAGOS, created 70181
    volume 0001, dataset 0001
creating file "plago_whole_02\000-DOCUMENT.bin"
*** tape label HDR2: RECFM=FB, BLKSIZE=00800, LRECL=00080
    job: LENNY   /STEP1
*** tapemark
wrote block   1,  800 bytes, 1 tries, parms 0, max AGC 1.35, 0 parity errs, CRC ok , LRC ok , avg speed 50.10 IPS, at time 3.5078912
wrote block   2,  800 bytes, 1 tries, parms 0, max AGC 1.37, 0 parity errs, CRC ok , LRC ok , avg speed 49.69 IPS, at time 3.5398886
wrote block   3,  800 bytes, 1 tries, parms 0, max AGC 1.25, 0 parity errs, CRC ok , LRC ok , avg speed 49.51 IPS, at time 3.5722214
wrote block   4,  800 bytes, 1 tries, parms 0, max AGC 1.32, 0 parity errs, CRC ok , LRC ok , avg speed 50.12 IPS, at time 3.6038989…
```

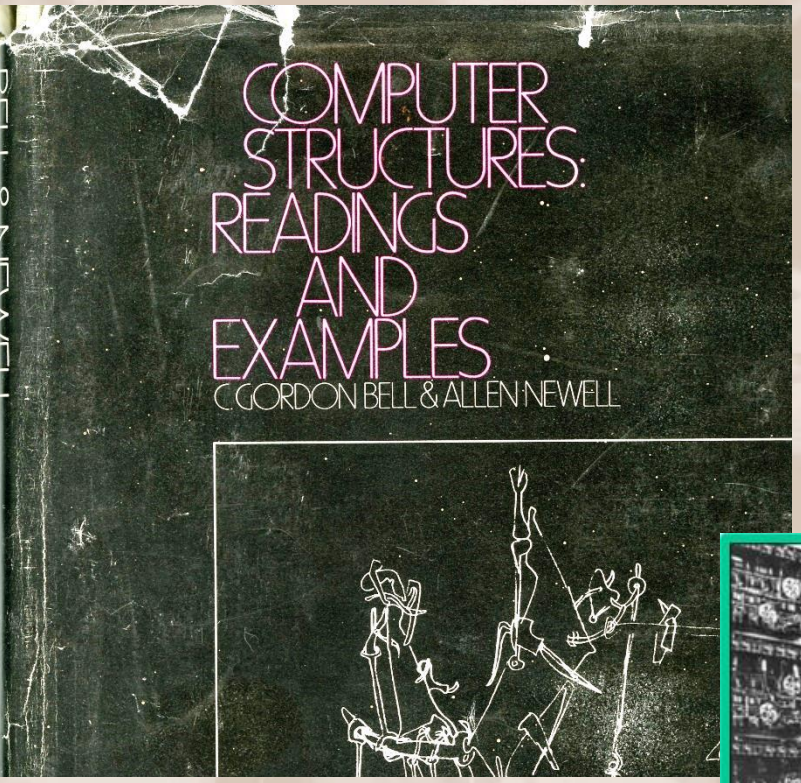
```
*********************************************************************** 00081400
*                                                                       00081500
******** FSB   FETCH SUBSCRIPTED REFERENCE ***************************** 00081600
*                                                                       00081700
*********************************************************************** 00081800
         SPACE                                                          00081900
EFSB     LH    TR4,0(LOCNTR)           OFFSET TO ID ENTRY               00082000
         A     TR4,IDBASE              ABS ADDR OF ID ENTRY             00082100
         LA    TR13,ENEXT              SET RETURN ADDRESS               00082200
         LH    TR5,2(LOCNTR)           GET NO. OF SUBSCRIPTS            00082300
         LA    LOCNTR,4(LOCNTR)                                         00082400
EFSB1    DS    0H    ENTER HERE FROM GTD RTN, TR5=NO. OF SUBS, TR4=ID   00082500
         LH    TR3,10(TR4)             OFFSET OF DV IN VSTK             00082600
         L     TR2,VBLKTBL             NO - GET ADDRESS OF BLOCK TABLE  00082700
         A     TR3,0(TR2,EOPERND)      ADD PROPER VSTK BASE             00082800
         TM    8(TR4),PARAM            IS THIS A PARAMTTER              00082900
         BZ    EFSB9                   NO...                            00083000
         A     TR3,0(,TR3)             YES-GET PTR TO ARGUMENT          00083100
         BNP   ERR22                   BR IF ARG WAS NOT PASSED         00083200
EFSB9    L     TR2,0(,TR3)             LOAD VIRTUAL ORIGIN              00083300
         LA    TR0,0(TR2)              * INTO ACCUMULATOR (TR0)         00083400
         BCTR  TR5,0                   SET TR5 = 4 * (NO. OF SUBS - 1)  00083500
         SLA   TR5,2                   *                                00083600
         SR    DSTK,TR5                PURGE SUBS, BUT LEAV A SLOT      00083700
         LR    TR2,DSTK                POINT TO FIRST SUBSCRIPT         00083800
         ST    CVSTK,SAV1              SAVE CURRENT VSTK POINTER        00083900
```

other tapes decoded

- Englebart NLS (SDS-940)
- SRI PDP-10
- Microdata Reality OS
- Lawrence Berkeley Lab
- SEL realtime OS
- Project Genie timesharing (UCB)
- BBN tapes
- etc.

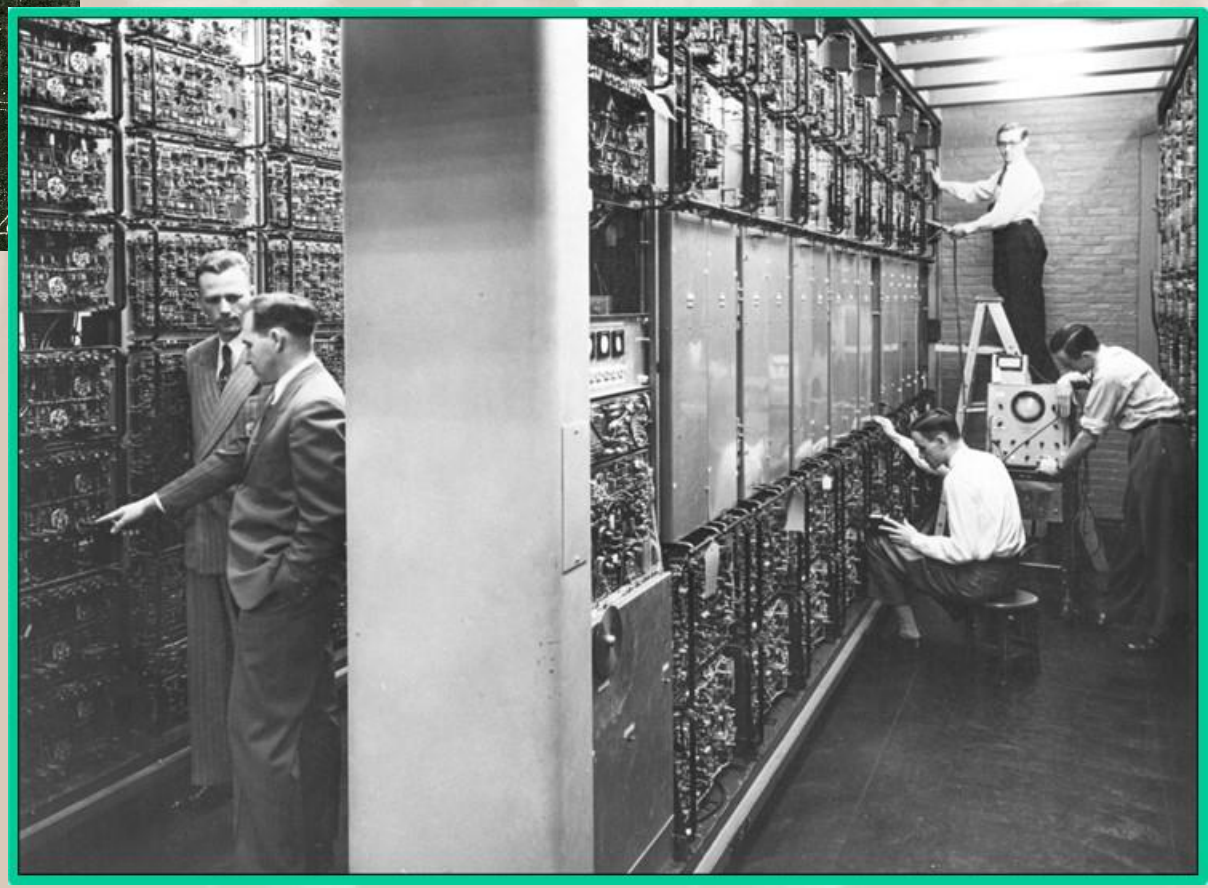
why do this?

- recover lost knowledge about early computers and software
- show techniques for digital archaeology
- provide authentic programs and data for restorations and simulations



case study: Whirlwind

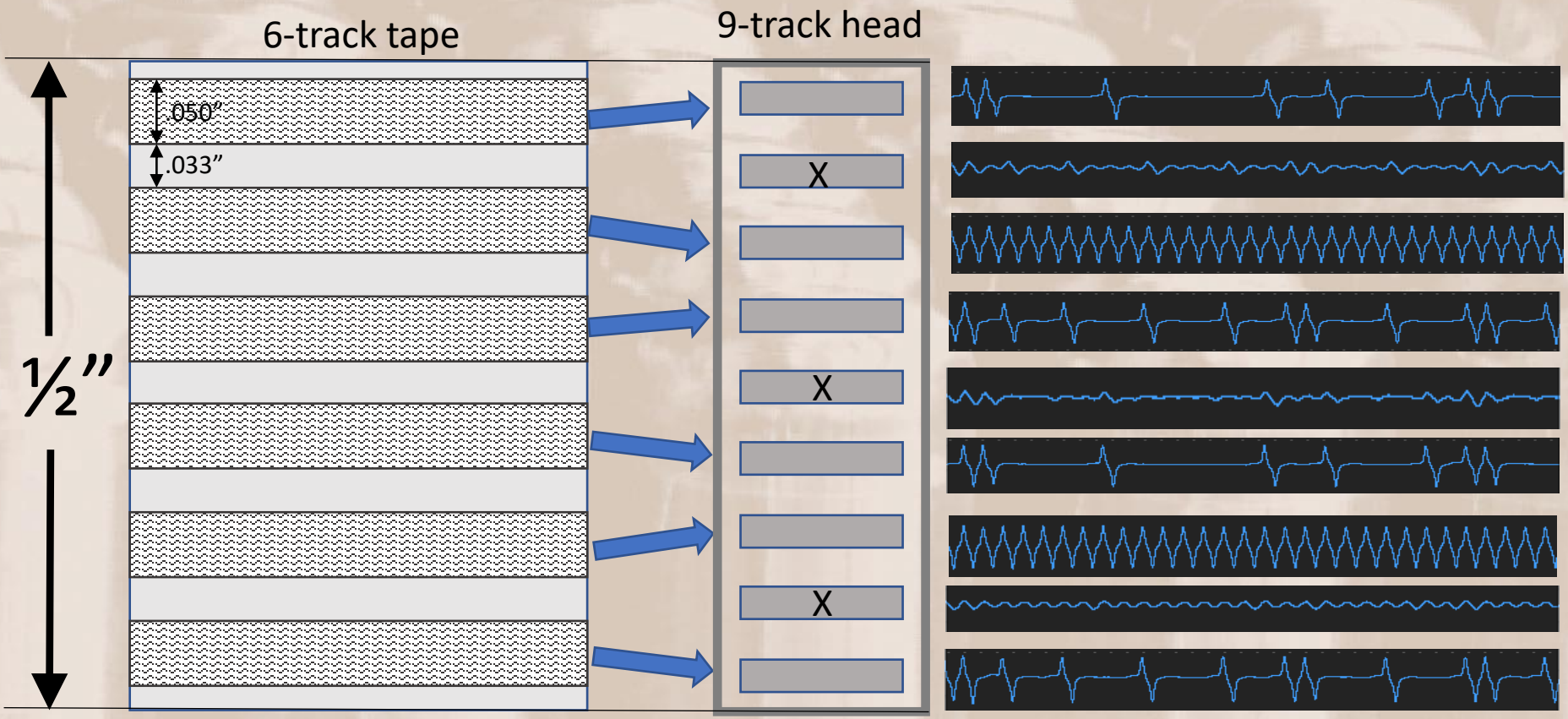
(the inspiration for CHM?)



Gordon Bell: Somebody should preserve the historic computers!

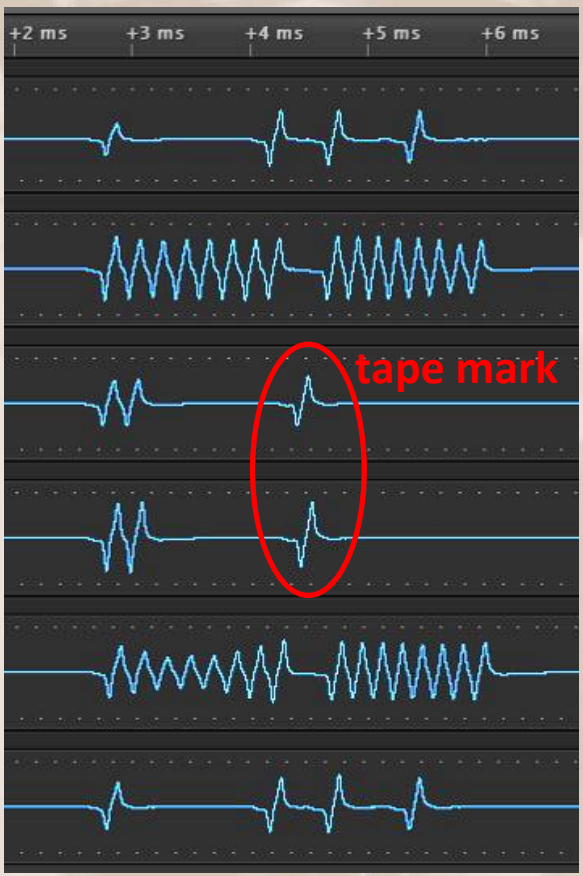
40+ years later, the 150 6-track Whirlwind tapes were still unread

6 track head -- NOT



the head misalignment is always only 0.014”
from the center of the 0.050” wide track

decoding Whirlwind bits



MSB

clock

LSB

LSB (copy)

clock (copy)

MSB (copy)

- basically NRZI
- 2 bits/character assembled into 16-bit words
- each track is duplicated
- no parity or other checks
- the flux transition polarity varies from tape to tape
- tape mark: LSB bits without a clock

dataflow

results

- decoded 61 tapes so far
  - about half the inventory
- 161,955 blocks
  - 0.13% had errors
- 10,274,572 bytes
- data and source code
  - very little text ☹

stay tuned for Guy's talk about using the bits

questions?

shustek@computerhistory.org