A Coherent Real Time Wind System
(description and operating manual)

Report No. 3 1989

MAY 1 2 1993

C.E.Meek (Mar.29'1989)

MAY 1 2 1993

(programs updated to ~~Feb22'90~~)

APR 1 1 1990
JUL 23 1990
DEC 1 3 1990

full update: Jan4'91 JAN 2 4 1992
MAY 1 2 1993

Institute of Space and Atmospheric Studies
University of Saskatchewan
Saskatoon, Sask. S7N 0W0
CANADA

INDEX (W=write-up,F=figure,L=listing)

Introduction: [page 2]

Section I.: General information
- Block diagram of COHRTW system (F) [page 3]
- Overview of system operation (W) [page 5]
- Description of CPU#2 screen display (W) [page 9]
- Illustration of CPU#2 screen display (F) [page 8]
- Doppler and AOA sign conventions and meaning (W) [page 11]
- Description of tape record (W) [page 13]
- Example of tape record (F) [page 16]
- Definition of complex correlation (W) [page 17]
- COHRTW operating instructions (W) [page 19]
- CPU#1 LED panel (F) [page 22]
- CPU#2 status byte description (F) [page 22]

Section II.: CPU#1 (Rx interface control)
- CPU#1 detailed description (W) [page 23]
- Subroutine list (L) [page 27]
- Port address list (L) [page 28]
- Zero-page memory use (L) [page 29]
- Bulk RAM map (F) [page 30]
- LTM store (F) [page 31]
- I/O locations (CPU#1 <=> CPU#2 link) (F) [page 32]
- ROM storage (tables, program, etc.) (F) [page 33]
- CPU#1 program annotated listing (L) [page 35]

Section III.: CPU#2 (correlation calculation)
- CPU#2 detailed description (W) [page 55]
- Subroutine list (L) [page 57]
- Port (PIA) address list (L) [page 58]
- Addresses for cross and auto storage vs. height (L) [page 59]
- Zero-page usage (flags, counters, etc.) (L) [page 60]
- Zero-page temporary bit-amplitude storage details (F) [page 61]
- Bulk storage map ($5EB0-BFFF) (F) [page 62]
- I/O memory used in CPU#1 <=> CPU#2 link (F) [page 63]
- Communication link CPU#2 <=> CPU#1 (F) [page 64]
- 5-min delay tripout on stuck system circuit (F) [page 64]
- Program listing (TOTALCPU#2W-D/N) (L) [page 65]

Section IV.: C128 (CPU#3) details
- C128 analysis (W) [page 103]
- C128 machine language (W) [page 107]
- BASIC program, PMFCA wind analysis (L) [page 109]
- C128 bulk storage (F) [page 115]
- Machine language program (L) [page 116]
- Interface connection between C128 and Apple (CPU#2) (F) [page 118]
- C128 printer interface circuit and connections (F) [page 118]

Section V.: Conversion to 3-antenna array +odds&ends [page 119-141]
- General information (W) [page 119]
- List of mods required (L) [page 123]
- Listing of modifed sections of program (L) [page 125]
- Sample of use of "Load and Go" feature (1) [page 139]
- Raw data port specs (W) [page 141]
- Raw data o/p description (F) [Page 143]
- Single card CPU:CPU#1 memory allocation (F) [Page 144]
- Collected upgrades to CNSR & Tromso: May'93 (Pg. 161-170)

---------- some page numbers left for expansion ----------

Section VI.: Pertinent extracts from 1979 RTW report. [page 219-243]

Includes derivation of the required correction to correlation values for the use of bit amplitudes, and derivation of the PMFCA analysis. This system used signal MAGNITUDES (i.e. non-coherent receiver), but most of the material applies directly to the COHRTW system. Also a 6502 machine language reference chart is appended.

Introduction

This report consists of many separate un-numbered notes and figures rather than a coherent write-up. The index is the best way to locate a given item. The printing is done so that each section in begins with a coloured divider page which contains the index for the section. Each "note" in a section starts on a right-hand page. Figures and tables occur on either side. There are no circuit diagrams (except for the C128 printer interface and watchdog circuit). The critical circuits (i.e. A/D and receiver gain) are an extension of the original RTW system Rx interface design, but with 8 channels instead of 1. Some of the circuits, e.g. Tx control, synch rate selection, Tape drive reset, were "tacked" together from existing bare boards and are simple enough that a diagram is of minimal value.

The RX interface design may be put into a separate report in the future.

Jan 4'91:

The system has evolved over the years since the original version of this report, most changes are connected with the interface computer (CPU#1). This started as an Apple II with a cable going to the interface and a loaded program; then an Apple II with an "auto-start" EPROM; then a single board CPU (2 MHz 6502). Each change affected some addresses, viz. port addresses and RAM addresses (apart from program modifications), which were usually optionally noted in the original report. Some "errors" still remain intentionally. One is the CPU#1 program listing - it cannot be listed where it actually resides ($F000-FFFF) because I didn't use an assembler. What is listed is the Apple II EPROM, moved to $2000-. Thus the port adresses are as for the Apple, not the single card CPU, however the RAM and EPROM addesses *in* the program are the same. I have corrected all memory maps (I hope) which previouly showed the "loaded Apple II" program addresses.

Some write-ups and figures have been modified to conform to the present system. Also a section on program conversion to a 3-antenna Rx array is added. Surprisingly few mods were needed.

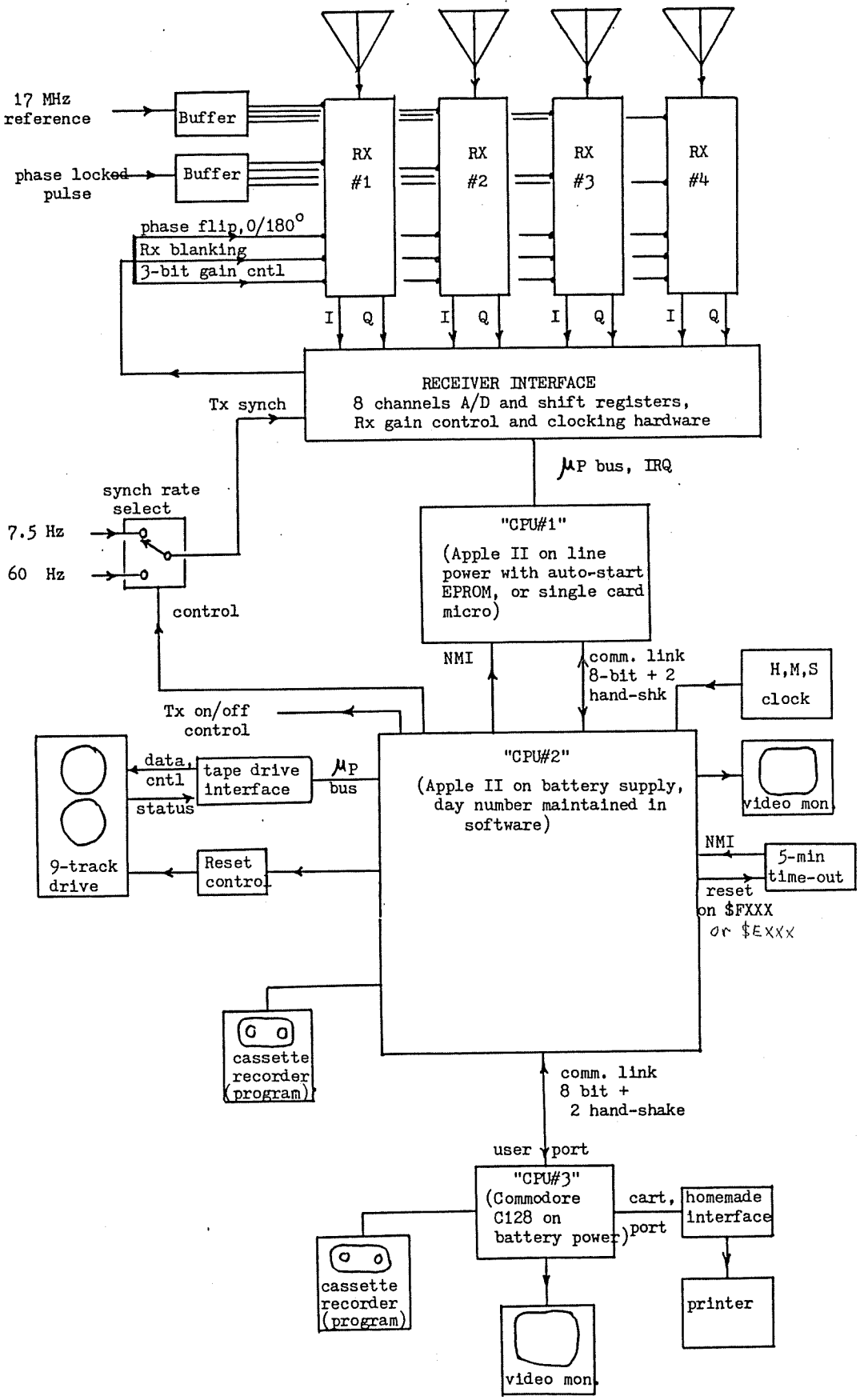
CM, Mar.29'89
Jan. 4'91

SECTION I.

SECTION I.

Section I.: General information

Block diagram of COHRTW system (F) [page 3]
Overview of system operation (W) [page 5]
Description of CPU#2 screen display (W) [page 7]
Illustration of CPU#2 screen display (F) [page 10]
Doppler and AOA sign conventions and meaning (W) [page 11]
Description of tape record (W) [page 13]
Example of tape record (F) [page 16]
Definition of complex correlation (W) [page 17]
COHRTW operating instructions (W) [page 19]
CPU#1 LED panel (F) [page 22]
CPU#2 status byte description (F) [page 22]



COHERENT REAL TIME WIND SYSTEM

Oct13'86

Coherent RTW System

Overview of system

The system contains three computers : the interface computer (#1- an Apple, or single board 2MHz 6502 system), the main control computer (#2- an Apple), and the numerical analysis computer (#3- a C128).

Computer#1/CPU#1(Apple):

The first runs the receiver interface hardware- i.e. receives command data from #2, sets the reference oscillator phase (0/180°) before each pulse, sends the present gains to the interface, controls the clocking of height gates, takes amplitude data from the interface, does A/D bit on/off checks for the 8 channels, does coherent integration and averages, accumulates long term running means (LTM's) for estimating receiver 'zero' or offset (which is nominally +5 volts, or 128 decimal), calculates bit amplitudes by comparing the coherent average with the appropriate LTM ('1' if above the LTM, '0' if below), and rolls these bits into an output store. When 8 bits are accumulated (i.e. 8 coherent averages, which takes 4.26 sec), the data (256 bytes = 32 hts x 4 Rcvrs x 2 components x 8 bits) are sent to computer #2 along with 8 bytes of status information (including bit check status, error flags etc).

In addition, #1 keeps track of mean signal magnitude vs height, and relative I and Q amplitudes at a selected height (to warn of differential component amplifier gain drift)

After a given number of sends has been completed (i.e. after a record, approx 4.5 min), #1 sends final parameters (including mean signal magnitude vs height, LTM's, the selected height, and I/Q or Q/I amplifier gain ratios for each receiver at this height), and waits for new instructions from #2.

There are 2 modes of operation in #1: local/test and remote. Local operation is initiated by holding down the "local/test" button while pushing reset. The system will then run by itself for a preset length of time (e.g. 4.5 min at 7.5 Hz prf) using preset clock-in parameters, gains etc., and at the end will switch back to remote operation. In Remote operation, all parameters except LTM's are taken from #2 (including the gains for this record, the clock-in parameters the number of sends to do- i.e. the number of groups of 8 coherent averages in the record, the number of pulses to integrate over, whether to do phase flipping with each pulse, etc.).

The LTM's are preset from the ROM when #1 is reset (manually, or on power-up) usually all $80 (equivalent to +5V). They will take 30-40 min to settle down to the actual receiver offsets.

A big advantage in doing immediate conversion to bit amplitudes (as opposed to the present RTW system, which takes the mean out of a 30 second block of data) is that no despiking is necessary. A spike only causes a problem with the bit in which it occurred. It would be useful to use medians (or some other sort of despiking) with each coherent average, however. At the moment this does not seem to be practical.

Computer#2/CPU#2(Apple):

This computer contains the date/time clock, the 9-track tape interface, and communication links to #1 and #3. It sends command data to #1 at the beginning of the record, receives bit amplitudes from #1, and does bit correlations. When the record is finished (by a countdown of the number of sends from #1) takes the final parameters from #1, finishes the correlation calculation, calculates the best gains for the next record from the mean signal magnitudes vs ht, displays all sorts of status messages gleaned from the last #1 info, takes the last analysed results from #3, sends all the correlation data for the record just completed to #3, dumps the analysed data to tape and checks the clock for starting the next record.

Note: If CPU#2 finds that CPU#3 is not ready to accept data, the "1st record counter" is reset; so that the next data from the C128 is thrown away (as if the system were just started)

Computer#3(C128):

This sits waiting for correlation data from #2. When it finds #2 ready, it sends the last analysed data store, and takes the correlation data for the just ended record. Then it sets the "jiffy timer" to zero, and does PMFCA analysis starting with the lowest height. Parameters calculated include the 3 cross phase differences at zero lag (which give mean angle of arrival), the cross correlations at zero lag, the noise factor (i.e. the magnitude of the auto correlation extrapolated to zero lag), the mean vertical Doppler velocity, and the normal PMFCA parameters (as on present RTW output). Record length is 864 bytes (as opposed to 592 bytes on RTW data). The jiffy timer is set to run out just before #2 is ready to send more data. If it runs out during analysis, the rest of the heights are skipped (leaving zeroes in the analysed data store).

Communication links

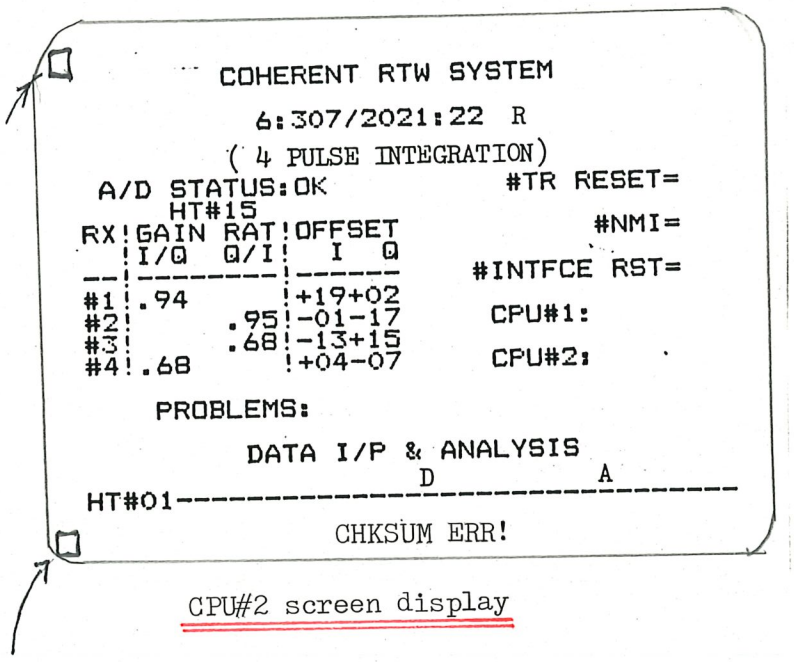
There are 2 two-way communication links: CPU#1 <=> CPU#2, and CPU#2 <=> CPU#3. These each consist of an 8 two-way data lines and 2 one-way handshake lines (one going each way). The handshake lines are programmed to be active on a negative going edge, and are normally high (5 V). Since each computer knows when it should be sending/receiving there is no conflict in the data direction for each end. However, it is possible that both ends of the data 'bus' will be in output mode for a short time while the computers reverse direction- i.e. two outputs hooked together. This is the arrangement used in the present RTW system, and has given no trouble- presumably the PIA lines have a relatively high impedence whether high or low. [If bus buffers were used, this might not be the case].

Since CPU#1 is essentially a 'blind' system - i.e. there is no way it can tell if it is receiving garbage - and since CPU#2 is receiving straight data from CPU#1, which could be any bytes, it is necessary to supply a method by which each computer can check whether the data link is working. This requires a check on each data line. This is similar to the bit-check performed on A/D data, and is done by sending two bytes: $00, and $FF (i.e. all bits off and all bits on) with every transmission. The receiving computer knows where these check bytes should be, and tests them. If CPU#1 finds an error, it lights an LED, and sets a flag in the "CPU#1 STATUS BYTE" going to CPU#2 at the end of every data transmission. CPU#2 displays this status on the screen.

If CPU#2 finds an error it displays the message "COMM#1 ERR" and stops. Note that if there is a link problem (i.e. a broken wire or fried bit in a PIA) the CPU#1 status byte will also be wrong, but the LED and the CPU#2 check should catch any error. [In the case of a broken wire, the receiving computer will find a '1', since the input 'floats high' - this will make CPU#1 status non zero, which will cause CPU#2 to display it as separate bits.]

A similar method is used in the CPU#2 <=> CPU#3 link. Here the check bytes are placed at the end of the date/time store in CPU#2. This store goes to CPU#3 and back unchanged, and the bytes coming back are checked by CPU#2. If there is an error, CPU#2 displays "COMM#3 ERR" and stops.

The date/time store represents the time as one byte per digit (which uses the lower 4 bits only), so an error in the top 4 bits would not be obvious in the CPU#3 display. [This is a potential problem with the present RTW system!]



CPU#2 screen display

aux. err.
msg loc.

CPU#2 Screen Display

This is the main information and debugging display in the system. Important parameters and errors from CPU#1 are passed on to this display. Fig 1. shows an example of the display(on page 8).

Clock line:

gives year(2 digits),day of year (Jan 1= 001), and hours, minutes and seconds in UT. The clock chip is on battery: a full clock set is done by calling the appropriate clock EPROM routine from the machine monitor before starting the program (e.g. if slot 3, then say C3286. See separate clock write-up for more detail). For minor adjustments while the program is running, use "+" or "-" keys. This changes the seconds units only. To the right of the clock is a "mode of operation" character; a flashing "W" means the system is in wait (in this mode ~~the day number is not updated -- and~~ the system does nothing except wait for the next key command); an "R" means a record is in progress; a blank means the system is checking the clock for the next record start.

Amount of integration:

The display is in decimal, and is the number of Tx pulses used to get one (bit) amplitude in CPU#1. This parameter is part of the program (location $BFE-where it must be entered in HEX!). Note that the Tx synch rate and the amount of integration must combine to produce a final coherent average data rate of 7.5/4 Hz. Thus CPU#2 receives data every 8x4/7.5 sec (~4 sec), and the "D" in the DATA ENTRY AND ANALYSIS display should move at this rate.

A/D status:

The A/D status is an indication of whether all bits coming from the A/D converters have been found to be on and off at least once in a record. From the left-most character the channels are I1,Q1,I2,Q2,I3,Q3,I4,Q4. A "?" means all bits for this channel have not been on and off yet in the present record - and the system is still checking them. An "I" or a "Q" means the check is successful, and the A/D is ok in this respect. This display is updated with each data block received from CPU#1 (every 4 sec); so it will start at flashing "????????" (however, the check may be partially/totally successful by the time CPU#2 displays it) and should change to "IQIQIQIQ" quite rapidly (since all 32 heights are checked). CPU#2 does not receive any information on which bit is bad in a particular channel. [If CPU#1 is an Apple with normal keyboard control, this information can be retrieved]. If all channels are ok "OK" appears instead of "IQIQIQIQ".

Left side of display:

This shows an estimate of the gain ratios of I/Q or Q/I (whichever is less than 1) in each receiver (#1 to #4) at a height (HT#) chosen in CPU#1 to give the most accurate value. This is a ratio of the mean absolute values(relative to the estimated offsets) of I and Q over the record. The best height is selected in CPU#1, on the basis that the signal magnitude at this height should be closest to 128 of any height.[The value of the height number is not important for "casual" use].

The display also shows the worst (i.e. furthest from 5V) receiver offsets for height #16 to 23 (8 heights) at the gains in use for the previous record. It is in terms of the difference of the mean value of all "1st pulse in coherent average" receiver amplitudes from 128 (+ve if above 128). If the receivers' offsets have been set up properly and are not drifting, the values should be somewhere in the range -10 to +10. The display routine was not designed to work for values outside -99 to +99; in this case there will be strange characters in the display- which means a very serious offset error! Note that these parameters may take several records from CPU#1 reset to settle down.

Right side:

"#TR RESET"= is the number of times the tape drive had to be reset -AND was reset successfully (blank if none). #NMI gives the number of times CPU#2 had to reset itself through an NMI generated by a time-out circuit; that is, it was stuck in the middle of communication with CPU#1, the C128 or the tape drive. When NMI occurs, CPU#2 status is displayed, showing where it was stuck as follows: 00010000="in tape drive routine (EPROM)", 00001000="in C128 link", 00000100="in interrupt (from CPU#1) receiving the final parameters at end of record", 00000010="in interrupt receiving a data block (in the middle of a record)", 00000001="in the middle of sending initial command data to CPU#1 (at the beginning of a record)".

CPU#1 status is displayed every time CPU#2 receives a data block from CPU#1. It indicates OK if no problems, or a bit pattern if any problems have occurred since the last full reset (not NMI) of CPU#1. [Most of (all) these conditions are shown by the CPU#1 LED display, but only for the present record] Possible patterns are 00000001= data overload (synch too fast for program).....(should be system test bad indic. but not in BK17). #INTFCE RST gives the number of times a power up or reset (which is done automatically on power up, or manually with cntl-reset keys) has occurred in CPU#1: CPU#1 sets an internal flag on reset which it clears just after the first data are sent to CPU#2)

Problems:

This indicates either a serious problem on which the system decides to "stop" - these are COMM#1 error,COMM#3 error (i.e. bad bits in link), DATA OVERLOAD or WRAP AT END (analysis is not fast enough to keep up with incoming data); or a temporary problem C128 NOT READY or DRIVE NOT READY or TAPE PAST EOT where it continues (although the tape dump is not done if the End Of Tape marker has been reached)

1. The drive is checked for "error" after the write routine; if error, then it displays the message and continues as if there were no problem. If no error, it clears the message. Error conditions are: tape off-line/not tensioned or no ring.
2. For C128 not ready, the system displays the message, resets the "1st record flag" -which means that the next data coming back from the C128 will be discarded, and continues. If it is ready, the message is cleared. This means that stopping the C128 for any length of time may cause the loss of at least 2 records. Also note that starting the C128 by RUN after a program change, when the rest of the system has been running for more than 1 record, will/may result in "garbage" going back to CPU#2 -unless you wait for this message to appear and then say RUN!
3. If the tape is at EOT, there will be no write to tape, but all other functions will be normal; so there will be printer output.

Check-sum

At the beginning of every record, a program check-sum is done (900-1FDF only) and compared with a stored byte. If bad, then the program has changed itself, so an error message is written at bottom centre screen. This will remain until the screen is cleared (e.g. by "W", "S"). It indicates a SERIOUS hardware fault in the Apple:CPU#2 (or someone neglected to update the stored check-sum byte when modifying the program). It does not stop the program.

CM Jun9'88

COHRTW: Sign conventions for AOA, Doppler
(compared with other Park equipment)

Define a complex signal as $A(t) = r.e^{i\phi}$ = I+iQ where

$$\phi = \tan^{-1}\left(\frac{Q}{I}\right)$$

(I and Q are the In-phase and Quadrature outputs of the receiver, sometimes called Re & Im or A & B on equipment).

1. COHRTW system:

Cross correlation is in the sense

$$A_4(t) * A_i^*(t + lag)$$

where the subscript is the antenna number (1,2,3,4 = North, South, Main, Centre), * represents complex conjugate, and lag goes from -16 to +15 : the sign of the lag has the same sense as the sign of tmax in the PMFCA analysis. The cross phases (saved for later AOA) are just the values at zero lag of

$$\tan^{-1}(Im/Re)$$

where Im and Re are the components of the complex cross correlation.

Auto correlation is in the sense

$$A_i(t) * A_i^*(t + lag)$$

[Note that ϕ is not necessarily in the sense Tx-Rx, it may be the same magnitude but opposite sign, depending on Rx construction - that is the I and Q might be reversed]

The C128 analysis defines Vz as positive if the auto phase is increasing with lag.

2. Meteor trail:

This is usually done with the prototype coherent receiver, but both built by Rob Strother-Stewart, and sense of I,Q seem to be the same.
Zero lag phase (cross) is found from

$$A_4\, A_i^*$$

and auto phase from:

$$A(t) * A^*(t + lag)$$

Thus COHRTW and Meteor-trail have the same analysis, and except for cycling correction to "zero lag" cross phase due to Doppler, the AOA results should have the same meaning.

3. RTVz system:

This used the same receiver as the meteor-trail system, and the auto was calculated according to

$$Q(t + lag) * I(t) - I(t+lag) * Q(t)$$

This corresponds to $A(t + lag) * A^{*}(t)$ in the previous repesentation for COHRTW and is seen to be reversed from the COHRTW value.
[note that the I,Q here were labelled R,I (for Re and Im) respectively in the Vz system program write-up]

The sign of the analysed velocity, which is the slope of the above expression going toward positive lags, was empirically found (by tilting the receiver array) to be reversed. All tape data are in this reversed condition!

4. Interferometer (tide system + prototype coherent Rx):

An FFT is done on each sequence, then cross phase found from:

Im Xspec iV4 = $Y_i . X_4 - X_i . Y_4$ (or $Q_i . I_4 - I_i . Q_4$ in COHRTW terms). Thus the cross is $A_i A^{*}_4$. But the I&Q had to be reversed later in the analysis to get the change in AOA and the sign of the Doppler to agree - so after reversing, the program is effectively using $A_4 . A^{*}_i$.

5. Conclusions:

The COHRTW and Meteor are correct in Vz/Vrad, that is a positive velocity indicates motion away from the radar. Also if the cross phase $A_4 . A^{*}_i$ goes more positive, then AOA is "leaning" more towards antenna 4; and so ϕ of the complex signal:

$$r\, e^{i\phi}$$

can be considered to be the total phase path with a negative sign, so $\phi_4 - \phi_3$ is really "phase-path-3" minus "phase-path-4"

6. Postscript:

An additional experimental test was carried out with the COHRTW running. 90^{o} of cable was added to antenna #1. The analysed 4V1 cross phases moved positively by 90^{o} ; which agrees with the conclusions in 5.

Description of final tape record

A tape block is 864 bytes organized as 32x3+24x32 bytes.

bytes#1-32:

This is the "date/time/param" store. The first 13 bytes are of the form 89:123/1230:05, where the punctuation is Ascii coding, but the digits are binary - except for the year which is BCD(1 byte). Other parameters are the delay (DLY) to the first height gate, the clockin delay (determines number of gates clocked into shift registers - should cover 64 or 65 gates: whichever makes the 1st gate equal to the 1st analysis height. This must be checked with a pulse generator triggered by the Tx synch.), and the gate separation (8 means 20 usec). The "delays" are really program parameters which determine the delay - they are not directly convertible to a delay value. These parameters are kept mostly as a means of data identification. Also stored is the amount of integration (# pulses per mean amplitude), and the length of the record (number of groups of 8 mean amps) - 2 bytes are reserved for the latter, but probably will never use the high byte. The last two bytes are always $00 and $FF - these are used internally for testing communication links. The site I.D. byte is always recorded, and also the A/D status (at the moment, but maybe not always)

bytes 33-64:

This is the "signal strength" store, starting at the lowest height (e.g. 49 Km). The signal strength is not quite as straightforward a parameter as that used in the RTW system. It is determined by taking the average of the square roots of the mean square value of (integrated) amplitude minus the appropriate running long-term-mean (i.e. receiver offset) over the 4 receivers and 2 (I&Q) components for each coherent integration group. That is, each set of 4x2 receiver amplitudes yields one RMS value, which is then accumumlated for 8 integrations and averaged. A grand average is done at the end of the record.

bytes 65-96:

This is the gain store, starting at the lowest height- the lowest 4 bits of each byte contain the gain (0=min to 7=max), the upper bits (if non-zero) contain a message showing why the height wasn't processed further. Gains for the R-S-S receivers are in steps of 10dB (not 8 dB as in the RTW system)and are not adjustable. Each gate has its own gain.

The message (4 bit number) has the following meanings: 1 = SD (signal) too small (less than 5), 2 = fast fading (auto below 0.5 at first lag), 3 = constant signal (all 1's or 0's in any of the 8 sequences).

Separate height results

Each height has 24 bytes for analysed data, starting with the lowest height. If no analysis has been done (i.e. the C128 ran out of time), all 24 bytes are zeros.

Byte#1:

The height number (lowest is 0)

bytes#2-7:

These are three 2 byte numbers (L,H) each containing the zero lag auto correlation and phase for antenna pairs 1v4,2v4,3v4 respectively. The formula giving the number is phase (deg) + INT(auto*100)x512. There is no apparent use for the zero lag autos at this time (except they could be used to determine pattern scale and orientation independently of the FCA analysis) - but why waste the free space.

Bytes# 8,9 (H,L):

This is the Doppler velocity (m/s) x 100 + $8000 (full range +-328 m/s). If these 2 bytes are zero, Vz wasn't calculated (or it could be an extreme! value- but very unlikely). Vz is calculated from either 1 (if the auto fell below 0.5 after the first lag) or 2 points of the auto-correlation phase(if it still above 0.5 at the second lag). If 2 points are available, the auto width and "noise factor" (NF) are also calculated, and the Vz is determined from the estimated phase slope at 0-lag from a 2 point parabolic fit calculation.

Bytes# 10-11:

The top two bits of byte#10 are flags set during the calculation of Vz. The MSB (bit 7) is set if the Vz is calculated from 1 point (in which case the auto width and noise factor are left zero); bit 6 is set if the "curvature" of the auto phase is greater than 30% (i.e. the slope from 0 to 1 lag and that from 1 to 2 lag phase differ by more than 30% - Vz is still calculated by the parabolic fit). If the auto magnitude is increasing from lag 1 to 2 ("impossible"), then there are no message flags, and the auto width and NF are left zero, but there is still a Vz from the 2-point fit.

The curvature criterion used here is similar in intent to that used in the "Vz System", but probably better. There the actual curvature from the parabolic fit was used, with a limit of 2, and it seemed that larger Vz's usually had greater curvature determined this way - which seems somewhat unacceptable. The present curvature here is in some sense "normalized" with respect to the Vz magnitude. (Curvature indicates that there is more than one significant Doppler frequency present - it does not necessarily indicate that the Vz should be rejected.)

The advantage of using the fitted slope at zero lag is that it is (theoretically) independent of the lag step.

The auto width parameter (if any) is taken from the 2-point (exact) fit to a Gaussian of the auto-magnitude. It is the calculated lag for ρ=0.61xNF (that is the NF is "eliminated" before the width is found). The units are lags x 4, maximum of 62, set equal to 63 if greater. The NF is the value of the auto-magnitude-times-100 extrapolated to zero lag from the Gaussian fit; maximum value is 127 (the top bit will be saved for possible future use as a message flag). These two parameters "may" be useful sometime.

Byte#12-13:

Byte#12 is $FF (fast fade) or, if the analysis got this far, ~~the~~ is 2*INT(NTD*100)+x, where x is 1 if the sum of the tmax is -ve, 0 otherwise. x is intended to show the sense of "vorticity" in the pattern. Byte#13 is the maximum lag used in searching for peaks in the cross correlations (usually $0F = 15, but sometimes a reduction is possible). (Note that after 1988day#348 NTD is now signed: value = INT(NTDx100)*2 + sign-bit, where sign-bit=1 if NTD is negative; before it is 200*NTD)

Byte#14-23:

These are the FCA parameters (0 if not calculated): #14-15(H,L)= V-true x10; #15-16(H,L)= phi-true(deg E of N); 17,18(H,L)=major axis (ρ=0.61) of pattern in metres; #19,20(H,L)= minor axis of pattern in metres; #21= tilt angle of major axis (deg E of N), #22=characteristic time(in sec) x10 (max. 25.5 sec).

Byte#24 contains 3 messages combined: IX*32+IRx4+IS with the same meaning as in the RTW system (but with the COHRTW system IX never=2)

Description of final data block from COHRTW system

C.M.Jun8'88

day#1=Jan1

date,time(GMT)

yr day# hr,min sec

8 : 1 5 9 / 1 6 4 5 : 0 1

#pulses integrated
"rec len"(mult x 8 to get pts/seq)→(L,H)
delay to 1st gate
clockin delay
gate separ.
check bytes
SITE I.D. (ASCII)

D/I:	08BA0105	09AF0106	0405BA00	01044000	373F0800	00000000	00000000	000000FF
SIG:	1D1D1D1E	1E202E37	5A7465B3	56652083	5A3F186F	CE8DC68B	A87C6453	948E7E75
GAIN:	27272727	27272707	07070606	06060404	03030101	02020404	05050505	06060606
49KM:	00000000	00000000	000000FF	0F000000	00000000	000000E0		
52KM:	01000000	00000000	000000FF	0F000000	00000000	000000E0		
55KM:	02000000	00000000	000000FF	0F000000	00000000	000000E0		
58KM:	03000000	00000000	000000FF	0F000000	00000000	000000E0		
61KM:	04000000	00000000	000000FF	0F000000	00000000	000000E0		
64KM:	05000000	00000000	000000FF	0F000000	00000000	000000E0		
67KM:	06000000	00000000	000000FF	0F000000	00000000	000000E0		
70KM:	07447F2B	7D13797F	9040453B9	0F000000	00000000	00000012		
73KM:	08368129	87167F7F	64405514	0F012F01	1C00AF00	82191500		
76KM:	092A7527	7120697E	E0035A26	0E017F01	11009F00	80AB1500		
79KM:	0A259124	970D9B7F	C0405E3A	0F000000	00000000	00000012		
82KM:	0B2BAF2E	AF14B380	0040617F	0F000000	00000000	00000091		
85KM:	0C306D42	87105B7F	92405A2D	0F000000	00000000	00000012		
88KM:	0D2B6127	950B8180	66405DAD	0F000000	00000000	00000091		
91KM:	0E307B28	950A8380	1A405E86	0F000000	00000000	00000070		
94KM:	0F418D0E	9B06AB7F	A940621A	0F029900	AE013100	B6A11B00		
97KM:	1043831C	9302A17F	4840620A	0F01D800	B400F600	A8A81400		
100KM:	11383D0C	4B15337F	6F406127	0F012500	8A007B00	55321100		
103KM:	122383EE	802A397F	D6406309	0F018300	7E00D800	683A1600		
106KM:	132DA910	B113B37F	9C056305	0F011A00	5A016B01	20AD2300		
109KM:	143CA114	A51AAF7F	7D046208	0F015C00	57012800	E4A31B16		
112KM:	155575F2	9631837E	EF406201	0B023A00	5400DC00	8A0C1100		
115KM:	16467719	3914837E	DA025F02	0F03D800	7100F900	647A3009		
118KM:	173A3FC8	4423597F	1E800001	0C01EE00	61007900	66930E16		
121KM:	18485BB4	521C357E	92405B7B	0F000000	00000000	00000091		
124KM:	194727AA	3A2F1A7D	F4800085	0D000000	00000000	00000091		
127KM:	1A53457E	42504E7C	D2800005	0A01EB00	5B007A00	53A60C00		
130KM:	1B583357	38505E7B	98800047	09000000	00000000	00000091		
133KM:	1C253E47	50546078	80800000	0901FB00	50008100	4C9B0B16		
136KM:	1D224223	52506477	B7800030	0A022F00	61007E00	61A00900		
139KM:	1E292A32	4E715478	0980000F	0A021D00	54007600	4EA10909		
142KM:	1F315020	4C5D4878	91800023	09020E00	60006F00	5B09090D		

rms sig(ht#0) rel. to off-set.

msg+gain(7) for ht#0
1:low sd
2:fast fade
3:const sig

ht#(0==49km)

4V3
zero lag x-phase&xcor
(L,H)=Int(ρ *100)*512+ Ø deg
ditto:4V2

ditto:4V1

(H,L)= Vz*100 +$8ØØØ (Vz in m/s)

bit check status (sometimes)

FCA msg=IX*32+ IR*4 +IS

(at ρ=0.61 in lags x 2)

msg+AWID,bit7= "1pt Vz",bit 6= "too crved"

"noise factor", if found, max 127

NTD*200(or 255 if not found)

now "signed", see text

reduced lag used for peak search (or 15 if can't get reduc.)

Vtr * 10(H,L)m/s

PHItr(deg EofN) (H,L)

maj axis(m)(H,L)
min axis(m)(H,L)

tilt of maj axis (deg EofN)

char.time(sec)*10 (=255 if .gt. 255)

NOTE: UNANALYSED HEIGHTS SHOW ALL ZEROES (INCLUDING THE HEIGHT NUMBER)

Complex Cross/Auto Correlation

The complex sequences A (t) and A (t+lag) are defined as

$A_i(t) = I_i(t) + i\ Q_i(t)$ and

$A_j(t+lag) = I_j(t+lag) + i\ Q_j(t+lag) = I'_j + Q'_j,$

where I and Q are the In-phase and Quadrature receiver components, and subscripts i,j are the antenna numbers.

Complex correlation is given by:

$$\rho_{ij}(lag) = \frac{\frac{\sum A_i A_j^*}{n - |lag|} - \frac{\sum A_i}{n} \cdot \frac{\sum A_j^*}{n}}{\sqrt{\frac{\sum A_i A_i^*}{n} - \frac{\sum A_i}{n} \cdot \frac{\sum A_i^*}{n}} \sqrt{\frac{\sum A_j A_j^*}{n} - \frac{\sum A_j}{n} \cdot \frac{\sum A_j^*}{n}}}$$

Let

$$V_i = \frac{\sum A_i A_i^*}{n} - \frac{\sum A_i}{n} \cdot \frac{\sum A_i^*}{n}$$

and

$$V_j = \frac{\sum A_j A_j^*}{n} - \frac{\sum A_j}{n} \cdot \frac{\sum A_j^*}{n}$$

where n is the number of amplitudes in the time sequence. Then the real and imaginary parts of the correlation are:

$$Re = \left[\frac{\sum_{n-|lag|} (I_i I'_j + Q_i Q'_j)}{n - lag} - \frac{\sum I_i}{n} \cdot \frac{\sum I_j}{n} - \frac{\sum Q_i}{n} \cdot \frac{\sum Q_j}{n} \right] \Big/ \sqrt{V_i V_j}$$

$$Im = \left[\frac{\sum_{n-|lag|} (Q_i I'_j - I_i Q'_j)}{n - |lag|} - \frac{\sum Q_i}{n} \cdot \frac{\sum I_j}{n} + \frac{\sum I_i}{n} \cdot \frac{\sum Q_j}{n} \right] \Big/ \sqrt{V_i V_j}$$

Since I and Q are bit-amplitude sequences (0 or 1), the I_i is the number of 1-bits in the I_i sequence for example, and $I_i\ I_j$ is the number of bit matches between I_i and I_j when they are shifted with respect to each other by a number of elements equal to the lag.

It can be seen that only two accumulators are needed for each lag of each cross or auto, viz. "II+QQ" and "QI-IQ". Also needed are the number of 1-bits in each of the 8 sequences ($I_1, Q_1, I_2, Q_2, I_3, Q_3, I_4, Q_4$). Also the Im part of the auto-correlation is 0 at zero lag, by definition.

Correction to correlation values:

Conversion of the amplitudes to bit-amplitudes introduces truncation noise - thus the magnitude of a bit-amplitude correlation is expected to be less than the corresponding amplitude correlation. Since it is *assumed* that the amplitude correlation is more appropriate, some sort of correction must be made because the V-true value is dependent on the correlation magnitudes. In the RTW system, an argument is made based on the amplitudes having a Gaussian distribution (actually is is more likely to be Rayleigh, since the "amplitudes" are really magnitudes of complex values). Here, the comlex amplitudes are likely to be symmetric, but who knows about Gaussian - anyway the same correction function is used here to the *magnitudes* of the complex correlations - athough a derivation has not been done, even assuming Gaussian distribution, with complex numbers to show whether the same function is appropriate.

A test was done with the Pitteway and Wright "point scatterer" model which shows that, at least in the one test done, the correlations for bit-amplitudes used in complex correlation works (i.e V-true are within 5%)

COHRTW OP-INSTRUCTIONS Dec'90

(TOTALCPU#2W-D/N)

GENERAL INFO:

Receiver#1,2,3,4 RF inputs are connected to North,South,Main,Centre (N-S dipoles), and outputs to I1,Q1,I2,Q2,I3,Q3,I4,Q4 on the receiver interface.
Note that the centre antenna must be connected through a 6dB preamp on the end of a power splitter which also feeds the RTW system.

DO NOT TURN OFF THE CPU#2 MONITOR- this causes a spike which sticks the system temporarily- or worse causes a comm. link error. Just turn down the brightness.(the system will recover by itself in approx 5 min if you do nothing else)

CHECKS

THE SYSTEM IS PROBABLY OK IF YOU SEE DATA BEING WRITTEN TO TAPE - this occurs every second occurrence (we are now dumping packed blocks) of X4:38 (M:S) or X9:38. If no dump, then look at PROBLEMS: entry on CPU#2 display. Note: a failure in the A/D check does not stop a tape dump, because it can happen (seldom) when there is no problem.

CHECK THE RECEIVERS BY PUSHING THE BUTTON ON THE INTERFACE MONITOR PANEL (left hand end of receiver interface) TO LOOK AT THE I OUTPUT FOR ALL RECEIVERS, AND USE THE OTHER BNC CONNECTOR TO LOOK AT THE Q OUTPUTS. THEY SHOULD LOOK SIMILAR - I.E. SIMILAR AMPLITUDE OF SIGNAL.

to change a tape

BEFORE DOING ANYTHING WITH THE TAPE DRIVE, PUSH "W" ON CPU#2 AND WAIT FOR A FLASHING "W" TO APPEAR ON THE SCREEN-OTHERWISE CPU#2 WILL OPERATE THE DRIVE WHEN YOU DON'T WANT ANY INTERFERENCE!

IF THE TAPE IS PAST THE End Of Tape MARKER DURING A WRITE (under system control), CPU#2 will still be running with "TAPE PAST EOT" displayed after "problems:"; but no tape write will be done. In this case, push "W" to put the system in wait (wait until you see the flashing W after the date/time), then push the tape drive "on-line" button to take drive off-line, push rewind, and push rewind again after tape has rewound to Beginning Of Tape (BOT) marker. Then mount a new tape following the threading diagram on the drive, and push LOAD button (this tensions the tape and looks for the BOT marker). When the tape is at the BOT marker, push "on-line" again(the on-line light should go on). Then push "S" on CPU#2. A return from wait clears the screen and re-writes the display so you won't see any parameters until the end of the next record.
- DON'T TURN OFF MONITOR (TV) FOR CPU#2.

****

...the following note probably does not apply since the tape reset circuit was modified, however.....

IF THE TAPE HAS RUN OFF THE END!:: it may be tangled on the take-up hub. VERY CAREFULLY untangle it making sure that more doesn't fall off! (or better still put a piece of scotch tape around the tape still on the reel- to make sure), then cut off any creased tape, thread it backwards and push LOAD, then REWIND. When the tape stops at the BOT (beginning of tape) marker, push rewind again and take off the tape.

****

IF THE TAPE IS STILL MOUNTED PROPERLY and you are in "wait" (flashing "W" on CPU#2 monitor) then push "E" a few times (file gaps); then push ON-LINE button (to take it off line - the on-line light should go off) and push rewind. When it reaches the beginning of tape push rewind again.

MOUNT THE NEW TAPE following the threading diagram inside the front door, and push LOAD. The drive automatically tensions the tape and moves forward to look for the BOT marker. You can push ON-LINE after (the ON-LINE light should come on) but the program will do this if you don't.

******

THEN push "S" on CPU#2 (the "middle" Apple) to restart.

****

The screen can be "re-drawn" with cleared parameters by going into and out of wait ("W" then "S" between records). This was added so that the #tape resets/NMIs/Interface resets can be cleared without stopping the system - otherwise they finally go off into strange characters (since only one digit is displayed) after a while (weeks).

****

Synch rates(for information only): At present(Jun'88 on...) the program selects 7.5 Hz synch (through an external relay)and runs 4 pulse integration in the day (10:00 AM to 8:59 PM) - it does *not* enable the Tx in the day, so it is depending on the RTW system to have selected 7.5 Hz Tx rate. While a tape dump is in progress or data are being packed in the tape output store, the system enables the Tx (which is set at 60Hz, i.e. rate "1" on the thumbwheel switch on the particular Tx pulse rate selector card, with the auto/man switch in the AUTO position). If the tape drive is at EOT there is no 60Hz "burst"; if the tape is not ready (but not at EOT) there is no 60Hz burst when an attemt is made to write, but there is when data are only being moved to the tape output store. If the write goes ahead, there is approx 1/2 sec of 60Hz. This occurs at approx. X4:34 or X9:34 (m:s).

At night, 60Hz synch is selected, and the Tx is enabled during the record (and the tape dump). After the dump, the Tx is "disabled" until the start of the next record as far as the COHRTW system is concerned- i.e. the Tx rate is determined by the other operating systems.

STARTING INSTRUCTIONS-all systems dead at start. If the Apple is an APPLE II+, it will come up in Basic ("]" prompt). If so you need to type CALL-151 ret., 3F2:69 FF ret., FB6FG ret.; so reset will go to the monitor instead of Basic.

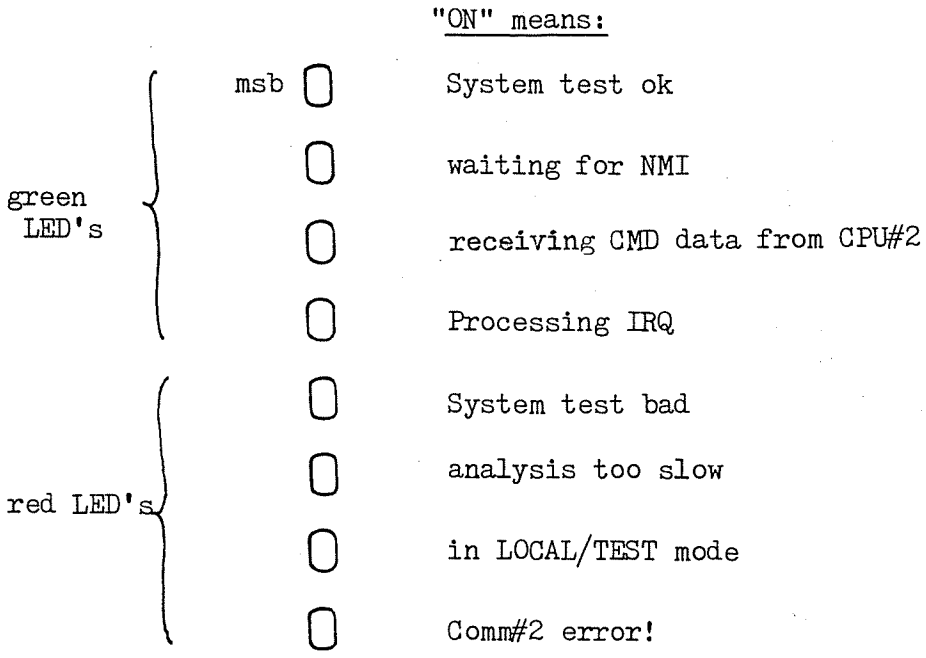
Load the APPLE (CPU#2) with TOTALCPU#2V'-D/N(900.1FEF) - this tape should be in the cassette recorder already. Type 900G return. [Note: the system can be started with 914G ret. which keeps the "present" gains, but you must KNOW that the gains are good! i.e. 0-7, unchanged from the last record run]

Then load the C128 with C128-PMFCA-V18J' (the latest version tape is left in the cassette recorder) and type RUN rtn.

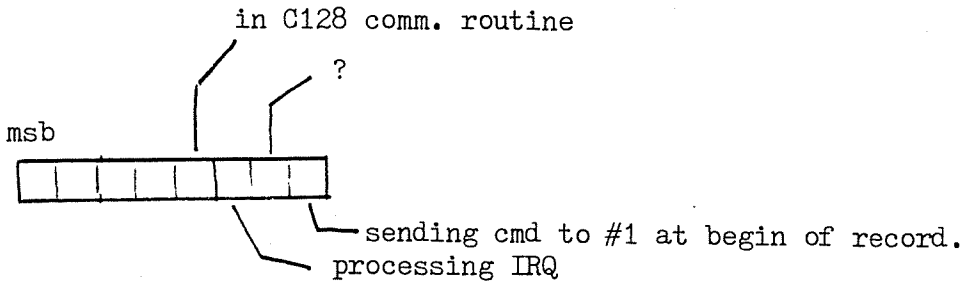
Set the clock on the Apple - use "+" and "-" keys to change units-of-seconds, and push "S" to start operation.[Note: a full clock set must be done from the mach. lang. monitor, but since the clock has its own battery, this will seldom be required)

If the tape drive is not on line (light), put it on line.

CPU#1 error (red) LEDs: Under normal conditions, none of these should be on. The 'local' LED indicates that the system is in self-operation (started manually by holding the 'test' button while pushing reset.), and will only be on for 4.5 min; then the system reverts to normal operation (controlled by CPU#2). If other red LEDs are on more than momentarily, there is a serious error. These will stay on until CPU#1 is reset (manually or by power-up). If red LEDs are on, then push the reset button (recessed) between records to see if the problem was temporary or is permanent - if permanent, there is a hardware fault which must be fixed.



CPU#1 LED Panel



CPU#2 Status byte: meaning on NMI

SECTION II.

SECTION II.

Section II.: CPU#1 (Rx interface control)

- CPU#1 detailed description (W) [page 23]
- Subroutine list (L) [page 27]
- Port address list (L) [page 28]
- Zero-page memory use (L) [page 29]
- Bulk RAM map (F) [page 30]
- LTM store (F) [page 31]
- I/O locations (CPU#1 <=> CPU#2 link) (F) [page 32]
- ROM storage (tables, program, etc.) (F) [page 33]
- CPU#1 program annotated listing (L) [page 35]

CPU#1 -detailed description

General operation

The interface computer runs two programs simultaneously: interrupt processing, and analysis [in contrast to the present non-coherent RTW system which does all data collection and analysis during interrupt processing.]

The interrupt is generated by the Tx synch pulse. During interrupt, the system controls the interface- gains, clocking, data collection, and storage (coherent integration), and also the sending of completed data records to CPU#2 - the latter is included in interrupt so the timing of the communication link to #2 is fixed (which may be important sometime).

The analysis does a running 8-channel A/D bit on/off check to test the hardware, does the coherent averages, the accumulation of long term means (LTM) Rx offset, the calculation of the vector magnitudes for the average signal, and conversion to bit-amplitudes with respect to the LTM. There are two flags needed for communication between interrupt processing and analysis. One is set in interrupt to tell the analysis that a coherent sum is complete and ready for analysis - this is reset when the analysis is complete. The other is set in analysis to tell the interrupt processing that a data record (the result of 8 coherent averages) is ready to send to #2 - this is reset when the data have been sent.

The record ends after a specified number of data records have been sent.[The required number has been sent to #1 from #2 at the beginning of the record, along with other parameters, e.g. gate separation, clockin delay, delay to 1st height gate, amount of coherent integration to do, etc.]

LTM considerations:

The only reason for doing a long term mean is that the prototype coherent receiver has an offset which varies with gain - due to a leakage of 2.219 MHz reference oscillator into the front end. If this problem is present in the new receivers, then an LTM is the only way of correcting for this fault. The LTM is meant to represent the receiver output level for no input (usually 5 volts = $80 hex through the A/D converters), which is set manually by adjusting the offset controls on the receiver op-amps. Over a long period of time, the signals should be randomly distributed about this level, and the LTM will approach the desired offset [assuming the offset is quite near 5 V ; otherwise the distribution will be skewed, and there will be a difference between the LTM and the real offset]. Then, even if the offset drifts slowly with time, the LTM will be a good approximation. This is important because the bit amplitudes are found by subtracting the LTM from the individual amplitudes - if the LTM is not correct and the signal is small, all bits may turn out to be the same in a sequence, which makes the cross correlation useless - ideally one wants an equal number of 1 and 0 bits in any amplitude sequence.

A way of avoiding the LTM altogether, is to flip the reference signal phase with each pulse, adding those signals measured with in-phase reference to the coherent accumulators, and subtracting those signals measured with out of phase reference. Since a real signal is locked to the transmitter phase, it will also reverse phase with the reference signal and the sum/diffference procedure will get rid of any constant offset in the receiver output, leaving a +ve or -ve sum. The sign bit of this sum is the desired bit-amplitude... very easy.

However, if there is some reference signal (e.g. ground clutter) getting in the front end of the receiver, this will look like a good reflection and will remain as a fixed offset in the result. Again, if this "offset" is greater than some small signals, the result will be a constant bit-amplitude sequence. This "offset" will vary with gain as well

A third option is to do the phase flipping, adding/subtracting etc. and still maintain an LTM for getting rid of this residual offset - but it means longer analysis time (I think) because +ve and -ve numbers have to be dealt with separately, and seems to have no advantages over the no-phase-flip analysis in terms of results. A minor advantage of the no-phase-flip analysis is that the LTMs can be monitored to see that the DC biases on the receiver outputs do not drift too far from the centre (5 Volts).

At present, the LTM is a running mean over 4096 amplitudes at each of the 8 receiver channels and 8 gains. If some gains (e.g. 7) occur more often, the LTM will represent a mean over a shorter period than for gains which do not occur as frequently. This does not appear to be a serious drawback, since signals at different heights may be considered independent in sign, and the 4096 amplitude average should approximate the receiver offset just as well no matter what the real time interval for the average is.

For example, if gain 7 is used for 4 heights, and the coherent integration time is 0.533 sec (which it will always be, since the pulse rate and amount of integration are adjusted to give this time), the average is over approx 9 min. Tests show that even if the initial LTM is very different from the real offset (i.e. several volts!), it only takes about 40 min to become a good approximation.

The latest EPROM version (BK28) allows the LTM to be floating (as above) or fixed (at $80 if CPU#1 was just reset) by setting a flag in the command data sent by CPU#2 at the beginning of a record, viz. this flag tells CPU#1 to skip the LTM update; an "LTM-mode" flag is returned to CPU#2 with the 8 status bytes tacked onto each data send (but not used at present).

Magnitudes:

Signal magnitudes are needed so the CPU#2 can decide on the best gains for the next record. Since a record may consist of greater than 256 coherent averages (e.g. 512avgs x 4 receivers for 4.5 min), and this would require at least 3 byte accumulators, some "pre-averaging" is done on each 8 coherent averages. This represents a sum of 8x4 magnitudes for each height and receiver. These average magnitudes are summed in another set of accumulators, and a "grand average" is done at the end of the record before sending them to CPU#2.

There are two ways of calculating a one-byte magnitude, given values of (I-LTM), (Q-LTM) from a coherent average. Assuming that both LTMs are exactly $80, the value of the maximum magnitude (i.e. both I and Q clipping at 0 or 10 volts) is ... $\sqrt{(\$80^2 + \$80^2)} = 181$; this wastes part of the magnitude byte. One way to solve this is to set the magnitude to $FF if $(I-LTM)^2 + (Q-LTM)^2 \geq \$4000$, and otherwise to take the square root of $4x\ ((I-LTM)^2 + (Q-LTM)^2)$. However, this eliminates some possible measurable magnitude variations, since clipping in only one component puts the value over the maximum. The other way is to always use the formula $mag = \sqrt{2x((I-LTM)^2 + (Q-LTM)^2)}$; however, if the LTM is not exactly $80, some values before the sqrt may be greater than 2 bytes, which cannot be handled by the subroutine....

~~Probably the first method will be used.~~

~~Added note:~~ The second method is used:

$$MAG = \sqrt{2 \times ((I-LTM)^2 + (Q-LTM)^2)}$$

A/D bit check:

The purpose of the bit check is to make sure that all bits of the A/D are operating. A malfunction here would be difficult, if not impossible to spot otherwise. It would just show as increased noise/fewer FCA values. Each A/D channel is checked independently. The first pulse in each coherent integral is stored in a separate "raw data" store for the bit check. When the bit-check is started, two check bytes are initialized for each channel, one to $00 and the other to $FF (i.e. all 0 bits, and all 1 bits). The initially valued $00 check byte is 'ORed' with the incoming amplitudes, and the result stored back in this check byte; the initially valued $FF check byte is 'ANDed' with the incoming amplitudes, and the result stored back in this check byte.

The A/D is ok if finally the initially valued $00 byte ends up as $FF, and the initially valued $FF byte ends up as $00. This means that all bits of the A/D output have been high and low at least once. The bit check continues with the 1st pulse of each coherent integral until this condition is satisfied for all A/D channels. Instead of keeping track of which bit(s) in a channel is bad, the "bit-check status byte" just indicates that a channel is bad by placing a '1' bit in the 8-bit byte.

Differential gain monitor

One serious problem noted with the prototype coherent receiver is that the gains of the I and Q channels can drift drastically! This causes large errors in the vertical Doppler measurement at least; the effect on the horizontal FCA velocity has not been investigated yet. In any case, it is necessary to put some sort of monitor on this, so that a clear warning is available automatically. Since, assuming a single CPU board for the interface computer, only a primitive display (LED's) is planned, CPU#2 must receive enough information from #1 to display a value of the gain ratio.

There are various possible ways of monitoring the gain ratio. For example, the ratio could be calculated after each coherent average, and the ratios averaged - however, it requires a divide each time (i.e. extra cpu time). The method used is to pick one height (arbitrarily for the 1st record), and sum the absolute values of (I-LTM) and (Q-LTM) in separate accumulators (uses 3 byte accumulators, two sets for each receiver). At the end of the record, the value of (100xmin)/max is the ratio, where min and max are the minimum and maxmium values of the two sums $\Sigma|(I-LTM_I)|$ and $\Sigma|(Q-LTM_\phi)|$; this number is less than or equal to 100 by definition, and so it fits in 7 bits. The msb (bit 7)is set if the sense is I/Q, and clear if Q/I.

The best height for the next record is calculated by looking for the mean magnitude closest to the middle of the possible magnitude range (-using $80 at present, but depends on how the magnitude is defined, and may change later). If the gain changes for the next record at this height, it may not be the best height for the gain ratio calculation.

System self test

A system self test is done on power-up and before each record (before waiting for an NMI from CPU#2). This verifies the EPROM program by doing a 2-byte check-sum and comparing the results with that stored on the EPROM, and checks all used RAM (except the LTM store) by storing and reading back $00 and $FF. As this is being done, the "system ok" green LED is turned off, and several of the red LEDs will flash momentarily (or stay on if an error is found). Error when "test-ok" is off and "test-fail" is on. Additional LEDs identify the error: no other LEDs on means RAM fail at least (couldn't even test EPROM checksum -gets stuck in loop); "analysis-too-slow" bad EPROM checksum; "in local/test mode" bad RAM.

LOAD and GO

This is a feature whereby a program can be loaded from CPU#2 and run. Since IRQ and NMI vectors are in hardware (EPROM), this feature is not intended to be used in conjuction with interrupt control (although it might be possible). In fact, I don't know what to use it for; it seemed like a good idea at the time. A sample CPU#2 program is shown in Section V. (pg. 139).

(Sub)routine list- CPU#1 (CPU#1BK28)

(note:these addresses are Fxxx on the single board computer EPROM)

ADDR	
$2000-	Power-up initialize
$206D-	Start
$2083-	NMI entry point(to begin record)
$209C-	Analyse
$20E0-	Running bit check
$2140-	Get Coherent averages
$21C0-	IRQ "pre-processing"
$21D0-	Accumulate long term means/offsets
$2240-	Get magnitudes/and bit amps
$2310-	Interrupt processing
$2390-	Software gating control
$23C0-	Load gains to shift regs
$23DA-	Read and store 1st data in coherent integral
$2450-	Accumulate 2nd and further amps in coherent integral
$24F8-	Send data and status to #2
$2540-	switch on selected LED(s)
$2550-	switch off selected LED(s)
$2560-	Read CMD data from #2
$25E0-	Initialize PIA's
$2620-	Reverse coherent store addresses
$2640-	Get I/Q or Q/I gain ratios at selected ht
$2720-	Record initialize
$2760-	Get mag avg and add to mag avg accum
$27A0-	SQRT
$27E0-	Divide(2 bytes by 1 byte)
$2810-	Multiply (1 byte by 1 byte)
$2830-	Load and send final parm at end of rec
$28B0-	Send gains, gate param., and test data to external-use port
$28E0-	Wait (software delay function)
$28F0-	Special command sorting (only 1 command avail. at present)
$2900-	System test (RAM and EPROM)
$29A0-	Load program from CPU#2 and run it
$2A00-	approx 0.5 sec delay
$2A10-	decrement "send" counter (2 byte)
$2A30-	prepare for divide(i.e. reduce 3-byte ÷ 2-byte problem to 2 ÷ 1)

CPU#1 (APPLE!) Port usage list

(note: on single board computer these addresses are 40xx)

C700 C701	Rx#1-Q
C702 C703	Rx#1-I
C704 C705	Rx#2-Q
C706 C707	Rx#2-I
C708 C709	Rx#3-Q
C70A C70B	Rx#3-I
C70C C70D	Rx#4-Q
C70E C70F	Rx#4-I
C710 C711	bit 7 = phase select,bit 6=Tx cntl (both o/p) bits 0-5 = switch cntl(s) i/p
C712 C713	gains and clocking cntl (o/p), IRQ control
C714 C715	LED display (o/p)
C716 C717	comm. link(port B) to #2, auto strobe on write
C718 C719	CA2 used for "buffer full" signal to external-use data port
C71A C71B	external-use raw data port, auto strobe on write

CPU#1 Zero page usage (flags & counters)

(* = non-scratch)

*$FF bit-check status
*$FE error condx since last reset
*$FD
*$FC
*$FB
*$FA
*$F9 $00 check bytes to test
*$F8 $FF comm. link

} status output to CPU#2 at end of every data transmission

$F7
$F6 used in height select for I/Q monitor
$F5
$F4
$F3
$F2
$F1
$F0

} used in mult/div routines

$EF fixed at initial value of EA ... why(!)
$EE fixed at initial value of $EB (why?)
$ED
$EC

} used in I/Q:Q/I calculation

*$EB Low byte of number of (8 coh avgs) in record (decremented after each send)
*$EA High byte of above(probably will not be used)
*$E9 GSEP = height gate separation, usually =8 (20 μsec)
*$E8 CLKIN sets window width for gates(i.e. # gates)
*$E7 DLY sets delay to 1st gate
*$E6 NIP = number of Tx pulses per cohherent avg.
*$E5 Tx cntl (=1 if off); not used at present (Tx is controlled by CPU#2)

} sent by CPU#2 at the beginning of every record

$E4
$E3 REC in progress flag(?)
*$E2 selected height pointer for I/Q calc. ($20=lowest)
*$E1 local/remote operations flag(=1 if loc.)
*$E0 temporary bit check status (stored in $FF at end)
*$C0-DF gains (DF is gain for lowest ht.) (sent by CPU#2 at beginning of rec.)
*$B0-BF bytes used in running A/D bit "on/off" check
*$AF "bit-check initialize" command
*$AE SL1-H
*$AD SL1-L } point to low bytes of coherent sum
*$AC SH1-H
*$AB SH1-L } point to high bytes of coherent sum
*$AA SL2-H
*$A9 SL2-L } point to low bytes of data being averaged
*$A8 SH2-H
*$A7 SH2-L } point to high bytes of data being averaged
*$A6 running pulse count for coherent sum
*$A5 DR#1:=1 if data ready for analysis,0 if anal. fini
*$A4 DR#2:=1 if data ready to go to CPU#2, 0 if sent
*$A3 PLS1 = 1 if 1st pulse in coh. sum
*$A2 phase select (=0 for normal, 0 phase)
*$A1 phase flip command(=1 if want alternate 0/180°)
*$A0 coh. avg. count (cmplx bit-amps),need 8 bit-amps before send
*$90-9F analysis scratch(reserved)
*$80-8F interrupt scratch (reserved,but may use at end of record)

JUL 17 1990

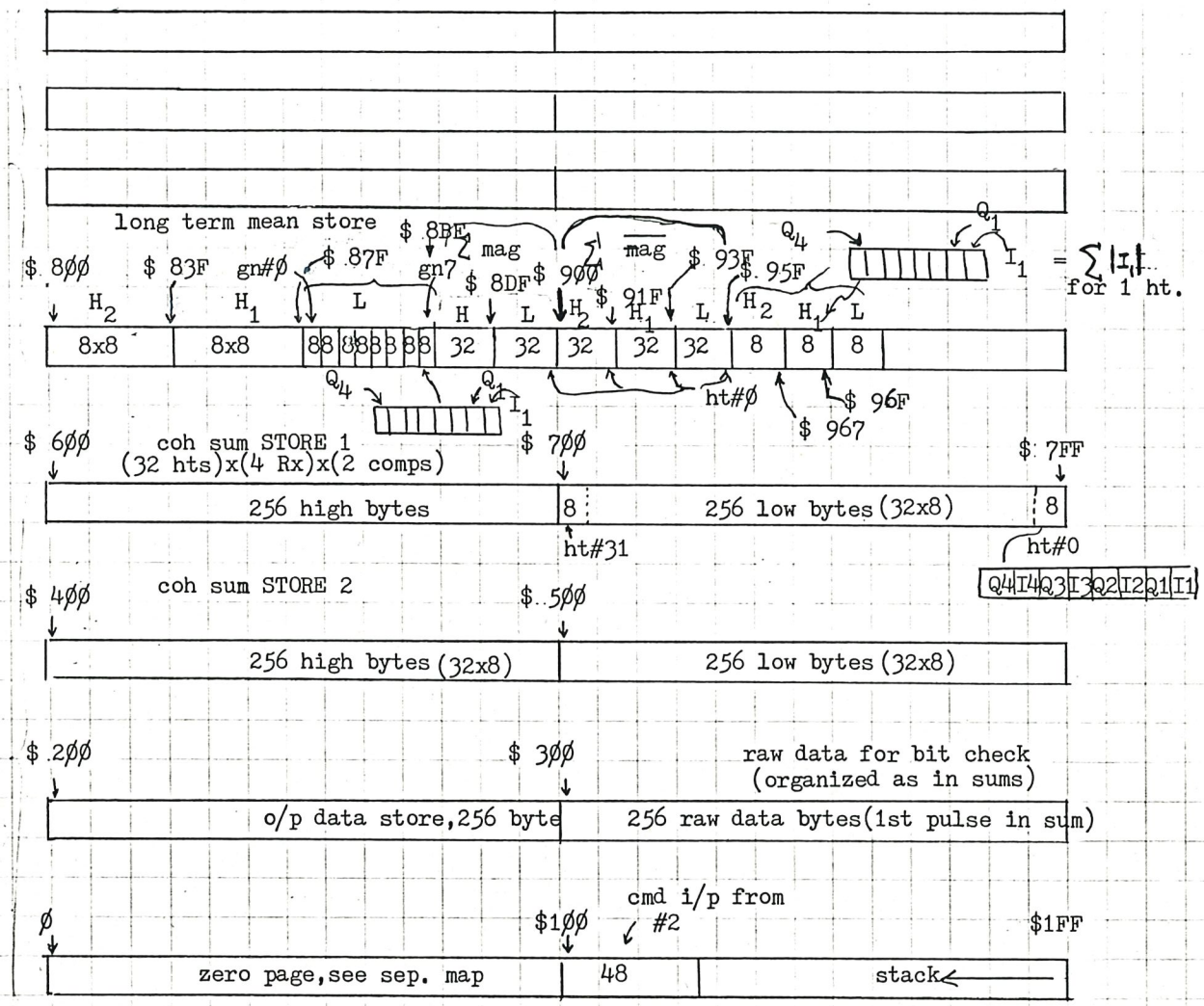
COHERENT RTW: INTERFACE COMPUTER (#1)

RAM ORGANIZATION - General info

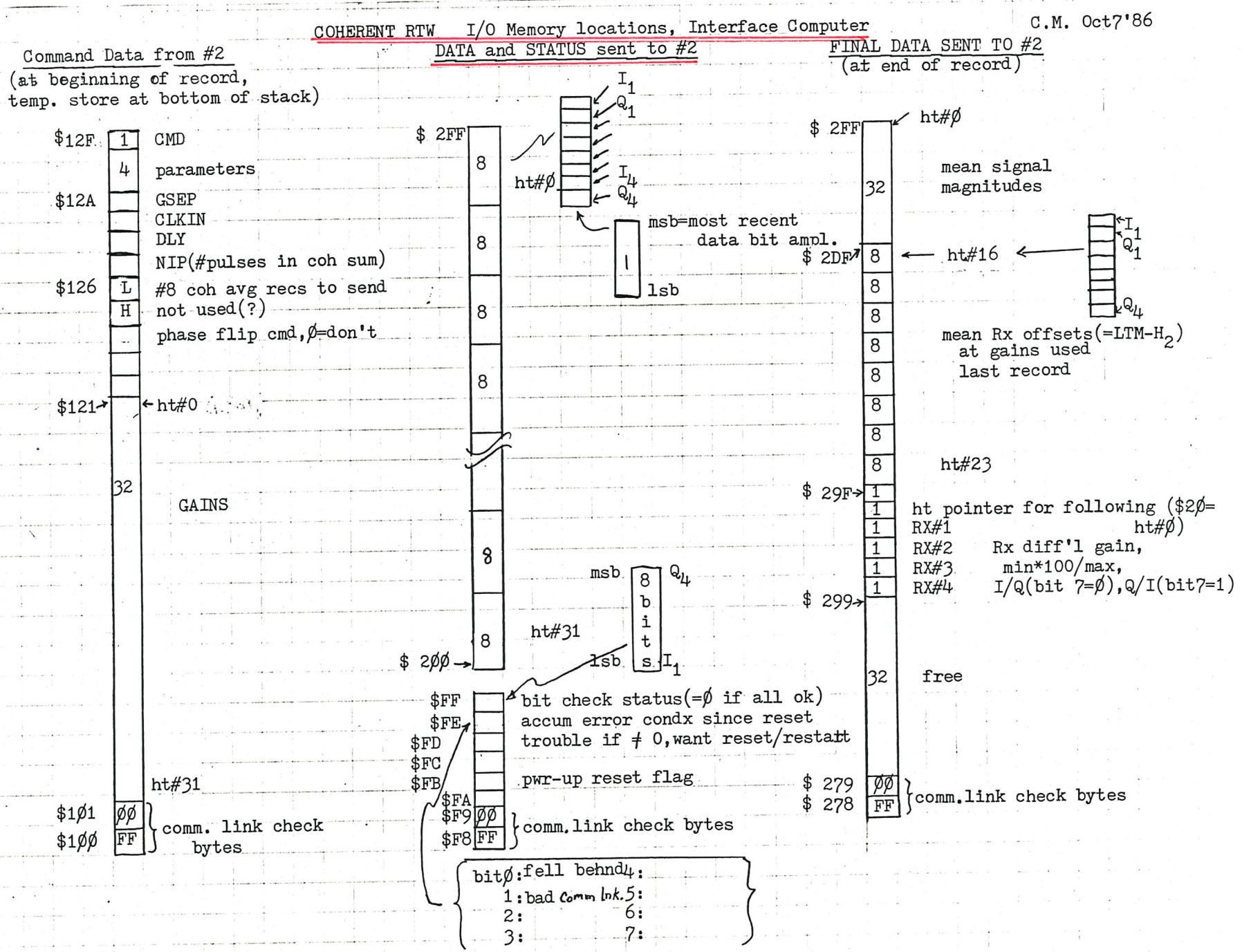
(for version CPU#1BK27'''-SBC on-)

C.M., Oct7'86

mod Jul17'90

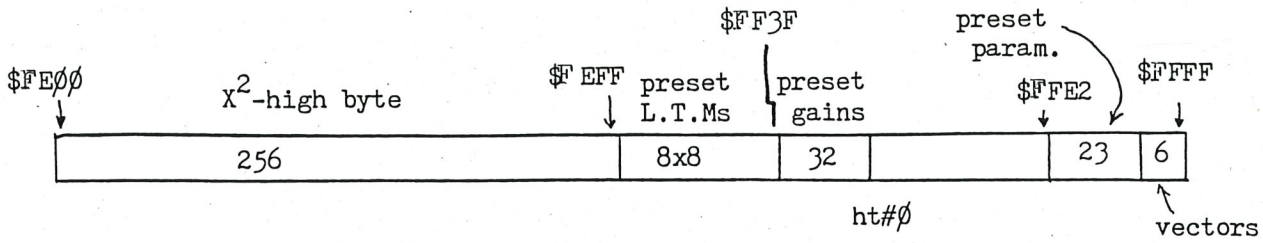


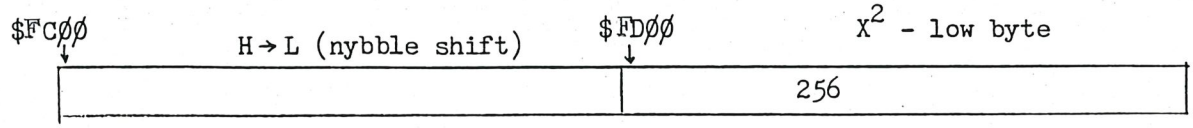
COHERENT RTW I/O Memory locations, Interface Computer C.M. Oct7'86

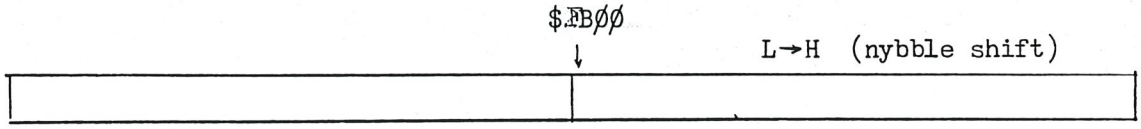


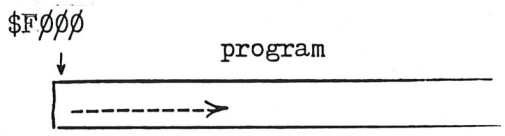
COHERENT RTW SYSTEM- INTERFACE COMPUTER (#1) C.M. Oct8'86

4K ROM storage

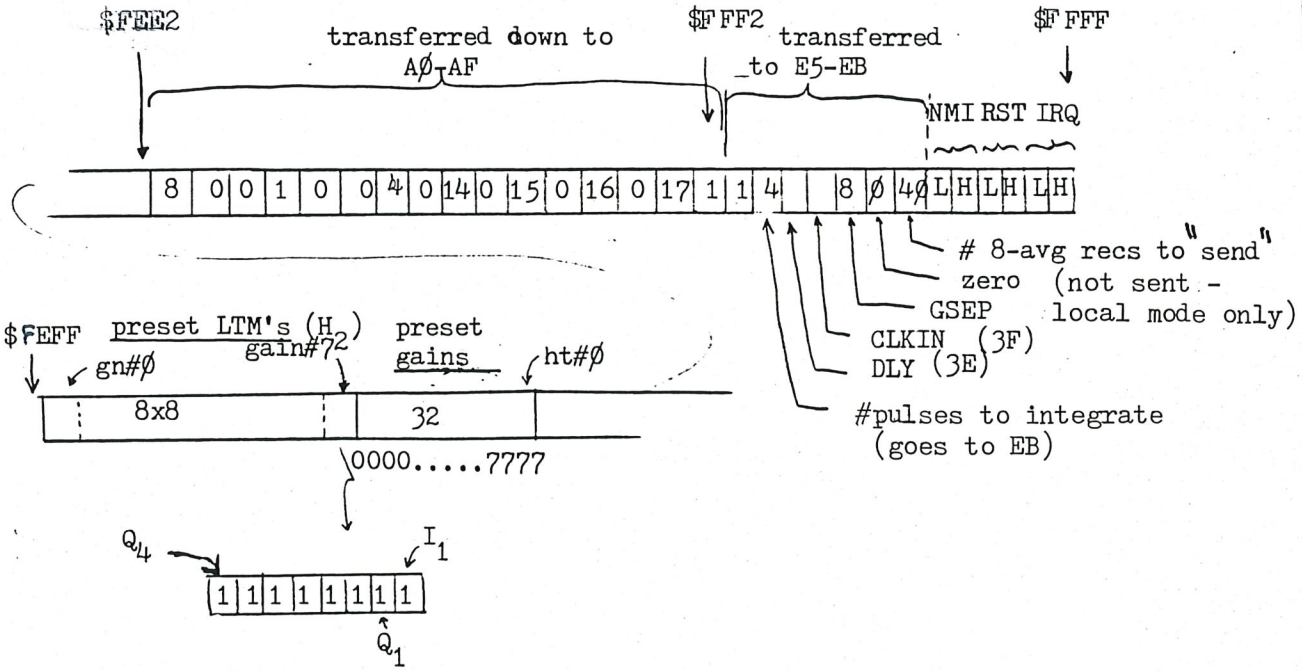








details of top page



COHRTW CPU#1BK28 (to work FØØØ.FFFF) in Apple

(change all Absolute addresses C7XX to 40XX and C6XX to 3FXX to convert to single board computer version = 'SBC' suffix)

4-antenna program

CM
~~Jan28'90~~
~~Apr2,'90~~
Jul17'90

JUL 23 1990

```
2000-  A2 FF      LDX  #$FF      Set stack pointer                  POWER-UP ENTRY
2002-  9A         TXS             to top of stack
2003-  78         SEI           - disable interrupts
2004-  D8         CLD           - clear decimal mode
2005-  A2 10      LDX  #$10
2007-  BD E2 FF   LDA  $FFE2.X   load pre-set parameters for
200A-  95 9F      STA  $9F.X     possible local/test oper.
200C-  CA         DEX
200D-  D0 F8      BNE  $2007
200F-  A2 20      LDX  #$20
2011-  BD 3F FF   LDA  $FF3F.X
2014-  95 BF      STA  $BF.X
2016-  CA         DEX
2017-  D0 F8      BNE  $2011
2019-  A2 07      LDX  #$07           more preset param.
201B-  BD F2 FF   LDA  $FFF2.X
201E-  95 E4      STA  $E4.X
2020-  CA         DEX
2021-  D0 F8      BNE  $201B
2023-  A9 01      LDA  #$01           set"power-up reset" flag.
2025-  85 FB      STA  $FB
2027-  A9 00      LDA  #$00         clear"accumulated error conditions" byte.
2029-  85 FE      STA  $FE
202B-  85 F9      STA  $F9          set comm link check bytes in "status o/p to #2"
202D-  A9 FF      LDA  #$FF               store.
202F-  85 F8      STA  $F8
2031-  A2 40      LDX  #$40
2033-  A9 00      LDA  #$00
2035-  9D 7F 08   STA  $087F.X L    load pre-set long-term-mean initial values
2038-  9D 3F 08   STA  $083F.X H1         (all = $8Ø)
203B-  BD FF FE   LDA  $FEFF.X
203E-  9D FF 07   STA  $07FF.X H2
2041-  CA         DEX
2042-  D0 EF      BNE  $2033
2044-  A0 05      LDY  #$05           approx 1 sec delay (let interface power settle)
2046-  20 E0 F8   JSR  $F8E0
2049-  88         DEY
204A-  D0 FA      BNE  $2046
204C-  EA         NOP
204D-  EA         NOP
204E-  EA         NOP
204F-  EA         NOP
2050-  EA         NOP
2051-  EA         NOP
2052-  EA         NOP
2053-  EA         NOP
2054-  EA         NOP
2055-  EA         NOP
2056-  EA         NOP
2057-  EA         NOP
2058-  EA         NOP
2059-  EA         NOP
205A-  EA         NOP
205B-  EA         NOP
205C-  EA         NOP
205D-  20 E0 F5   JSR  $F5E0      Initialize PIAs;and turn off Tx (not used).
2060-  AD 10 C7   LDA  $C710      check whether local/test button pushed
2063-  49 20      EOR  #$20       (bit 5 of PIA, = Ø if pressed, =$20 if not)
2065-  29 20      AND  #$20
2067-  85 E1      STA  $E1
2069-  A9 10      LDA  #$10       set initial height for I/Q,Q/I diff'l gain stuff
206B-  85 E2      STA  $E2
```

JUL 17 1990

```
206D-  78          SEI          START
206E-  D8          CLD
206F-  20 E0 F5    JSR  $F5E0   Initialize PIAs
2072-  20 00 F9    JSR  $F900   do EPROM and RAM system test (LEDs show results)
2075-  A5 E1       LDA  $E1     Want local operation?
2077-  D0 0A       BNE  $2083   yes,so skip NMI wait, and go to start record
2079-  A9 40       LDA  #$40    No,so turn on "waiting for NMI" LED
207B-  20 40 F5    JSR  $F540
207E-  EA          NOP
207F-  EA          NOP
2080-  4C 80 F0    JMP  $F080   endless loop to wait for NMI
2083-  A2 FF   NMI LDX  #$FF    NMI entry, reset stack pointer to top of stack
2085-  9A          TXS
2086-  A9 40       LDA  #$40    turn off "waiting for NMI" LED
2088-  20 50 F5    JSR  $F550
208B-  EA          NOP
208C-  20 20 F7    JSR  $F720   Initialize various parameters and read command data
208F-  20 C0 F3    JSR  $F3C0   <- put gains in shift regs. from CPU#2 ready for 1st Tx pulse
2092-  AD 12 C7    LDA  $C712   <- clear any existing interrupt flag (Tx synch)
2095-  58          CLI          enable interrupts
2096-  EA          NOP
2097-  EA          NOP
2098-  EA          NOP
2099-  EA          NOP
209A-  EA          NOP
209B-  EA          NOP
209C-  A5 EA       LDA  $EA     - last data sent to #2?  ANALYZE
209E-  30 CD       BMI  $206D   - yes,so go back to START to wait for next cmd from #2
20A0-  A5 A5       LDA  $A5     - data ready for analysis?
20A2-  F0 F8       BEQ  $209C   - no, so don't analyze
20A4-  20 E0 F0    JSR  $F0E0   yes, do/continue running bit check on raw data
20A7-  EA          NOP
20A8-  EA          NOP
20A9-  EA          NOP
20AA-  20 40 F1    JSR  $F140   get mean signals (i.e. coherent avg)

20AD-  A5 FD       LDA  $FD     } check to see if cmd says skip LTM update.
20AF-  30 03       BMI  $20B4   }
20B1-  20 D0 F1    JSR  $F1D0   <- long-term-means (used for conversion to bit amps)
20B4-  EA          NOP
20B5-  EA          NOP
20B6-  20 40 F2    JSR  $F240   get magnitudes, and bit amps
20B9-  EA          NOP
20BA-  EA          NOP
20BB-  EA          NOP
20BC-  EA          NOP
20BD-  EA          NOP
20BE-  EA          NOP
20BF-  EA          NOP
20C0-  EA          NOP
20C1-  EA          NOP
20C2-  A9 00       LDA  #$00    set "data analysed" flag  (checked in interrupt)
20C4-  85 A5       STA  $A5
20C6-  C6 A0       DEC  $A0     next bit-amp, = last?
20C8-  D0 D2       BNE  $209C   no, so continue analysis
20CA-  A9 08       LDA  #$08    re-load bit-amp counter  (8-bit amps per send)
20CC-  85 A0       STA  $A0
20CE-  20 60 F7    JSR  $F760   add mean mags to Σmean mag store
20D1-  A9 01       LDA  #$01    set "data ready to send to #2" flag
20D3-  85 A4       STA  $A4
20D5-  D0 C5       BNE  $209C   Forced branch
```

```
20E0-  A5 AF      LDA   $AF       start bit check?
20E2-  D0 05      BNE   $20E9     yes,so initialize
20E4-  A5 E0      LDA   $E0       no; bit check fini ok?
20E6-  D0 16      BNE   $20FE     no,so keep tesing
20E8-  60         RTS
20E9-  A2 08      LDX   #$08
20EB-  A9 FF      LDA   #$FF
20ED-  95 AF      STA   $AF.X
20EF-  A9 00      LDA   #$00      initialize comparison bytes
20F1-  95 B7      STA   $B7.X
20F3-  CA         DEX
20F4-  D0 F5      BNE   $20EB
20F6-  A9 00      LDA   #$00      clear "start bit check"command
20F8-  85 AF      STA   $AF
20FA-  A9 FF      LDA   #$FF      set command status = bad
20FC-  85 E0      STA   $E0
20FE-  A0 FF      LDY   #$FF      point to raw data byte (256 of them)
2100-  A2 08      LDX   #$08      8 channels (I1,Q1,I2,Q2,I3,Q3,I4,Q4)
2102-  B9 00 03   LDA   $0300,Y   Raw data (from 1st pulse in coh. integr.)
2105-  48         PHA
2106-  35 AF      AND   $AF.X     do all channels at 32 heights
2108-  95 AF      STA   $AF.X
210A-  68         PLA
210B-  15 B7      ORA   $B7.X
210D-  95 B7      STA   $B7.X
210F-  88         DEY
2110-  CA         DEX
2111-  D0 EF      BNE   $2102
2113-  C0 FF      CPY   #$FF
2115-  D0 E9      BNE   $2100
2117-  EA         NOP
2118-  EA         NOP
2119-  EA         NOP
211A-  EA         NOP
                                  test result of bit check
211B-  A2 08      LDX   #$08      channel loop
211D-  B5 AF      LDA   $AF.X     -1st check
211F-  D0 07      BNE   $2128      bad, should be $ØØ, so skip other check
2121-  B5 B7      LDA   $B7.X      ok, try 2nd check
2123-  C9 FF      CMP   #$FF       all bits should be high if OK
2125-  18         CLC              clear carry for roll into status byte if OK
2126-  F0 01      BEQ   $2129      OK, so store Ø bit in status
2128-  38         SEC              bad,so set 1 bit in status
2129-  66 E0      ROR   $E0        roll into status byte
212B-  CA         DEX              next channel
212C-  D0 EF      BNE   $211D
212E-  A5 E0      LDA   $E0        save present bit-check status in status bytes
2130-  85 FF      STA   $FF         going to #2 with each data sent
2132-  60         RTS
```

Running Bit Check

check all A/D data (8 channels) for bits on&off Done on 1st pulse in coh. integ. data'

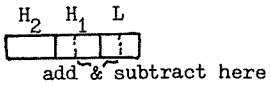
```
2140-  A5 E6        LDA  $E6       = 2^n, find n
2142-  A2 00        LDX  #$00                    Get mean signals(Coh. Avg.)
2144-  4A           LSR                          N.B Coh. integral assumed to
2145-  F0 03        BEQ  $214A                   be over 2^n amps
2147-  E8           INX
2148-  D0 FA        BNE  $2144
214A-  E0 00        CPX  #$00      n=0? (i.e. 2^n=1?)
214C-  D0 01        BNE  $214F     no,so do average
214E-  60           RTS            yes,so no average needed: skip out
214F-  86 9F        STX  $9F       = n
2151-  A5 E6        LDA  $E6       2^n > 16 ?
2153-  C9 10        CMP  #$10
2155-  B0 1B        BCS  $2172     yes, so want to pre-shift by 4-bits using tables
2157-  A0 00        LDY  #$00      no,so just shift data to right by n bits
2159-  B1 A7        LDA  ($A7),Y   = sum-H byte
215B-  85 9E        STA  $9E
215D-  B1 A9        LDA  ($A9),Y   = sum-L byte
215F-  A6 9F        LDX  $9F       shift down n times
2161-  18           CLC
2162-  66 9E        ROR  $9E
2164-  6A           ROR
2165-  CA           DEX
2166-  D0 F9        BNE  $2161
2168-  91 A9        STA  ($A9),Y   and store result-L  (H=0)
216A-  88           DEY
216B-  D0 EC        BNE  $2159
216D-  60           RTS
216E-  EA           NOP
216F-  EA           NOP
2170-  EA           NOP
2171-  EA           NOP
2172-  D0 1B        BNE  $218F     2^n not equal to 16,so use shift AND nybble transfers
2174-  A0 00        LDY  #$00          2^n = 16, just need nybble transfers (by tables)
2176-  B1 A9        LDA  ($A9),Y   H byte of sum
2178-  AA           TAX
2179-  BD 00 FC     LDA  $FC00,X   H→L  nybble xfer
217C-  85 9E        STA  $9E
217E-  B1 A7        LDA  ($A7),Y   L byte of sum
2180-  AA           TAX
2181-  BD 00 FB     LDA  $FB00,X   L→H  nybble xfer
2184-  05 9E        ORA  $9E        plus , then
2186-  91 A9        STA  ($A9),Y    store result in L-byte-of-sum pos'n
2188-  88           DEY             next avg
2189-  D0 EB        BNE  $2176
218B-  60           RTS
218C-  EA           NOP
218D-  EA           NOP
218E-  EA           NOP
218F-  A5 9F        LDA  $9F           2^n > 16 so need shift  & then nybble xefrs
2191-  38           SEC
2192-  E9 04        SBC  #$04      get n-4 = # shifts needed
2194-  85 9F        STA  $9F
2196-  A0 00        LDY  #$00
2198-  B1 A7        LDA  ($A7),Y     sum-H
219A-  85 9E        STA  $9E
219C-  B1 A9        LDA  ($A9),Y     sum-L
219E-  A6 9F        LDX  $9F         shift n-4 times to right
21A0-  18           CLC
21A1-  66 9E        ROR  $9E
21A3-  6A           ROR
21A4-  CA           DEX              next shift
21A5-  D0 F9        BNE  $21A0
21A7-  AA           TAX              = L byte of result
21A8-  BD 00 FC     LDA  $FC00,X       now shift down an additional 4-bits by nybble xfers
21AB-  85 9D        STA  $9D
21AD-  A6 9E        LDX  $9E
21AF-  BD 00 FB     LDA  $FB00,X
21B2-  05 9D        ORA  $9D
21B4-  91 A9        STA  ($A9),Y
21B6-  88           DEY
21B7-  D0 DF        BNE  $2198
21B9-  60           RTS
```

```
21C0-  85 45  IRQ  STA  $45     IRQ entry,save A   This duplicates Apple Monitor IRQ
21C2-  68          PLA          get processor status, processing
21C3-  48          PHA          and put it back
21C4-  0A          ASL          Was IRQ disabled?        N.B. There should never be
21C5-  0A          ASL                                 a BREAK programmed, but should
21C6-  0A          ASL                                 test anyway- i.e. prog. error.
21C7-  30 03       BMI  $21CC   yes, so must have been a "break" (a $ØØ taken as a program
21C9-  6C E0 FF    JMP  ($FFE0) Ok, IRQ,so go to routine      statement
21CC-  4C 00 F0    JMP  $F000   was BREAK, so just go back to pwr-up reset
```

ACCUMULATE LONG-TERM-MEANS

long term mean (LTM)

$$= \frac{4096*\text{old}-\text{old}+\text{new}}{4096}$$




```
21D0-  A9 FF       LDA  #$FF    coh. avg. pointer
21D2-  85 9F       STA  $9F
21D4-  A9 20       LDA  #$20    height,start at ht#0
21D6-  85 9E       STA  $9E
21D8-  A6 9E       LDX  $9E
21DA-  B5 BF       LDA  $BF,X   get gain for this ht
21DC-  0A          ASL
21DD-  0A          ASL          gain*8 (part of ltm pointer)
21DE-  0A          ASL
21DF-  85 97       STA  $97
21E1-  A9 08       LDA  #$08    Rx/component counter
21E3-  85 9C       STA  $9C
21E5-  A5 9F       LDA  $9F     last part of ltm pointer
21E7-  29 07       AND  #$07
21E9-  05 97       ORA  $97
21EB-  85 9D       STA  $9D     full ltm pointer
21ED-  A6 9D       LDX  $9D
21EF-  BD 00 08    LDA  $0800,X  H2 byte of LTM (="old")
21F2-  48          PHA           save "old"
21F3-  A8          TAY           save value as index for nybble shifts
21F4-  BD 80 08    LDA  $0880,X  L byte of "old"
21F7-  38          SEC
21F8-  F9 00 FB    SBC  $FB00,Y  minus L→H of old
21FB-  9D 80 08    STA  $0880,X  save in L byte                  subtract old
21FE-  68          PLA           get "old"
21FF-  A8          TAY
2200-  BD 40 08    LDA  $0840,X  H1 of LTM
2203-  F9 00 FC    SBC  $FC00,Y   minus H→L of "old"
2206-  9D 40 08    STA  $0840,X   save in H1
2209-  B0 06       BCS  $2211     no borrow
220B-  EA          NOP
220C-  EA          NOP
220D-  EA          NOP
220E-  DE 00 08    DEC  $0800,X   borrow from H2
2211-  A4 9F       LDY  $9F
2213-  B1 A9       LDA  ($A9),Y   new                            add new
2215-  48          PHA
2216-  A8          TAY
2217-  BD 80 08    LDA  $0880,X      L of LTM
221A-  18          CLC
221B-  79 00 FB    ADC  $FB00,Y      plus L→H of new
221E-  9D 80 08    STA  $0880,X      save in L
2221-  68          PLA               get new back
2222-  A8          TAY
2223-  BD 40 08    LDA  $0840,X      H1 of LTM
2226-  79 00 FC    ADC  $FC00,Y      plus H→L of new
2229-  9D 40 08    STA  $0840,X      save H1
222C-  90 03       BCC  $2231        no carry
222E-  FE 00 08    INC  $0800,X      carry over to H2
2231-  C6 9F       DEC  $9F       next coh. avg.
2233-  C6 9C       DEC  $9C       next Rx/component
2235-  D0 AE       BNE  $21E5
2237-  C6 9E       DEC  $9E       next height
2239-  D0 9D       BNE  $21D8
223B-  60          RTS
```

Get Magnitudes and bit-amps

```
2240-  A0 FF     LDY  #$FF     coh. avg. pointer
2242-  A9 20     LDA  #$20     height counter
2244-  85 9F     STA  $9F
2246-  A9 04     LDA  #$04     Rx counter, starting at rx#1
2248-  85 9E     STA  $9E
224A-  A5 9F     LDA  $9F      get pointer to LTM for this gain/Rx/component
224C-  AA        TAX
224D-  B5 BF     LDA  $BF.X    gain
224F-  0A        ASL
2250-  0A        ASL
2251-  0A        ASL
2252-  85 97     STA  $97
2254-  98        TYA
2255-  29 07     AND  #$07
2257-  05 97     ORA  $97
2259-  AA        TAX           = pointer to LTM
225A-  B1 A9     LDA  ($A9),Y  coh. avg. I_i
225C-  84 9A     STY  $9A
225E-  38        SEC
225F-  FD 00 08  SBC  $0800.X  minus LTM for I_i (at the present gain)
2262-  B0 04     BCS  $2268    :result=positive, so no fuss
2264-  49 FF     EOR  #$FF     :result -ve,so must get abs value; keep carry for bit-amp
2266-  69 01     ADC  #$01     nb: carry was clear if -ve, so adc ok.
2268-  86 9D     STX  $9D      A=abs (I_i-LTM) here
226A-  A6 9A     LDX  $9A
226C-  7E 00 02  ROR  $0200.X  roll carry into bit amp byte ('1' if .ge. LTM)
226F-  A6 9F     LDX  $9F      get back height index.
2271-  E4 E2     CPX  $E2      Is this the height for monitoring I/Q ??
2273-  D0 01     BNE  $2276    no
2275-  48        PHA           yes, so save |I_i-LTM| on stack for later use
2276-  AA        TAX           Look up square from tables
2277-  BD 00 FE  LDA  $FE00.X  X^2(H)
227A-  85 9C     STA  $9C
227C-  BD 00 FD  LDA  $FD00.X  X^2(L)          = |I_i-LTM|^2
227F-  85 9B     STA  $9B
2281-  88        DEY
2282-  A6 9D     LDX  $9D
2284-  CA        DEX
2285-  B1 A9     LDA  ($A9),Y   Q̄_i
2287-  84 9A     STY  $9A
2289-  38        SEC
228A-  FD 00 08  SBC  $0800.X   Q̄_i-LTM
228D-  B0 04     BCS  $2293
228F-  49 FF     EOR  #$FF      |Q̄_i- LTM|
2291-  69 01     ADC  #$01
2293-  A6 9A     LDX  $9A
2295-  7E 00 02  ROR  $0200,X   save bit-amp for Q_i
2298-  A6 9F     LDX  $9F
229A-  E4 E2     CPX  $E2       is this the selected height for monitor I/Q stuff?
229C-  D0 01     BNE  $229F     no,so skip
229E-  48        PHA            yes, so save |Q_i-LTM| on stack for later use
229F-  AA        TAX
22A0-  18        CLC
22A1-  BD 00 FD  LDA  $FD00.X   X^2(L)          get |Q_i-LTM|^2 and add to |I_i-LTM|^2
22A4-  65 9B     ADC  $9B
22A6-  85 9B     STA  $9B
22A8-  BD 00 FE  LDA  $FE00,X   X^2(H)
22AB-  65 9C     ADC  $9C
22AD-  85 9C     STA  $9C       high byte of result
22AF-  C9 80     CMP  #$80      will √(2*result) be > $FF ?
22B1-  90 0C     BCC  $22BF     No.
22B3-  A9 FF     LDA  #$FF      Yes, so set = $FF
22B5-  D0 10     BNE  $22C7
```

-8 x NOP -

```
22BF-  18        CLC
22C0-  26 9B     ROL  $9B       x 2
22C2-  26 9C     ROL  $9C
22C4-  20 A0 F7  JSR  $F7A0     Get SQRT = "magnitude of signal"
22C7-  EA        NOP
22C8-  A6 9F     LDX  $9F       now add magnitude to Σmagnitude store
22CA-  18        CLC
22CB-  7D DF 08  ADC  $08DF.X   L                (coh. avgs for all Rx)
22CE-  9D DF 08  STA  $08DF.X     (after sum of 8, these are avged and added
22D1-  90 03     BCC  $22D6         to Σmag store)
22D3-  FE BF 08  INC  $08BF.X   H
22D6-  A4 9A     LDY  $9A
22D8-  88        DEY
22D9-  C6 9E     DEC  $9E       next Rx
22DB-  F0 03     BEQ  $22E0
22DD-  4C 4A F2  JMP  $F24A
22E0-  C6 9F     DEC  $9F       next height
22E2-  F0 03     BEQ  $22E7      = last height fini.
22E4-  4C 46 F2  JMP  $F246
```

APR 2 1990 JUL 17 1990 (7)

```
22E7-  A2 01     LDX  #$01          now get back stack data for I/Q stuff
22E9-  A0 08     LDY  #$08            and accumulate sums of abs values
22EB-  68        PLA
22EC-  7D 6F 09  ADC  $096F,X       L of Σ|I_i-LTM|  or Σ|Q_i-LTM|
22EF-  9D 6F 09  STA  $096F,X
22F2-  90 08     BCC  $22FC
22F4-  FE 67 09  INC  $0967,X       H_1 (high byte of accum)
22F7-  D0 03     BNE  $22FC
22F9-  FE 5F 09  INC  $095F,X       H_2 ( higher byte...)
22FC-  E8        INX
22FD-  88        DEY
22FE-  D0 EB     BNE  $22EB
2300-  60        RTS
```

```
2310-  08        PHP                          INTERRUPT PROCESSING
2311-  A5 45     LDA  $45        (needed for Apple RAM version, but kept anyway)
2313-  48        PHA
2314-  98        TYA                  save all regs. on stack
2315-  48        PHA
2316-  8A        TXA
2317-  48        PHA
2318-  20 90 F3  JSR  $F390      control delay to 1st gate,#gates etc., data now in S.regs!
231B-  A9 10     LDA  #$10
231D-  20 40 F5  JSR  $F540      turn on "in IRQ LED
2320-  A5 EA     LDA  $EA        last data in Rec sent already? (i.e. no run in progress!)
2322-  30 4D     BMI  $2371      yes, so skip out of interrupt
2324-  A5 A4     LDA  $A4        Data ready for o/p to #2?
2326-  F0 19     BEQ  $2341      no,so continue
2328-  20 F8 F4  JSR  $F4F8      yes,so send data to #2   (results of 8 coh. avgs.)
232B-  A9 00     LDA  #$00
232D-  85 A4     STA  $A4        clear "data ready to send"flag
232F-  85 FB     STA  $FB        turn off "power-up reset" status byte
2331-  20 10 FA  JSR  $FA10      decrement "send counter"; was this the last send?
2334-  10 0B     BPL  $2341      no
2336-  20 40 F6  JSR  $F640      Yes,so do I/Q,Q/I calculation,  (See also CNSR upgrades)
2339-  20 30 F8  JSR  $F830      and send last info/param. to #2
233C-  4C 71 F3  JMP  $F371      and skip out of interrupt ::= END OF RECORD
233F-  EA        NOP
2340-  EA        NOP

2341-  A5 A3     LDA  $A3        is this the 1st pulse in the coh. integral?
2343-  F0 09     BEQ  $234E      no
2345-  20 DA F3  JSR  $F3DA      yes,so read PIAs and store amps in L byte of sum
2348-  A9 00     LDA  #$00
234A-  85 A3     STA  $A3        clear "1st pulse" flag
234C-  F0 03     BEQ  $2351       Forced branch
234E-  20 50 F4  JSR  $F450      Read PIAs and add amps to accumulators
2351-  C6 A6     DEC  $A6         Next pulse in coh. integral; = last?
2353-  D0 1C     BNE  $2371       No
2355-  A5 A5     LDA  $A5         yes, was analysis fini. on previous coh. avg.??
2357-  F0 0B     BEQ  $2364        yes, so Ok
2359-  A5 FE     LDA  $FE         NO!!! so set error flag in status byte going to #2
235B-  09 01     ORA  #$01
235D-  85 FE     STA  $FE
235F-  A9 04     LDA  #$04        turn on "DATA OVERLOAD" LED
2361-  20 40 F5  JSR  $F540       ....and keep going.... CPU#2 will"remember"the overload
2364-  20 20 F6  JSR  $F620      reverse coh. sum store addresses
2367-  A9 01     LDA  #$01
2369-  85 A5     STA  $A5        set "data ready for analysis" flag
236B-  85 A3     STA  $A3        set "1st pulse in coh. integr." flag
236D-  A5 E6     LDA  $E6
236F-  85 A6     STA  $A6        reload pulse counter for Coh. integral
2371-  20 C0 F3  JSR  $F3C0      reload gains for next pulse
2374-  A5 A1     LDA  $A1        phase flip required??
2376-  F0 08     BEQ  $2380      no
2378-  AD 10 C7  LDA  $C710      yes,so take last phase and reverse it
237B-  49 80     EOR  #$80                    (bit 7 = phase control)
237D-  8D 10 C7  STA  $C710
2380-  A9 10     LDA  #$10
2382-  20 50 F5  JSR  $F550      turn off "in IRQ" LED
2385-  AD 12 C7  LDA  $C712      clear interrupt flag
2388-  68        PLA
2389-  AA        TAX
238A-  68        PLA                 restore regs from stack
238B-  A8        TAY
238C-  68        PLA
238D-  28        PLP
238E-  40        RTI
```

SOFTWARE GATING CONTROL

```
2390-  A4 E7      LDY   $E7     delay to 1st gate
2392-  88         DEY
2393-  D0 FD      BNE   $2392
2395-  A5 E9      LDA   $E9     turn on clocking with selected gate sep. ($8==20μsec)
2397-  8D 12 C7   STA   $C712   by storing in here with bit 7 low   For Apple: 1.026 MHz !
239A-  A4 E8      LDY   $E8      3      clockin delay: should be a multiple of 20 cycles
239C-  A2 02      LDX   #$02     2                  of Apple clock
239E-  EA         NOP            2
239F-  EA         NOP            2
23A0-  CA         DEX            2
23A1-  D0 FD      BNE   $23A0   3/2                    Note: can only control
23A3-  88         DEY            2                     clocking On/Off  at
23A4-  D0 F6      BNE   $239C   3/2                    20 μsec intervals
23A6-  EA         NOP            2
23A7-  EA         NOP            2
23A8-  EA         NOP            2
23A9-  EA         NOP            2
23AA-  A5 E9      LDA   $E9      3    keep same gate separation
23AC-  09 80      ORA   #$80     2    set bit 7 high
23AE-  8D 12 C7   STA   $C712    4    = stop clocking (PB7 = high)
23B1-  60         RTS
```

Load Shift registers with Gains

```
23C0-  A0 02      LDY   #$02    twice; to fill 64 shifts
23C2-  A2 20      LDX   #$20
23C4-  B5 BF      LDA   $BF,X
23C6-  09 C0      ORA   #$C0
23C8-  8D 12 C7   STA   $C712
23CB-  CA         DEX
23CC-  D0 F6      BNE   $23C4
23CE-  88         DEY
23CF-  D0 F1      BNE   $23C2
23D1-  60         RTS
```

READ AND STORE 1ST DATA IN COHERENT INTEGRAL

(also send to external system, if any)

```
23DA-  A0 00      LDY   #$00
23DC-  A9 00      LDA   #$00     clear H-byte of sum
23DE-  91 AB      STA   ($AB),Y
23E0-  88         DEY
23E1-  D0 FB      BNE   $23DE
23E3-  A0 FF      LDY   #$FF
23E5-  AD 02 C7   LDA   $C702    I1
23E8-  91 AD      STA   ($AD),Y  store in L-byte of accum
23EA-  99 00 03   STA   $0300,Y  save for A/D bit check
23ED-  8D 1A C7   STA   $C71A    send out to external system
23F0-  88         DEY
23F1-  AD 00 C7   LDA   $C700    Q1          (this address also strobes shift regs after)
23F4-  91 AD      STA   ($AD),Y
23F6-  99 00 03   STA   $0300,Y
23F9-  8D 1A C7   STA   $C71A
23FC-  88         DEY
23FD-  AD 06 C7   LDA   $C706    I2
2400-  91 AD      STA   ($AD),Y
2402-  99 00 03   STA   $0300,Y
2405-  8D 1A C7   STA   $C71A
2408-  88         DEY
2409-  AD 04 C7   LDA   $C704    Q2          & strobe
240C-  91 AD      STA   ($AD),Y
240E-  99 00 03   STA   $0300,Y
2411-  8D 1A C7   STA   $C71A
2414-  88         DEY
2415-  AD 0A C7   LDA   $C70A    I3
2418-  91 AD      STA   ($AD),Y
241A-  99 00 03   STA   $0300,Y
241D-  8D 1A C7   STA   $C71A
2420-  88         DEY
2421-  AD 08 C7   LDA   $C708    Q3          & strobe
2424-  91 AD      STA   ($AD),Y
2426-  99 00 03   STA   $0300,Y
2429-  8D 1A C7   STA   $C71A
242C-  88         DEY
242D-  AD 0E C7   LDA   $C70E    I4
2430-  91 AD      STA   ($AD),Y
2432-  99 00 03   STA   $0300,Y
2435-  8D 1A C7   STA   $C71A
2438-  88         DEY
2439-  AD 0C C7   LDA   $C70C    Q4          & strobe
243C-  91 AD      STA   ($AD),Y
243E-  99 00 03   STA   $0300,Y
2441-  8D 1A C7   STA   $C71A
2444-  88         DEY
2445-  C0 FF      CPY   #$FF
2447-  F0 03      BEQ   $244C
2449-  4C E5 F3   JMP   $F3E5          next height
244C-  AD 18 C7   LDA   $C718    (signal external system that end of "buffer")
244F-  60         RTS
```

(9)

ACCUMULATE 2ND AND FURTHER pulses in coherent sum

```
2450-   18          CLC
2451-   A0 FF       LDY   ##$FF
2453-   AD 02 C7    LDA   $C702  I1
2456-   8D 1A C7    STA   $C71A
2459-   71 AD       ADC   ($AD),Y    L-byte of accum
245B-   91 AD       STA   ($AD),Y
245D-   90 06       BCC   $2465
245F-   A9 00       LDA   ##$00
2461-   71 AB       ADC   ($AB),Y
2463-   91 AB       STA   ($AB),Y    H-byte of accum
2465-   88          DEY
2466-   AD 00 C7    LDA   $C700   Q1      & strobe
2469-   8D 1A C7    STA   $C71A
246C-   71 AD       ADC   ($AD),Y
246E-   91 AD       STA   ($AD),Y
2470-   90 06       BCC   $2478
2472-   A9 00       LDA   ##$00
2474-   71 AB       ADC   ($AB),Y
2476-   91 AB       STA   ($AB),Y
2478-   88          DEY
2479-   AD 06 C7    LDA   $C706   I2
247C-   8D 1A C7    STA   $C71A
247F-   71 AD       ADC   ($AD),Y
2481-   91 AD       STA   ($AD),Y
2483-   90 06       BCC   $248B
2485-   A9 00       LDA   ##$00
2487-   71 AB       ADC   ($AB),Y
2489-   91 AB       STA   ($AB),Y
248B-   88          DEY
248C-   AD 04 C7    LDA   $C704     Q2    & strobe
248F-   8D 1A C7    STA   $C71A
2492-   71 AD       ADC   ($AD),Y
2494-   91 AD       STA   ($AD),Y
2496-   90 06       BCC   $249E
2498-   A9 00       LDA   ##$00
249A-   71 AB       ADC   ($AB),Y
249C-   91 AB       STA   ($AB),Y
249E-   88          DEY
249F-   AD 0A C7    LDA   $C70A   I3
24A2-   8D 1A C7    STA   $C71A
24A5-   71 AD       ADC   ($AD),Y
24A7-   91 AD       STA   ($AD),Y
24A9-   90 06       BCC   $24B1
24AB-   A9 00       LDA   ##$00
24AD-   71 AB       ADC   ($AB),Y
24AF-   91 AB       STA   ($AB),Y
24B1-   88          DEY
24B2-   AD 08 C7    LDA   $C708     Q3    & strobe
24B5-   8D 1A C7    STA   $C71A
24B8-   71 AD       ADC   ($AD),Y
24BA-   91 AD       STA   ($AD),Y
24BC-   90 06       BCC   $24C4
24BE-   A9 00       LDA   ##$00
24C0-   71 AB       ADC   ($AB),Y
24C2-   91 AB       STA   ($AB),Y
24C4-   88          DEY
24C5-   AD 0E C7    LDA   $C70E   I4
24C8-   8D 1A C7    STA   $C71A
24CB-   71 AD       ADC   ($AD),Y
24CD-   91 AD       STA   ($AD),Y
24CF-   90 06       BCC   $24D7
24D1-   A9 00       LDA   ##$00
24D3-   71 AB       ADC   ($AB),Y
24D5-   91 AB       STA   ($AB),Y
24D7-   88          DEY
24D8-   AD 0C C7    LDA   $C70C     Q4    and strobe
24DB-   8D 1A C7    STA   $C71A
24DE-   71 AD       ADC   ($AD),Y
24E0-   91 AD       STA   ($AD),Y
24E2-   90 06       BCC   $24EA
24E4-   A9 00       LDA   ##$00
24E6-   71 AB       ADC   ($AB),Y
24E8-   91 AB       STA   ($AB),Y
24EA-   88          DEY
24EB-   C0 FF       CPY   ##$FF
24ED-   F0 03       BEQ   $24F2
24EF-   4C 53 F4    JMP   $F453   next height
24F2-   AD 18 C7    LDA   $C718   signal "end of buffer" (not used by COHRTW system)
24F5-   60          RTS
```

```
24F8-   A5 E1       LDA   $E1   local/test mode?      Send data and status to #2
24FA-   F0 01       BEQ   $24FD no,so do send
24FC-   60          RTS         yes, so no send.
24FD-   EA          NOP
24FE-   EA          NOP
24FF-   EA          NOP
2500-   A9 28       LDA   #$28       select DDR
2502-   8D 17 C7    STA   $C717
2505-   A9 FF       LDA   #$FF       all bits o/p
2507-   8D 16 C7    STA   $C716
250A-   A9 2C       LDA   #$2C       select port, strobe CB2 on STA port
250C-   8D 17 C7    STA   $C717
250F-   A2 FF       LDX   #$FF       send bit amp data, 256 bytes(=8bits/Rx/Comp/ht)
2511-   BD 00 02    LDA   $0200.X
2514-   8D 16 C7    STA   $C716      load data (auto stribe) into port
2517-   2C 17 C7    BIT   $C717      wait for response from #2
251A-   10 FB       BPL   $2517
251C-   AD 16 C7    LDA   $C716      clear interrrupt flag
251F-   CA          DEX              next byte
2520-   E0 FF       CPX   #$FF       last byte?
2522-   D0 ED       BNE   $2511      no, so loop
2524-   EA          NOP
2525-   A2 08       LDX   #$08       Ok, now send 8 byte of status info, same procedure
2527-   B5 F7       LDA   $F7.X
2529-   8D 16 C7    STA   $C716
252C-   2C 17 C7    BIT   $C717
252F-   10 FB       BPL   $252C
2531-   AD 16 C7    LDA   $C716
2534-   CA          DEX
2535-   D0 F0       BNE   $2527        interrupt flag left clear at end
2537-   EA          NOP
2538-   60          RTS
```

```
2540-   0D 14 C7    ORA   $C714      TURN ON LED(s) INDICATED BY
2543-   8D 14 C7    STA   $C714      HIGH BITS IN A
2546-   60          RTS
```

```
2550-   49 FF       EOR   #$FF       TURN OFF LED(s) INDICATED BY
2552-   2D 14 C7    AND   $C714        HIGH BITS IN A
2555-   8D 14 C7    STA   $C714
2558-   60          RTS
```

```
2560-   A5 E1       LDA   $E1   local/test mode?      READ COMMAND DATA FROM #2
2562-   F0 01       BEQ   $2565 no
2564-   60          RTS         yes, so skip out
2565-   A9 20       LDA   #$20
2567-   20 40 F5    JSR   $F540      turn on "in Receive CMD" LED
256A-   A9 28       LDA   #$28       select DDR
256C-   8D 17 C7    STA   $C717
256F-   A9 00       LDA   #$00       all bits i/p
2571-   8D 16 C7    STA   $C716
2574-   A9 2D       LDA   #$2D       select port, strobe on STA port
2576-   8D 17 C7    STA   $C717
2579-   AD 16 C7    LDA   $C716      clear interrupt flag
257C-   A0 30       LDY   #$30       loop to read 48 bytes
257E-   8D 16 C7    STA   $C716      strobe CB2
2581-   AD 17 C7    LDA   $C717      wait for #2 response (bit 7 of cntl reg.)
2584-   10 FB       BPL   $2581
2586-   AD 16 C7    LDA   $C716      read data (which also clears the irq flag)
2589-   99 FF 00    STA   $00FF.Y    save it temporarily at bottom of stack
258C-   88          DEY
258D-   D0 EF       BNE   $257E
258F-   AD 01 01    LDA   $0101      test whether comm. link check bytes correct
2592-   F0 0D       BEQ   $25A1      ok
2594-   A9 01       LDA   #$01       Bad: turn on "Comm. Err" LED
2596-   20 40 F5    JSR   $F540
2599-   A5 FE       LDA   $FE        and save error status condition in status byte for #2
259B-   09 02       ORA   #$02
259D-   85 FE       STA   $FE
259F-   D0 07       BNE   $25A8      forced branch
25A1-   AD 00 01    LDA   $0100      test other check byte
25A4-   C9 FF       CMP   #$FF
25A6-   D0 EC       BNE   $2594      Bad.
25A8-   A2 20       LDX   #$20       Move gains to normal storage area
25AA-   BD 01 01    LDA   $0101.X
25AD-   95 BF       STA   $BF.X
25AF-   CA          DEX
25B0-   D0 F8       BNE   $25AA
```

```
25B2-  A2 04     LDX  #$04      get other param (DLY,GSEP,CLKIN, rec length ...etc)
25B4-  BD 26 01  LDA  $0126,X
25B7-  95 E5     STA  $E5,X
25B9-  CA        DEX
25BA-  D0 F8     BNE  $25B4
25BC-  AD 26 01  LDA  $0126
25BF-  85 EB     STA  $EB       L (#SENDS)
25C1-  85 EE     STA  $EE
25C3-  AD 25 01  LDA  $0125
25C6-  85 EA     STA  $EA       H  ( " )   (USUALLY = 0 )
25C8-  85 EF     STA  $EF
25CA-  AD 24 01  LDA  $0124
25CD-  85 A1     STA  $A1       } load bit-amp counter   (8 bit-amps per send)
25CF-  A9 08     LDA  #$08
25D1-  85 A0     STA  $A0       { ok, NOW is normal COHRTW run wanted, or special cmd?
25D3-  AD 2F 01  LDA  $012F     { = normal
25D6-  F0 03     BEQ  $25DB     = special (e.g. accept program and run it)
25D8-  4C F0 F8  JMP  $F8F0     turn off "in receive cmd" LED
25DB-  A9 20     LDA  #$20
25DD-  4C 50 F5  JMP  $F550
```

INITIALIZE PIAs

```
25E0-  A2 1C     LDX  #$1C
25E2-  E0 18     CPX  #$18
25E4-  F0 08     BEQ  $25EE
25E6-  A9 00     LDA  #$00      select DDR
25E8-  9D FF C6  STA  $C6FF,X
25EB-  9D FE C6  STA  $C6FE,X   set all bits i/p  (will change a couple later)
25EE-  CA        DEX
25EF-  CA        DEX
25F0-  D0 F0     BNE  $25E2
25F2-  A9 FF     LDA  #$FF      set = o/p (gains and clocking control)
25F4-  8D 12 C7  STA  $C712
25F7-  8D 14 C7  STA  $C714     LEDs  (all o/p)
25FA-  8D 1A C7  STA  $C71A     external raw data port(all o/p)
25FD-  A9 C0     LDA  #$C0      Tx cntl,test switch(s) etc (nput/output )
25FF-  8D 10 C7  STA  $C710
2602-  A2 1C     LDX  #$1C
2604-  E0 18     CPX  #$18
2606-  F0 05     BEQ  $260D
2608-  A9 2D     LDA  #$2D      set control regs. of PIAs (nd select ports)
260A-  9D FF C6  STA  $C6FF,X       all cntl regs:  irq on ↓ CA1
260D-  CA        DEX                   (but irq line only connected on Rx gains/
260E-  CA        DEX                    clocking board)
260F-  D0 F3     BNE  $2604
2611-  A9 00     LDA  #$00      turn off all LEDs
2613-  8D 14 C7  STA  $C714
2616-  A9 40     LDA  #$40      turn off Tx (not used)
2618-  8D 10 C7  STA  $C710
261B-  60        RTS
```

REVERSE COHERENT SUM STORE ADDRESSES

```
2620-  A2 04     LDX  #$04
2622-  B5 A6     LDA  $A6,X
2624-  48        PHA
2625-  B5 AA     LDA  $AA,X
2627-  95 A6     STA  $A6,X
2629-  68        PLA
262A-  95 AA     STA  $AA,X
262C-  CA        DEX
262D-  D0 F3     BNE  $2622
262F-  60        RTS
```

GET I/Q or Q/I for selected height

o/p= min*100/max

(top bit set in byte if Q/I)

```
2640-  A9 04     LDA  #$04
2642-  85 EC     STA  $EC       Rx counter
2644-  A0 08     LDY  #$08      point to Σx
2646-  B9 5F 09  LDA  $095F,Y   H2 (I1)
2649-  88        DEY
264A-  D9 5F 09  CMP  $095F,Y   H2 (Q1)
264D-  D0 1E     BNE  $266D
264F-  C8        INY
2650-  B9 67 09  LDA  $0967,Y   H1
2653-  88        DEY
2654-  D9 67 09  CMP  $0967,Y       first see whether I>Q or not
2657-  D0 14     BNE  $266D
2659-  C8        INY
265A-  B9 6F 09  LDA  $096F,Y
265D-  88        DEY                                  >
265E-  D9 6F 09  CMP  $096F,Y
2661-  D0 0A     BNE  $266D
2663-  EA        NOP                I & Q exactly equal, so ratio = 100
2664-  EA        NOP
2665-  A6 EC     LDX  $EC
2667-  A9 64     LDA  #$64      = 100
2669-  C8        INY            and leave Y pointing at I1
266A-  4C FD F6  JMP  $F6FD     skip out to store result
266D-  90 46     BCC  $26B5     see if I>Q
```

(12)

```
266F-  C8          INY                    ----------------------------------
2670-  84 ED       STY   $ED                        I > Q
2672-  A0 00       LDY   #$00             divide by 2 until I fits in 1 byte, and
2674-  A6 ED       LDX   $ED               save the number of shifts in Y
2676-  18          CLC
2677-  7E 5F 09    ROR   $095F,X
267A-  7E 67 09    ROR   $0967,X
267D-  7E 6F 09    ROR   $096F,X
2680-  C8          INY
2681-  BD 5F 09    LDA   $095F,X
2684-  D0 F0       BNE   $2676
2686-  BD 67 09    LDA   $0967,X
2689-  D0 EB       BNE   $2676
268B-  BD 6F 09    LDA   $096F,X
268E-  48          PHA                 = I/2^n
268F-  CA          DEX
2690-  18          CLC
2691-  7E 5F 09    ROR   $095F,X
2694-  7E 67 09    ROR   $0967,X        divide Q by the same amount
2697-  7E 6F 09    ROR   $096F,X
269A-  88          DEY
269B-  D0 F3       BNE   $2690
269D-  BD 6F 09    LDA   $096F,X
26A0-  85 F0       STA   $F0           = Q/2^n
26A2-  A9 64       LDA   #$64          = 100
26A4-  85 F1       STA   $F1
26A6-  20 10 F8    JSR   $F810         multiply to get 100Q/2^n   in F3(H),F4(L)
26A9-  68          PLA                 get back I/2^n
26AA-  85 F1       STA   $F1
26AC-  20 E0 F7    JSR   $F7E0         divide
26AF-  A5 F0       LDA   $F0           = result = 100Q/I
26B1-  09 80       ORA   #$80          set bit-7 to indicate sense of ratio,
26B3-  D0 48       BNE   $26FD          go to store result
                   ------------------------------------------
26B5-  C8          INY                                Q > I
26B6-  84 ED       STY   $ED
26B8-  A0 00       LDY   #$00                exactly the same procedur as above
26BA-  A6 ED       LDX   $ED                 but I and Q reversed, and o/p is
26BC-  CA          DEX
26BD-  18          CLC                          100 I/2^n
26BE-  7E 5F 09    ROR   $095F,X                ------     with bit-7 cleared
26C1-  7E 67 09    ROR   $0967,X               Q/2^n
26C4-  7E 6F 09    ROR   $096F,X
26C7-  C8          INY
26C8-  BD 5F 09    LDA   $095F,X
26CB-  D0 F0       BNE   $26BD
26CD-  BD 67 09    LDA   $0967,X
26D0-  D0 EB       BNE   $26BD
26D2-  BD 6F 09    LDA   $096F,X
26D5-  48          PHA
26D6-  E8          INX
26D7-  18          CLC
26D8-  7E 5F 09    ROR   $095F,X
26DB-  7E 67 09    ROR   $0967,X
26DE-  7E 6F 09    ROR   $096F,X
26E1-  88          DEY
26E2-  D0 F3       BNE   $26D7
26E4-  BD 6F 09    LDA   $096F,X
26E7-  85 F0       STA   $F0
26E9-  A9 64       LDA   #$64
26EB-  85 F1       STA   $F1
26ED-  20 10 F8    JSR   $F810
26F0-  68          PLA
26F1-  85 F1       STA   $F1
26F3-  20 E0 F7    JSR   $F7E0
26F6-  A5 F0       LDA   $F0           result
26F8-  EA          NOP
26F9-  EA          NOP
26FA-  EA          NOP
26FB-  EA          NOP
26FC-  EA          NOP
26FD-  A6 EC       LDX   $EC           store flagged ratio
26FF-  9D 99 02    STA   $0299,X
2702-  A4 ED       LDY   $ED
2704-  88          DEY                   move pointer to next Rx
2705-  88          DEY
2706-  C6 EC       DEC   $EC
2708-  F0 03       BEQ   $270D
270A-  4C 46 F6    JMP   $F646
270D-  A5 E2       LDA   $E2           save the selected height pointer in the o/p store
270F-  8D 9E 02    STA   $029E                ($2Ø = lowest height = ht#0 ) for CPU#2 display
2712-  60          RTS                 ----------------------------------
```

(13)

```
2720-  A9 01     LDA  #$01   set bit-check request        RECORD INITIALIZE
2722-  85 AF     STA  $AF
2724-  85 A3     STA  $A3    set "1st pulse in coh.∫"flag
2726-  A5 E1     LDA  $E1    local/test mode?
2728-  F0 05     BEQ  $272F  no
272A-  A9 02     LDA  #$02   yes,so turn on "local oper." LED
272C-  20 40 F5  JSR  $F540
272F-  A2 B8     LDX  #$B8      clear Σmag, Σmag ; and H2,H1,L (stores for
2731-  A9 00     LDA  #$00                          selected ht.)
2733-  9D BF 08  STA  $08BF,X
2736-  CA        DEX
2737-  D0 FA     BNE  $2733
2739-  20 60 F5  JSR  $F560    go to read command data from CPU#2
273C-  AD 2E 01  LDA  $012E     save "skip LTMs" info (bit 7=1 if skip)
273F-  85 FD     STA  $FD
2741-  20 B0 F8  JSR  $F8B0     send gains and I.D. & test data to external port
2744-  EA        NOP
2745-  EA        NOP
2746-  EA        NOP
2747-  EA        NOP
2748-  A5 E6     LDA  $E6    load coherent ∫ pulse counter
274A-  85 A6     STA  $A6
274C-  A9 00     LDA  #$00   clear "data ready for analysis" flag
274E-  85 A5     STA  $A5
2750-  A9 00     LDA  #$00   clear "data ready to send" flag
2752-  85 A4     STA  $A4
2754-  60        RTS
```

```
2760-  A2 20     LDX  #$20       height cntr          ACCUMULATE MEAN MAGNITUDES
2762-  BD DF 08  LDA  $08DF,X   Σmag(L)
2765-  85 9F     STA  $9F                             (done once for every 8 coh. avgs)
2767-  BD BF 08  LDA  $08BF,X   Σmag(H)
276A-  85 9E     STA  $9E                             NB: mag is done over 4 Rx's and
276C-  18        CLC                                   8 coh. avgs.; so have to div-by-
276D-  66 9E     ROR  $9E       div-by-2               -32 to get  mag
276F-  66 9F     ROR  $9F
2771-  A4 9F     LDY  $9F
2773-  B9 00 FC  LDA  $FC00,Y
2776-  A4 9E     LDY  $9E       and use nybble transfers
2778-  19 00 FB  ORA  $FB00,Y   to get Σmag /32 = mag
277B-  18        CLC
277C-  7D 3F 09  ADC  $093F,X   add result to Σmag  accumulator
277F-  9D 3F 09  STA  $093F,X    L
2782-  90 08     BCC  $278C
2784-  FE 1F 09  INC  $091F,X    H1 of Σmag
2787-  D0 03     BNE  $278C
2789-  FE FF 08  INC  $08FF,X  H2
278C-  A9 00     LDA  #$00          at the same time, clear Σmag store to be rdy
278E-  9D DF 08  STA  $08DF,X        for the next 8 coh. avgs
2791-  9D BF 08  STA  $08BF,X
2794-  CA        DEX                --- next height
2795-  D0 CB     BNE  $2762
2797-  60        RTS
```

```
27A0-  A9 00     LDA  #$00   bottom of interval          SQUARE ROOT
27A2-  85 98     STA  $98
27A4-  A9 FF     LDA  #$FF   top of interval            method is to look up squares of
27A6-  85 99     STA  $99                               end points of an interval
27A8-  A2 09     LDX  #$09   do 8 iterations            containing the value, and
27AA-  CA        DEX                                    successively reduce the inter-
27AB-  F0 21     BEQ  $27CE     done!                   val size by ½ each iteration,
27AD-  A5 98     LDA  $98                               keeping the value inside.
27AF-  18        CLC            get centre of interval
27B0-  65 99     ADC  $99
27B2-  6A        ROR
27B3-  A8        TAY            in Y
27B4-  B9 00 FE  LDA  $FE00,Y    and look up Y²(H)
27B7-  C5 9C     CMP  $9C       Compare with H-byte of input number
27B9-  90 0F     BCC  $27CA     Y² is smaller than (i/p)²,so replace lower limit with Y
27BB-  D0 09     BNE  $27C6     Y² is greater "  "  "  "  upper  "  "  "
27BD-  B9 00 FD  LDA  $FD00,Y   high bytes are equal, so test low byte
27C0-  C5 9B     CMP  $9B
27C2-  F0 0A     BEQ  $27CE     still equal, so Y is the SQRT! how about that!
27C4-  90 04     BCC  $27CA     Y² is smaller
27C6-  84 99     STY  $99       Y² is larger,so replace upper limit with Y
27C8-  D0 E0     BNE  $27AA     -forced branch
27CA-  84 98     STY  $98       Y² is smaller,so replace lower limit with Y
27CC-  D0 DC     BNE  $27AA     --forced branch
27CE-  98        TYA            leave the SQRT value in A
27CF-  60        RTS
```

```
27E0-  A9 00      LDA   #$00
27E2-  85 F0      STA   $F0
27E4-  85 F2      STA   $F2        !!
27E6-  A2 08      LDX   #$08
27E8-  18         CLC
27E9-  66 F1      ROR   $F1
27EB-  66 F2      ROR   $F2
27ED-  A5 F4      LDA   $F4
27EF-  38         SEC
27F0-  E5 F2      SBC   $F2
27F2-  85 F5      STA   $F5
27F4-  A5 F3      LDA   $F3
27F6-  E5 F1      SBC   $F1
27F8-  90 06      BCC   $2800
27FA-  85 F3      STA   $F3
27FC-  A5 F5      LDA   $F5
27FE-  85 F4      STA   $F4
2800-  26 F0      ROL   $F0
2802-  CA         DEX
2803-  D0 E3      BNE   $27E8
2805-  60         RTS
```

DIVIDE ROUTINE

F3(H),F4(L) ÷ F1 = FØ

remainder in F4

n.b. result must be 1 byte

old version causes minor errors in sig. values

```
2810-  A2 08      LDX   #$08
2812-  A9 00      LDA   #$00
2814-  85 F3      STA   $F3
2816-  18         CLC
2817-  2A         ROL
2818-  26 F3      ROL   $F3
281A-  06 F0      ASL   $F0
281C-  90 08      BCC   $2826
281E-  18         CLC
281F-  65 F1      ADC   $F1
2821-  90 03      BCC   $2826
2823-  E6 F3      INC   $F3
2825-  18         CLC
2826-  CA         DEX
2827-  D0 EE      BNE   $2817
2829-  85 F4      STA   $F4
282B-  60         RTS
```

MULTIPLY ROUTINE

FØ*F1 = F3(H), F4(L)

```
2830-  A0 20      LDY   #$20       height loop
2832-  98         TYA
2833-  AA         TAX
2834-  20 30 FA   JSR   $FA30
2837-  EA         NOP
2838-  EA         NOP
2839-  EA         7 x NOP
2840-  20 E0 F7   JSR   $F7E0      divide to get mean (sum of mean mags)
2843-  A5 F0      LDA   $F0
2845-  99 DF 02   STA   $02DF,Y    store result in o/p store going to CPU#2
2848-  88         DEY              next height.
2849-  D0 E7      BNE   $2832
284B-  A0 40      LDY   #$40       get LTMs (8 gains-times-8 channels= 64 LTMS)
284D-  B9 FF 07   LDA   $07FF,Y
2850-  99 9F 02   STA   $029F,Y
2853-  88         DEY
2854-  D0 F7      BNE   $284D
2856-  A9 00      LDA   #$00
2858-  8D 79 02   STA   $0279      set comm. link check bytes $ØØ & $FF
285B-  A9 FF      LDA   #$FF
285D-  8D 78 02   STA   $0278
2860-  A5 E1      LDA   $E1        in local/test mode?
2862-  D0 18      BNE   $287C      yes, so don't send anything
2864-  EA         NOP
2865-  EA         NOP
2866-  A0 88      LDY   #$88       Ok, send $88 bytes
2868-  B9 77 02   LDA   $0277,Y    read from store
286B-  8D 16 C7   STA   $C716      and store in port (auto strobe on STA)
286E-  AD 17 C7   LDA   $C717      wait for handshake
2871-  10 FB      BPL   $286E
2873-  AD 16 C7   LDA   $C716      clear flag (handshake)
2876-  88         DEY              next byte
2877-  D0 EF      BNE   $2868
2879-  EA         NOP
```

LOAD AND SEND FINAL PARAM.

(15)

```
287A-   EA          NOP
287B-   EA          NOP
287C-   A9 FF       LDA   #$FF
287E-   85 F6       STA   $F6
2880-   A2 20       LDX   #$20     height
2882-   BD DF 02    LDA   $02DF.X  get magnitude
2885-   38          SEC
2886-   E9 80       SBC   #$80     minus 128
2888-   B0 04       BCS   $288E    +ve
288A-   49 FF       EOR   #$FF     -ve,so get abs value
288C-   69 01       ADC   #$01
288E-   C5 F6       CMP   $F6      closer to 128 than any previous difference?
2890-   B0 04       BCS   $2896    no
2892-   86 E2       STX   $E2      yes, so save height pointer in permanent store, and
2894-   85 F6       STA   $F6      save difference.
2896-   CA          DEX            Next height
2897-   D0 E9       BNE   $2882
2899-   A9 00       LDA   #$00     Clear "test mode" flag
289B-   85 E1       STA   $E1          (if any), so next record is under normal control.
289D-   8D 14 C7    STA   $C714    turn off LEDs
28A0-   A9 40       LDA   #$40     turn off Tx (not used), and set phase = 0°
28A2-   8D 10 C7    STA   $C710
28A5-   60          RTS
```

look for best height to do I/Q,Q/I stuff for next record, want the height with magnitude closest to 128

```
28B0-   A2 20       LDX   #$20
28B2-   B5 BF       LDA   $BF.X    32
28B4-   8D 1A C7    STA   $C71A    gains
28B7-   CA          DEX
28B8-   D0 F8       BNE   $28B2
28BA-   A2 03       LDX   #$03
28BC-   B5 E6       LDA   $E6.X
28BE-   8D 1A C7    STA   $C71A
28C1-   CA          DEX
28C2-   D0 F8       BNE   $28BC
28C4-   A9 D0       LDA   #$D0
28C6-   A2 00       LDX   #$00
28C8-   F0 06       BEQ   $28D0
28CA-   8A          TXA
28CB-   8D 1A C7    STA   $C71A
28CE-   49 FF       EOR   #$FF
28D0-   8D 1A C7    STA   $C71A
28D3-   CA          DEX
28D4-   E0 91       CPX   #$91
28D6-   D0 F2       BNE   $28CA
28D8-   AD 18 C7    LDA   $C718
28DB-   A9 00       LDA   #$00
28DD-   EA          NOP
28DE-   EA          NOP
28DF-   EA          NOP
```

SEND GAINS AND PARAM TO EXTERNAL RAW DATA PORT (not used by COHRTW)

DLY,CLKIN,GSEP (not necess. in tha t order!)

I.D. "P" = $DØ hex = "PARK"
other I.D.s will be
"X" portable ($D8)
~~"G" Calgary ($C3)~~ "S" SYLVAN L. ($D3)
"L" London ($CC)

the rest of the 256 byte "buffer" is filled with test data, to see if link to external system is ok. Starts: 00,FF,FF,00,FE,01... (I think)

send "buffer full" signal

and wait for ~.16 sec, so external system doesn't get hit with 2 buffers full too fast

```
28E0-   38          SEC
28E1-   48          PHA
28E2-   E9 01       SBC   #$01
28E4-   D0 FC       BNE   $28E2
28E6-   68          PLA
28E7-   E9 01       SBC   #$01
28E9-   D0 F6       BNE   $28E1
28EB-   60          RTS
```

WAIT

routine from apple monitor
length of delay dependes on A

```
28F0-   AD 2F 01    LDA   $012F
28F3-   EA          NOP
28F4-   EA          NOP
28F5-   EA          NOP
28F6-   4C A0 F9    JMP   $F9A0
```

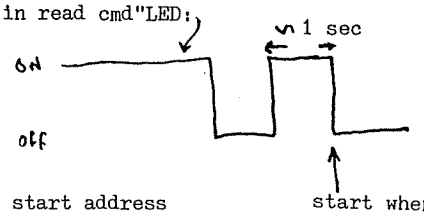
Special Command Sorting

There is only one special command at present so this routine does nothing.

```
2900-  A9 80      LDA  #$80   turn off"test ok" LED    System SELF-TEST
2902-  20 50 F5   JSR  $F550
2905-  A5 FE      LDA  $FE    indicate "test bad" in status going to #2
2907-  09 04      ORA  #$04
2909-  85 FE      STA  $FE
290B-  A9 08      LDA  #$08   turn on "test bad" LED
290D-  20 40 F5   JSR  $F540
2910-  A9 00      LDA  #$00   Do checksum FØØØ-FEFF  to test EPROM
2912-  85 19      STA  $19
2914-  A9 F0      LDA  #$F0
2916-  85 1A      STA  $1A
2918-  A0 00      LDY  #$00
291A-  A9 00      LDA  #$00
291C-  85 1B      STA  $1B
291E-  85 1C      STA  $1C
2920-  B1 19      LDA  ($19),Y
2922-  18         CLC
2923-  65 1B      ADC  $1B
2925-  85 1B      STA  $1B
2927-  90 02      BCC  $292B
2929-  E6 1C      INC  $1C
292B-  C8         INY
292C-  D0 F2      BNE  $2920
292E-  E6 1A      INC  $1A
2930-  A5 1A      LDA  $1A
2932-  C9 FF      CMP  #$FF
2934-  D0 EA      BNE  $2920
2936-  A5 1B      LDA  $1B
2938-  CD DE FF   CMP  $FFDE   L  compare with correct checksum stored on EPROM
293B-  F0 02      BEQ  $293F
293D-  D0 07      BNE  $2946
293F-  A5 1C      LDA  $1C
2941-  CD DF FF   CMP  $FFDF   H
2944-  F0 05      BEQ  $294B
2946-  A9 04      LDA  #$04   test BAD!! so light LED ("analysis too slow")
2948-  20 40 F5   JSR  $F540
294B-  A2 02      LDX  #$02      Now test RAM by storing 00 (1st loop) and FF(2nd loop)
294D-  A9 00      LDA  #$00         in each RAM byte and reading back
294F-  85 1B      STA  $1B   = test byte  (1st loop)
2951-  A0 00      LDY  #$00
2953-  84 19      STY  $19    start at $2ØØ   (can't mess up stack!!, so dodge it)
2955-  A9 02      LDA  #$02
2957-  85 1A      STA  $1A
2959-  A5 1B      LDA  $1B    get test byte
295B-  91 19      STA  ($19),Y  save it
295D-  B1 19      LDA  ($19),Y  get it back
295F-  C5 1B      CMP  $1B      is it the same?
2961-  D0 34      BNE  $2997    no, grrr! so bad ram , skip out
2963-  C8         INY           ok, next byte to test
2964-  D0 F3      BNE  $2959
2966-  E6 1A      INC  $1A      next page of RAM
2968-  A5 1A      LDA  $1A
296A-  C9 08      CMP  #$08      trick to dodge over LTM store (otherwise they won't be
296C-  D0 04      BNE  $2972                    LTMs!!)  $800-8BF
296E-  A9 C0      LDA  #$C0
2970-  85 19      STA  $19            end at $9FF
2972-  C9 0A      CMP  #$0A
2974-  D0 E3      BNE  $2959
2976-  A9 FF      LDA  #$FF        for next loop, store and read $FF
2978-  85 1B      STA  $1B
297A-  CA         DEX
297B-  D0 D4      BNE  $2951
297D-  AD 14 C7   LDA  $C714   RAM ok
2980-  29 0E      AND  #$0E    Was the "TEST BAD" LED the only one 'on' ?
2982-  C9 08      CMP  #$08
2984-  D0 10      BNE  $2996   No, so the EPROM test must have been bad, so leave as is
2986-  A9 0E      LDA  #$0E    Yes, so turn off all red LEDs (except comm link err.)
2988-  20 50 F5   JSR  $F550
298B-  A9 80      LDA  #$80    and turn on "test ok" LED
298D-  20 40 F5   JSR  $F540
2990-  A5 FE      LDA  $FE
2992-  29 FB      AND  #$FB    and clear message from status byte
2994-  85 FE      STA  $FE
2996-  60         RTS          ---------------
2997-  A9 02      LDA  #$02    RAM test bad, so light "In local/test mode" LED
2999-  20 40 F5   JSR  $F540          and leave "test BAD" LED on.
299C-  60         RTS
```

(17)

```
29A0-  AD 2E 01   LDA  $012E H                    "Load & Go"
29A3-  85 01      STA  $01        program start addr.
29A5-  AD 2D 01   LDA  $012D L     from header     Load program from CPU#2 and
29A8-  85 00      STA  $00                         run it.
29AA-  A0 00      LDY  #$00                          (48 byte header has already
29AC-  20 E0 F8   JSR  $F8E0      ~0.3 sec delay        been read)
29AF-  20 E0 F8   JSR  $F8E0
29B2-  EA         NOP
29B3-  EA         NOP
29B4-  8D 16 C7   STA  $C716      strobe to #2 (continuation of read started
29B7-  AD 17 C7   LDA  $C717  wait for hndshk          in header read)
29BA-  10 FB      BPL  $29B7
29BC-  AD 16 C7   LDA  $C716      read byte
29BF-  91 00      STA  ($00),Y    store it
29C1-  E6 00      INC  $00        next byte
29C3-  D0 02      BNE  $29C7
29C5-  E6 01      INC  $01        next page
29C7-  CE 2B 01   DEC  $012B      decrement byte count L
29CA-  D0 EB      BNE  $29B4
29CC-  CE 2C 01   DEC  $012C      decrement byte count H
29CF-  10 E3      BPL  $29B4
29D1-  EA         NOP
29D2-  EA         NOP                  NB:"in read cmd"  LED remains on during
29D3-  EA         NOP                     program read
29D4-  A2 02      LDX  #$02            Now show that program has been read and
29D6-  D0 0B      BNE  $29E3             started:
29D8-  20 00 FA   JSR  $FA00
29DB-  A9 20      LDA  #$20       "in read cmd"LED:
29DD-  20 40 F5   JSR  $F540
29E0-  20 00 FA   JSR  $FA00
29E3-  A9 20      LDA  #$20
29E5-  20 50 F5   JSR  $F550
29E8-  CA         DEX
29E9-  D0 ED      BNE  $29D8
29EB-  AD 2E 01   LDA  $012E
29EE-  85 01      STA  $01        load start address
29F0-  AD 2D 01   LDA  $012D
29F3-  85 00      STA  $00
29F5-  6C 00 00   JMP  ($0000)    and GO
29F8-  FF         ???
29F9-  FF         ???
29FA-  FF         ???
29FB-  FF         ???
29FC-  FF         ???
29FD-  FF         ???
29FE-  FF         ???
29FF-  FF         ???
2A00-  A0 05      LDY  #$05            DELAY
2A02-  20 E0 F8   JSR  $F8E0
2A05-  88         DEY                  approx 1 sec for 1MHz CPU clock
2A06-  D0 FA      BNE  $2A02
2A08-  60         RTS
2A09-  FF         ???
2A0A-  FF         ???
2A0B-  FF         ???
```



start when LED goes off for the 2nd time

```
2A10-  A5 EB      LDA  $EB     L byte      Decrement "Send" Counter
2A12-  D0 02      BNE  $2A16
2A14-  C6 EA      DEC  $EA     H byte      EA is left -ve ($FF,or$FE)
2A16-  C6 EB      DEC  $EB                 on result = zero
2A18-  D0 08      BNE  $2A22
2A1A-  A5 EA      LDA  $EA
2A1C-  F0 02      BEQ  $2A20          I know this looks complicated,
2A1E-  10 04      BPL  $2A24          but it works and I can't think of
2A20-  C6 EA      DEC  $EA            a more obvious/shorter way!
2A22-  A5 EA      LDA  $EA
2A24-  60         RTS            I.E.It has to work for ØØ,4Ø; Ø1,4Ø; Ø1,ØØ
2A25-  FF         ???             for example.                    H  L
2A26-  FF         ???
2A27-  FF         ???
2A28-  FF         ???
2A29-  FF         ???
2A2A-  FF         ???
2A2B-  FF         ???
2A2C-  FF         ???
2A2D-  FF         ???
```

```
2A30-   A5 EE       LDA   $EE  L of # sends                  Prepare for divide
2A32-   85 F1       STA   $F1  keep in divisor
2A34-   A5 EF       LDA   $EF  H of # sends
2A36-   F0 12       BEQ   $2A4A     if = zero, then simple 2byte/1byte div.,so out
2A38-   18          CLC
2A39-   7E FF 08    ROR   $08FF,X      shift down Σmag      and # sends until #
2A3C-   7E 1F 09    ROR   $091F,X        sends is one byte, then use normal divide
2A3F-   7E 3F 09    ROR   $093F,X
2A42-   18          CLC
2A43-   66 EF       ROR   $EF
2A45-   66 F1       ROR   $F1
2A47-   4C 34 FA    JMP   $FA34        loop
2A4A-   BD 1F 09    LDA   $091F,X      ok, so to save space&time put
2A4D-   85 F3       STA   $F3          H in dividing routine variable  here
2A4F-   BD 3F 09    LDA   $093F,X
2A52-   85 F4       STA   $F4           and L
2A54-   60          RTS                now every thing is ready for  divide
2A55-   FF          ???
2A56-   FF          ???
2A57-   FF          ???
```

(9)

```
      00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617 18191A1B 1C1D1E1F
2B00:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2B20:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2B40:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2B60:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2B80:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2BA0:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2BC0:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2BE0:00102030 40506070 8090A0B0 C0D0E0F0 00102030 40506070 8090A0B0 C0D0E0F0
2C00:00000000 00000000 00000000 00000000 01010101 01010101 01010101 01010101
2C20:02020202 02020202 02020202 02020202 03030303 03030303 03030303 03030303
2C40:04040404 04040404 04040404 04040404 05050505 05050505 05050505 05050505
2C60:06060606 06060606 06060606 06060606 07070707 07070707 07070707 07070707
2C80:08080808 08080808 08080808 08080808 09090909 09090909 09090909 09090909
2CA0:0A0A0A0A 0A0A0A0A 0A0A0A0A 0A0A0A0A 0B0B0B0B 0B0B0B0B 0B0B0B0B 0B0B0B0B
2CC0:0C0C0C0C 0C0C0C0C 0C0C0C0C 0C0C0C0C 0D0D0D0D 0D0D0D0D 0D0D0D0D 0D0D0D0D
2CE0:0E0E0E0E 0E0E0E0E 0E0E0E0E 0E0E0E0E 0F0F0F0F 0F0F0F0F 0F0F0F0F 0F0F0F0F
2D00:00010409 10192431 40516479 90A9C4E1 00214469 90B9E411 4071A4D9 104984C1
2D20:004184C9 1059A4F1 4091E439 90E944A1 0061C429 90F964D1 40B12499 10890481
2D40:00810489 109924B1 40D164F9 9029C461 00A144E9 9039E491 40F1A459 10C98441
2D60:00C18449 10D9A471 4011E4B9 90694421 00E1C4A9 90796451 40312419 10090401
2D80:00010409 10192431 40516479 90A9C4E1 00214469 90B9E411 4071A4D9 104984C1
2DA0:0041B4C9 1059A4F1 4091E439 90E944A1 0061C429 90F964D1 40B12499 10890481
2DC0:00810489 109924B1 40D164F9 9029C461 00A144E9 9039E491 40F1A459 10C98441
2DE0:00C18449 10D9A471 4011E4B9 90694421 00E1C4A9 90796451 40312419 10090401
2E00:00000000 00000000 00000000 00000000 01010101 01010102 02020202 03030303
2E20:04040404 05050505 06060607 07070808 0909090A 0A0A0B0B 0C0C0D0D 0E0E0F0F
2E40:10101111 12121313 14141515 16171718 19191A1A 1B1C1C1D 1E1E1F20 21212223
2E60:24242526 27272829 2A2B2B2C 2D2E2F30 31313233 34353637 38393A3B 3C3D3E3F
2E80:40414243 44454647 48494A4B 4C4D4E4F 51525354 55565759 5A5B5C5D 5F606162
2EA0:64656667 696A6B6C 6E6F7072 73747677 797A7B7D 7E7F8182 84858788 8A8B8D8E
2EC0:90919394 9697999A 9C9D9FA0 A2A4A5A7 A9AAACAD AFB1B2B4 B6B7B9BB BDBEC0C2
2EE0:C4C5C7C9 CBCCCED0 D2D4D5D7 D9DBDDDF E1E2E4E6 E8EAECEE F0F2F4F6 F8FAFCFE
2F00:80808080 80808080 80808080 80808080 80808080 80808080 80808080 80808080
2F20:80808080 80808080 80808080 80808080 80808080 80808080 80808080 80808080
2F40:00000000 01010101 02020202 03030303 04040404 05050505 06060606 07070707
2F60:00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
2F80:00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
2FA0:00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
2FC0:00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
2FE0:10F308[08] [00][00]0100 00[04]6004 00050006 00070101 043E3F08 [00]483F0 00F0C0F1
```

L→H table

H→L table

X^2-L table

X^2-H table

INIT. LTMS

Init. gains

VECTOR: IRQ final addr. (used after brk/irq test)

VECTORS: NMI RST IRQ

EPROM checksum

(NOT CORRECT HERE!)

store 1 and store 2 addresses

data used in local/test oper.

→ #pulses to integr.
coh avg cntr.
flip cmd
select (0°)
1st pulse in integr. flag
8-avg recs to send (ie.rec.length)

other:

SECTION III.

Section III.: CPU#2 (correlation calculation)

- CPU#2 detailed description (W) [page 55]
- Subroutine list (L) [page 57]
- Port (PIA) address list (L) [page 58]
- Addresses for cross and auto storage vs. height (L) [page 59]
- Zero-page usage (flags, counters, etc.) (L) [page 60]
- Zero-page temporary bit-amplitude storage details (F) [page 61]
- Bulk storage map ($5EB0-BFFF) (F) [page 62]
- I/O memory used in CPU#1 <=> CPU#2 link (F) [page 63]
- Communication link CPU#2 <=> CPU#1 (F) [page 64]
- 5-min delay tripout on stuck system circuit (F) [page 64]
- Program listing (TOTALCPU#2R'-D/N) (L) [page 65]

CPU#2 detailed description

General operation:

On start CPU#2 initializes PIA's, waits for a record start time (usually 0 or 5 min), sets the REC flag, clears correlation accumulators etc. A non-maskable interrupt (NMI) is then sent to #1 and it waits for a interrupt flag from CPU#1. It then sends parameters to #1 (including gains, amount of coherent integration, number of sets of 8 coherent averages per record etc...). Then the PIA control register is changed so that the next link with #1 will start with a hardware interrupt generated by #1. In this and subsequent interrupts (which are spaced by 8x0.533=4.27 sec) it takes 256 bytes data (bit-amplitudes) and 8 bytes status from #1, and when it has counted out the full record (as #1 is doing) it takes the final parameters ($88 bytes)[At this point #1 is finished, and sits waiting for an NMI.], moves the data input pointer ahead entering zeroes so that the analysis will be finished once it catches up with the data pointer, and clears the REC flag.

While the data are being read through interrupt processing, the correlations are continuing in analysis. The present correlation routine requires at least 10 bytes of bit-amps to run(viz 8 bytes data plus 2 bytes overlap- the amount of overlap required is defined by the maximum lag, which is 16), so it runs around all the heights looking for one with enough data to run correlations. When enough data have been loaded in the interrupt processing,it does cross, and auto numbers of 1-matches ,and #1's calculation. The auto correlations are totals over all antennas, but the #1's calculation (which would normally be just 0-lag auto) is separated betwen antennas so that separate means can be used in the calculation of cross correlations in CPU#3.

Since 8 bytes of bit amps (which takes 8x4.27 sec) are required per correlation calculation, the total time available for one 'partial correlation' analysis at all height is 34 sec - however, this analysis has to be completed after #1 is finished sending data, which will take one or two more passes, so it's best to have the correlation calculation as fast as possible. By linear programming (program memory is no object) it was possible to get the analysis down to 4 sec. So the analysis is finished 4-8 sec after #1 stops sending data. The end of analysis is determined by REC=0 and the lack of data to analyse in the previous pass over all heights [another analysis pass is forced by incrementing the # analyses counter at the end of the record in interrupt]. Then the gains for the next record are set, the FCA results are taken from CPU#3 (the C128) and all the correlation data are sent to #3(this will take 1-2 sec). CPU#3 must be ready for this!, otherwise the system displays the "C128 NOT READY" message, resets the 1st record counter (i.e next data from C128 will be discarded), and continues.

Data entry and analysis pointers

A 32 byte store is assigned to each height/receiver/component(I or Q) for bit-amplitude storage. Data entry for all heights is always at the same relative position in each store. The data entry pointer (as seen by the analysis), Doff, is always to the next data entry position, and is decremented each time new data are received from CPU#1.[the first data are placed at the top of the store- index $1F]. If a decrement will make the index negative, it is reloaded with $1F, so that the data are entered in a rotating fashion. After the decrement, a check is made to see whether the next data will write over some data still needed for analysis- if so, CPU#2 displays "DATA OVERLOAD" and "stops" (see footnote, pg.). There is no possibility of this happening in normal operation because the analysis is much (8 times) faster than data entry- it was added as a safety check in tests.

There is a separate analysis pointer for each height: Aoff(h). These are initialized to the top of the store ($1F) at the beginning of the record. The running analysis requires 8 bytes(64 bit-amps) + 2 bytes (16 bit-amps overlap for non-zero lags), so it must wait until enough data have been entered -and usually the analysis will be scanning all heights waiting for enough data. As the analysis is completed for each height, its pointer is moved down by 8 bytes. I.e. the analysis pointer is to the *least* recent data needed. Since the data store is a multiple of 8 bytes, the analysis pointer is always sitting at one of 4 values: viz. $1F, $17, $0F, or $07.

At the end of the record (determined in interrupt, just after the last data are received from CPU#1), the data entry pointer, Doff, can be anywhere depending on the length of record. [For the *normal* 4.5 min data records = 64 data sets received from CPU#1, it will be pointing to $1F.] 10 bytes are always needed in analysis, and to get the correct correlation results, any extra data length required to complete the analysis must be zeroes so there is no contribution to the number of 1-matches. Two zero bytes (16 bits) are needed beyond the end of data anyway for overlap. It can be seen (yes it can....) that adding the least required amount of dummy data will result in a value of Doff = $1D, $15, $0D, or $05. Note that this value is still one beyond the 'dummy' data entered. The table below shows this final value of Doff versus that at the end of the record:

Doff at End of Rec	Final Doff
$17-$1E	$15
$0F-$16	$0D
$07-$0E	$05
$1F,$00-$06	$1D

Note that this may require moving Doff down by as much as 9 bytes all at once. If the analysis were slow, this might overwrite data in use, but since the analysis is so fast, this is another case which will *never* occur except in tests. However, if it does, CPU#2 displays "DATA WRAP AT END" and "stops" (see footnote, pg. 56)

Footnote: At present (CPU#2V') the system doesn't stop; it resets as if the watchdog timeout had triggered, and increments an unused (LH top or bottom) corner screen byte to indicate trouble. The appropriate "problems:" message is left on the screen (unless overwritten by another later).

(Sub)Routine list- COHRTW CPU#2 (CPU#2W)

ADDR.	
$900-	Initialize/start
$914-	Initialize/start keeping old gains
$940-	entry point after internal NMI
$99A-	Start normal operation
$9BD-	Analyse
$A70-	Wait & check key commands (EOF,Break, Man.start)
$AC0-	Get key & check for break (cntl-B)
$ADD-	Update/adjust clock with watchdog reset
$AE0-	Update/adjust clock (no watchdog reset)
$B30-	Verify program by check-sum (1 byte)
$B50-	Decrement "receive data" counter
$B68-	Display amount of integration
$B80-	Clear Correl store
$BA0-	set-up tables
$BE0-	Record initialize
$C90-	Interrupt processing
$D34-	Initialize tape interface
$D40-	Send CMD data to #1
$D80-	Reset gains(single hts)
$DB0-	Sort data from #1
$E12-	Turn on Tx rate (to mark tape store/dump)
$E20-	Display bit check status
$E50-	Display analysis ht
$E70-	Cross correlation (1 ht)
$15C0-	Count#1's for all I,Q
$1640-	Count#1's in a seq of bytes
$1670-	Auto correl (1 ht)
$1920-	Transfer 10 bytes of each bin seq to zero page
$1950-	NMI (on internal timeout) processing
$1970-	Tape dump(incl. rdy chk and reset, and F.G. call
$19D4-	Reset tape drive(load/on-line)
$19F8-	Initialze tape reset/Tx cntl pia,Tx=off
$1A10-	C128 communication
$1AB0-	set Doff param for finishing analysis
$1B10-	check for record start time
$1B30-	Display final param from CPU#1
$1B40-	Display worst Rx offsets
$1BF0-1C18	entries for TROUBLE mess.,and stop
$1C40-	Display diff'l gain for the 4 Rx's, and the selected ht
$1D10-	Setup fixed screen display, incl char data
$1ED0-	Set synch rate,Tx on/off,amt. of integ. based on GMT
$1F07-	check for End Of Tape
$1F20-	increment scrn cntr for "#TR RST"
$1F30-	increment scrn cntr for "INTFCE RST"
$1F50-	Put file gap on tape and clear FG flag
$1F70-	Move C128 output to tape store and dump to tape if full.
$1FB0-	Pack analysed C128 data in tape store.
$1FD1-	Display "CHKSUM-ERR!" message, don't stop
$1FF0-	Increment screen byte - as "problem occurred" message

$1CB4 CONVERT A TO BCD ASCII

CPU#2 PIA list

C0C0 C0C1	comm. link to CPU#3 (C128)	
C0C2 C0C3	spare port (but has to be wired up!)	
C500 C501	comm. link to cpu#1	
C502 C503	CB2 used as NMI cntl cpu#1	
C0F0 C0F1	drive status	tape interface (slot 7)
C0F2 C0F3	read data	
C0F4 C0F5	write data	
C0F6 C0F7	control	

COHRTW-CPU#2

LOCATION OF CROSS AND AUTO CORRELATIONS

HT#		CROSS	AUTO
32	(142Km)	8C40-8DBF	8440-847F
31	(139Km)	8DC0-8F3F	8480-84BF
30	(136Km)	8F40-90BF	84C0-84FF
29	(133Km)	90C0-923F	8500-853F
28	(130Km)	9240-93BF	8540-857F
27	(127Km)	93C0-953F	8580-85BF
26	(124Km)	9540-96BF	85C0-85FF
25	(121Km)	96C0-983F	8600-863F
24	(118Km)	9840-99BF	8640-867F
23	(115Km)	99C0-9B3F	8680-86BF
22	(112Km)	9B40-9CBF	86C0-86FF
21	(109Km)	9CC0-9E3F	8700-873F
20	(106Km)	9E40-9FBF	8740-877F
19	(103Km)	9FC0-A13F	8780-87BF
18	(100Km)	A140-A2BF	87C0-87FF
17	(97Km)	A2C0-A43F	8800-883F
16	(94Km)	A440-A5BF	8840-887F
15	(91Km)	A5C0-A73F	8880-88BF
14	(88Km)	A740-A8BF	88C0-88FF
13	(85Km)	A840-AA3F	8900-893F
12	(82Km)	AA40-ABBF	8940-897F
11	(79Km)	ABC0-AD3F	8980-89BF
10	(76Km)	AD40-AEBF	89C0-89FF
9	(73 Km)	AEC0-B03F	8A00-8A3F
8	(70Km)	B040-B1BF	8A40-8A7F
7	(67Km)	B1C0-B33F	8A80-8ABF
6	(64Km)	B340-B4BF	8AC0-8AFF
5	(61Km)	B4C0-B63F	8B00-8B3F
4	(58Km)	B6C0-B7BF	8B40-8B7F
3	(55Km)	B7C0-B93F	8B80-8BBF
2	(52Km)	B940-BABF	8BC0-8BFF
1	(49Km)	BAC0-BC3F	8C00-8B3F

CPU#2 Zero-page usage (flags and counters)

$19-1F Tape dump parameters(used by ROM program)
$4E-9D bit-amp stores (all antennas, see separate map)
$9E-A9 rolling bit-amps(I comp.)
$AA-AB PBIN, pointer
$AC-AD XCOR/ACOR pointer (also used in tape dump
$AE-AF temporary bin,also xcor cntr for stacking records)
$B0 Counter
$B1 tmp store for Y
$B2 sum(+) for 1-matches
$B3 lag counter
$B4 -ve lag flag
$B5 # analyses counter (# heights done per height cycle)
$B6 "hang-up message", tells where prog stuck
$B7 sum(-) for 1-matches
$B8 height counter
$B9 Aoff(h) for the present h
$BA 1st record flag (tells whether real data from C128)
$BB REC (=1 if record in progress)
$BC ~~DV=1 if looking for day number update~~ not used, new clock board has day#
$BD "pulse" counter (low), actually counts # data xmissions from
          CPU#1, so number of coh. avgs is *this* times 8
$BE "pulse counter (high), not used & prob. never
$BF-C0 Fixed pointer to bit-amps at this analaysis height
$C1-C2 Fixed point ot Xcorrel accum (4V3 Re) at this height
$C3-C4 Fixed point to Acorrel accum (Re) at this height)
$C5 FIle gap flag, set in UPDATE, cleared after FG on tape
$C6 Doff, entry point for *next* bit data from CPU#1
$C7-C8 Interrupt scratch pointer, C7 also used in "Displ. b-chk stat.
$C9 temporary use in SQRT
$CA } {used in interrupt for finish-up
$CB } { operations and display CPU#1
$CC } {   status.
$CD-CE Used in counting # 1's in bit sequences
$CF used in tape store/dump and NOWHERE ELSE
$DE-E9 rolling bit-amps (Q comp.)

CM
Sept.24'86

Coherent RTW analysis: zero page binary info.

Aoffh ↓

← time

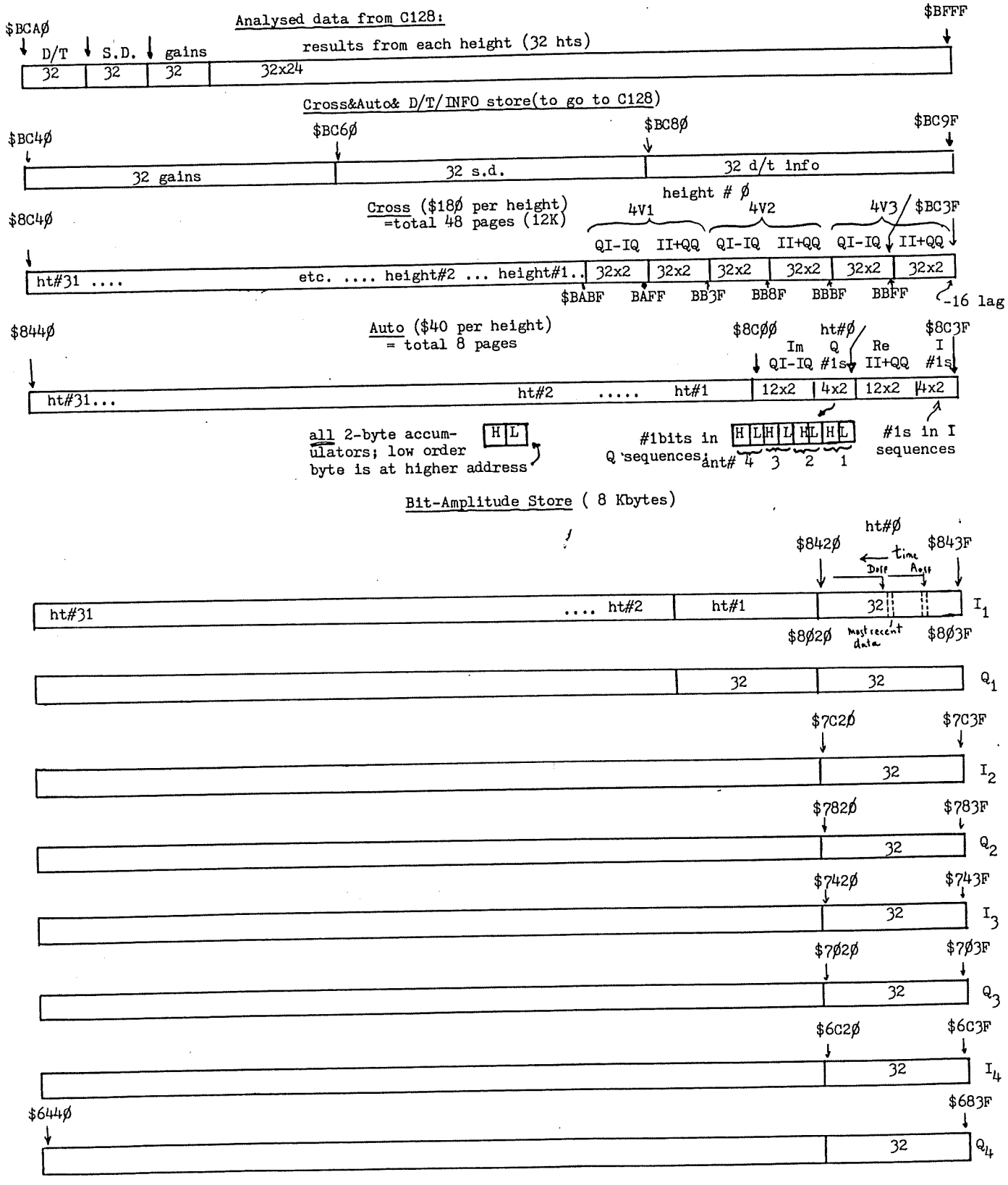
3 antennas
Program
Sense of rotation
in XCOR

94 95 96 97 98 99 9A 9B 9C 9D	I1
8A 8B 8C 8D 8E 8F 9Ø 91 92 93	Q1
8Ø 81 82 83 84 85 86 87 88 89	I2
76 77 78 79 7A 7B 7C 7D 7E 7F	Q2
6C 6D 6E 6F 7Ø 71 72 73 74 75	I3
62 63 64 65 66 67 68 69 6A 6B	Q3
58 59 5A 5B 5C 5D 5E 5F 6Ø 61	I4
4E 4F 5Ø 51 52 53 54 55 56 57	Q4
9E 9F AØ A1 A2 A3 A4 A5 A6 A7 A8 A9	rolling I
DE DF EØ E1 E2 E3 E4 E5 E6 E7 E8 E9	rolling Q

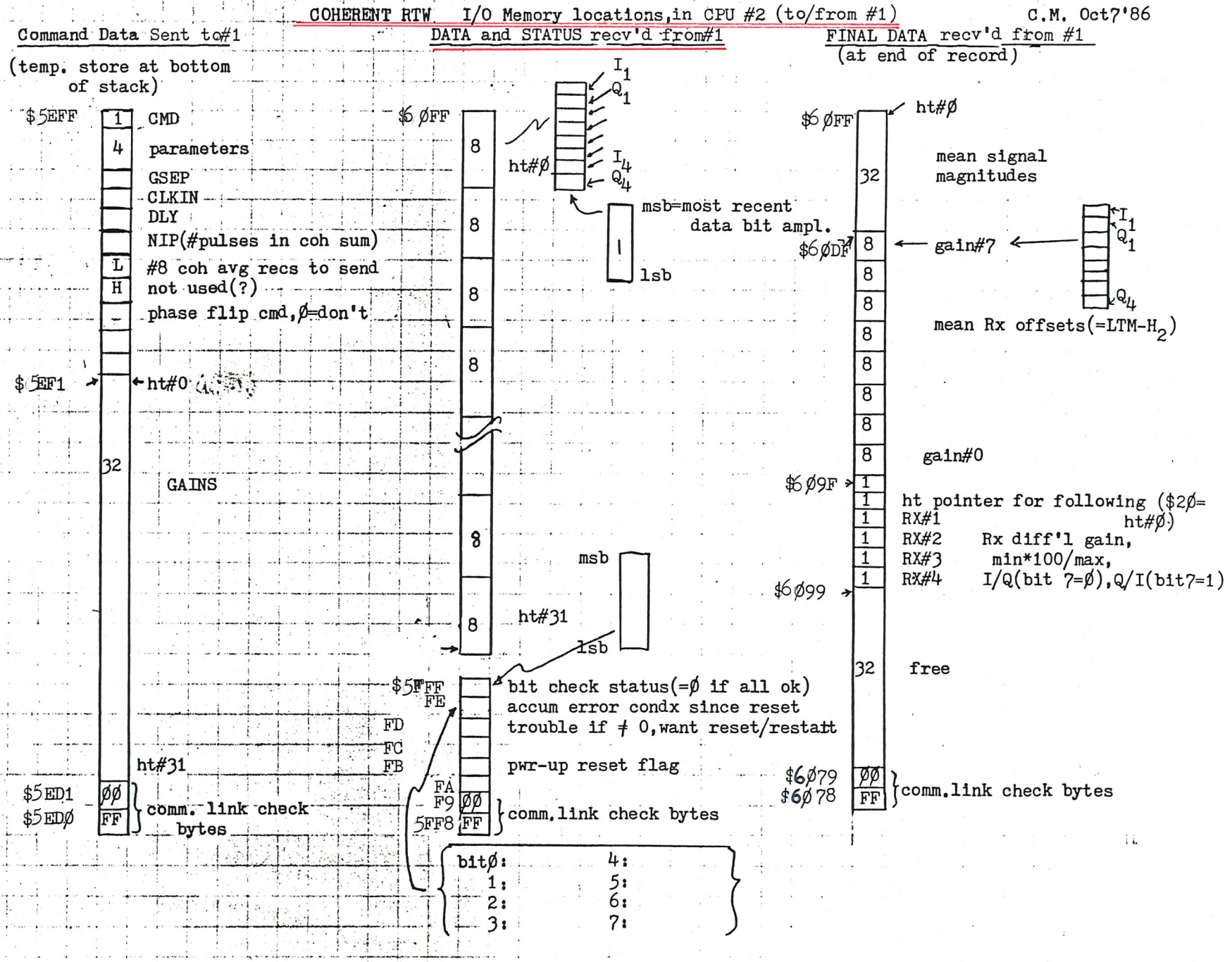
←-ROL

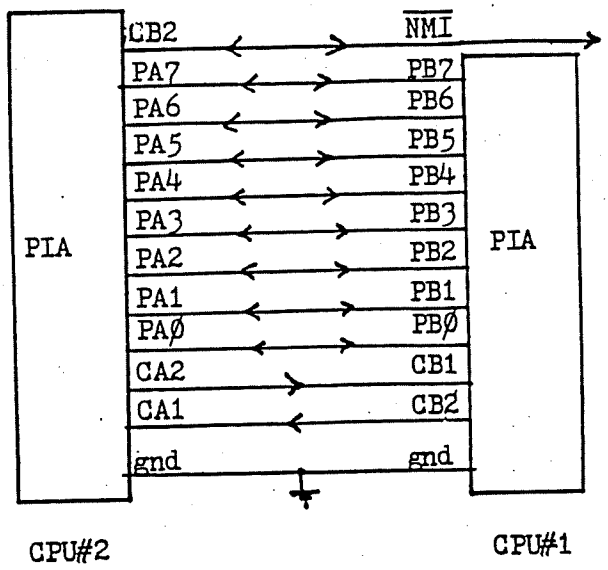
x x x x x x x x	and for -ve lags
x x x x x x x x x x	and for +ve lags
x x x x x x x x x x	ROL for "before -ve" lags or 0
x x x x x x x x x x	ROLs after 0 and +ve lags

COHRTW- CPU#2 Bulk storage map

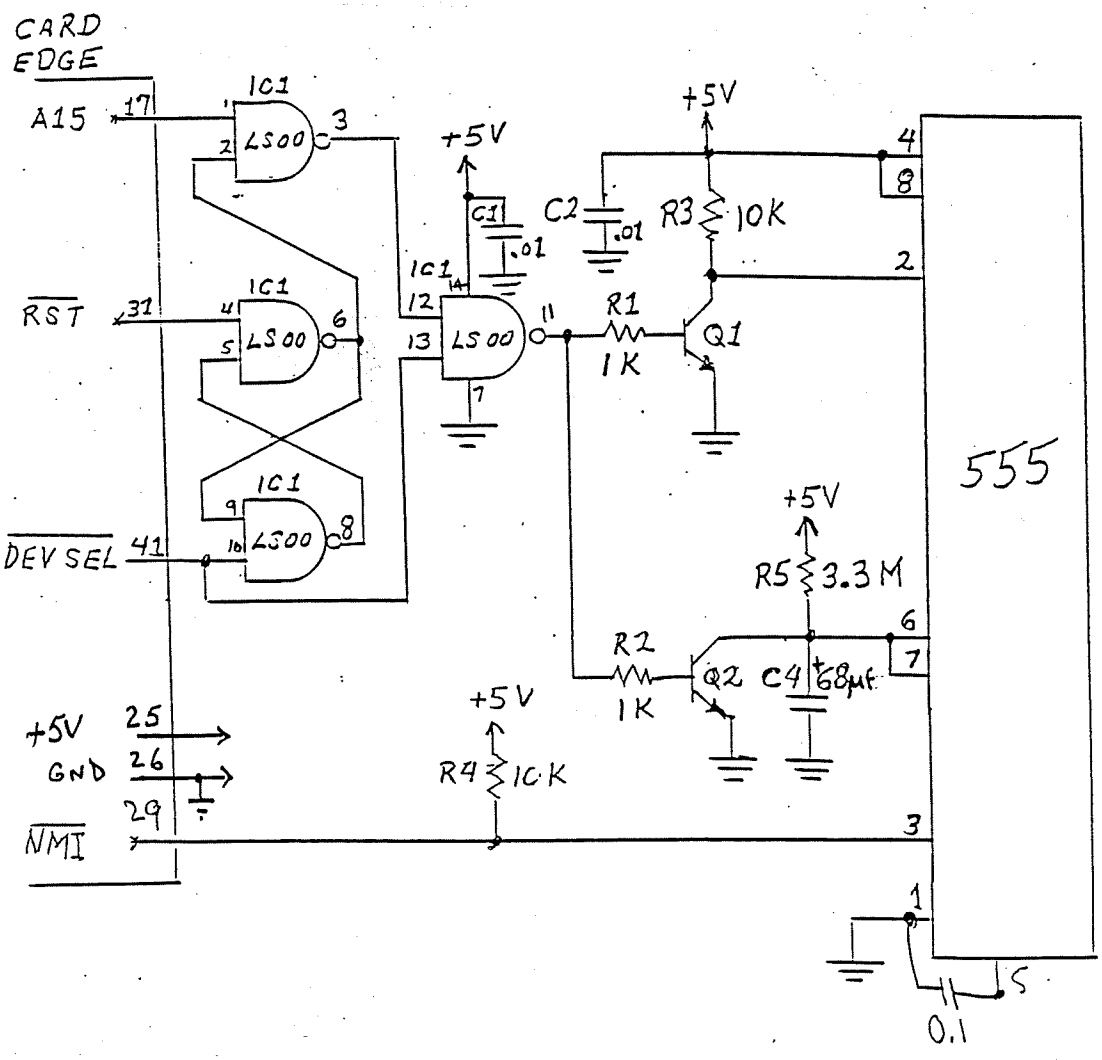


COHERENT RTW I/O Memory locations,in CPU #2 (to/from #1) C.M. Oct7'86





Communication link: CPU#2 -to- CPU#1



5 min timer circuit: On reset, the circuit will be re-triggered by any call to addresses $8000-$FFFF (i.e. line A15). On a call to the address $CØXY (Y = Ø-F, where X is the slot number plus 8; e.g. for slot 3, any address $CØBØ to $CØBF) the circuit will only be triggered on further such calls (until the next reset). If the circuit has not been re-triggered for 5 min, it will pull the NMI line low .

JAN 4 1991 APR 11 1990 NOV 30 1990 FEB 22 1990 DEC 13 1990 JUL 23 1990 CM,APR11'90

COHRTW APPLE #2 PROGRAM

TOTALCPU#2W-D/N

```
0900-  78          SEI        disable interrupts                    MAIN INITIALIZE
0901-  D8          CLD        binary arith (not decimal)
0902-  A9 07       LDA  #$07
0904-  A0 20       LDY  #$20
0906-  A2 04       LDX  #$04     initialize gains to 7,7,7,6,....1,0,0,0,0
0908-  99 3F BC    STA  $BC3F,Y
090B-  88          DEY
090C-  CA          DEX
090D-  D0 F9       BNE  $0908
090F-  38          SEC
0910-  E9 01       SBC  #$01
0912-  10 F2       BPL  $0906
0914-  78          SEI           can restart from here without re-init. gains
0915-  D8          CLD
0916-  A9 4C       LDA  #$4C
0918-  8D FB 03    STA  $03FB
091B-  A9 50       LDA  #$50       NMI JMP instruction (NMI operated by hardware
091D-  8D FC 03    STA  $03FC        tripout if system stuck for > ~5 min)
0920-  A9 19       LDA  #$19
0922-  8D FD 03    STA  $03FD
0925-  A9 90       LDA  #$90
0927-  8D FE 03    STA  $03FE     set IRQ vector
092A-  A9 0C       LDA  #$0C
092C-  8D FF 03    STA  $03FF
092F-  A9 00       LDA  #$00
0931-  85 BB       STA  $BB     REC not in progress
0933-  85 BC       STA  $BC     (don't need BC with new clock board)
0935-  20 10 1D    JSR  $1D10      clear screen and set up screen display
0938-  A9 D7       LDA  #$D7       "W" and jump to "WAIT"
093A-  20 78 0A    JSR  $0A78        (for no good reason, since new clock board)
093D-  AD B0 C0    LDA  $C0B0      clear irq flag (C128 comm. link)
0940-  A9 2D  NMI→ LDA  #$2D       Entry for NMI (which operates if"stuck"system)
0942-  8D 01 C5    STA  $C501      CA2     high, no irq (CA1↓= irq) , auto
0945-  8D B1 C0    STA  $C0B1      CA2 strobe on port read
0948-  AD 00 C5    LDA  $C500 ←    clear any handshake from CPU#1
094B-  78          SEI
094C-  D8          CLD
094D-  EA          NOP
094E-  A9 3C       LDA  #$3C  }
0950-  8D 03 C5    STA  $C503 }  CB2 high: = No NMI sent to CPU#1
0953-  A9 00       LDA  #$00  }
0955-  85 BB       STA  $BB   }  no run in progress
0957-  20 A0 0B    JSR  $0BA0    set-up various lookup tables
095A-  A5 00       LDA  $00   )
095C-  8D CF 5E    STA  $5ECF )
095F-  A5 01       LDA  $01   )
0961-  8D CD 5E    STA  $5ECD )  not needed with new clock board
0964-  A5 02       LDA  $02   )
0966-  8D CC 5E    STA  $5ECC )
0969-  A5 03       LDA  $03   )
096B-  8D CB 5E    STA  $5ECB )
096E-  20 34 0D    JSR  $0D34 ← initialize tape interface so drive can be put
0971-  A2 30       LDX  #$30          'on line'
0973-  A9 00       LDA  #$00       clear command store going to CPU#1
0975-  9D CF 5E    STA  $5ECF,X          (necessary! to set proper op-mode)
0978-  CA          DEX
0979-  D0 FA       BNE  $0975
097B-  A9 00       LDA  #$00 drive#  }
097D-  85 1D       STA  $1D          } set tape drive param. so
097F-  A9 82       LDA  #$82 drive char. } can use before 1st dump.
0981-  85 1E       STA  $1E
0983-  A9 01       LDA  #$01     set "1st record" flag
0985-  85 BA       STA  $BA
0987-  AD F0 C0    LDA  $C0F0       clear EOT flag(if any)
098A-  EA          NOP
098B-  EA          NOP
098C-  A9 00  NMI→ LDA  #$00        clear
098E-  85 C5       STA  $C5          file gap flag
0990-  85 B6       STA  $B6          and CPU#2 status (=no message)
0992-  A0 20       LDY  #$20       clear  C128   date/time store
0994-  99 7F BC    STA  $BC7F,Y     so unused bytes will look unused
0997-  88          DEY
0998-  D0 FA       BNE  $0994
```

```
099A-  78          SEI      disable interrupts
099B-  D8          CLD      clear decimal mode                    START
099C-  EA          NOP
099D-  20 60 0E    JSR   $0E60 clear analysis ht# scrn space,& "W" msg space
09A0-  20 70 0A    JSR   $0A70 check for "wait" command(incl. man. start cmd)
09A3-  20 E0 0A    JSR   $0AE0  update clock
09A6-  20 10 1B    JSR   $1B10  check for start time (BB=1 if start req'd)
09A9-  A5 BB       LDA   $BB
09AB-  F0 F3       BEQ   $09A0  no start, so loop
09AD-  20 E0 0B    JSR   $0BE0  yes, start; so initialize for record
09B0-  20 40 0D    JSR   $0D40  and send command data to CPU#1
09B3-  58          CLI          and enable interrupts from #1
09B4-  A9 D2       LDA   #$D2   and write "R" on the screen after the clock display
09B6-  8D 9A 05    STA   $059A
09B9-  EA          NOP               then go to analysis.... loops until done
09BA-  EA          NOP
09BB-  EA          NOP
09BC-  EA          NOP
-------------------------------------------------------------------------------
09BD-  A9 00       LDA   #$00   } #analyses cntr                    ANALYSE
09BF-  85 B5       STA   $B5    }
09C1-  A9 20       LDA   #$20   } ht cntr
09C3-  85 B8       STA   $B8    }
09C5-  A9 20       LDA   #$20   }
09C7-  85 BF       STA   $BF    } fixed bin pointer to I1 data,ht#ø(lowest)
09C9-  A9 84       LDA   #$84   }
09CB-  85 C0       STA   $C0    }
09CD-  A9 FF       LDA   #$FF   }
09CF-  85 C1       STA   $C1    } fixed XCOR. pointer to 4V3 Re at ht#ø
09D1-  A9 BB       LDA   #$BB   }
09D3-  85 C2       STA   $C2    }
09D5-  A9 1F       LDA   #$1F   }
09D7-  85 C3       STA   $C3    }
09D9-  A9 8C       LDA   #$8C   } fixed ACOR. pointer to Re at ht#ø
09DB-  85 C4       STA   $C4    }
09DD-  20 E0 0A    JSR   $0AE0  update screen clock and clock store
09E0-  A4 B8       LDY   $B8
09E2-  B9 FF 63    LDA   $63FF,Y
09E5-  85 B9       STA   $B9         see if there is enough data at this height
09E7-  38          SEC                     for analysis
09E8-  E5 C6       SBC   $C6
09EA-  EA          NOP
09EB-  B0 02       BCS   $09EF
09ED-  49 E0       EOR   #$E0        wrap-around at $1F
09EF-  C9 0A       CMP   #$0A        need 10 bytes
09F1-  90 31       BCC   $0A24       don't have 10,so skip this height for now
09F3-  A5 B8       LDA   $B8         ok, store "A" on screen,showing relative data
09F5-  20 50 0E    JSR   $0E50          entry and analysis positions.
09F8-  A4 B9       LDY   $B9
09FA-  A9 C1       LDA   #$C1   "A"
09FC-  99 D7 05    STA   $05D7,Y
09FF-  20 20 19    JSR   $1920       load all binary to zero page
0A02-  20 70 0E    JSR   $0E70       XCOR
0A05-  20 C0 15    JSR   $15C0       #1's in I & Q
0A08-  20 70 16    JSR   $1670       ACOR
0A0B-  EA          NOP                    --end of analysis for this ht, so
0A0C-  A4 B9       LDY   $B9
0A0E-  A9 A0       LDA   #$A0   }    clear the "A" from the screen
0A10-  99 D7 05    STA   $05D7,Y}
0A13-  A4 B8       LDY   $B8    }
0A15-  B9 FF 63    LDA   $63FF,Y}    move Aoff(h) down(="finished with this data")
0A18-  38          SEC          }
0A19-  E9 08       SBC   #$08   }
0A1B-  B0 02       BCS   $0A1F  }
0A1D-  49 E0       EOR   #$E0   <-   (wrap at $1F)
0A1F-  99 FF 63    STA   $63FF,Y}
0A22-  E6 B5       INC   $B5    <-   count number of hts analysed
```

```
0A24-   A5 BF       LDA   $BF      }
0A26-   38          SEC             }
0A27-   E9 20       SBC   #$20      }   DEC 13 1990
0A29-   85 BF       STA   $BF      }    modify fixed Bin pointer, Fixed XCOR pointer
0A2B-   B0 02       BCS   $0A2F    } BIN  and fixed ACOR pointer for next height
0A2D-   C6 C0       DEC   $C0      }
0A2F-   C6 C2       DEC   $C2      }
0A31-   A5 C1       LDA   $C1      }
0A33-   38          SEC             }
0A34-   E9 80       SBC   #$80     } XCOR   (subtract $18Ø)
0A36-   85 C1       STA   $C1      }
0A38-   B0 02       BCS   $0A3C    }
0A3A-   C6 C2       DEC   $C2      }
0A3C-   A5 C3       LDA   $C3      }
0A3E-   38          SEC             }
0A3F-   E9 40       SBC   #$40      }
0A41-   85 C3       STA   $C3      }
0A43-   B0 02       BCS   $0A47    } ACOR
0A45-   C6 C4       DEC   $C4      }
0A47-   C6 B8       DEC   $B8  ← next height
0A49-   D0 92       BNE   $09DD
0A4B-   A5 BB       LDA   $BB    record still in progress?
0A4D-   D0 05       BNE   $0A54  yes, so loop
0A4F-   78          SEI          no, so disable interrupts
0A50-   A5 B5       LDA   $B5  ← any data left to be analysed?
0A52-   F0 03       BEQ   $0A57  no,so quit
0A54-   4C BD 09    JMP   $09BD  yes,so try one more height scan, until no analysès
0A57-   20 30 1B    JSR   $1B30  Display final params from  CPU#1
0A5A-   20 10 1A    JSR   $1A10  take results of last data from C128,and send all correls
0A5D-   20 80 0D    JSR   $0D80  Reset gains
```

10xNOP

```
0A6A-   20 70 1F    JSR   $1F70  call "store/dump" (checks for 1st rec.)
0A6D-   4C 9A 09    JMP   $099A  and back to START to  wait for the next record time
0A70-   20 C0 0A    JSR   $0AC0  get key
0A73-   C9 D7       CMP   #$D7   ="W" ?                          WAIT
0A75-   F0 01       BEQ   $0A78  yes,so stay in
0A77-   60          RTS          no,so return
0A78-   A9 57  entry LDA  #$57   write flashing [W] after clock display
0A7A-   8D 9A 05    STA   $059A
0A7D-   20 DD 0A    JSR   $0ADD  update clock, re-trigger watchdog circuit
0A80-   A9 00       LDA   #$00   but make sure day# doesn't change
0A82-   85 BC       STA   $BC
0A84-   20 C0 0A    JSR   $0AC0  and get Key cmd
0A87-   C9 D3       CMP   #$D3    ="S" ?  (start)
0A89-   D0 0C       BNE   $0A97  no, try next command
0A8B-   20 10 1D    JSR   $1D10    clear and re-draw screen
0A8E-   AD F0 C0    LDA   $C0F0    clear EOT flag(if any)
0A91-   A9 A0       LDA   #$A0     clear flashing "W" from screen
0A93-   8D 9A 05    STA   $059A
0A96-   60          RTS             and return from wait
0A97-   C9 C5       CMP   #$C5     key="E"?
0A99-   D0 06       BNE   $0AA1    no
0A9B-   20 50 1F    JSR   $1F50    yes,so do file gap and update zero
0A9E-   18          CLC             page manual date store, and stay in
0A9F-   90 DC       BCC   $0A7D    wait
0AA1-   C9 CD       CMP   #$CD     Key="M"?   (manual record start)
0AA3-   D0 D8       BNE   $0A7D    no
0AA5-   E6 BB       INC   $BB      yes,so set "rec in progress" flag
0AA7-   D0 E8       BNE   $0A91    and return(without clearing display,but
0AA9-   FF          ???            clearing "W")
0AAA-   FF          ???
0AAB-   FF          ???
0AAC-   FF          ???
```

```
0AC0-   A9 00       LDA   #$00   leave ØØ in A if no key          GET KEY
0AC2-   2C 00 C0    BIT   $C000  kbd strobe?
0AC5-   10 0C       BPL   $0AD3  no,so skip out
0AC7-   2C 10 C0    BIT   $C010  yes,clear strobe
0ACA-   AD 00 C0    LDA   $C000  and read key
0ACD-   09 80       ORA   #$80
0ACF-   C9 82       CMP   #$82   key= CB ??
0AD1-   F0 04       BEQ   $0AD7  yes, so "BREAK"
0AD3-   8D 51 C0    STA   $C051  no, so keep monitor in'TEXT'mode
0AD6-   60          RTS          and return
0AD7-   78          SEI          disable interrputs
0AD8-   4C 59 FF    JMP   $FF59  and jump to Apple monitor - i.e. STOP
```

JAN 4 1991 APR 11 1990 NOV 30 1990 DEC 13 1990 (4)

```
0ADD-   AD FF FF     LDA   $FFFF    re-trigger watch dog circuit (5 min time-out)
0AE0-   20 2E C3 entry JSR $C32E    full clock read          UPDATE CLOCK STORE
0AE3-   2C 00 C0     BIT   $C000    kbd strobe?
0AE6-   10 17        BPL   $0AFF    no,
0AE8-   AD 00 C0     LDA   $C000    read keyboard
0AEB-   C9 AD        CMP   #$AD     = "-" ?
0AED-   F0 0A        BEQ   $0AF9
0AEF-   C9 AB        CMP   #$AB   = "+" ?
0AF1-   D0 0C        BNE   $0AFF    not + or -,so leave strobe on.
0AF3-   20 1C C3     JSR   $C31C    add 1 to S1 (sec. units)
0AF6-   18           CLC
0AF7-   90 03        BCC   $0AFC    forced branch
0AF9-   20 22 C3     JSR   $C322    subtr. 1 from S1
0AFC-   2C 10 C0     BIT   $C010    clear kbd strobe
0AFF-   A2 0E        LDX   #$0E     write clock on screen (note:year is written as
0B01-   A0 00        LDY   #$00        2 sep. charcters here, but it is stored in
0B03-   BD 03 03     LDA   $0303,X-    BCD = Y10*16+Y1 ) in memory (& on tape)
0B06-   30 02        BMI   $0B0A    already ascii
0B08-   09 B0        ORA   #$B0     clock digit,so convert to ascii
0B0A-   99 8B 05     STA   $058B,Y and write on screen
0B0D-   C8           INY
0B0E-   CA           DEX
0B0F-   D0 F2        BNE   $0B03
```

Note Clock and clock EPROM in slot#3. To do full clock reset from monitor, use C328Greturn sep. (for other features, see manual)

3xNOP

Note: clock EPROM call stores clock in $300-$312 thus-
$300: $D_1,D_{10},M_1,M_{10},S_1,S_{10}$,"t",$M_1,M_{10},H_1,H_{10}$,"/",
D_1,D_{10},D_{100},":",Y_1,Y_{10}, day-of-week(1-7)

```
0B14-   A9 4C        LDA   #$4C     re-load IRQ and NMI vectors
0B16-   8D FB 03     STA   $03FB   (trying to overcome a minor
0B19-   A9 50        LDA   #$50     glitch which may destroy these)
0B1B-   8D FC 03     STA   $03FC
0B1E-   A9 19        LDA   #$19
0B20-   8D FD 03     STA   $03FD
0B23-   A9 90        LDA   #$90
0B25-   8D FE 03     STA   $03FE
0B28-   A9 0C        LDA   #$0C
0B2A-   8D FF 03     STA   $03FF
0B2D-   60           RTS
0B2E-   FF           ???
```

```
0B30-   A2 1F        LDX   #$1F     Do 1 byte check-sum
0B32-   A9 00        LDA   #$00
0B34-   85 A0        STA   $A0      just $900:1FDF, see if program
0B36-   A0 DF        LDY   #$DF       was modified in operation.
0B38-   86 A1        STX   $A1
0B3A-   18           CLC
0B3B-   71 A0        ADC   ($A0),Y    ( called from record-init. )
0B3D-   88           DEY
0B3E-   C0 FF        CPY   #$FF     NB. WHen program updates are put in,
0B40-   D0 F8        BNE   $0B3A      the stored checksum(in $1FEB) must also
0B42-   CA           DEX              be updated by running this routine, and
0B43-   E0 08        CPX   #$08       getting the new checksum from $F9.
0B45-   D0 F1        BNE   $0B38
0B47-   CD EB 1F     CMP   $1FEB    stored check-sum
0B4A-   F0 03        BEQ   $0B4F    OK
0B4C-   4C D1 1F     JMP   $1FD1    bad, so save actual check-sum in F9,and
0B4F-   60           RTS              type error msg. (and continue!)
```

```
0B50-   A5 BD        LDA   $BD      L byte of"rec length"  Decrement "Receive data"
0B52-   D0 02        BNE   $0B56                            Counter
0B54-   C6 BE        DEC   $BE      H
0B56-   C6 BD        DEC   $BD        'N' flag set if end of count
0B58-   D0 08        BNE   $0B62      down
0B5A-   A5 BE        LDA   $BE
0B5C-   F0 02        BEQ   $0B60
0B5E-   10 04        BPL   $0B64
0B60-   C6 BE        DEC   $BE
0B62-   A5 BE        LDA   $BE
0B64-   60           RTS
```

```
0B68-   AD 92 BC     LDA   $BC92    get #pulses/integ   DISPLAY AMOUNT OF INTEGRATION
0B6B-   20 B0 1C     JSR   $1CB0    change to BCD ascii
0B6E-   8D 8B 06     STA   $068B    write L on screen
0B71-   8A           TXA            get H
0B72-   C9 B0        CMP   #$B0     high-digit=0?  puts the XX in the
0B74-   D0 02        BNE   $0B78    no                ( XX pulse integration) " message
0B76-   A9 A0        LDA   #$A0     yes,so clear      on the screen
0B78-   8D 8A 06     STA   $068A      on screen
0B7B-   60           RTS
```

```
0B80-  A0 00      LDY  #$00
0B82-  A9 40      LDA  #$40
0B84-  85 A0      STA  $A0
0B86-  A9 BB      LDA  #$BB
0B88-  85 A1      STA  $A1
0B8A-  A9 00      LDA  #$00
0B8C-  91 A0      STA  ($A0),Y
0B8E-  88         DEY
0B8F-  D0 FB      BNE  $0B8C
0B91-  C6 A1      DEC  $A1
0B93-  A5 A1      LDA  $A1
0B95-  C9 83      CMP  #$83
0B97-  D0 F1      BNE  $0B8A
0B99-  60         RTS
```

CLEAR CORREL STORE

(just the correl store, not S.D. or gains or D/T)

```
0B9A-  FF         ???
0B9B-  FF         ???
0B9C-  FF         ???
0B9D-  FF         ???
0B9E-  FF         ???
0B9F-  FF         ???
```

```
0BA0-  A0 00      LDY  #$00    entries 0-255,each table
0BA2-  98         TYA
0BA3-  4A         LSR
0BA4-  4A         LSR
0BA5-  4A         LSR
0BA6-  4A         LSR
0BA7-  99 00 62   STA  $6200,Y
0BAA-  98         TYA
0BAB-  0A         ASL
0BAC-  0A         ASL
0BAD-  0A         ASL
0BAE-  0A         ASL
0BAF-  99 00 61   STA  $6100,Y
0BB2-  A9 00      LDA  #$00
0BB4-  85 A0      STA  $A0
0BB6-  A2 08      LDX  #$08
0BB8-  98         TYA
0BB9-  0A         ASL
0BBA-  90 02      BCC  $0BBE
0BBC-  E6 A0      INC  $A0
0BBE-  CA         DEX
0BBF-  D0 F8      BNE  $0BB9
0BC1-  A5 A0      LDA  $A0
0BC3-  99 00 63   STA  $6300,Y
0BC6-  88         DEY
0BC7-  D0 D9      BNE  $0BA2
0BC9-  A0 0F      LDY  #$0F
0BCB-  98         TYA
0BCC-  C9 0A      CMP  #$0A
0BCE-  90 03      BCC  $0BD3
0BD0-  18         CLC
0BD1-  69 07      ADC  #$07
0BD3-  69 B0      ADC  #$B0
0BD5-  99 B0 5E   STA  $5EB0,Y
0BD8-  88         DEY
0BD9-  10 F0      BPL  $0BCB
0BDB-  20 F8 19   JSR  $19F8
0BDE-  60         RTS
```

SET-UP TABLES

H → L nybble shift, and clear

L → H nybble shift and clear

#1 bits per byte (used in correl)

ASCII screen char. table for hex nybble input

initialize tape drive reset/Tx cntl port and turn off Tx

RECORD INITIALIZE
(done before every record)

```
0BE0-  78          SEI          just in case...
0BE1-  D8          CLD          ditto
0BE2-  20 80 0B    JSR  $0B80   clear correl store
0BE5-  A9 7D       LDA  #$7D    } set delay to 1st ht. gate param.
0BE7-  8D 8F BC    STA  $BC8F   } (goes to CPU#1)
0BEA-  8D F8 5E    STA  $5EF8   set clockin delay(must let 65 gates through!)
0BED-  A9 7F       LDA  #$7F    }
0BEF-  8D 8E BC    STA  $BC8E   } NOTE: USE $37,3F for 1 MHz clock; 7D,7F for 2 MHz clock (single card comparator)
0BF2-  8D F9 5E    STA  $5EF9   }
0BF5-  A9 08       LDA  #$08    }
0BF7-  8D 8D BC    STA  $BC8D   } set gate separation (20 µs)
0BFA-  8D FA 5E    STA  $5EFA   }
0BFD-  20 D0 1E    JSR  $1ED0   select synch, tx on/off&amt.of integ. dep. on GMT
0C00-  20 30 0B    JSR  $0B30   verify program checksum
                   2xNOP
0C05-  A9 40       LDA  #$40    }
0C07-  8D 91 BC    STA  $BC91   } set record length (number of groups of 8 amps)
0C0A-  8D F6 5E    STA  $5EF6   }
0C0D-  85 BD       STA  $BD     }
0C0F-  A9 00       LDA  #$00    } H (always zero, unless very long records
0C11-  8D 90 BC    STA  $BC90   }
0C14-  8D F5 5E    STA  $5EF5   }
0C17-  85 BE       STA  $BE     }
0C19-  A9 00       LDA  #$00    } don't want auto phase flip in CPU#1 interface cntl.
0C1B-  8D F4 5E    STA  $5EF4   } (never!,unless CPU#1 program upgraded)
0C1E-  A0 20       LDY  #$20    }
0C20-  B9 3F BC    LDA  $BC3F,Y } store present gains in cmd store going to
0C23-  99 D1 5E    STA  $5ED1,Y } CPU#1
0C26-  88          DEY          }
0C27-  D0 F7       BNE  $0C20   }
0C29-  A9 00       LDA  #$00    }
0C2B-  8D D1 5E    STA  $5ED1   } set comm. link check bytes for CPU#1 comm link
0C2E-  A9 FF       LDA  #$FF    }
0C30-  8D D0 5E    STA  $5ED0   }
0C33-  A9 38       LDA  #$38    } select DDR( CPU#1 link)
0C35-  8D 01 C5    STA  $C501   }
0C38-  A9 FF       LDA  #$FF    } all bits o/p
0C3A-  8D 00 C5    STA  $C500   }
0C3D-  A9 3C       LDA  #$3C    } manual control of CA2, CA2=high
0C3F-  8D 01 C5    STA  $C501   }

0C42-  20 E0 0A    JSR  $0AE0   update clock
0C45-  AD 10 03    LDA  $0310   Y1
0C48-  AE 11 03    LDX  $0311   Y10
0C4B-  1D 00 61    ORA  $6100,X change to BCD
0C4E-  A0 0D       LDY  #$0D    } save whole clock in clock store
0C50-  99 92 BC    STA  $BC92,Y } (a little odd ball, but it does what I want!)
0C53-  B9 02 03    LDA  $0302,Y }
0C56-  88          DEY          }
0C57-  D0 F7       BNE  $0C50   }
0C59-  A0 20       LDY  #$20    }
0C5B-  A9 1F       LDA  #$1F    } set all Aoff(h) to top of BIN store
0C5D-  99 FF 63    STA  $63FF,Y }
0C60-  88          DEY          }
0C61-  D0 FA       BNE  $0C5D   }
0C63-  85 C6       STA  $C6     clear any leftover interrupt flag from #1
0C65-  AD 00 C5    LDA  $C500   display amount of integration in use
0C68-  20 68 0B    JSR  $0B68   Request normal CPU#1 operation ("command=Ø")
0C6B-  A9 00       LDA  #$00
0C6D-  8D FF 5E    STA  $5EFF

0C70-  A9 34       LDA  #$34    }
0C72-  8D 03 C5    STA  $C503   } toggle CB2 = NMI in CPU#1
0C75-  A9 3C       LDA  #$3C    }
0C77-  8D 03 C5    STA  $C503   }
0C7A-  A9 00       LDA  #$00    }
0C7C-  8D 81 BC    STA  $BC81   } put check bytes in clock store for the
0C7F-  A9 FF       LDA  #$FF    } benefit of the C128 comm. link check
0C81-  8D 80 BC    STA  $BC80   }
0C84-  A9 D0       LDA  #$D0 <- "P"  Site I.D. going onto analysed data
0C86-  8D 8A BC    STA  $BC8A        "P" = Park
0C89-  60          RTS
0C8A-  FF          ???
0C8B-  FF          ???
```

INTERRUPT PROCESSING

```
0C90-   08          PHP
0C91-   A5 45       LDA   $45    Apple monitor stores A here
0C93-   48                        on irq,so get it back
0C94-   8A          TXA
0C95-   48          PHA
0C96-   98          TYA
0C97-   48          PHA
0C98-   A5 BB       LDA   $BB     record in progress?
0C9A-   D0 03       BNE   $0C9F   yes
0C9C-   4C 21 0D    JMP   $0D21   no,so skip out
0C9F-   A9 F8       LDA   ##$F8   } set pointer to temporary input store
0CA1-   85 C7       STA   $C7     }
0CA3-   A9 60       LDA   ##$60   }
0CA5-   85 C8       STA   $C8     }
0CA7-   A2 02       LDX   ##$02   } #pages,or part pages, to read($108 bytes here)
0CA9-   A0 07       LDY   ##$07   }
0CAB-   A9 02       LDA   ##$02   } set CPU#2 status byte to indicate possibe hangup
0CAD-   85 B6       STA   $B6     }        location
0CAF-   AD 01 C5    LDA   $C501   } wait for interrupt(is set 1st time because of hrdware
0CB2-   10 FB       BPL   $0CAF   }                    interrupt)
0CB4-   AD 00 C5    LDA   $C500   load data, irq flag cleared automatically , auto stdout
0CB7-   91 C7       STA   ($C7),Y and save in temporary store
0CB9-   88          DEY
0CBA-   C0 FF       CPY   ##$FF
0CBC-   D0 F1       BNE   $0CAF   next byte
0CBE-   C6 C8       DEC   $C8
0CC0-   CA          DEX
0CC1-   D0 EC       BNE   $0CAF   next page
0CC3-   AD F9 5F    LDA   $5FF9    ----end of communication
0CC6-   F0 03       BEQ   $0CCB          test comm. link check bytes
0CC8-   4C F0 1B    JMP   $1BF0 <--- trouble so"comm#1 error" and stop program
0CCB-   AD F8 5F    LDA   $5FF8
0CCE-   C9 FF       CMP   ##$FF
0CD0-   D0 F6       BNE   $0CC8
0CD2-   20 C8 1C    JSR   $1CC8 <--- write CPU#1 status byte on screen
0CD5-   A9 04       LDA   ##$04   } set CPU#2 status
0CD7-   85 B6       STA   $B6     }
0CD9-   20 B0 0D    JSR   $0DB0   sort data from #1, and store in proper sequences
0CDC-   20 20 0E    JSR   $0E20   display A/D bit check status
0CDF-   20 50 0B    JSR   $0B50   next data set (countdown to end of record)
0CE2-   10 3D       BPL   $0D21   not ended yet,so skip out
0CE4-   20 F8 19    JSR   $19F8 turn off Tx at end of rec(just aft. last data from #1)
                                  ok, end of record, so take last param. from #1
~~0CE6-  EA          NOP~~
0CE7-   A0 88       LDY   ##$88   read 136 bytes
0CE9-   AD 01 C5    LDA   $C501   wait for flag
0CEC-   10 FB       BPL   $0CE9
0CEE-   AD 00 C5    LDA   $C500   load byte
0CF1-   99 77 60    STA   $6077,Y and store it in temporary store
0CF4-   88          DEY
0CF5-   D0 F2       BNE   $0CE9
0CF7-   AD 79 60    LDA   $6079     look at comm. link check bytes
0CFA-   F0 03       BEQ   $0CFF
0CFC-   4C F0 1B    JMP   $1BF0 <--- bad:"COMM#1 ERROR" and stop program
0CFF-   AD 78 60    LDA   $6078
0D02-   C9 FF       CMP   ##$FF
0D04-   D0 F6       BNE   $0CFC
0D06-   20 B0 1A    JSR   $1AB0     ok,now move Doff along, entering zeroes, so
                                          analysis will be complete when it runs
                                           out of data
0D09-   EA          NOP
0D0A-   A9 3C       LDA   ##$3C   }
0D0C-   8D 01 C5    STA   $C501   } set CA2 high, manual cntl of CA2
0D0F-   A9 00       LDA   ##$00   }
0D11-   85 BB       STA   $BB     } END OF RECORD, so set REC=ØØ
0D13-   A0 20       LDY   ##$20   }
0D15-   B9 DF 60    LDA   $60DF,Y } put magnitude (="signal") data from CPU#1 in
0D18-   99 5F BC    STA   $BC5F,Y }       output store(going to C128)
0D1B-   88          DEY           }
0D1C-   D0 F7       BNE   $0D15   }
0D1E-   E6 B5       INC   $B5 <--- make sure analysis doesn't accidentally finish
                                          too soon(i.e. force at least one more
                                          height scan)
0D20-   EA          NOP
```

(8)

```
0D21-  20 30 1F   JSR  $1F30     check for CPU#1 power-up/reset and
0D24-  EA         NOP                        incr. screen cntr if so
0D25-  EA         NOP   LDA FFFF (RETRIG WATCHDOG)  or LDA C0B1
0D26-  EA         NOP

0D27-  A9 00      LDA  #$00      set CPU#2 status
0D29-  85 B6      STA  $B6
0D2B-  EA         NOP
0D2C-  68         PLA
0D2D-  A8         TAY
0D2E-  68         PLA
0D2F-  AA         TAX
0D30-  68         PLA
0D31-  28         PLP
0D32-  40         RTI
0D33-  FF         ???

0D34-  A9 3F      LDA  #$3F      INITIALIZE TAPE INTERFACE
0D36-  85 F9      STA  $F9       absolute return
0D38-  A9 0D      LDA  #$0D       adress!!!!!     (so have control of drive before
0D3A-  85 FA      STA  $FA        careful!!       1st write)
0D3C-  4C 21 C7   JMP  $C721     if modifying
0D3F-  60         RTS

0D40-  A9 01      LDA  #$01      SEND CMD TO CPU#1
0D42-  85 B6      STA  $B6       set CPU#2 status
0D44-  A0 30      LDY  #$30      48 bytes to send
0D46-  AD 01 C5   LDA  $C501     wait for CPU#1 ready
0D49-  10 FB      BPL  $0D46
0D4B-  B9 CF 5E   LDA  $5ECF,Y   get byte
0D4E-  8D 00 C5   STA  $C500     and send it to #1
0D51-  AD 00 C5   LDA  $C500     clear interrupt flag
0D54-  A9 34      LDA  #$34 }CA2↓
0D56-  8D 01 C5   STA  $C501 }   and send "data ~~taken~~ ready" signal to #1
0D59-  A9 3C      LDA  #$3C }CA2↑   (by toggling CA2)
0D5B-  8D 01 C5   STA  $C501 }
0D5E-  88         DEY            next byte
0D5F-  D0 E5      BNE  $0D46
0D61-  EA         NOP
0D62-  EA         NOP            ----now reverse port so next data comes from #1
0D63-  EA         NOP
0D64-  A9 29      LDA  #$29      select DDR
0D66-  8D 01 C5   STA  $C501
0D69-  A9 00      LDA  #$00      all bit input
0D6B-  8D 00 C5   STA  $C500
0D6E-  A9 2D      LDA  #$2D      select port, IRQ on ↓ of CA1
0D70-  8D 01 C5   STA  $C501
0D73-  A9 00      LDA  #$00      set CPU#2 status "out of "send" routine
0D75-  85 B6      STA  $B6
0D77-  60         RTS

0D78-  FF         ???
0D79-  FF         ???
0D7A-  FF         ???
0D7B-  FF         ???
0D7C-  FF         ???
0D7D-  FF         ???
0D7E-  FF         ???
0D7F-  FF         ???

0D80-  A2 20      LDX  #$20      32 heights      RESET GAINS
0D82-  BD 5F BC   LDA  $BC5F,X   get magnitude ("signal")   (assumes mags 0-255!)
0D85-  C9 80      CMP  #$80      mag                                and 10dB
0D87-  B0 10      BCS  $0D99     mag >128 ,so try to decrease gain  steps
0D89-  C9 28      CMP  #$28      mag > 40  (and <128 ) so ok, leave as is
0D8B-  B0 14      BCS  $0DA1
0D8D-  BD 3F BC   LDA  $BC3F,X   can the gain be
0D90-  C9 07      CMP  #$07      increased?
0D92-  F0 0D      BEQ  $0DA1     no,so skip
0D94-  FE 3F BC   INC  $BC3F,X   yes,so increase the gain at this height
0D97-  D0 08      BNE  $0DA1     and forced branch out
0D99-  BD 3F BC   LDA  $BC3F,X   want to decrease gain,
0D9C-  F0 03      BEQ  $0DA1     can't, so skip
0D9E-  DE 3F BC   DEC  $BC3F,X   decrease gain 1 step
0DA1-  CA         DEX            next height
0DA2-  D0 DE      BNE  $0D82
0DA4-  60         RTS
```

Note: strict signal limits have been used to keep clipping minimal. We want to use the auxiliary raw o/p data quite often. If raw data are not used, limits could be $A0 and $32 (160 and 50)

```
0DB0-  A4 C6      LDY  $C6       Doff                    SORT DATA FROM CPU#1
0DB2-  A2 FF      LDX  ##$FF     256 bytes to distribute
0DB4-  A9 20      LDA  ##$20
0DB6-  85 C7      STA  $C7       point to ht#0, I1
0DB8-  A9 84      LDA  ##$84
0DBA-  85 C8      STA  $C8
0DBC-  A9 08      LDA  ##$08
0DBE-  85 C9      STA  $C9
0DC0-  BD 00 60   LDA  $6000,X
0DC3-  91 C7      STA  ($C7),Y
0DC5-  CA         DEX
0DC6-  E0 FF      CPX  ##$FF
0DC8-  F0 20      BEQ  $0DEA     done,so exit
0DCA-  A5 C8      LDA  $C8       not fini,so
0DCC-  38         SEC
0DCD-  E9 04      SBC  ##$04     modify addr. to next component,same ht
0DCF-  85 C8      STA  $C8
0DD1-  C6 C9      DEC  $C9       next component(in order I1,Q1,I2,Q2,I3,Q3,I4,Q4)
0DD3-  D0 EB      BNE  $0DC0
0DD5-  A5 C8      LDA  $C8         next height
0DD7-  18         CLC
0DD8-  69 20      ADC  ##$20       modify addr. to next ht.,component I1
0DDA-  85 C8      STA  $C8
0DDC-  A5 C7      LDA  $C7
0DDE-  38         SEC
0DDF-  E9 20      SBC  ##$20
0DE1-  85 C7      STA  $C7
0DE3-  B0 02      BCS  $0DE7
0DE5-  C6 C8      DEC  $C8
0DE7-  18         CLC
0DE8-  90 D2      BCC  $0DBC     forced branch
0DEA-  A4 C6      LDY  $C6
0DEC-  A9 A0      LDA  ##$A0     clear last"D" from screen
0DEE-  99 D7 05   STA  $05D7,Y
0DF1-  C6 C6      DEC  $C6       move Doff down
0DF3-  10 06      BPL  $0DFB     no wrap around occurred
0DF5-  A5 C6      LDA  $C6       yes, so wrap at $1F(actually just need LDA#1F
0DF7-  49 E0      EOR  ##$E0                                          STA$C6 )
0DF9-  85 C6      STA  $C6
0DFB-  A4 C6      LDY  $C6       write new "D" on screen "analysis and data entry"
0DFD-  A9 C4      LDA  ##$C4     "D"                      display
0DFF-  99 D7 05   STA  $05D7,Y   to screen addr.
0E02-  A5 C6      LDA  $C6
0E04-  A0 20      LDY  ##$20       now run through all heights to see
0E06-  D9 FF 63   CMP  $63FF,Y     whether next data entry will overwrite
0E09-  D0 03      BNE  $0E0E       data being used in analysis .
0E0B-  4C 08 1C   JMP  $1C08     yes,it will,so:"DATA OVERLOAD" and STOP
0E0E-  88         DEY
0E0F-  D0 F5      BNE  $0E06
0E11-  60         RTS

0E12-  AD FF 5F   LDA  $5FFF     check A/D status          TURN ON TX RATE
0E15-  D0 08      BNE  $0E1F     not ok,so don't "buzz" Tx during dump
0E17-  A9 17      LDA  ##$17     ok,so turn on Tx (make bit 3 low)
0E19-  2D 02 C5   AND  $C502
0E1C-  8D 02 C5   STA  $C502
0E1F-  60         RTS

0E20-  AD FF 5F   LDA  $5FFF   bit-chk status byte from#1 DISPLAY BIT CHECK STATUS
0E23-  D0 0C      BNE  $0E31   not ok
0E25-  A0 08      LDY  ##$08   is ok,so just display "OK"
0E27-  B9 69 1E   LDA  $1E69,Y
0E2A-  99 0D 07   STA  $070D,Y
0E2D-  88         DEY
0E2E-  D0 F7      BNE  $0E27
0E30-  60         RTS
0E31-  85 C7      STA  $C7     use this reg. so can keep 5FFF for checking
0E33-  A0 08      LDY  #$08    write 8char. in "60Hz buzz" signal
0E35-  06 C7      ASL  $C7
0E37-  A9 7F      LDA  ##$7F   "?" (flashing) = not OK
0E39-  B0 02      BCS  $0E3D   was the bit=1? if yes,keep flash. [?]
0E3B-  A9 D1      LDA  ##$D1     no,so       "Q" = OK
0E3D-  99 0D 07   STA  $070D,Y  and write on screen
0E40-  88         DEY
0E41-  06 C7      ASL  $C7       ?
0E43-  A9 7F      LDA  ##$7F
0E45-  B0 02      BCS  $0E49                 for I component
0E47-  A9 C9      LDA  ##$C9     I
0E49-  99 0D 07   STA  $070D,Y
0E4C-  88         DEY
0E4D-  D0 E6      BNE  $0E35
0E4F-  60         RTS
```

```
0E50-  A5 B8      LDA  $B8      height pointer          DISPLAY ANALYSIS HEIGHT
0E52-  20 B0 1C   JSR  $1CB0    convert to ascii BCD
0E55-  8D 56 06   STA  $0656    L ascii                 (ht#0 displays 31!)
0E58-  8A         TXA
0E59-  8D 55 06   STA  $0655    H ascii
0E5C-  60         RTS
-------------------------------------------------------------------------
0E60-  A9 A0      LDA  #$A0     " "                     CLEAR ANALYSIS HEIGHT
0E62-  8D 55 06   STA  $0655                            (and"W" locations)
0E65-  8D 56 06   STA  $0656    screen locations
0E68-  8D 9A 05   STA  $059A
0E6B-  60         RTS
-------------------------------------------------------------------------
0E6C-  FF         ???
0E6D-  FF         ???
0E6E-  FF         ???
0E6F-  FF         ???
-------------------------------------------------------------------------
0E70-  A2 0A      LDX  #$0A                             COMPLEX XCORREL(76msec)
0E72-  B5 57      LDA  $57,X
0E74-  95 9F      STA  $9F,X  move I4,Q4 to zero         4V3,4V2,4V1 Re and Im
0E76-  B5 4D      LDA  $4D,X   page"rolling"stores
0E78-  95 DF      STA  $DF,X
0E7A-  CA         DEX
0E7B-  D0 F5      BNE  $0E72
0E7D-  A0 40      LDY  #$40   lag counter
0E7F-  A9 FF      LDA  #$FF      = negative lag flag(=0 for zero and pos. lags)
0E81-  85 B4      STA  $B4
0E83-  A5 C1      LDA  $C1
0E85-  85 AC      STA  $AC    } set XCOR Accum. base addr. to 4V3 Re
0E87-  A5 C2      LDA  $C2
0E89-  85 AD      STA  $AD
0E8B-  18         CLC
0E8C-  A9 00      LDA  #$00   clear temporary #1's accumulator      I4vsI3
0E8E-  85 B2      STA  $B2                                         -ve lags,
0E90-  A5 A7      LDA  $A7                                    ( 8 bytes 'AND')
0E92-  25 75      AND  $75       get bit matches
0E94-  AA         TAX
0E95-  BD 00 63   LDA  $6300,X  look up #1 bits in byte from table
0E98-  65 B2      ADC  $B2
0E9A-  85 B2      STA  $B2
0E9C-  A5 A6      LDA  $A6
0E9E-  25 74      AND  $74
0EA0-  AA         TAX
0EA1-  BD 00 63   LDA  $6300,X
0EA4-  65 B2      ADC  $B2
0EA6-  85 B2      STA  $B2
0EA8-  A5 A5      LDA  $A5
0EAA-  25 73      AND  $73
0EAC-  AA         TAX
0EAD-  BD 00 63   LDA  $6300,X
0EB0-  65 B2      ADC  $B2
0EB2-  85 B2      STA  $B2
0EB4-  A5 A4      LDA  $A4
0EB6-  25 72      AND  $72
0EB8-  AA         TAX
0EB9-  BD 00 63   LDA  $6300,X
0EBC-  65 B2      ADC  $B2
0EBE-  85 B2      STA  $B2
0EC0-  A5 A3      LDA  $A3
0EC2-  25 71      AND  $71
0EC4-  AA         TAX
0EC5-  BD 00 63   LDA  $6300,X
0EC8-  65 B2      ADC  $B2
0ECA-  85 B2      STA  $B2
0ECC-  A5 A2      LDA  $A2
0ECE-  25 70      AND  $70
0ED0-  AA         TAX
0ED1-  BD 00 63   LDA  $6300,X
0ED4-  65 B2      ADC  $B2
0ED6-  85 B2      STA  $B2
0ED8-  A5 A1      LDA  $A1
0EDA-  25 6F      AND  $6F
0EDC-  AA         TAX
0EDD-  BD 00 63   LDA  $6300,X
0EE0-  65 B2      ADC  $B2
0EE2-  85 B2      STA  $B2
0EE4-  A5 A0      LDA  $A0
0EE6-  25 6E      AND  $6E
0EE8-  AA         TAX
0EE9-  BD 00 63   LDA  $6300,X
0EEC-  65 B2      ADC  $B2
0EEE-  85 B2      STA  $B2
```

```
0EF0-   A5 E7       LDA   $E7       Q4 byte          starting Q4vsQ3, -ve lags
0EF2-   25 6B       AND   $6B       Q3 byte                          (8 bytes 'anded')
0EF4-   AA          TAX
0EF5-   BD 00 63    LDA   $6300,X
0EF8-   65 B2       ADC   $B2
0EFA-   85 B2       STA   $B2
0EFC-   A5 E6       LDA   $E6
0EFE-   25 6A       AND   $6A
0F00-   AA          TAX
0F01-   BD 00 63    LDA   $6300,X
0F04-   65 B2       ADC   $B2
0F06-   85 B2       STA   $B2
0F08-   A5 E5       LDA   $E5
0F0A-   25 69       AND   $69
0F0C-   AA          TAX
0F0D-   BD 00 63    LDA   $6300,X
0F10-   65 B2       ADC   $B2
0F12-   85 B2       STA   $B2
0F14-   A5 E4       LDA   $E4
0F16-   25 68       AND   $68
0F18-   AA          TAX
0F19-   BD 00 63    LDA   $6300,X
0F1C-   65 B2       ADC   $B2
0F1E-   85 B2       STA   $B2
0F20-   A5 E3       LDA   $E3
0F22-   25 67       AND   $67
0F24-   AA          TAX
0F25-   BD 00 63    LDA   $6300,X
0F28-   65 B2       ADC   $B2
0F2A-   85 B2       STA   $B2
0F2C-   A5 E2       LDA   $E2
0F2E-   25 66       AND   $66
0F30-   AA          TAX
0F31-   BD 00 63    LDA   $6300,X
0F34-   65 B2       ADC   $B2
0F36-   85 B2       STA   $B2
0F38-   A5 E1       LDA   $E1
0F3A-   25 65       AND   $65
0F3C-   AA          TAX
0F3D-   BD 00 63    LDA   $6300,X
0F40-   65 B2       ADC   $B2
0F42-   85 B2       STA   $B2
0F44-   A5 E0       LDA   $E0
0F46-   25 64       AND   $64
0F48-   AA          TAX
0F49-   BD 00 63    LDA   $6300,X
0F4C-   65 B2       ADC   $B2
0F4E-   85 B2       STA   $B2
0F50-   24 B4       BIT   $B4       if still on -ve lags,then skip
0F52-   30 30       BMI   $0F84
0F54-   A5 9F       LDA   $9F       +ve lags,so tack on I4vsI3(2 bytes)
0F56-   25 6D       AND   $6D
0F58-   AA          TAX
0F59-   BD 00 63    LDA   $6300,X
0F5C-   65 B2       ADC   $B2
0F5E-   85 B2       STA   $B2
0F60-   A5 9E       LDA   $9E
0F62-   25 6C       AND   $6C
0F64-   AA          TAX
0F65-   BD 00 63    LDA   $6300,X
0F68-   65 B2       ADC   $B2
0F6A-   85 B2       STA   $B2
0F6C-   A5 DF       LDA   $DF       and tack on Q4vsQ3 (2 bytes)
0F6E-   25 63       AND   $63
0F70-   AA          TAX
0F71-   BD 00 63    LDA   $6300,X
0F74-   65 B2       ADC   $B2
0F76-   85 B2       STA   $B2
0F78-   A5 DE       LDA   $DE
0F7A-   25 62       AND   $62
0F7C-   AA          TAX
0F7D-   BD 00 63    LDA   $6300,X
0F80-   65 B2       ADC   $B2
0F82-   85 B2       STA   $B2
0F84-   71 AC       ADC   ($AC),Y     add I4vsI3+Q4vsQ3 to xcor. accumulator  Re
0F86-   91 AC       STA   ($AC),Y                for this lag
0F88-   90 08       BCC   $0F92
0F8A-   88          DEY
0F8B-   A9 00       LDA   #$00        add 1 to H byte of accum (INC not avail.in this
0F8D-   71 AC       ADC   ($AC),Y                              mode on 6502)
0F8F-   91 AC       STA   ($AC),Y
0F91-   C8          INY
```

```
OF92-   A5 AC       LDA   $AC          move xcor accum pointer down to
OF94-   38          SEC                     4vs3 Im (=Q4vsI3-I4vsQ3)
OF95-   E9 40       SBC   #$40
OF97-   85 AC       STA   $AC
OF99-   B0 02       BCS   $OF9D
OF9B-   C6 AD       DEC   $AD
OF9D-   18          CLC
OF9E-   A9 00       LDA   #$00
OFA0-   85 B2       STA   $B2       = accum for Q4vsI3
OFA2-   85 B7       STA   $B7       =accum for I4vsQ3
OFA4-   A5 E7       LDA   $E7        Start Q4vsI3, (8 bytes 'anded')
OFA6-   25 75       AND   $75
OFA8-   AA          TAX
OFA9-   BD 00 63    LDA   $6300,X
OFAC-   65 B2       ADC   $B2
OFAE-   85 B2       STA   $B2
OFB0-   A5 E6       LDA   $E6
OFB2-   25 74       AND   $74
OFB4-   AA          TAX
OFB5-   BD 00 63    LDA   $6300,X
OFB8-   65 B2       ADC   $B2
OFBA-   85 B2       STA   $B2
OFBC-   A5 E5       LDA   $E5
OFBE-   25 73       AND   $73
OFC0-   AA          TAX
OFC1-   BD 00 63    LDA   $6300,X
OFC4-   65 B2       ADC   $B2
OFC6-   85 B2       STA   $B2
OFC8-   A5 E4       LDA   $E4
OFCA-   25 72       AND   $72
OFCC-   AA          TAX
OFCD-   BD 00 63    LDA   $6300,X
OFD0-   65 B2       ADC   $B2
OFD2-   85 B2       STA   $B2
OFD4-   A5 E3       LDA   $E3
OFD6-   25 71       AND   $71
OFD8-   AA          TAX
OFD9-   BD 00 63    LDA   $6300,X
OFDC-   65 B2       ADC   $B2
OFDE-   85 B2       STA   $B2
OFE0-   A5 E2       LDA   $E2
OFE2-   25 70       AND   $70
OFE4-   AA          TAX
OFE5-   BD 00 63    LDA   $6300,X
OFE8-   65 B2       ADC   $B2
OFEA-   85 B2       STA   $B2
OFEC-   A5 E1       LDA   $E1
OFEE-   25 6F       AND   $6F
OFF0-   AA          TAX
OFF1-   BD 00 63    LDA   $6300,X
OFF4-   65 B2       ADC   $B2
OFF6-   85 B2       STA   $B2
OFF8-   A5 E0       LDA   $E0
OFFA-   25 6E       AND   $6E
OFFC-   AA          TAX
OFFD-   BD 00 63    LDA   $6300,X
1000-   65 B2       ADC   $B2
1002-   85 B2       STA   $B2
1004-   A5 A7       LDA   $A7       start I4vsQ3 (8 bytes anded)
1006-   25 6B       AND   $6B
1008-   AA          TAX
1009-   BD 00 63    LDA   $6300,X
100C-   65 B7       ADC   $B7
100E-   85 B7       STA   $B7
1010-   A5 A6       LDA   $A6
1012-   25 6A       AND   $6A
1014-   AA          TAX
1015-   BD 00 63    LDA   $6300,X
1018-   65 B7       ADC   $B7
101A-   85 B7       STA   $B7
101C-   A5 A5       LDA   $A5
101E-   25 69       AND   $69
1020-   AA          TAX
1021-   BD 00 63    LDA   $6300,X
1024-   65 B7       ADC   $B7
1026-   85 B7       STA   $B7
1028-   A5 A4       LDA   $A4
102A-   25 68       AND   $68
102C-   AA          TAX
102D-   BD 00 63    LDA   $6300,X
1030-   65 B7       ADC   $B7
1032-   85 B7       STA   $B7
```

```
1034-   A5 A3       LDA   $A3
1036-   25 67       AND   $67
1038-   AA          TAX
1039-   BD 00 63    LDA   $6300,X
103C-   65 B7       ADC   $B7
103E-   85 B7       STA   $B7
1040-   A5 A2       LDA   $A2
1042-   25 66       AND   $66
1044-   AA          TAX
1045-   BD 00 63    LDA   $6300,X
1048-   65 B7       ADC   $B7
104A-   85 B7       STA   $B7
104C-   A5 A1       LDA   $A1
104E-   25 65       AND   $65
1050-   AA          TAX
1051-   BD 00 63    LDA   $6300,X
1054-   65 B7       ADC   $B7
1056-   85 B7       STA   $B7
1058-   A5 A0       LDA   $A0
105A-   25 64       AND   $64
105C-   AA          TAX
105D-   BD 00 63    LDA   $6300,X
1060-   65 B7       ADC   $B7
1062-   85 B7       STA   $B7
1064-   24 B4       BIT   $B4       negative lags still?
1066-   30 30       BMI   $1098    yes,so skip
1068-   A5 DF       LDA   $DF       +ve lags,so tack on 2 more and's
106A-   25 6D       AND   $6D        last 2 bytes of Q4vsI3
106C-   AA          TAX
106D-   BD 00 63    LDA   $6300,X
1070-   65 B2       ADC   $B2
1072-   85 B2       STA   $B2
1074-   A5 DE       LDA   $DE
1076-   25 6C       AND   $6C
1078-   AA          TAX
1079-   BD 00 63    LDA   $6300,X
107C-   65 B2       ADC   $B2
107E-   85 B2       STA   $B2
1080-   A5 9F       LDA   $9F       last 2 bytes of I4vsQ3
1082-   25 63       AND   $63
1084-   AA          TAX
1085-   BD 00 63    LDA   $6300,X
1088-   65 B7       ADC   $B7
108A-   85 B7       STA   $B7
108C-   A5 9E       LDA   $9E
108E-   25 62       AND   $62
1090-   AA          TAX
1091-   BD 00 63    LDA   $6300,X
1094-   65 B7       ADC   $B7
1096-   85 B7       STA   $B7
1098-   38          SEC                 now add (B2 minus B7) to the accumulator
1099-   B1 AC       LDA   ($AC),Y
109B-   E5 B7       SBC   $B7
109D-   B0 10       BCS   $10AF
109F-   65 B2       ADC   $B2
10A1-   91 AC       STA   ($AC),Y
10A3-   B0 19       BCS   $10BE
10A5-   88          DEY
10A6-   B1 AC       LDA   ($AC),Y
10A8-   E9 00       SBC   #$00
10AA-   91 AC       STA   ($AC),Y
10AC-   C8          INY
10AD-   D0 0F       BNE   $10BE
10AF-   18          CLC
10B0-   65 B2       ADC   $B2
10B2-   91 AC       STA   ($AC),Y
10B4-   90 08       BCC   $10BE
10B6-   88          DEY
10B7-   A9 00       LDA   #$00
10B9-   71 AC       ADC   ($AC),Y
10BB-   91 AC       STA   ($AC),Y
10BD-   C8          INY             ---- end of accumulator operation
10BE-   38          SEC
10BF-   A5 AC       LDA   $AC       move Xcor pointer down to 4V2 Re
10C1-   E9 40       SBC   #$40           and do all 4V2 ops
10C3-   85 AC       STA   $AC
10C5-   B0 02       BCS   $10C9
10C7-   C6 AD       DEC   $AD
10C9-   18          CLC
```

```
10CA-   A9 00       LDA   #$00        Start I4vsI2
10CC-   85 B2       STA   $B2
10CE-   A5 A7       LDA   $A7
10D0-   25 89       AND   $89
10D2-   AA          TAX
10D3-   BD 00 63    LDA   $6300,X
10D6-   65 B2       ADC   $B2
10D8-   85 B2       STA   $B2
10DA-   A5 A6       LDA   $A6
10DC-   25 88       AND   $88
10DE-   AA          TAX
10DF-   BD 00 63    LDA   $6300,X
10E2-   65 B2       ADC   $B2
10E4-   85 B2       STA   $B2
10E6-   A5 A5       LDA   $A5
10E8-   25 87       AND   $87
10EA-   AA          TAX
10EB-   BD 00 63    LDA   $6300,X
10EE-   65 B2       ADC   $B2
10F0-   85 B2       STA   $B2
10F2-   A5 A4       LDA   $A4
10F4-   25 86       AND   $86
10F6-   AA          TAX
10F7-   BD 00 63    LDA   $6300,X
10FA-   65 B2       ADC   $B2
10FC-   85 B2       STA   $B2
10FE-   A5 A3       LDA   $A3
1100-   25 85       AND   $85
1102-   AA          TAX
1103-   BD 00 63    LDA   $6300,X
1106-   65 B2       ADC   $B2
1108-   85 B2       STA   $B2
110A-   A5 A2       LDA   $A2
110C-   25 84       AND   $84
110E-   AA          TAX
110F-   BD 00 63    LDA   $6300,X
1112-   65 B2       ADC   $B2
1114-   85 B2       STA   $B2
1116-   A5 A1       LDA   $A1
1118-   25 83       AND   $83
111A-   AA          TAX
111B-   BD 00 63    LDA   $6300,X
111E-   65 B2       ADC   $B2
1120-   85 B2       STA   $B2
1122-   A5 A0       LDA   $A0
1124-   25 82       AND   $82
1126-   AA          TAX
1127-   BD 00 63    LDA   $6300,X
112A-   65 B2       ADC   $B2
112C-   85 B2       STA   $B2
112E-   A5 E7       LDA   $E7        start Q4vsQ2
1130-   25 7F       AND   $7F
1132-   AA          TAX
1133-   BD 00 63    LDA   $6300,X
1136-   65 B2       ADC   $B2
1138-   85 B2       STA   $B2
113A-   A5 E6       LDA   $E6
113C-   25 7E       AND   $7E
113E-   AA          TAX
113F-   BD 00 63    LDA   $6300,X
1142-   65 B2       ADC   $B2
1144-   85 B2       STA   $B2
1146-   A5 E5       LDA   $E5
1148-   25 7D       AND   $7D
114A-   AA          TAX
114B-   BD 00 63    LDA   $6300,X
114E-   65 B2       ADC   $B2
1150-   85 B2       STA   $B2
1152-   A5 E4       LDA   $E4
1154-   25 7C       AND   $7C
1156-   AA          TAX
1157-   BD 00 63    LDA   $6300,X
115A-   65 B2       ADC   $B2
115C-   85 B2       STA   $B2
115E-   A5 E3       LDA   $E3
1160-   25 7B       AND   $7B
1162-   AA          TAX
1163-   BD 00 63    LDA   $6300,X
1166-   65 B2       ADC   $B2
1168-   85 B2       STA   $B2
```

```
116A-   A5 E2        LDA   $E2
116C-   25 7A        AND   $7A
116E-   AA           TAX
116F-   BD 00 63     LDA   $6300,X
1172-   65 B2        ADC   $B2
1174-   85 B2        STA   $B2
1176-   A5 E1        LDA   $E1
1178-   25 79        AND   $79
117A-   AA           TAX
117B-   BD 00 63     LDA   $6300,X
117E-   65 B2        ADC   $B2
1180-   85 B2        STA   $B2
1182-   A5 E0        LDA   $E0
1184-   25 78        AND   $78
1186-   AA           TAX
1187-   BD 00 63     LDA   $6300,X
118A-   65 B2        ADC   $B2
118C-   85 B2        STA   $B2
118E-   24 B4        BIT   $B4       check for -ve lag
1190-   30 30        BMI   $11C2
1192-   A5 9F        LDA   $9F        finish +ve lag calculation
1194-   25 81        AND   $81
1196-   AA           TAX
1197-   BD 00 63     LDA   $6300,X
119A-   65 B2        ADC   $B2
119C-   85 B2        STA   $B2
119E-   A5 9E        LDA   $9E
11A0-   25 80        AND   $80
11A2-   AA           TAX
11A3-   BD 00 63     LDA   $6300,X
11A6-   65 B2        ADC   $B2
11A8-   85 B2        STA   $B2
11AA-   A5 DF        LDA   $DF
11AC-   25 77        AND   $77
11AE-   AA           TAX
11AF-   BD 00 63     LDA   $6300,X
11B2-   65 B2        ADC   $B2
11B4-   85 B2        STA   $B2
11B6-   A5 DE        LDA   $DE
11B8-   25 76        AND   $76
11BA-   AA           TAX
11BB-   BD 00 63     LDA   $6300,X
11BE-   65 B2        ADC   $B2
11C0-   85 B2        STA   $B2
11C2-   71 AC        ADC   ($AC),Y
11C4-   91 AC        STA   ($AC),Y
11C6-   90 08        BCC   $11D0
11C8-   88           DEY
11C9-   A9 00        LDA   #$00
11CB-   71 AC        ADC   ($AC),Y
11CD-   91 AC        STA   ($AC),Y
11CF-   C8           INY             ------
11D0-   A5 AC        LDA   $AC     point to 4V2 Im
11D2-   38           SEC
11D3-   E9 40        SBC   #$40
11D5-   85 AC        STA   $AC
11D7-   B0 02        BCS   $11DB
11D9-   C6 AD        DEC   $AD
11DB-   18           CLC
11DC-   A9 00        LDA   #$00
11DE-   85 B2        STA   $B2
11E0-   85 B7        STA   $B7
11E2-   A5 E7        LDA   $E7       start Q4 vs I2
11E4-   25 89        AND   $89
11E6-   AA           TAX
11E7-   BD 00 63     LDA   $6300,X
11EA-   65 B2        ADC   $B2
11EC-   85 B2        STA   $B2
11EE-   A5 E6        LDA   $E6
11F0-   25 88        AND   $88
11F2-   AA           TAX
11F3-   BD 00 63     LDA   $6300,X
11F6-   65 B2        ADC   $B2
11F8-   85 B2        STA   $B2
11FA-   A5 E5        LDA   $E5
11FC-   25 87        AND   $87
11FE-   AA           TAX
11FF-   BD 00 63     LDA   $6300,X
1202-   65 B2        ADC   $B2
1204-   85 B2        STA   $B2
```

```
1206-   A5 E4        LDA   $E4
1208-   25 86        AND   $86
120A-   AA           TAX
120B-   BD 00 63     LDA   $6300,X
120E-   65 B2        ADC   $B2
1210-   85 B2        STA   $B2
1212-   A5 E3        LDA   $E3
1214-   25 85        AND   $85
1216-   AA           TAX
1217-   BD 00 63     LDA   $6300,X
121A-   65 B2        ADC   $B2
121C-   85 B2        STA   $B2
121E-   A5 E2        LDA   $E2
1220-   25 84        AND   $84
1222-   AA           TAX
1223-   BD 00 63     LDA   $6300,X
1226-   65 B2        ADC   $B2
1228-   85 B2        STA   $B2
122A-   A5 E1        LDA   $E1
122C-   25 83        AND   $83
122E-   AA           TAX
122F-   BD 00 63     LDA   $6300,X
1232-   65 B2        ADC   $B2
1234-   85 B2        STA   $B2
1236-   A5 E0        LDA   $E0
1238-   25 82        AND   $82
123A-   AA           TAX
123B-   BD 00 63     LDA   $6300,X
123E-   65 B2        ADC   $B2
1240-   85 B2        STA   $B2
1242-   A5 A7        LDA   $A7        start I4vsQ2
1244-   25 7F        AND   $7F
1246-   AA           TAX
1247-   BD 00 63     LDA   $6300,X
124A-   65 B7        ADC   $B7
124C-   85 B7        STA   $B7
124E-   A5 A6        LDA   $A6
1250-   25 7E        AND   $7E
1252-   AA           TAX
1253-   BD 00 63     LDA   $6300,X
1256-   65 B7        ADC   $B7
1258-   85 B7        STA   $B7
125A-   A5 A5        LDA   $A5
125C-   25 7D        AND   $7D
125E-   AA           TAX
125F-   BD 00 63     LDA   $6300,X
1262-   65 B7        ADC   $B7
1264-   85 B7        STA   $B7
1266-   A5 A4        LDA   $A4
1268-   25 7C        AND   $7C
126A-   AA           TAX
126B-   BD 00 63     LDA   $6300,X
126E-   65 B7        ADC   $B7
1270-   85 B7        STA   $B7
1272-   A5 A3        LDA   $A3
1274-   25 7B        AND   $7B
1276-   AA           TAX
1277-   BD 00 63     LDA   $6300,X
127A-   65 B7        ADC   $B7
127C-   85 B7        STA   $B7
127E-   A5 A2        LDA   $A2
1280-   25 7A        AND   $7A
1282-   AA           TAX
1283-   BD 00 63     LDA   $6300,X
1286-   65 B7        ADC   $B7
1288-   85 B7        STA   $B7
128A-   A5 A1        LDA   $A1
128C-   25 79        AND   $79
128E-   AA           TAX
128F-   BD 00 63     LDA   $6300,X
1292-   65 B7        ADC   $B7
1294-   85 B7        STA   $B7
1296-   A5 A0        LDA   $A0
1298-   25 78        AND   $78
129A-   AA           TAX
129B-   BD 00 63     LDA   $6300,X
129E-   65 B7        ADC   $B7
12A0-   85 B7        STA   $B7
12A2-   24 B4        BIT   $B4        skip if -ve lag
12A4-   30 30        BMI   $12D6
```

```
12A6-   A5 DF       LDA   $DF      finish up positive lag calculation
12A8-   25 B1       AND   $B1
12AA-   AA          TAX
12AB-   BD 00 63    LDA   $6300,X
12AE-   65 B2       ADC   $B2
12B0-   85 B2       STA   $B2
12B2-   A5 DE       LDA   $DE
12B4-   25 B0       AND   $B0
12B6-   AA          TAX
12B7-   BD 00 63    LDA   $6300,X
12BA-   65 B2       ADC   $B2
12BC-   85 B2       STA   $B2
12BE-   A5 9F       LDA   $9F
12C0-   25 77       AND   $77
12C2-   AA          TAX
12C3-   BD 00 63    LDA   $6300,X
12C6-   65 B7       ADC   $B7
12C8-   85 B7       STA   $B7
12CA-   A5 9E       LDA   $9E
12CC-   25 76       AND   $76
12CE-   AA          TAX
12CF-   BD 00 63    LDA   $6300,X
12D2-   65 B7       ADC   $B7
12D4-   85 B7       STA   $B7
12D6-   38          SEC           add (B2 minus B7) to 4V2 Im accum.
12D7-   B1 AC       LDA   ($AC),Y
12D9-   E5 B7       SBC   $B7
12DB-   B0 10       BCS   $12ED
12DD-   65 B2       ADC   $B2
12DF-   91 AC       STA   ($AC),Y
12E1-   B0 19       BCS   $12FC
12E3-   88          DEY
12E4-   B1 AC       LDA   ($AC),Y
12E6-   E9 00       SBC   #$00
12E8-   91 AC       STA   ($AC),Y
12EA-   C8          INY
12EB-   D0 0F       BNE   $12FC
12ED-   18          CLC
12EE-   65 B2       ADC   $B2
12F0-   91 AC       STA   ($AC),Y
12F2-   90 08       BCC   $12FC
12F4-   88          DEY
12F5-   A9 00       LDA   #$00
12F7-   71 AC       ADC   ($AC),Y
12F9-   91 AC       STA   ($AC),Y
12FB-   C8          INY           -----end of add
12FC-   38          SEC
12FD-   A5 AC       LDA   $AC      move pointer to 4V1 Re
12FF-   E9 40       SBC   #$40
1301-   85 AC       STA   $AC
1303-   B0 02       BCS   $1307
1305-   C6 AD       DEC   $AD
1307-   18          CLC
1308-   A9 00       LDA   #$00
130A-   85 B2       STA   $B2
130C-   A5 A7       LDA   $A7      start I4vsI1
130E-   25 9D       AND   $9D
1310-   AA          TAX
1311-   BD 00 63    LDA   $6300,X
1314-   65 B2       ADC   $B2
1316-   85 B2       STA   $B2
1318-   A5 A6       LDA   $A6
131A-   25 9C       AND   $9C
131C-   AA          TAX
131D-   BD 00 63    LDA   $6300,X
1320-   65 B2       ADC   $B2
1322-   85 B2       STA   $B2
1324-   A5 A5       LDA   $A5
1326-   25 9B       AND   $9B
1328-   AA          TAX
1329-   BD 00 63    LDA   $6300,X
132C-   65 B2       ADC   $B2
132E-   85 B2       STA   $B2
1330-   A5 A4       LDA   $A4
1332-   25 9A       AND   $9A
1334-   AA          TAX
1335-   BD 00 63    LDA   $6300,X
1338-   65 B2       ADC   $B2
133A-   85 B2       STA   $B2
```

```
133C-   A5 A3       LDA   $A3
133E-   25 99       AND   $99
1340-   AA          TAX
1341-   BD 00 63    LDA   $6300,X
1344-   65 B2       ADC   $B2
1346-   85 B2       STA   $B2
1348-   A5 A2       LDA   $A2
134A-   25 98       AND   $98
134C-   AA          TAX
134D-   BD 00 63    LDA   $6300,X
1350-   65 B2       ADC   $B2
1352-   85 B2       STA   $B2
1354-   A5 A1       LDA   $A1
1356-   25 97       AND   $97
1358-   AA          TAX
1359-   BD 00 63    LDA   $6300,X
135C-   65 B2       ADC   $B2
135E-   85 B2       STA   $B2
1360-   A5 A0       LDA   $A0
1362-   25 96       AND   $96
1364-   AA          TAX
1365-   BD 00 63    LDA   $6300,X
1368-   65 B2       ADC   $B2
136A-   85 B2       STA   $B2
136C-   A5 E7       LDA   $E7       start. Q4vsQ1
136E-   25 93       AND   $93
1370-   AA          TAX
1371-   BD 00 63    LDA   $6300,X
1374-   65 B2       ADC   $B2
1376-   85 B2       STA   $B2
1378-   A5 E6       LDA   $E6
137A-   25 92       AND   $92
137C-   AA          TAX
137D-   BD 00 63    LDA   $6300,X
1380-   65 B2       ADC   $B2
1382-   85 B2       STA   $B2
1384-   A5 E5       LDA   $E5
1386-   25 91       AND   $91
1388-   AA          TAX
1389-   BD 00 63    LDA   $6300,X
138C-   65 B2       ADC   $B2
138E-   85 B2       STA   $B2
1390-   A5 E4       LDA   $E4
1392-   25 90       AND   $90
1394-   AA          TAX
1395-   BD 00 63    LDA   $6300,X
1398-   65 B2       ADC   $B2
139A-   85 B2       STA   $B2
139C-   A5 E3       LDA   $E3
139E-   25 8F       AND   $8F
13A0-   AA          TAX
13A1-   BD 00 63    LDA   $6300,X
13A4-   65 B2       ADC   $B2
13A6-   85 B2       STA   $B2
13A8-   A5 E2       LDA   $E2
13AA-   25 8E       AND   $8E
13AC-   AA          TAX
13AD-   BD 00 63    LDA   $6300,X
13B0-   65 B2       ADC   $B2
13B2-   85 B2       STA   $B2
13B4-   A5 E1       LDA   $E1
13B6-   25 8D       AND   $8D
13B8-   AA          TAX
13B9-   BD 00 63    LDA   $6300,X
13BC-   65 B2       ADC   $B2
13BE-   85 B2       STA   $B2
13C0-   A5 E0       LDA   $E0
13C2-   25 8C       AND   $8C
13C4-   AA          TAX
13C5-   BD 00 63    LDA   $6300,X
13C8-   65 B2       ADC   $B2
13CA-   85 B2       STA   $B2
13CC-   24 B4       BIT   $B4
13CE-   30 30       BMI   $1400     skip if -ve lag
```

```
13D0-   A5 9F       LDA   $9F        finish positive lags
13D2-   25 95       AND   $95
13D4-   AA          TAX
13D5-   BD 00 63    LDA   $6300,X
13D8-   65 B2       ADC   $B2
13DA-   85 B2       STA   $B2
13DC-   A5 9E       LDA   $9E
13DE-   25 94       AND   $94
13E0-   AA          TAX
13E1-   BD 00 63    LDA   $6300,X
13E4-   65 B2       ADC   $B2
13E6-   85 B2       STA   $B2
13E8-   A5 DF       LDA   $DF
13EA-   25 8B       AND   $8B
13EC-   AA          TAX
13ED-   BD 00 63    LDA   $6300,X
13F0-   65 B2       ADC   $B2
13F2-   85 B2       STA   $B2
13F4-   A5 DE       LDA   $DE
13F6-   25 8A       AND   $8A
13F8-   AA          TAX
13F9-   BD 00 63    LDA   $6300,X
13FC-   65 B2       ADC   $B2
13FE-   85 B2       STA   $B2
1400-   71 AC       ADC   ($AC),Y
1402-   91 AC       STA   ($AC),Y
1404-   90 08       BCC   $140E
1406-   88          DEY
1407-   A9 00       LDA   #$00
1409-   71 AC       ADC   ($AC),Y
140B-   91 AC       STA   ($AC),Y
140D-   C8          INY
140E-   A5 AC       LDA   $AC          move Xcor accum pointer to 4V1 Im
1410-   38          SEC
1411-   E9 40       SBC   #$40
1413-   85 AC       STA   $AC
1415-   B0 02       BCS   $1419
1417-   C6 AD       DEC   $AD
1419-   18          CLC
141A-   A9 00       LDA   #$00
141C-   85 B2       STA   $B2
141E-   85 B7       STA   $B7
1420-   A5 E7       LDA   $E7      start Q4vsI1
1422-   25 9D       AND   $9D
1424-   AA          TAX
1425-   BD 00 63    LDA   $6300,X
1428-   65 B2       ADC   $B2
142A-   85 B2       STA   $B2
142C-   A5 E6       LDA   $E6
142E-   25 9C       AND   $9C
1430-   AA          TAX
1431-   BD 00 63    LDA   $6300,X
1434-   65 B2       ADC   $B2
1436-   85 B2       STA   $B2
1438-   A5 E5       LDA   $E5
143A-   25 9B       AND   $9B
143C-   AA          TAX
143D-   BD 00 63    LDA   $6300,X
1440-   65 B2       ADC   $B2
1442-   85 B2       STA   $B2
1444-   A5 E4       LDA   $E4
1446-   25 9A       AND   $9A
1448-   AA          TAX
1449-   BD 00 63    LDA   $6300,X
144C-   65 B2       ADC   $B2
144E-   85 B2       STA   $B2
1450-   A5 E3       LDA   $E3
1452-   25 99       AND   $99
1454-   AA          TAX
1455-   BD 00 63    LDA   $6300,X
1458-   65 B2       ADC   $B2
145A-   85 B2       STA   $B2
145C-   A5 E2       LDA   $E2
145E-   25 98       AND   $98
1460-   AA          TAX
1461-   BD 00 63    LDA   $6300,X
1464-   65 B2       ADC   $B2
1466-   85 B2       STA   $B2
```

```
1468-   A5 E1      LDA   $E1
146A-   25 97      AND   $97
146C-   AA         TAX
146D-   BD 00 63   LDA   $6300,X
1470-   65 B2      ADC   $B2
1472-   85 B2      STA   $B2
1474-   A5 E0      LDA   $E0
1476-   25 96      AND   $96
1478-   AA         TAX
1479-   BD 00 63   LDA   $6300,X
147C-   65 B2      ADC   $B2
147E-   85 B2      STA   $B2
1480-   A5 A7      LDA   $A7        start I4vsQ1
1482-   25 93      AND   $93
1484-   AA         TAX
1485-   BD 00 63   LDA   $6300,X
1488-   65 B7      ADC   $B7
148A-   85 B7      STA   $B7
148C-   A5 A6      LDA   $A6
148E-   25 92      AND   $92
1490-   AA         TAX
1491-   BD 00 63   LDA   $6300,X
1494-   65 B7      ADC   $B7
1496-   85 B7      STA   $B7
1498-   A5 A5      LDA   $A5
149A-   25 91      AND   $91
149C-   AA         TAX
149D-   BD 00 63   LDA   $6300,X
14A0-   65 B7      ADC   $B7
14A2-   85 B7      STA   $B7
14A4-   A5 A4      LDA   $A4
14A6-   25 90      AND   $90
14A8-   AA         TAX
14A9-   BD 00 63   LDA   $6300,X
14AC-   65 B7      ADC   $B7
14AE-   85 B7      STA   $B7
14B0-   A5 A3      LDA   $A3
14B2-   25 8F      AND   $8F
14B4-   AA         TAX
14B5-   BD 00 63   LDA   $6300,X
14B8-   65 B7      ADC   $B7
14BA-   85 B7      STA   $B7
14BC-   A5 A2      LDA   $A2
14BE-   25 8E      AND   $8E
14C0-   AA         TAX
14C1-   BD 00 63   LDA   $6300,X
14C4-   65 B7      ADC   $B7
14C6-   85 B7      STA   $B7
14C8-   A5 A1      LDA   $A1
14CA-   25 8D      AND   $8D
14CC-   AA         TAX
14CD-   BD 00 63   LDA   $6300,X
14D0-   65 B7      ADC   $B7
14D2-   85 B7      STA   $B7
14D4-   A5 A0      LDA   $A0
14D6-   25 8C      AND   $8C
14D8-   AA         TAX
14D9-   BD 00 63   LDA   $6300,X
14DC-   65 B7      ADC   $B7
14DE-   85 B7      STA   $B7
14E0-   24 B4      BIT   $B4
14E2-   30 30      BMI   $1514      skip if -ve lags
14E4-   A5 DF      LDA   $DF        finish up +ve lag calculation
14E6-   25 95      AND   $95
14E8-   AA         TAX
14E9-   BD 00 63   LDA   $6300,X
14EC-   65 B2      ADC   $B2
14EE-   85 B2      STA   $B2
14F0-   A5 DE      LDA   $DE
14F2-   25 94      AND   $94
14F4-   AA         TAX
14F5-   BD 00 63   LDA   $6300,X
14F8-   65 B2      ADC   $B2
14FA-   85 B2      STA   $B2
14FC-   A5 9F      LDA   $9F
14FE-   25 8B      AND   $8B
1500-   AA         TAX
1501-   BD 00 63   LDA   $6300,X
1504-   65 B7      ADC   $B7
1506-   85 B7      STA   $B7
1508-   A5 9E      LDA   $9E
150A-   25 8A      AND   $8A
150C-   AA         TAX
150D-   BD 00 63   LDA   $6300,X
1510-   65 B7      ADC   $B7
1512-   85 B7      STA   $B7
```

```
1514-   38          SEC             add Q4vsI1-I4vsQ1 to xcor accum 4V1 Im
1515-   B1 AC       LDA   ($AC),Y
1517-   E5 B7       SBC   $B7
1519-   B0 10       BCS   $152B
151B-   65 B2       ADC   $B2
151D-   91 AC       STA   ($AC),Y
151F-   B0 19       BCS   $153A
1521-   88          DEY
1522-   B1 AC       LDA   ($AC),Y
1524-   E9 00       SBC   #$00
1526-   91 AC       STA   ($AC),Y
1528-   C8          INY
1529-   D0 0F       BNE   $153A
152B-   18          CLC
152C-   65 B2       ADC   $B2
152E-   91 AC       STA   ($AC),Y
1530-   90 08       BCC   $153A
1532-   88          DEY
1533-   A9 00       LDA   #$00
1535-   71 AC       ADC   ($AC),Y
1537-   91 AC       STA   ($AC),Y
1539-   C8          INY             ------end of add
153A-   24 B4       BIT   $B4        now shift sequence by 1 bit (# bytes to rol
153C-   10 3F       BPL   $157D                diff for -ve or (0&+ve) lags
153E-   18          CLC
153F-   26 A9       ROL   $A9           ROLL I4 1 bit to left
1541-   26 A8       ROL   $A8
1543-   26 A7       ROL   $A7                 -ve lag
1545-   26 A6       ROL   $A6
1547-   26 A5       ROL   $A5
1549-   26 A4       ROL   $A4
154B-   26 A3       ROL   $A3
154D-   26 A2       ROL   $A2
154F-   26 A1       ROL   $A1
1551-   26 A0       ROL   $A0
1553-   18          CLC
1554-   26 E9       ROL   $E9
1556-   26 E8       ROL   $E8           ROLL Q4 1 bit to left
1558-   26 E7       ROL   $E7
155A-   26 E6       ROL   $E6               -ve lag
155C-   26 E5       ROL   $E5
155E-   26 E4       ROL   $E4
1560-   26 E3       ROL   $E3
1562-   26 E2       ROL   $E2
1564-   26 E1       ROL   $E1
1566-   26 E0       ROL   $E0
1568-   88          DEY             decrement lag cntr/xcor accum pointer
1569-   88          DEY
156A-   C0 20       CPY   #$20
156C-   D0 0C       BNE   $157A     next lag=zero lag?
156E-   A9 00       LDA   #$00      yes,so set 0&+ve lag flag
1570-   85 B4       STA   $B4
1572-   85 9E       STA   $9E
1574-   85 9F       STA   $9F       and clear 2 bytes below rolling binary
1576-   85 DE       STA   $DE
1578-   85 DF       STA   $DF
157A-   4C 83 0E    JMP   $0E83     and skip around
157D-   88          DEY             !do the following after 0 or +ve lags only
157E-   88          DEY
157F-   F0 2D       BEQ   $15AE   finished all lags,so quit
1581-   18          CLC
1582-   26 A7       ROL   $A7         roll I4 left by 1 bit
1584-   26 A6       ROL   $A6              +ve lags
1586-   26 A5       ROL   $A5
1588-   26 A4       ROL   $A4
158A-   26 A3       ROL   $A3
158C-   26 A2       ROL   $A2
158E-   26 A1       ROL   $A1
1590-   26 A0       ROL   $A0
1592-   26 9F       ROL   $9F
1594-   26 9E       ROL   $9E
1596-   18          CLC
1597-   26 E7       ROL   $E7         roll Q4 left by 1 bit
1599-   26 E6       ROL   $E6
159B-   26 E5       ROL   $E5              +ve lags
159D-   26 E4       ROL   $E4
159F-   26 E3       ROL   $E3
15A1-   26 E2       ROL   $E2
15A3-   26 E1       ROL   $E1
15A5-   26 E0       ROL   $E0
15A7-   26 DF       ROL   $DF
15A9-   26 DE       ROL   $DE
15AB-   4C 83 0E    JMP   $0E83
15AE-   60          RTS                     whew!
```

(21)

```
15C0-  A9 59        LDA  #$59
15C2-  85 B7        STA  $B7     point to I4(8 bytes)      COUNT #1's in I1,Q1 (2 ms)
15C4-  A5 BF        LDA  $BF
15C6-  85 AA        STA  $AA     ?? used?                  n.b. C3 points at Re auto
15C8-  A5 C0        LDA  $C0                                  accum
15CA-  85 AB        STA  $AB
15CC-  A9 1A        LDA  #$1A
15CE-  85 B0        STA  $B0     "lag" pointer to Re auto accum.(=#1's in I4)
15D0-  A5 B7        LDA  $B7       count #1's for seq. pointed at by $B7
15D2-  20 40 16     JSR  $1640
15D5-  A4 B0        LDY  $B0
15D7-  18           CLC            and add to #1's store
15D8-  71 C3        ADC  ($C3),Y
15DA-  91 C3        STA  ($C3),Y
15DC-  90 08        BCC  $15E6
15DE-  88           DEY
15DF-  A9 00        LDA  #$00
15E1-  71 C3        ADC  ($C3),Y
15E3-  91 C3        STA  ($C3),Y
15E5-  C8           INY
15E6-  A5 B7        LDA  $B7
15E8-  18           CLC            move to next antenna, I comp.
15E9-  69 14        ADC  #$14
15EB-  85 B7        STA  $B7
15ED-  C8           INY
15EE-  C8           INY
15EF-  84 B0        STY  $B0
15F1-  C0 22        CPY  #$22      last I comp?
15F3-  D0 DB        BNE  $15D0
15F5-  A9 20        LDA  #$20      now Im accum (for #1's in Q sequences)
15F7-  85 B0        STA  $B0
15F9-  A9 8B        LDA  #$8B      point to Q1
15FB-  85 B7        STA  $B7
15FD-  EA           NOP
15FE-  EA           NOP
15FF-  EA           NOP
1600-  A5 C4        LDA  $C4
1602-  85 AF        STA  $AF
1604-  A5 C3        LDA  $C3
1606-  38           SEC            construct pointer to Im accum.
1607-  E9 20        SBC  #$20
1609-  85 AE        STA  $AE
160B-  B0 02        BCS  $160F
160D-  C6 AF        DEC  $AF
160F-  A5 B7        LDA  $B7       get #1's
1611-  20 40 16     JSR  $1640
1614-  A4 B0        LDY  $B0
1616-  18           CLC
1617-  71 AE        ADC  ($AE),Y
1619-  91 AE        STA  ($AE),Y
161B-  90 08        BCC  $1625
161D-  88           DEY
161E-  A9 00        LDA  #$00
1620-  71 AE        ADC  ($AE),Y'
1622-  91 AE        STA  ($AE),Y
1624-  C8           INY
1625-  A5 B7        LDA  $B7
1627-  38             SEC          next Q sequence
1628-  E9 14        SBC  #$14
162A-  85 B7        STA  $B7
162C-  88           DEY            next antenna/Rx
162D-  88           DEY
162E-  84 B0        STY  $B0
1630-  C0 18        CPY  #$18      last Rx?
1632-  D0 DB        BNE  $160F     no
1634-  60           RTS            yes,so quit
```

```
1640-  85 CD        STA  $CD       addr. L          COUNT #1's in 8 BYTE SEQS
1642-  A9 00        LDA  #$00
1644-  85 CE        STA  $CE       addr. H
1646-  85 B2        STA  $B2       accum.
1648-  18           CLC
1649-  A0 08        LDY  #$08      8 bytes
164B-  B1 CD        LDA  ($CD),Y
164D-  AA           TAX
164E-  BD 00 63     LDA  $6300,X   look up #1-bits
1651-  65 B2        ADC  $B2
1653-  85 B2        STA  $B2
1655-  88           DEY
1656-  D0 F3        BNE  $164B
1658-  60           RTS
```

```
1670-  A9 04     LDA  #$04
1672-  85 B4     STA  $B4      antenna counter          AUTO CORREL (46ms)
1674-  A9 89     LDA  #$89
1676-  85 AA     STA  $AA      point to Q1 initially
1678-  A2 08     LDX  #$08     move 8 bytes to zero-page 'rolling' stores
167A-  B5 95     LDA  $95,X    I1
167C-  95 9F     STA  $9F,X       (moves ANT#1,but later other ANT# seqs moved
167E-  B5 8B     LDA  $8B,X    Q1              into this store)
1680-  95 DF     STA  $DF,X
1682-  CA        DEX
1683-  D0 F5     BNE  $167A
1685-  A9 00     LDA  #$00        clear 2 bytes below rolling binary
1687-  85 9E     STA  $9E
1689-  85 9F     STA  $9F
168B-  85 DE     STA  $DE
168D-  85 DF     STA  $DF
168F-  A0 18     LDY  #$18        lag pointer(note top 4 lags contain #1's data,not acor
1691-  A5 C3     LDA  $C3  }
1693-  85 AC     STA  $AC  }      point to Re
1695-  A5 C4     LDA  $C4  }
1697-  85 AD     STA  $AD  }
1699-  18        CLC
169A-  26 A7     ROL  $A7
169C-  26 A6     ROL  $A6
169E-  26 A5     ROL  $A5
16A0-  26 A4     ROL  $A4
16A2-  26 A3     ROL  $A3
16A4-  26 A2     ROL  $A2
16A6-  26 A1     ROL  $A1
16A8-  26 A0     ROL  $A0
16AA-  26 9F     ROL  $9F
16AC-  26 9E     ROL  $9E
16AE-  18        CLC
16AF-  26 E7     ROL  $E7
16B1-  26 E6     ROL  $E6
16B3-  26 E5     ROL  $E5
16B5-  26 E4     ROL  $E4
16B7-  26 E3     ROL  $E3
16B9-  26 E2     ROL  $E2
16BB-  26 E1     ROL  $E1
16BD-  26 E0     ROL  $E0
16BF-  26 DF     ROL  $DF
16C1-  26 DE     ROL  $DE
16C3-  18        CLC          start IivsIi
16C4-  A9 00     LDA  #$00
16C6-  85 B2     STA  $B2
16C8-  A5 A7     LDA  $A7
16CA-  25 9D     AND  $9D
16CC-  AA        TAX
16CD-  BD 00 63  LDA  $6300,X
16D0-  65 B2     ADC  $B2
16D2-  85 B2     STA  $B2
16D4-  A5 A6     LDA  $A6
16D6-  25 9C     AND  $9C
16D8-  AA        TAX
16D9-  BD 00 63  LDA  $6300,X
16DC-  65 B2     ADC  $B2
16DE-  85 B2     STA  $B2
16E0-  A5 A5     LDA  $A5
16E2-  25 9B     AND  $9B
16E4-  AA        TAX
16E5-  BD 00 63  LDA  $6300,X
16E8-  65 B2     ADC  $B2
16EA-  85 B2     STA  $B2
16EC-  A5 A4     LDA  $A4
16EE-  25 9A     AND  $9A
16F0-  AA        TAX
16F1-  BD 00 63  LDA  $6300,X
16F4-  65 B2     ADC  $B2
16F6-  85 B2     STA  $B2
16F8-  A5 A3     LDA  $A3
16FA-  25 99     AND  $99
16FC-  AA        TAX
16FD-  BD 00 63  LDA  $6300,X
1700-  65 B2     ADC  $B2
1702-  85 B2     STA  $B2
1704-  A5 A2     LDA  $A2
1706-  25 98     AND  $98
1708-  AA        TAX
1709-  BD 00 63  LDA  $6300,X
170C-  65 B2     ADC  $B2
170E-  85 B2     STA  $B2
```

```
1710-   A5 A1        LDA   $A1
1712-   25 97        AND   $97
1714-   AA           TAX
1715-   BD 00 63     LDA   $6300,X
1718-   65 B2        ADC   $B2
171A-   85 B2        STA   $B2
171C-   A5 A0        LDA   $A0
171E-   25 96        AND   $96
1720-   AA           TAX
1721-   BD 00 63     LDA   $6300,X
1724-   65 B2        ADC   $B2
1726-   85 B2        STA   $B2
1728-   A5 E7        LDA   $E7        start QivsQi
172A-   25 93        AND   $93
172C-   AA           TAX
172D-   BD 00 63     LDA   $6300,X
1730-   65 B2        ADC   $B2
1732-   85 B2        STA   $B2
1734-   A5 E6        LDA   $E6
1736-   25 92        AND   $92
1738-   AA           TAX
1739-   BD 00 63     LDA   $6300,X
173C-   65 B2        ADC   $B2
173E-   85 B2        STA   $B2
1740-   A5 E5        LDA   $E5
1742-   25 91        AND   $91
1744-   AA           TAX
1745-   BD 00 63     LDA   $6300,X
1748-   65 B2        ADC   $B2
174A-   85 B2        STA   $B2
174C-   A5 E4        LDA   $E4
174E-   25 90        AND   $90
1750-   AA           TAX
1751-   BD 00 63     LDA   $6300,X
1754-   65 B2        ADC   $B2
1756-   85 B2        STA   $B2
1758-   A5 E3        LDA   $E3
175A-   25 8F        AND   $8F
175C-   AA           TAX
175D-   BD 00 63     LDA   $6300,X
1760-   65 B2        ADC   $B2
1762-   85 B2        STA   $B2
1764-   A5 E2        LDA   $E2
1766-   25 8E        AND   $8E
1768-   AA           TAX
1769-   BD 00 63     LDA   $6300,X
176C-   65 B2        ADC   $B2
176E-   85 B2        STA   $B2
1770-   A5 E1        LDA   $E1
1772-   25 8D        AND   $8D
1774-   AA           TAX
1775-   BD 00 63     LDA   $6300,X
1778-   65 B2        ADC   $B2
177A-   85 B2        STA   $B2
177C-   A5 E0        LDA   $E0
177E-   25 8C        AND   $8C
1780-   AA           TAX
1781-   BD 00 63     LDA   $6300,X
1784-   65 B2        ADC   $B2
1786-   85 B2        STA   $B2
1788-   EA           NOP
1789-   EA           NOP
178A-   EA           NOP
178B-   EA           NOP
178C-   A5 9F        LDA   $9F        add in last two bytes (this routine was lifted
178E-   25 95        AND   $95                from 4V1 xcor)
1790-   AA           TAX
1791-   BD 00 63     LDA   $6300,X
1794-   65 B2        ADC   $B2
1796-   85 B2        STA   $B2
1798-   A5 9E        LDA   $9E
179A-   25 94        AND   $94
179C-   AA           TAX
179D-   BD 00 63     LDA   $6300,X
17A0-   65 B2        ADC   $B2
17A2-   85 B2        STA   $B2
17A4-   A5 DF        LDA   $DF
17A6-   25 8B        AND   $8B
17A8-   AA           TAX
17A9-   BD 00 63     LDA   $6300,X
17AC-   65 B2        ADC   $B2
17AE-   85 B2        STA   $B2
```

```
17B0-   A5 DE       LDA   $DE
17B2-   25 8A       AND   $8A
17B4-   AA          TAX
17B5-   BD 00 63    LDA   $6300,X
17B8-   65 B2       ADC   $B2
17BA-   85 B2       STA   $B2
17BC-   71 AC       ADC   ($AC),Y    add to Re accum (all antennas in same accum)
17BE-   91 AC       STA   ($AC),Y
17C0-   90 08       BCC   $17CA
17C2-   88          DEY
17C3-   A9 00       LDA   #$00
17C5-   71 AC       ADC   ($AC),Y
17C7-   91 AC       STA   ($AC),Y
17C9-   C8          INY
17CA-   A5 AC       LDA   $AC        }
17CC-   38          SEC              }   move acor pointer to Im
17CD-   E9 20       SBC   #$20       }
17CF-   85 AC       STA   $AC        }
17D1-   B0 02       BCS   $17D5      }
17D3-   C6 AD       DEC   $AD        }
17D5-   18          CLC
17D6-   A9 00       LDA   #$00
17D8-   85 B2       STA   $B2
17DA-   85 B7       STA   $B7
17DC-   A5 E7       LDA   $E7        start QivsIi
17DE-   25 9D       AND   $9D
17E0-   AA          TAX
17E1-   BD 00 63    LDA   $6300,X
17E4-   65 B2       ADC   $B2
17E6-   85 B2       STA   $B2
17E8-   A5 E6       LDA   $E6
17EA-   25 9C       AND   $9C
17EC-   AA          TAX
17ED-   BD 00 63    LDA   $6300,X
17F0-   65 B2       ADC   $B2
17F2-   85 B2       STA   $B2
17F4-   A5 E5       LDA   $E5
17F6-   25 9B       AND   $9B
17F8-   AA          TAX
17F9-   BD 00 63    LDA   $6300,X
17FC-   65 B2       ADC   $B2
17FE-   85 B2       STA   $B2
1800-   A5 E4       LDA   $E4
1802-   25 9A       AND   $9A
1804-   AA          TAX
1805-   BD 00 63    LDA   $6300,X
1808-   65 B2       ADC   $B2
180A-   85 B2       STA   $B2
180C-   A5 E3       LDA   $E3
180E-   25 99       AND   $99
1810-   AA          TAX
1811-   BD 00 63    LDA   $6300,X
1814-   65 B2       ADC   $B2
1816-   85 B2       STA   $B2
1818-   A5 E2       LDA   $E2
181A-   25 98       AND   $98
181C-   AA          TAX
181D-   BD 00 63    LDA   $6300,X
1820-   65 B2       ADC   $B2
1822-   85 B2       STA   $B2
1824-   A5 E1       LDA   $E1
1826-   25 97       AND   $97
1828-   AA          TAX
1829-   BD 00 63    LDA   $6300,X
182C-   65 B2       ADC   $B2
182E-   85 B2       STA   $B2
1830-   A5 E0       LDA   $E0
1832-   25 96       AND   $96
1834-   AA          TAX
1835-   BD 00 63    LDA   $6300,X
1838-   65 B2       ADC   $B2
183A-   85 B2       STA   $B2
```

```
183C-   A5 A7       LDA   $A7        start IivsQi
183E-   25 93       AND   $93
1840-   AA          TAX
1841-   BD 00 63    LDA   $6300,X
1844-   65 B7       ADC   $B7
1846-   85 B7       STA   $B7
1848-   A5 A6       LDA   $A6
184A-   25 92       AND   $92
184C-   AA          TAX
184D-   BD 00 63    LDA   $6300,X
1850-   65 B7       ADC   $B7
1852-   85 B7       STA   $B7
1854-   A5 A5       LDA   $A5
1856-   25 91       AND   $91
1858-   AA          TAX
1859-   BD 00 63    LDA   $6300,X
185C-   65 B7       ADC   $B7
185E-   85 B7       STA   $B7
1860-   A5 A4       LDA   $A4
1862-   25 90       AND   $90
1864-   AA          TAX
1865-   BD 00 63    LDA   $6300,X
1868-   65 B7       ADC   $B7
186A-   85 B7       STA   $B7
186C-   A5 A3       LDA   $A3
186E-   25 8F       AND   $8F
1870-   AA          TAX
1871-   BD 00 63    LDA   $6300,X
1874-   65 B7       ADC   $B7
1876-   85 B7       STA   $B7
1878-   A5 A2       LDA   $A2
187A-   25 8E       AND   $8E
187C-   AA          TAX
187D-   BD 00 63    LDA   $6300,X
1880-   65 B7       ADC   $B7
1882-   85 B7       STA   $B7
1884-   A5 A1       LDA   $A1
1886-   25 8D       AND   $8D
1888-   AA          TAX
1889-   BD 00 63    LDA   $6300,X
188C-   65 B7       ADC   $B7
188E-   85 B7       STA   $B7
1890-   A5 A0       LDA   $A0
1892-   25 8C       AND   $8C
1894-   AA          TAX
1895-   BD 00 63    LDA   $6300,X
1898-   65 B7       ADC   $B7
189A-   85 B7       STA   $B7
189C-   EA          NOP
189D-   EA          NOP
189E-   EA          NOP
189F-   EA          NOP
18A0-   A5 DF       LDA   $DF        finish up last 2 bytes
18A2-   25 95       AND   $95
18A4-   AA          TAX
18A5-   BD 00 63    LDA   $6300,X
18A8-   65 B2       ADC   $B2
18AA-   85 B2       STA   $B2
18AC-   A5 DE       LDA   $DE
18AE-   25 94       AND   $94
18B0-   AA          TAX
18B1-   BD 00 63    LDA   $6300,X
18B4-   65 B2       ADC   $B2
18B6-   85 B2       STA   $B2
18B8-   A5 9F       LDA   $9F
18BA-   25 8B       AND   $8B
18BC-   AA          TAX
18BD-   BD 00 63    LDA   $6300,X
18C0-   65 B7       ADC   $B7
18C2-   85 B7       STA   $B7
18C4-   A5 9E       LDA   $9E
18C6-   25 8A       AND   $8A
18C8-   AA          TAX
18C9-   BD 00 63    LDA   $6300,X
18CC-   65 B7       ADC   $B7
18CE-   85 B7       STA   $B7  ------------------
```

(27)

```
18D0-  38          SEC                    add QivsIi-IivsQi to Im accum.
18D1-  B1 AC       LDA   ($AC),Y
18D3-  E5 B7       SBC   $B7
18D5-  B0 10       BCS   $18E7
18D7-  65 B2       ADC   $B2
18D9-  91 AC       STA   ($AC),Y
18DB-  B0 19       BCS   $18F6
18DD-  88          DEY
18DE-  B1 AC       LDA   ($AC),Y
18E0-  E9 00       SBC   #$00
18E2-  91 AC       STA   ($AC),Y
18E4-  C8          INY
18E5-  D0 0F       BNE   $18F6
18E7-  18          CLC
18E8-  65 B2       ADC   $B2
18EA-  91 AC       STA   ($AC),Y
18EC-  90 08       BCC   $18F6
18EE-  88          DEY
18EF-  A9 00       LDA   #$00
18F1-  71 AC       ADC   ($AC),Y
18F3-  91 AC       STA   ($AC),Y
18F5-  C8          INY     ----------end of add
18F6-  88          DEY
18F7-  88          DEY
18F8-  F0 03       BEQ   $18FD
18FA-  4C 91 16    JMP   $1691
18FD-  C6 B4       DEC   $B4          next antenna
18FF-  F0 18       BEQ   $1919
1901-  A5 AA       LDA   $AA
1903-  38          SEC                    move next antenna to ant#1 store
1904-  E9 14       SBC   #$14                   (I and Q data)
1906-  85 AA       STA   $AA
1908-  A9 00       LDA   #$00
190A-  85 AB       STA   $AB
190C-  A0 14       LDY   #$14
190E-  B1 AA       LDA   ($AA),Y
1910-  99 89 00    STA   $0089,Y
1913-  88          DEY
1914-  D0 F8       BNE   $190E
1916-  4C 78 16    JMP   $1678
1919-  60          RTS
```

TRANSFER BINARY TO ZERO PAGE

```
1920-  A5 BF       LDA   $BF
1922-  85 AA       STA   $AA          bin I1
1924-  A5 C0       LDA   $C0
1926-  85 AB       STA   $AB
1928-  A2 50       LDX   #$50         8x10 bytes
192A-  A9 0A       LDA   #$0A         10 bytes/ant/component
192C-  85 B2       STA   $B2
192E-  A4 B9       LDY   $B9
1930-  B1 AA       LDA   ($AA),Y
1932-  95 4D       STA   $4D,X
1934-  CA          DEX
1935-  F0 12       BEQ   $1949
1937-  88          DEY
1938-  10 02       BPL   $193C
193A-  A0 1F       LDY   #$1F         wrap around pointer in original data store
193C-  C6 B2       DEC   $B2
193E-  D0 F0       BNE   $1930
1940-  A5 AB       LDA   $AB
1942-  38          SEC
1943-  E9 04       SBC   #$04         next component (I or Q)
1945-  85 AB       STA   $AB
1947-  D0 E1       BNE   $192A        forced branch
1949-  60          RTS
```

NMI PROCESSING

```
1950-  AD 4C 04    LDA   $044C
1953-  C9 A0       CMP   #$A0    scrn=" "?
1955-  D0 07       BNE   $195E   no                    (NMI operated by 5 min
1957-  A9 B1       LDA   #$B1    yes,so                  time-out counter in
1959-  8D 4C 04    STA   $044C   write "1" on screen     hardware.)
195C-  D0 03       BNE   $1961   forced branch
195E-  EE 4C 04    INC   $044C   just incr. screen
1961-  20 ED 1C    JSR   $1CED   and display cpu#2 status (i.e. where was it stuck?)
1964-  20 F8 19    JSR   $19F8   turn off Tx
1967-  4C 40 09    JMP   $0940    and return to START (NMI entry)
```

```
1970-  A9 10       LDA  #$10    set CPU#2 status                TAPE DUMP
1972-  85 B6       STA  $B6
1974-  20 07 1F    JSR  $1F07   check for end of tape(pop addr. and rtn from here
1977-  20 12 0E    JSR  $0E12   turn on Tx rate          if so!)
                                (to mark dump)

197A-  EA          NOP
197B-  A9 00       LDA  #$00
197D-  85 19       STA  $19        set starting addr. for dump ($4000)
197F-  A9 40       LDA  #$40
1981-  85 1A       STA  $1A
1983-  A9 C0       LDA  #$C0       set length(#byte=$6C0= 2 recs of $360ea)
1985-  85 1B       STA  $1B
1987-  A9 06       LDA  #$06
1989-  85 1C       STA  $1C   }
198B-  A9 00       LDA  #$00  }       select drive #0
198D-  85 1D       STA  $1D   }
198F-  A9 82       LDA  #$82  }------ set drive characteristics (DEC TS03)
1991-  85 1E       STA  $1E   }
1993-  20 0C C7    JSR  $C70C <------ write tape block to tape (EPROM routine)
1996-  A5 C5       LDA  $C5        file gap req'd?
1998-  F0 0C       BEQ  $19A6      no
199A-  AD 0A 03    LDA  $030A      yes, is H10 = 0?( i.e. end of day for analysed
199D-  D0 07       BNE  $19A6      no,so no FG yet,       data)
199F-  A9 00       LDA  #$00       yes,so clear FG flag
19A1-  85 C5       STA  $C5
19A3-  20 50 1F    JSR  $1F50      and write file gap
19A6-  20 F8 19    JSR  $19F8      and turn off Tx
19A9-  EA          NOP
19AA-  A5 1F       LDA  $1F        check status of tape write.
19AC-  F0 12       BEQ  $19C0      zero if OK,so skip
19AE-  20 18 1C    JSR  $1C18 <------ not ok,so write "tape drive not ready" on scrn.
19B1-  A5 B6       LDA  $B6        is this the first try?
19B3-  D0 01       BNE  $19B6      yes
19B5-  60          RTS             no,tried twice, so quit and try again for next
19B6-  20 D1 19    JSR  $19D1      go through drive reset procedure      record.
19B9-  A9 00       LDA  #$00  }------ clear CPU#2 status (also marks end of 1st write try
19BB-  85 B6       STA  $B6   }
19BD-  4C 93 19    JMP  $1993      and go back for 2nd try.
19C0-  20 E0 1B    JSR  $1BE0 <------ Write was ok,so clear any "not ready" msg,
19C3-  A5 B6       LDA  $B6   <------ check to see if this was the 2nd try
19C5-  D0 03       BNE  $19CA      no, it was the first,
19C7-  20 20 1F    JSR  $1F20 <------ yes,so drive was successfully init.on 2nd try,
                                      so increment screen cntr

19CA-  A9 00       LDA  #$00  }------ clear status
19CC-  85 B6       STA  $B6   }
19CE-  60          RTS             end of dump routine
19CF-  FF          ???
19D0-  FF          ???
```

```
19D1-  A9 1E       LDA  #$1E   "load" low          RESET TAPE DRIVE
19D3-  20 EC 19    JSR  $19EC  send&delay
19D6-  A9 1F       LDA  #$1F   "load"high     note:PB0,PB1,PB2 must normally be
19D8-  20 EC 19    JSR  $19EC  send&delay          high (active low)
19DB-  A9 1B       LDA  #$1B   "test" low
19DD-  20 EC 19    JSR  $19EC  send&delay     PB0=load,PB1=on-line,PB2=test
19E0-  A9 1F       LDA  #$1F   "test" high          (drive controls)
19E2-  20 EC 19    JSR  $19EC
19E5-  A9 1D       LDA  #$1D   "on-line" low
19E7-  20 EC 19    JSR  $19EC
19EA-  A9 1F       LDA  #$1F   "on-line"high
19EC-  8D 02 C5    STA  $C502  send to port
19EF-  A0 04       LDY  #$04
19F1-  20 A8 FC    JSR  $FCA8   approx 0.5 s delay
19F4-  88          DEY
19F5-  D0 FA       BNE  $19F1
19F7-  60          RTS
```

```
19F8-  A9 38       LDA  #$38   select DDR          INITIALIZE TAPE RESET/TX
19FA-  8D 03 C5    STA  $C503                       CONTROL PORT
19FD-  A9 1F       LDA  #$1F   lowest 5 bits o/p
19FF-  8D 02 C5    STA  $C502
1A02-  A9 3C       LDA  #$3C   select port(CB2 high)   (bit 3 = tx cntl )
1A04-  8D 03 C5    STA  $C503                          ( high=off )
1A07-  A9 1F       LDA  #$1F   all o/p high (as after reset)
1A09-  8D 02 C5    STA  $C502      (so Tx=off)
1A0C-  60          RTS
```

COMM. LINK TO C128

```
1A10-  A9 28     LDA  #$28   } select DDR
1A12-  8D B1 C0  STA  $C0B1  }
1A15-  A9 00     LDA  #$00   } all bits input
1A17-  8D B0 C0  STA  $C0B0  }
1A1A-  A9 2C     LDA  #$2C   } set cntl reg. to strobe CA2 on read
1A1C-  8D B1 C0  STA  $C0B1  }
1A1F-  AD B1 C0  LDA  $C0B1    C128 ready?
1A22-  30 07     BMI  $1A2B    yes
1A24-  A9 01     LDA  #$01   } no??, so reset 1st rec flag(so discard next data back)
1A26-  85 BA     STA  $BA    }
1A28-  4C 00 1C  JMP  $1C00  } and go to display msg "C128 not ready",and return
1A2B-  20 E0 1B  JSR  $1BE0    ok,it is ready,so clear any previous msg
1A2E-  A9 08     LDA  #$08   } and set CPU#2 status "in C128 comm."
1A30-  85 B6     STA  $B6    }
1A32-  A9 00     LDA  #$00   } read analysed data from C128
1A34-  85 A0     STA  $A0    }    and store in BCAØ-BFFF (from bottom up!)
1A36-  A9 BC     LDA  #$BC   }
1A38-  85 A1     STA  $A1    }
1A3A-  A0 A0     LDY  #$A0   }
1A3C-  AD B1 C0  LDA  $C0B1  } wait for flag
1A3F-  10 FB     BPL  $1A3C  }
1A41-  AD B0 C0  LDA  $C0B0  } read data port (auto stobe out)
1A44-  91 A0     STA  ($A0),Y}
1A46-  C8        INY         }
1A47-  D0 F3     BNE  $1A3C  }
1A49-  E6 A1     INC  $A1    }
1A4B-  A5 A1     LDA  $A1    }
1A4D-  C9 C0     CMP  #$C0   }
1A4F-  D0 EB     BNE  $1A3C  }
1A51-  EA        NOP
1A52-  AD BE BC  LDA  $BCBE  } look at comm link chaek bytes
1A55-  F0 03     BEQ  $1A5A  } this one's ok,try the other
1A57-  4C F8 1B  JMP  $1BF8  } bad!- so write "COMM#3 ERR" and stop program
1A5A-  AD BF BC  LDA  $BCBF  } should be $FF ,
1A5D-  C9 FF     CMP  #$FF   } is it?
1A5F-  D0 F6     BNE  $1A57  } no!, so pull the brake cord!
1A61-  EA        NOP         } ok,
1A62-  EA        NOP
1A63-  A9 38     LDA  #$38   } select DDR
1A65-  8D B1 C0  STA  $C0B1  }
1A68-  A9 FF     LDA  #$FF   } all bits output
1A6A-  8D B0 C0  STA  $C0B0  }
1A6D-  A9 3C     LDA  #$3C   } select port, manual cntl of CA2
1A6F-  8D B1 C0  STA  $C0B1  }
```

OR JUMP TO NMI VECTOR TO RESTART

```
1A72-  AD FF 5F  LDA  $5FFF    A/D status
1A75-  8D 8C BC  STA  $BC8C    save it in tape d/t store
1A78-  EA        NOP
1A79-  A9 40     LDA  #$40     send data from $BC9F down to $844Ø
1A7B-  85 A0     STA  $A0        to C128   ($386Ø bytes)
1A7D-  A9 BC     LDA  #$BC
1A7F-  85 A1     STA  $A1
1A81-  A0 5F     LDY  #$5F
1A83-  B1 A0     LDA  ($A0),Y
1A85-  8D B0 C0  STA  $C0B0  store data in port
1A88-  A9 34     LDA  #$34   }
1A8A-  8D B1 C0  STA  $C0B1  } toggle CA2
1A8D-  A9 3C     LDA  #$3C   }
1A8F-  8D B1 C0  STA  $C0B1  }
1A92-  AD B1 C0  LDA  $C0B1    wait for response
1A95-  10 FB     BPL  $1A92
1A97-  AD B0 C0  LDA  $C0B0    clear response flag
1A9A-  88        DEY
1A9B-  C0 FF     CPY  #$FF
1A9D-  D0 E4     BNE  $1A83
1A9F-  C6 A1     DEC  $A1
1AA1-  A5 A1     LDA  $A1
1AA3-  C9 83     CMP  #$83
1AA5-  D0 DC     BNE  $1A83
1AA7-  A9 00     LDA  #$00   } clear CPU#2 status msg
1AA9-  85 B6     STA  $B6    }
1AAB-  60        RTS
```

```
1AB0-  A5 C6      LDA  $C6    Doff
1AB2-  18         CLC
1AB3-  69 01      ADC  #$01   = last data pos'n
1AB5-  C9 20      CMP  #$20
1AB7-  D0 02      BNE  $1ABB
1AB9-  A9 00      LDA  #$00
1ABB-  4A         LSR
1ABC-  4A         LSR
1ABD-  4A         LSR
1ABE-  F0 08      BEQ  $1AC8
1AC0-  0A         ASL
1AC1-  0A         ASL
1AC2-  0A         ASL
1AC3-  18         CLC
1AC4-  69 05      ADC  #$05
1AC6-  90 02      BCC  $1ACA
1AC8-  A9 1D      LDA  #$1D
1ACA-  85 CA      STA  $CA
1ACC-  A9 20      LDA  #$20
1ACE-  85 C7      STA  $C7
1AD0-  A9 84      LDA  #$84
1AD2-  85 C8      STA  $C8
1AD4-  A9 08      LDA  #$08
1AD6-  85 CB      STA  $CB
1AD8-  A2 20      LDX  #$20
1ADA-  BD FF 63   LDA  $63FF,X
1ADD-  85 CC      STA  $CC
1ADF-  A4 C6      LDY  $C6
1AE1-  A9 00      LDA  #$00
1AE3-  91 C7      STA  ($C7),Y
1AE5-  88         DEY
1AE6-  10 02      BPL  $1AEA
1AE8-  A0 1F      LDY  #$1F
1AEA-  C4 CC      CPY  $CC
1AEC-  D0 03      BNE  $1AF1
1AEE-  4C 10 1C   JMP  $1C10
1AF1-  C4 CA      CPY  $CA
1AF3-  D0 EE      BNE  $1AE3
1AF5-  A5 C7      LDA  $C7
1AF7-  38         SEC
1AF8-  E9 20      SBC  #$20
1AFA-  85 C7      STA  $C7
1AFC-  B0 02      BCS  $1B00
1AFE-  C6 C8      DEC  $C8
1B00-  CA         DEX
1B01-  D0 D7      BNE  $1ADA
1B03-  C6 CB      DEC  $CB
1B05-  D0 D1      BNE  $1AD8
1B07-  A5 CA      LDA  $CA
1B09-  85 C6      STA  $C6
1B0B-  60         RTS
1B0C-  FF         ???
1B0D-  FF         ???
1B0E-  FF         ???
1B0F-  FF         ???
1B10-  EA         NOP
1B11-  EA         NOP
1B12-  EA         NOP
1B13-  EA         NOP
1B14-  AD 04 03   LDA  $0304  S1
1B17-  0D 05 03   ORA  $0305  S10
1B1A-  F0 0B      BEQ  $1B27  =ØØ,so out
1B1C-  AD 07 03   LDA  $0307  ok,M1
1B1F-  F0 04      BEQ  $1B25  =0,so start
1B21-  C9 05      CMP  #$05   not Ø, is it 5?
1B23-  D0 02      BNE  $1B27  not = 5, so don't start
1B25-  E6 BB      INC  $BB    ok,=0 or 5,so request record start
1B27-  60         RTS
1B28-  FF         ???
1B29-  FF         ???
1B2A-  FF         ???
1B2B-  FF         ???
1B2C-  FF         ???
1B2D-  FF         ???
1B2E-  FF         ???
1B2F-  FF         ???
1B30-  20 40 1C   JSR  $1C40  I/Q,Q/I
1B33-  20 40 1B   JSR  $1B40  LTMs
1B36-  EA         NOP
1B37-  EA         NOP
1B38-  EA         NOP
1B39-  60         RTS
```

SEC
SBC#3
BCS 1ACA (forced)

FINAL "DATA" ENTRY

(write zeroes and move pointer Doff along to finish record)

called from IRQ process. at end of record.

final pos'n of data+1 pointer

there are only a few pos'ns they can be at, logically-(4 of them?) so program uses trick to get quickest result?

oops! just about to write over data still used by analysis-so :"WRAP AT END" and stop

next ht.,

CHECK FOR RECORD START TIME

(but don't clear BB, because will wipe out possible man. start request)

don't start if SEC=ØØ (because clock time unstable here!)

DISPLAY FINAL PARAMETERS FROM #1

```
1B40-  A0 08     LDY  #$08 Rx/component pointer   DISPLAY WORST OFFSETS (LTMs)
1B42-  A2 08     LDX  #$08  gain#                      (diff.from $80 in BCD )
1B44-  A9 D7     LDA  #$D7 }                             ( -ve if below )
1B46-  85 A0     STA  $A0  }
1B48-  A9 60     LDA  #$60 } base point to LTM store received from
1B4A-  85 A1     STA  $A1  }    CPU#1 , at gain#7 to start
1B4C-  A9 00     LDA  #$00
1B4E-  85 A3     STA  $A3     to hold signed values of offset
1B50-  85 A2     STA  $A2     save running max offset here
1B52-  B1 A0     LDA  ($A0),Y  LTM
1B54-  38        SEC
1B55-  E9 80     SBC  #$80     get offset rel.to $80
1B57-  48        PHA            save signed value on stack
1B58-  B0 05     BCS  $1B5F    +ve value
1B5A-  49 FF     EOR  #$FF     -ve value,so get ABS
1B5C-  18        CLC  <------- (don't need)
1B5D-  69 01     ADC  #$01
1B5F-  C5 A2     CMP  $A2      compare with last max offset(same Rx)
1B61-  90 08     BCC  $1B6B    this one is less,so don't keep
1B63-  85 A2     STA  $A2      this one is greater,so keep it
1B65-  68        PLA           and store the corresponding signed value
1B66-  85 A3     STA  $A3
1B68-  18        CLC
1B69-  90 01     BCC  $1B6C    forced branch
1B6B-  68        PLA           dummy operation to clear stack
1B6C-  A5 A0     LDA  $A0      move base addr. to point at next gain
1B6E-  38        SEC
1B6F-  E9 08     SBC  #$08
1B71-  85 A0     STA  $A0
1B73-  B0 02     BCS  $1B77
1B75-  C6 A1     DEC  $A1
1B77-  CA        DEX           next gain
1B78-  D0 D8     BNE  $1B52
1B7A-  A5 A3     LDA  $A3      take max signed value of offset
1B7C-  99 D7 60  STA  $60D7,Y  and save in gain#7 store
1B7F-  88        DEY           next Rx/component
1B80-  D0 C0     BNE  $1B42
1B82-  EA        NOP
```

```
1B83-  EA          NOP
1B84-  EA          NOP                    now display on screen
1B85-  EA          NOP
1B86-  A9 B6       LDA   #$B6
1B88-  85 A0       STA   $A0              screen addr. for Rx #1
1B8A-  A9 05       LDA   #$05
1B8C-  85 A1       STA   $A1
1B8E-  A0 08       LDY   #$08             Rx/Component pointer
1B90-  A9 04       LDA   #$04             Rx counter
1B92-  85 A3       STA   $A3
1B94-  B9 D7 60    LDA   $60D7,Y          get  Δ I max  } for this Rx
1B97-  48          PHA
1B98-  88          DEY
1B99-  B9 D7 60    LDA   $60D7,Y          get  ΔQ max   }
1B9C-  48          PHA
1B9D-  88          DEY
1B9E-  84 A2       STY   $A2
1BA0-  A0 05       LDY   #$05             point to Q-ascii-L on screen
1BA2-  A9 02       LDA   #$02
1BA4-  85 A4       STA   $A4              run through Q then I
1BA6-  EA          NOP
1BA7-  EA          NOP
1BA8-  68          PLA                    retrieve Q first
1BA9-  10 0B       BPL   $1BB6            +ve, so leave it
1BAB-  49 FF       EOR   #$FF             -ve,so get ABS value
1BAD-  18          CLC
1BAE-  69 01       ADC   #$01
1BB0-  20 B0 1C    JSR   $1CB0            convert to ascii BCD(L in A,H in X)
1BB3-  38          SEC                    set flag to show -ve
1BB4-  B0 04       BCS   $1BBA            forced branch
1BB6-  20 B0 1C    JSR   $1CB0            get ascii BCD for +ve value
1BB9-  18          CLC                    clear flag,to show +ve
1BBA-  91 A0       STA   ($A0),Y          store L ascii on screen
1BBC-  88          DEY
1BBD-  8A          TXA
1BBE-  91 A0       STA   ($A0),Y          store H ascii on screen
1BC0-  88          DEY
1BC1-  A9 AB       LDA   #$AB             "+"
1BC3-  90 02       BCC   $1BC7  <-------  check sign of offset
1BC5-  A9 AD       LDA   #$AD             "-"
1BC7-  91 A0       STA   ($A0),Y          store appropriate sign on screen
1BC9-  88          DEY
1BCA-  C6 A4       DEC   $A4              next component (I)
1BCC-  D0 DA       BNE   $1BA8                                  note: offsets ≥ 100
1BCE-  A4 A2       LDY   $A2              restore Y contents     will show strange
1BD0-  A5 A0       LDA   $A0                                     ascii characters!
1BD2-  18          CLC                    modify screen addr. to next line (Rx)
1BD3-  69 80       ADC   #$80
1BD5-  85 A0       STA   $A0
1BD7-  90 02       BCC   $1BDB
1BD9-  E6 A1       INC   $A1
1BDB-  C6 A3       DEC   $A3  <-------  next Rx
1BDD-  D0 B5       BNE   $1B94
1BDF-  60          RTS

1BE0-  A0 10       LDY   #$10                    CLEAR "PROBLEM" MSG
1BE2-  A9 A0       LDA   #$A0
1BE4-  99 5E 04    STA   $045E,Y          used if C128 becomes ready or
1BE7-  88          DEY                    tape drive becomes ready
1BE8-  D0 FA       BNE   $1BE4
1BEA-  60          RTS
```

PRINT TROUBLE MESSAGE

then stop or continue depending....

```
1BF0-  A9 1E     LDA  ##$1E   }
1BF2-  48        PHA          } "COMM#1 ERR"
1BF3-  A9 73     LDA  #$73    }  and stop
1BF5-  4C 20 1C  JMP  $1C20   }

1BF8-  A9 1E     LDA  ##$1E   }
1BFA-  48        PHA          } "COMM#3 ERR"
1BFB-  A9 7E     LDA  #$7E    }  and stop
1BFD-  4C 20 1C  JMP  $1C20   }

1C00-  A9 1E     LDA  ##$1E   }
1C02-  48        PHA          } "C128 NOT READY"
1C03-  A9 89     LDA  #$89    }  and continue
1C05-  4C 1D 1C  JMP  $1C1D   }

1C08-  A9 1E     LDA  ##$1E   }
1C0A-  48        PHA          } "DATA OVERLOAD"
1C0B-  A9 98     LDA  #$98    }   and stop
1C0D-  4C 20 1C  JMP  $1C20   }

1C10-  A9 1E     LDA  ##$1E   }
1C12-  48        PHA          } "WRAP AT END"
1C13-  A9 A6     LDA  #$A6    }  and stop
1C15-  4C 20 1C  JMP  $1C20   }

1C18-  A9 1E     LDA  ##$1E   }
1C1A-  48        PHA          } "DRIVE NOT READY"
1C1B-  A9 B2     LDA  #$B2    }  and continue
1C1D-  18        CLC          clear carry to indicate RTS req'd after display
1C1E-  90 01     BCC  $1C21
1C20-  38        SEC          set carry to indicate STOP after display
1C21-  85 A0     STA  $A0     }
1C23-  68        PLA          }
1C24-  85 A1     STA  $A1     }
1C26-  A0 00     LDY  #$00    }
1C28-  B1 A0     LDA  ($A0),Y } display message "PROBLEMS:xxxxxxxxxx "
1C2A-  F0 06     BEQ  $1C32   }
1C2C-  99 5F 04  STA  $045F,Y }
1C2F-  C8        INY          }
1C30-  D0 F6     BNE  $1C28   }
1C32-  90 07     BCC  $1C3B   check for stop or continue
1C34-  78        SEI          want to stop, so set interrupt disable,
1C35-  20 F8 19  JSR  $19F8   turn off Tx,
1C38-  4C E0 1F  JMP  $1FF0   and jump to ~~Apple monitor (i.e. stop progrm)~~
1C3B-  60        RTS          No! keep going and indicate trouble by
                              incementing a screen byte
                              then re-start as if after NMI
```

```
1C40-  A9 AD      LDA  #$AD                    DISPLAY I/Q or Q/I and
1C42-  85 A0      STA  $A0                     SELECTED HT# FOR CALCULATION
1C44-  A9 05      LDA  #$05   screen addr.
1C46-  85 A1      STA  $A1
1C48-  A9 04      LDA  #$04
1C4A-  85 A2      STA  $A2    Rx(screen line)cntr
1C4C-  A0 07      LDY  #$07
1C4E-  A9 A0      LDA  #$A0     " "
1C50-  91 A0      STA  ($A0),Y  clear line
1C52-  88         DEY
1C53-  10 FB      BPL  $1C50
1C55-  A0 02      LDY  #$02     I/Q screen pos'n
1C57-  A6 A2      LDX  $A2
1C59-  BD 99 60   LDA  $6099,X  ratio
1C5C-  30 0C      BMI  $1C6A     top bit set(clear) so must be I/Q(Q/I) value
1C5E-  A0 07      LDY  #$07     Q/I screen pos'n
1C60-  C9 64      CMP  #$64     if value=100,just display 1, no decimal point
1C62-  D0 06      BNE  $1C6A
1C64-  A9 B1      LDA  #$B1     "1"
1C66-  91 A0      STA  ($A0),Y
1C68-  D0 10      BNE  $1C7A
1C6A-  29 7F      AND  #$7F     strip top bit
1C6C-  20 B0 1C   JSR  $1CB0    get ascii BCD
1C6F-  91 A0      STA  ($A0),Y  store L char.
1C71-  8A         TXA
1C72-  88         DEY
1C73-  91 A0      STA  ($A0),Y  store H char
1C75-  88         DEY
1C76-  A9 AE      LDA  #$AE     "."
1C78-  91 A0      STA  ($A0),Y  store decimal point
1C7A-  A5 A0      LDA  $A0
1C7C-  18         CLC
1C7D-  69 80      ADC  #$80     mod scrn addr. to next line/Rx
1C7F-  85 A0      STA  $A0
1C81-  90 02      BCC  $1C85
1C83-  E6 A1      INC  $A1
1C85-  C6 A2      DEC  $A2
1C87-  D0 C3      BNE  $1C4C
1C89-  A9 20      LDA  #$20
1C8B-  38         SEC                          display ht# used(Ø=lowest gate)
1C8C-  ED 9E 60   SBC  $609E                     in I/Q,Q/I calc.
1C8F-  20 B0 1C   JSR  $1CB0  ascii BCD
1C92-  8E 8A 07   STX  $078A  H
1C95-  8D 8B 07   STA  $078B  L
1C98-  EA         NOP
1C99-  A0 20      LDY  #$20
1C9B-  A9 A0      LDA  #$A0
1C9D-  99 D6 05   STA  $05D6,Y  clear "data and analysis" line
1CA0-  88         DEY             (nothing to do with I/Q, by the way.)
1CA1-  D0 FA      BNE  $1C9D
1CA3-  60         RTS
1CA4-  FF         ???
1CA5-  FF         ???
1CA6-  FF         ???
1CA7-  FF         ???
1CA8-  FF         ???
1CA9-  FF         ???
1CAA-  FF         ???
1CAB-  FF         ???
1CAC-  FF         ???
1CAD-  FF         ???
1CAE-  FF         ???
1CAF-  FF         ???
1CB0-  A2 FF      LDX  #$FF                    CONVERT A to BCD ASCII
1CB2-  38         SEC
1CB3-  E9 0A      SBC  #$0A
1CB5-  E8         INX
1CB6-  B0 FB      BCS  $1CB3
1CB8-  69 0A      ADC  #$0A
1CBA-  09 B0      ORA  #$B0
1CBC-  48         PHA
1CBD-  8A         TXA
1CBE-  09 B0      ORA  #$B0
1CC0-  AA         TAX            ascii H char.
1CC1-  68         PLA            ascii L char.
1CC2-  60         RTS
1CC3-  FF         ???
1CC4-  FF         ???
1CC5-  FF         ???
1CC6-  FF         ???
1CC7-  FF         ???
```

```
1CC8-  A2 08      LDX  #$08
1CCA-  BD 69 1E   LDA  $1E69,X   } leave "OK" if status=Ø
1CCD-  9D 46 06   STA  $0646,X   }
1CD0-  CA         DEX            }
1CD1-  D0 F7      BNE  $1CCA     }
1CD3-  AD FE 5F   LDA  $5FFE     get status byte
1CD6-  F0 12      BEQ  $1CEA     ok,=Ø, so skip out
1CD8-  85 CA      STA  $CA       something wrong,so display all bits(msb at left!)
1CDA-  A2 08      LDX  #$08
1CDC-  A9 B0      LDA  #$B0  "0"
1CDE-  46 CA      LSR  $CA
1CE0-  90 02      BCC  $1CE4
1CE2-  A9 B1      LDA  #$B1  "1"
1CE4-  9D 46 06   STA  $0646,X
1CE7-  CA         DEX
1CE8-  D0 F2      BNE  $1CDC
1CEA-  D8         CLD
1CEB-  60         RTS
1CEC-  FF         ???
```

DISPLAY CPU#1 STATUS BYTE

(located in$FE in CPU#1

e.g. 01100010 (msb)

N.B.!!! have to set Binary arith mode before any arith. in Interrupt, because clock program, which may be interrupted, uses decimal mode

```
1CED-  A5 B6      LDA  $B6   cpu#2 status byte
1CEF-  EA         NOP
1CF0-  F0 12      BEQ  $1D04 =Ø,so not in a comm, link or tape wr.??
1CF2-  85 A0      STA  $A0
1CF4-  A2 08      LDX  #$08  should never happen,
1CF6-  A9 B0      LDA  #$B0    so skip out
1CF8-  46 A0      LSR  $A0
1CFA-  90 02      BCC  $1CFE
1CFC-  A9 B1      LDA  #$B1  display all bits, msb at left
1CFE-  9D 46 07   STA  $0746,X
1D01-  CA         DEX
1D02-  D0 F2      BNE  $1CF6
1D04-  60         RTS
```

DISPLAY CPU#2 STATUS BYTE

only called during NMI recovery, i.e. to show where it was stuck.

```
1D05-  FF         ???
1D06-  FF         ???
1D07-  FF         ???
1D08-  FF         ???
1D09-  FF         ???
1D0A-  FF         ???
1D0B-  FF         ???
1D0C-  FF         ???
1D0D-  FF         ???
1D0E-  FF         ???
1D0F-  FF         ???
1D10-  20 58 FC   JSR  $FC58  clear screen(mon.rout.)
1D13-  A9 40      LDA  #$40   }
1D15-  85 A0      STA  $A0    } point to ascii data tables
1D17-  A9 1D      LDA  #$1D   }
1D19-  85 A1      STA  $A1    }
1D1B-  A0 00      LDY  #$00
1D1D-  B1 A0      LDA  ($A0),Y
1D1F-  85 A2      STA  $A2       L
1D21-  C8         INY
1D22-  B1 A0      LDA  ($A0),Y   H
1D24-  F0 19      BEQ  $1D3F     check for end of fixed display data
1D26-  85 A3      STA  $A3       H
1D28-  C8         INY            -(next) ascii char
1D29-  B1 A0      LDA  ($A0),Y   ascii char
1D2B-  F0 05      BEQ  $1D32     end of this string,so get new addr.
1D2D-  91 A2      STA  ($A2),Y   write ascii char on screen
1D2F-  18         CLC
1D30-  90 F6      BCC  $1D28     forced
1D32-  C8         INY            }
1D33-  98         TYA            } modify base screen addr. to point to 1st
1D34-  18         CLC            }   byte after Ø (=low screen addr. byte for
1D35-  65 A0      ADC  $A0       }     next string)
1D37-  85 A0      STA  $A0       }
1D39-  90 E0      BCC  $1D1B     }
1D3B-  E6 A1      INC  $A1       }
1D3D-  D0 DC      BNE  $1D1B     }
1D3F-  60         RTS
```

1st 2 bytes in data string are screen address minus 2 (L&H), rest are ascii, ends with a zero

*1D40.1ECF

Character strings
for display

```
1D40- 88 04 C3 CF C8 C5 D2 C5
1D48- CE D4 A0 D2 D4 D7 A0 D3
1D50- D9 D3 D4 C5 CD 00 01 07
1D58- C1 AF C4 A0 D3 D4 C1 D4
1D60- D5 D3 BA 00 18 07 A3 D4
1D68- D2 A0 D2 C5 D3 C5 D4 BD
1D70- 00 28 04 D2 D8 A1 C7 C1
1D78- C9 CE A0 D2 C1 D4 A1 CF
1D80- C6 C6 D3 C5 D4 00 45 04
1D88- A3 CE CD C9 BD 00 AA 04
1D90- A1 C9 AF D1 A0 A0 D1 AF
1D98- C9 A1 A0 A0 C9 A0 A0 D1
1DA0- 00 28 05 AD AD A1 AD AD
1DA8- AD AD AD AD AD AD A1 AD
1DB0- AD AD AD AD AD 00 3E 05
1DB8- A3 C9 CE D4 C6 C3 C5 A0
1DC0- D2 D3 D4 BD 00 A8 05 A3
1DC8- B1 A1 00 B3 05 A1 00 28
1DD0- 06 A3 B2 A1 00 33 06 A1
1DD8- 00 3F 06 C3 D0 D5 A3 B1
1DE0- BA 00 A8 06 A3 B3 A1 00
1DE8- B3 06 A1 00 28 07 A3 B4
1DF0- A1 00 33 07 A1 00 3F 07
1DF8- C3 D0 D5 A3 B2 BA 00 54
1E00- 04 D0 D2 CF C2 CC C5 CD
1E08- D3 BA 00 59 05 C4 C1 D4
1E10- C1 A0 C9 AF D0 A0 A6 A0
1E18- C1 CE C1 CC D9 D3 C9 D3
1E20- 00 50 06 C8 D4 A3 A0 A0
1E28- AD AD AD AD AD AD AD AD
1E30- AD AD AD AD AD AD AD AD
1E38- AD AD AD AD AD AD AD AD
1E40- AD AD AD AD AD AD AD AD
1E48- 00 85 07 C8 D4 A3 00 87
1E50- 06 A8 A0 A0 A0 D0 D5 CC
1E58- D3 C5 A0 C9 CE D4 C5 C7
1E60- D2 C1 D4 C9 CF CE A9 00
1E68- 00 00 CF CB A0 A0 A0 A0
1E70- A0 A0 00 C3 CF CD CD A3
1E78- B1 A0 C5 D2 D2 00 C3 CF
1E80- CD CD A3 B3 A0 C5 D2 D2
1E88- 00 43 71 72 78 A0 4E 4F
1E90- 54 A0 52 45 41 44 59 00
1E98- C4 C1 D4 C1 A0 CF D6 C5
1EA0- D2 CC CF C1 C4 00 D7 D2
1EA8- C1 D0 A0 C1 D4 A0 C5 CE
1EB0- C4 00 44 52 49 56 45 A0
1EB8- 4E 4F 54 A0 52 45 41 44
1EC0- 59 00 54 41 50 45 A0 50
1EC8- 41 53 54 A0 45 4F 54 00
```

fixed screen display

end of fixed display

separate strings:
1E6A: "OK "
1E73: "COMM#1 ERR"
1E7D: "COMM#3 ERR"
1E89: "C128 NOT READY" (flashing)
1E98: "DATA OVERLOAD"
1EA6: "WRAP AT END"
1EB2: "DRIVE NOT READY"(flashing)
1EC2 ; "TAPE PAST EOT"(flashing)

```
1ED0-  20 F8 19   JSR  $19F8 tx off (ie.7.5Hz background)   SET TX RATE&AMOUNT OF INTEGR.
1ED3-  AD 59 C0   LDA  $C059 synch rate=7.5Hz
1ED6-  A9 04      LDA  #$04  amount of integr.=4            (depend of time of day)
1ED8-  48         PHA        save on stack
1ED9-  AD ØA Ø3   LDA  $Ø3ØA   H10
1EDC-  0A         ASL
1EDD-  0A         ASL
1EDE-  0A         ASL
1EDF-  0A         ASL
1EE0-  0D Ø9 Ø3   ORA  $Ø3Ø9   H1
1EE3-  C9 03      CMP  #$03        compare with Ø3GMT
1EE5-  90 16      BCC  $1EFD     less, so stay with 7.5Hz,4 puls integr
1EE7-  C9 16      CMP  #$16        compare with 16GMT
1EE9-  B0 12      BCS  $1EFD     greater or equal, so stay with 7.5Hz, 4 pulse in
1EEB-  AD 58 C0   LDA  $C058     "nighttime"=03Z-15Z incl.,so set 60Hz synch
1EEE-  20 17 0E   JSR  $0E17     turn on tx (60 Hz rate)
1EF1-  A0 05      LDY  #$05      wait for ½ sec
1EF3-  20 A8 FC   JSR  $FCA8
1EF6-  88         DEY
1EF7-  D0 FA      BNE  $1EF3
1EF9-  68         PLA            clear stack( discard the 4)
1EFA-  A9 20      LDA  #$20      use 32 pulse integr.
1EFC-  48         PHA            save on stack
1EFD-  68         PLA            get #pulse integ. off stack
1EFE-  8D 92 BC   STA  $BC92     save in D/T store
1F01-  8D F7 5E   STA  $5EF7     put into cmd store going to CPU#1
1F04-  60         RTS
1F05-  FF         ???
1F06-  FF         ???

1F07-  AD F1 C0   LDA  $C0F1 drive status      Check for EOT
1F0A-  29 40      AND  #$40  get eot flag                    slot 7 is drive
1F0C-  F0 11      BEQ  $1F1F not EOT,so return               CRA6 is EOT flag
1F0E-  A9 1F      LDA  #$1F EOT, so don't want to write!
1F10-  48         PHA          push return addr on stack,
1F11-  A9 1C      LDA  #$1C    and jump to display "drive at eot" msg
1F13-  48         PHA
1F14-  18         CLC          clear flag,so doesn't go to ff69 at end of msg
1F15-  A9 1E      LDA  #$1E
1F17-  48         PHA          load start addr. of message
1F18-  A9 C2      LDA  #$C2
1F1A-  4C 21 1C   JMP  $1C21   and display msg after "problems", and rts
1F1D-  68         PLA            to here; now pop last rtn addr. off stack
1F1E-  68         PLA            (don't want to go back to tape dump!)
1F1F-  60         RTS            and return to the routine that called tape dump

1F20-  EE 24 07   INC  $0724 inc scrn addr     INCREMENT"#TR RST"on SCREEN
1F23-  AD 24 07   LDA  $0724  was screen val
1F26-  C9 A1      CMP  #$A1   equal to"space"before inc?
1F28-  D0 05      BNE  $1F2F   no, so inc. will give next digit
1F2A-  A9 B1      LDA  #$B1    yes, so put "1"
1F2C-  8D 24 07   STA  $0724
1F2F-  60         RTS

1F30-  AD FB 5F   LDA  $5FFB   get"pwr-up flag"   INCREMENT #INTFC RST on SCREEN
1F33-  F0 0F      BEQ  $1F44   not set,so return.
1F35-  EE 4C 05   INC  $054C   set,so inc. scrn.   (count number of CPU#1 resets)
1F38-  AD 4C 05   LDA  $054C Was scrn addr.
1F3B-  C9 A1      CMP  #$A1  a "space" before?
1F3D-  D0 05      BNE  $1F44  no, so ok
1F3F-  A9 B1      LDA  #$B1   yes,so put "1"
1F41-  8D 4C 05   STA  $054C
1F44-  60         RTS

1F50-  AD FF FF   LDA  $FFFF   }
1F53-  AD FF FF   LDA  $FFFF   }                    FILE GAP!
1F56-  AD FF FF   LDA  $FFFF   }
1F59-  AD FF FF   LDA  $FFFF   }   dummy filler
1F5C-  AD FF FF   LDA  $FFFF   }
1F5F-  AD FF FF   LDA  $FFFF   }
1F62-  EA         NOP          }
1F63-  EA         NOP
1F64-  A9 00      LDA  #$00        clear
1F66-  85 C5      STA  $C5          file gap flag,
1F68-  20 12 C7   JSR  $C712        and call file gap routine
1F6B-  60         RTS
1F6C-  FF         ???
1F6D-  FF         ???
1F6E-  FF         ???
1F6F-  FF         ???
```

```
1F70-  A5 BA      LDA  $BA      1st rec?          TAPE STORE/DUMP
1F72-  F0 07      BEQ  $1F7B    no
1F74-  A9 00      LDA  #$00     yes,so set CF=0
1F76-  85 CF      STA  $CF      to mean 1st rec in 2.
1F78-  85 BA      STA  $BA      and clear 1st rec flg
1F7A-  60         RTS           and return
1F7B-  A5 CF      LDA  $CF      ok,we have data,is it
1F7D-  C9 01      CMP  #$01     the 2nd rec in a pair?
1F7F-  F0 1C      BEQ  $1F9D    yes,so it goes at top,and want dump to tape
1F81-  20 12 0E   JSR  $0E12    no,so turn on Tx for marker anyway
1F84-  E6 CF      INC  $CF      set flg for next call,
1F86-  A9 40      LDA  #$40
1F88-  85 AD      STA  $AD      and set addr. to put c128 output at bottom of
1F8A-  A9 00      LDA  #$00         dump store ($4000)
1F8C-  85 AC      STA  $AC
1F8E-  20 B0 1F   JSR  $1FB0    and go to move c128 o/p from$BCAØ-BFFF
1F91-  A9 00      LDA  #$00     add a delay
1F93-  20 A8 FC   JSR  $FCA8    so Tx marker will stay on a little longer
1F96-  EA         NOP
1F97-  EA         NOP
1F98-  EA         NOP
1F99-  20 F8 19   JSR  $19F8    and turn off Tx   (i.e. back to backgnd rate)
1F9C-  60         RTS
1F9D-  A9 00      LDA  #$00     This is the last data before dump
1F9F-  85 CF      STA  $CF      so set flg for next oper
1FA1-  A9 43      LDA  #$43
1FA3-  85 AD      STA  $AD      set addr. for move to $4360-up
1FA5-  A9 60      LDA  #$60
1FA7-  85 AC      STA  $AC
1FA9-  20 B0 1F   JSR  $1FB0    and move c128 output to there.
1FAC-  20 70 19   JSR  $1970    now go to dump$4ØØØ-46BF (2 recs) to tape
1FAF-  60         RTS
```

```
1FB0-  A9 A0      LDA  #$A0                          MOVE
1FB2-  85 AE      STA  $AE      set "from" addr ("to" addr already set)
1FB4-  A9 BC      LDA  #$BC
1FB6-  85 AF      STA  $AF
1FB8-  A0 00      LDY  #$00
1FBA-  B1 AE      LDA  ($AE),Y     move$BCAØ-BFFF to $4000-435F or 46C0-46BF
1FBC-  91 AC      STA  ($AC),Y
1FBE-  E6 AC      INC  $AC
1FC0-  D0 02      BNE  $1FC4
1FC2-  E6 AD      INC  $AD
1FC4-  E6 AE      INC  $AE
1FC6-  D0 F2      BNE  $1FBA
1FC8-  E6 AF      INC  $AF
1FCA-  A5 AF      LDA  $AF
1FCC-  C9 C0      CMP  #$C0
1FCE-  D0 EA      BNE  $1FBA
1FD0-  60         RTS
1FD1-  FF         ???
```

```
1FD1-  85 F9      STA  $F9   keep real chksum
1FD3-  A0 0B      LDY  #$0B                  Type Check-sum error msg
1FD5-  B9 DF 1F   LDA  $1FDF,Y get flashing ascii
1FD8-  99 DB 07   STA  $07DB,Y store at bottom centre of screen
1FDB-  88         DEY
1FDC-  D0 F7      BNE  $1FD5
1FDE-  60         RTS

1FE0- 43 48 4B 53 55 4D 60 45      ascii for flashing
1FE8- 52 52 61 BC FF FF FF FF      "CHKSUM ERR!"
*1FF0L                             stored checksum in 1FEB
```

```
1FF0-  EE 80 04   INC  $0480 top corner (LH)
1FF3-  4C 40 09   JMP  $0940                  Auxilliary Error counts
1FF6-  EE D0 07   INC  $07D0 bott. cnr.(LH)
1FF9-  4C 40 09   JMP  $0940
```

Increment screen address, and jump back to start as if after NMI

SECTION IV.

SECTION IV.

Section IV.: C128 (CPU#3) details

C128 analysis (W) [page 103]
C128 machine language (W) [page 107]
BASIC program, PMFCA wind analysis (L) [page 109]
C128 bulk storage (F) [page 115]
Machine language program (L) [page 116]
Interface connection between C128 and Apple (CPU#2) (F) [page 118]
C128 printer interface circuit and connections (F) [page 118]

C128 Analysis

This is a description of the analysis done in the C128.

Initialization includes a reference to the variable names roughly in order of decreasing use (line 25)- the idea is that the more used names will be near the top of the variable name list, to save execution time. The antenna pair directions are geographic in degrees East of North for the vectors from antenna #4 (centre) to #3 (P1), #4 to #2 (P2) and #4 to #1(P3). This is the order correlations are stored in the C(3,32) matrix. Constants based on these are also set (to save time by avoiding sin and cos later). The machine language programs are read from BASIC statements and poked into memory, and memory limits are set so BASIC storage occurs below the data (from CPU#2) store.

Then the program goes into the comm. link program with a USR function. When a set of data (all heights) has been read, the TI$ variable is set to 0 : this sets the "jiffy timer" (counts cycles of 60Hz) to zero, and can be read at the beginning of each height analysis to see whether there is enough time left to analyse the next height. The date/time and parameters (rec length) are read and displayed. If there has been an hour change, a subroutine is called which averages the data for the last hour number and sends it to the printer (and screen). Also, running height data pointers for cross (IC), auto-correl(IA), analysed data (ID), gain(IG), and signal variance(IV) are

set. These are incremented/decremented with each height loop.

For each height (until the time goes over the set limit): The height, signal and gain are displayed. The height number (0=49Km) is poked into the 1st byte of the height record. If the signal variance is less than 5, the height is skipped (i.e. the signal is very close to constant and analysis may produce spurious values)- a marker bit is placed in the top 4 bits of the gain byte to indicated this rejection. In this case all except the first byte of the 24 bytes height record will be left as zero.

If the variance is ok, then the auto bytes are read (PEEKed) out of memory - not all, just the 1st 4 which are special and give the #1-bits in each sequence (AR=I comp.,AI=Q comp.), and the first 6 (of 12 available) lags (AR=real,AI=imaginary). The numbers are read as positive 2 byte values, but the AI must be translated as "signed" 2-byte integers since they can be negative. At the same time, each sequence is tested for no 1-bits or all 1-bits - this means constant signal in the bit-amplitude sequence, - in which case another flag is set on top of the gain, and the height is skipped.

If the #1-bits is ok, the magnitudes of the combined mean and variance (both of which are complex numbers) of the bit-amplitude sequences (all antennas) are found from the #1-bits-per-sequence information. Using these and the #1-matches for non-zero lags, the mean lagged auto correlation values can be found - this includes the required correction to the correlation value because of the conversion of amplitudes to bit sequences. The magnitudes of the complex auto values are calculated for increasing lag until the value falls below 0.5, and the phases for the 1st 2 lags are saved. If there are no auto's greater than 0.5, the height is marked "fast fading" (again a bit on top of the gain byte, and also in the message byte at the end of the 24 byte height record). Linear interpolation is done to find the "exact" lag at 0.5, or if the correlation doesn't fall below 0.5, the greatest lag is used to define one point on the auto correlation. This is the value used in FCA.

If there is only one auto greater than 0.5, then there is a flag set, and the Vz calculation just uses the 1st lag auto, and there is no "noise factor" calculated. If there are two (or more), a Gaussian which peaks at zero lag is fitted (exact solution) to the 1st two points. If the curvature is zero or negative (i.e. auto constant or increasing with lag for these 2 points), the fit is not used. If it is ok, then the noise factor(NF), which is the extrapolated auto at zero lag, is poked into the height record; and the width (at ρ=0.61) in units of lags x 2 (max value clipped at 31) is also saved in the height record. The NF is clipped at 127 (the fit can find values greater than 1.0 even though they are theoretically meaningless) so that the top bit of this byte can be used for another flag if desired. A parabola is fitted to the 2 phase points, and the slope of the parabola at 0 lag gives Vz. If the difference of the phase slopes between 0-1 and 1-2 lags, as a fraction of the slope from 0-1 lag, is greater than 0.3,* the value is flagged as "too curved" - i.e. the 1st 2 lags have a difference of 30% in phase slope. [Earlier the limit was 20% , but most were more curved, so the it was changed]. The 2-byte stored Vz data has the value Vz*100+$8000 ($8000=32768), where Vz is in m/s.

* may have changed this again

.....we are now at line 500 .....

The complex mean and variance needed for calculating the cross correlations for each antenna pair are found next. Actually these values are correct only for zero lag, but are a good approximation if the number of lags is much smaller than the sequence length (as it is here). Thus bit-correlation value greater than 1 are possible, but the function which corrects correlation values for the use of bit-amplitudes cannot have a value greater than 1, and solves a potential problem here. Next, the zero lag cross correlations data are PEEKed from the correlation store, and the phases, and corrected magnitudes are found and POKEd into the height record as 2-byte integers: value=INT(ρ*100)*512+phase(deg). That is, the phase is contained in the lowest 9 bits of the 16-bit word. An attempt is then made to predict (from the zero lag auto values) the maximum lag needed to cover any significant correlation peaks (as is done in the RTW system). The idea is to try to reduce the number of correlation data which have to be translated into correlation values and searched for peaks. Those values are then PEEKed out of memory, and converted to the squared correlation sequences (uncorrected for bit-correlation), which are searched for peaks (to save time by avoiding too many square roots). The lowest acceptable correlation (0.1 "real" correlation) is translated into this "uncorrected" format, so that low peaks can be rejected right away. (the limit is 0.1*0.1/2.25).

A search for local peaks is done over each cross correlation sequence, and the indices for the two greatest peaks saved. A 3-point parabolic fit is done around each of these indicies to find the exact lag and peak value, which is then translated into a "real" correlation value. If the greatest peak is greater than 1.8 times the secondary one (if any), the latter is discarded. The "weed" routine looks at these peaks as follows: if there is only one significant peak in each cross correlation, the normalizeed time discrepancy (NTD) of the lags of these peaks must be less than 0.3 to continue; if there are 2 peaks in one, and 1 in each of the other correlations, the NTD must be less than 0.2 for one of the sets of 3 peaks; if there are 2 peaks in two or more correlations, the NTD for the greatest peaks must be less than 0.1. Note that the NTD of the greatest peaks is saved in the height record (even if that set of 3 peaks is not used, as might happen in the second case). The value of NTD saved is 2*INT(NTD*100)+x, where x is0 if the sum of tmax was +ve, and 1 otherwise. This sign bit indicates the sense of vorticity (i.e. pattern rotation), whose measurement was suggested by Miguel Larsen. But statistics show that on the average (weeks), the "sign-sequences" are random. Still, it may be of interest/use sometime.

Note that all tmax can sometimes be exactly zero; in this case there is no NTD (or wind), and the program replaces the height number with 255 as an error message.

If the data have not been rejected yet, a normal PMFCA is done. Note that after the peaks have been found, the analysis is exactly the same as in the RTW system. The only final difference is that the value of Vtrue*10 is saved. [Originally it was thought that the coherent system might produce more "accurate" values of Vtrue, however preliminary results do not support this. The magnitudes are lower than the RTW system - some sort of bias, so it makes sense to keep the factor of 10 anyway.] The other parameters (axes,tilt,Tc) are stored with the same accuracy as the RTW system.

The communication link (to CPU#2) routine is placed in the cassette buffer (196 bytes available) $1B00-1B5F. This location is always(?) accessible from Basic without change in bank structure. However, the bank structure (BANK 0) must be changed in the routine to access the user port. The user port is one port of a CIA (Port B), and the handshake line on the connector is bit#2 of port A. This is a normal port line, and is set to "output" by setting the DDR (data dirction register) for port A. The line is toggled by writing 0(L) then 4(H) to the port address. (there is no "auto strobe" as in a PIA). The sense line is called FLAG2 (this resembles CA1/CB1 in a PIA in that it cannot be used as an output line), and the interrupt flag is only set on a negative edge. To see whether the flag is set (high), the interrupt control register is read (reading it clears all such flags), and the appropriate bit referring to FLAG2 is masked off and tested.

Both ends of the link use a negative edge for a handshake.

The interrupt is disabled at the beginning of the routine to prevent interrupts (60Hz) by the "jiffy clock" (time of day), because it is uncertain whether this routine preserves and restores the present Bank structure. It was also found to cause trouble on PETs/VIC20s and C64s because it will sometimes add an extra delay in the program just as the comm. link direction is being changed - which seems to upset the Apple (that is, they get out of "synch").

The printer program was added later.There was not enough space to put it in the cassette buffer, so it was put in high memory, just below the main correlation store. The problem with entry to the routine is that BASIC must internally switch bank structure continually to access the BASIC ROMs, which are "behind" the 64K RAM specified by BANK0. Thus, although you can look in low memory for the print character output vector, it is uncertain what the Bank structure is when it is called, since this is determined internally by the BASIC monitor. Since the bank structure isn't known (except by gurus), you can't jump directly to your program in high memory, because it may be in a different bank! Anyway, to get around this problem, entry and exit points were placed at the end of the cassette buffer- which seems always available to BASIC, so that the bank structure (whatever it is during the BASIC output call) can be preserved for the return to normal output, while the required bank structure (one which sees the machine language printer routine and port) can be used during the routine. Interrupts have to be disabled (as before), so the jiffy timer (which was used in the Tromso C64 routine for time-out on printer-ready after line-feed command) cannot be used. Instead a delay loop is added to the program. The length of the delay depends on the CPU clock frequency during the routine. It is not known whether this is 1MHz (which is the speed always used for CIAs) or 2 MHz (which is set by the BASIC command "FAST" at the beginning of the BASIC program). Just in case it is 2 MHz, a 2 MHz PIA was used (the C128 doesn't 'know' there is a port here, so who knows what speed it will use). A fixed delay is invoked after sending each character (don't know how fast the printer will accept them , and it is uncertain whether the "busy" line from printer refers to character acceptance). An attempt to use the printer "busy" line to indicate the end of a line feed through CB1 on the PIA failed for unknown reasons(it always tested READY- have to read the port to clear from the character rdy??) - so at present there is just a fixed delay (which means printing is not as fast as it could be). Also, characters less than $20 are changed to spaces - which means the printer mode(e.g. characters per inch) can't be changed from BASIC.

LISTING OF BASIC PROGRAM FOR WIND ANALYSIS

C128PMFCA-V18J' (modif. lines from V18I' noted in margin)

```
• 10 REM-C128 COH-RTW PMFCA, AUG17,1989,VERSION 18J'
• 20 BANK0:FAST:DEFFNS(T)=SGN(T<0)-SGN(T>=0):REM-NB  C128 LOG. TRUE=-1 !!!
  25 TT=0:I=0:J=0:IA=0:IC=0:X=0:Y=0:MR=0:MI=0:S0=0:S1=0:S2=0:N=0:K=0:NN=0
  30 DIMC(3,32),M(3),MX(3,2),T(3,2),R(3,2),D%(16),MN(2,3),V(3)
  40 DIMMZ%(18),ME(6,2),NM(6),AR(10),AI(10)
  50 REM-ANT PAIR DIR 4=>3,4=>2,4=>1
  60 D=156:P=57.29578:P1=90/P:P2=210/P:P3=330/P:PZ=1.5708:TT=256:TU=TT*TT
  70 TV=TU-1:TW=TU/2-1
  80 D2=D*D:X1=D*COS(P1):X2=D*SIN(P1):X3=D*COS(P2)
  90 X4=D*SIN(P2):X5=D*COS(P3):X6=D*SIN(P3)
 100 U1=COS(2*P1):U2=COS(2*P2):U3=COS(2*P3)
 110 V1=SIN(2*P1):V2=SIN(2*P2):V3=SIN(2*P3)
 120 TS=.533:CC=.5:LX=.1:CY=0.00:FC=100*135/360/2/TS
 130 GOSUB2330:REM-LOAD MACH LANG,SET MEM LMTS
 135 GOSUB3500:REM-POKE PTR PROG
 140 REM-.....START PROCESSING.....
 144 PRINT"WAITING FOR DATA....":X=USR(X)
 150 TI$="000000":IC=52288:IA=40000:IO=53152:IG=52320:IV=52352
 160 REM-READ D/T
 170 ID=52384:FORI=1TO16:D%(I)=PEEK(ID-I):NEXT
 180 DY%=D%(3)*100+D%(4)*10+D%(5):HR%=D%(7)*10+D%(8):MN%=D%(9)*10+D%(10)
 185 IFLH%<>HR%THENGOSUB2040
 190 NN=8*(D%(16)*TT+D%(15))
 195 S=D%(1)/16:S=S*16-6*INT(S)
 200 PRINT"D/T=";S":"DY%"/"HR%:MN%" #PTS="NN" COHINT="D%(14)
 210 REM-.....HEIGHT LOOP.....
 220 FORIH=1TO32:IG=IG-1:IV=IV-1:NT=255:RL=15:IS=0:IR=0:IX=0
 230 IFTI>16200THENPRINT"OUT OF TIME":GOTO140:REM-OVER 4.5 MIN SO QUIT
 240 PRINT"HKM=";IH*3+46;" SIG=";PEEK(IV);" GN=";PEEK(IG)
 250 POKEIO-1,IH-1
 255 IFPEEK(IV)<5THENPRINT" SD TOO LOW":POKEIG,PEEK(IG)+16:GOTO2000
 260 REM-.....PEEK REOD PART OF AUTO.....
 270 FORI=1TO10:J=I*2-1:AR(I)=PEEK(IA-J)+PEEK(IA-J-1)*TT
 280 X=PEEK(IA-J-32)+PEEK(IA-J-33)*TT:IFX>TWTHENX=X-TU
 287 IFI<5THENIFAR(I)=NNORX=NNTHENPRINT" CONST SIG":POKEIG,PEEK(IG)+48:GOTO2000
 288 IFI<5THENIFAR(I)=0ORX=0THENPRINT" CONST SIG":POKEIG,PEEK(IG)+48:GOTO2000
 290 AI(I)=X:NEXTI
 300 SR=(AR(1)+AR(2)+AR(3)+AR(4))/NN/4:SI=(AI(1)+AI(2)+AI(3)+AI(4))/NN/4
 310 MR=SR*SR+SI*SI:VR=SR*(1-SR)+SI*(1-SI)
 330 FORI=5TO10:N=4*(NN-I+4):X=(AR(I)/N-MR)/VR:Y=AI(I)/N/VR
 340 S1=SIN(PZ*SQR(X*X+Y*Y)):AR(I)=S1:IFI<7THENGOSUB1410:AI(I)=AA
 350 IFS1<CCTHEN370
 355 S0=S1:NEXTI
 360 RA=S1:TA=(10-4)*TS:GOTO390
 370 IFI=5THENNT=255:IX=7:PRINT" FAST FADE":POKEIG,PEEK(IG)+32:GOTO2000
 380 TA=(I-5+(S0-CC)/(S0-S1))*TS:RA=CC
 390 REM-VZ
 400 IFI=6THENB=INT(AI(5)*FC):POKEIO-10,128:PRINT" 1PT ";:GOTO420
 405 X=LOG(AR(5)):Y=LOG(AR(6)):IFX<=YTHEN410
• 407 NF=EXP(X+(X-Y)/3)*100:S=SQR(0.66/(X-Y))*4:IFNF>127THENNF=127
• 408 POKEIO-11,NF:IFS>63THENS=63
 409 POKEIO-10,S
 410 X=AI(5):Y=AI(6):B=INT((4*X-Y)/2*FC):REM VZ*100
• 415 IFX<>0THENIFABS(Y/X-2)>.6THENPOKEIO-10,PEEK(IO-10)+64:PRINT" TOO CRVD ";
 420 PRINT" VZ=";B/100;" M/S"
 430 X=B+TW+1
 440 IFX>TVTHENX=TV
 450 IFX<0THENX=0
 460 Y=INT(X/TT):POKEIO-8,Y:POKEIO-9,X-Y*TT
 500 REM-.....GET MEANS AND VAR FOR XCORL.....
 510 MR=AR(4)/NN:MI=AI(4)/NN:VR=MR*(1-MR)+MI*(1-MI)
 530 FORI=1TO3:X=AR(4-I)/NN:Y=AI(4-I)/NN
 550 MN(1,I)=X*MR+Y*MI:MN(2,I)=X*MI-Y*MR:V(I)=SQR((X*(1-X)+Y*(1-Y))*VR)
 560 NEXTI
```

V18J - cont'd

```
570 REM-.............GET REDUCED LAG,RL,M1,M2.............
580 S0=2:K=I0:ID=IC
590 FORJ=1TO3:MR=ID-33:MI=MR-64
600 X=((PEEK(MR)+PEEK(MR-1)*TT)/NN-MN(1,J))/V(J)
610 Y=PEEK(MI)+PEEK(MI-1)*TT:IFY>TWTHENY=Y-TU
620 Y=(Y/NN-MN(2,J))/V(J)
630 S1=SIN(PZ*SQR(X*X+Y*Y)):GOSUB1410:Y=INT(S1*100)*512+A
640 K=K-2:POKEK-1,INT(Y/TT):POKEK,Y-INT(Y/TT)*TT:IFS1<S0THENS0=S1
650 ID=ID-128:NEXTJ
660 IFS0>0.01THENRL=INT(TA*((LOG(S0)-2)/LOG(RA))/TS+3.5)
670 IFRL>15THENRL=15
680 M1=16-RL:M2=17+RL
690 REM-.............PEEK USED CROSS ...............
700 ID=IC-1:REM-CORRELS J=1,3 ARE 4V3,4V2,4V1
710 FORJ=1TO3:MR=ID-(M1-1)*2:MI=MR-64:S1=MN(1,J):S2=MN(2,J):S0=V(J)^2
720 FORI=M1TOM2:N=NN-ABS(17-I):X=(PEEK(MR)+PEEK(MR-1)*TT)/N-S1
730 Y=PEEK(MI)+PEEK(MI-1)*TT:IFY>TWTHENY=Y-TU
735 REM-USING SQUARE OF XCOR HERE!!
740 Y=Y/N-S2:C(J,I)=(X*X+Y*Y)/S0
760 MR=MR-2:MI=MI-2:NEXTI
770 ID=ID-128:NEXTJ
780 REM-.............FIND PEAKS.............
790 X=LX*LX/2.25:M1=M1+1:M2=M2+1:IFM2>32THENM2=32
800 IFM1<3THENM1=3
810 FORI=1TO3:N=0:MI=0:S0=C(I,M1-2):S1=C(I,M1-1)
820 FORJ=M1TOM2:S2=C(I,J)
825 IFS1<X OR S1<=S0 OR S1<S2THENS0=S1:S1=S2:NEXTJ:GOTO890
830 N=N+1:IFN=1THENMR=S1:K1=J-1:S0=S1:S1=S2:NEXTJ:GOTO890
840 IFS1>=MRTHENMI=MR:MR=S1:K2=K1:K1=J-1:S0=S1:S1=S2:NEXTJ:GOTO890
850 IFS1>MITHENMI=S1:K2=J-1
880 S0=S1:S1=S2:NEXTJ
890 MX(I,1)=K1:R(I,1)=MR:IFN>1THENMX(I,2)=K2:R(I,2)=MI:N=2
900 M(I)=N:NEXTI
930 GOSUB1470
940 IFIX=1ORIX=2THENGOTO1840
955 PRINT" ";
960 PRINTMX(1,1);MX(2,1);MX(3,1);INT(R(1,1)*100);INT(R(2,1)*100);INT(R(3,1)*100)
970 IFIR=4ORFL=1THENFL=0:GOTO1840
980 REM..... WLSFIT .....
990 S1=R1*R1:S2=R2*R2:S3=R3*R3
1000 Q1=X1*X1*S1+X3*X3*S2+X5*X5*S3
1010 Q=X2*X1*S1+X3*X4*S2+X5*X6*S3
1020 Q2=X2*X2*S1+X4*X4*S2+X6*X6*S3
1030 D1=Q1*Q2-Q*Q:A1=Q2/D1:A=-Q/D1:A2=Q1/D1
1040 X=T1*(X1*A1+A*X2)*S1+T2*(X3*A1+X4*A)*S2
1050 X=X+T3*(X5*A1+X6*A)*S3:Y=T1*(X1*A+X2*A2)*S1
1060 Y=Y+T2*(X3*A+X4*A2)*S2+T3*(X5*A+X6*A2)*S3
1070 Q=X*X+Y*Y
1080 GOSUB1410
1090 VA=1/SQR(Q)
1100 REM
1110 T1=D*COS(P1-A1)/VA:T2=D*COS(P2-A1)/VA:T3=D*COS(P3-A1)/VA
```

V18J. cont'd

```
1120 REM-..................... PMFCA.....................
1130 Q=-2*LOG(RA)/TA/TA
1140 MA=(-2*LOG(R1)+Q*T1*T1)/D2
1150 MB=(-2*LOG(R2)+Q*T2*T2)/D2
1160 MC=(-2*LOG(R3)+Q*T3*T3)/D2
1170 N2=MC-MB:N3=MA-MC:N1=MB-MA
1180 Y=N2*U1+N3*U2+N1*U3:X=-N2*V1-N3*V-N1*V3
1190 GOSUB1410
1200 S0=A1/2:TL=A/2
1210 MR=COS(P1-S0):MI=COS(P2-S0):S1=SIN(P1-S0):S2=SIN(P2-S0)
1220 R2=(MA*MI*MI-MB*MR*MR)/N1:R2=R2/(1+R2)
1230 B2=(MR*MR/R2+S1*S1)/MA:A2=R2*B2
1240 IFA2<=0ORB2<=0THENPRINT" IMAG AXIS":IX=5:GOTO1840
1250 AX=SQR(A2):BX=SQR(B2)
1260 X=A2*(T2*S1-T1*S2):Y=B2*(T1*MI-T2*MR)
1270 GOSUB1410
1280 PT=A+TL:PT=PT-180*(1+SGN(PT-360))
1290 Q1=COS(A1):Q2=SIN(A1)
1300 VT=Q*T1/D/(Q1*MR/A2+Q2*S1/B2)/2
1310 IFVT<0THENPT=PT-180*SGN(PT-180):VT=-VT
1320 MI=1/(Q-4*VT*VT*(Q1*Q1/A2+Q2*Q2/B2))
1330 IFMI<=0THENPRINT" IMAG TC":IX=6:GOTO1840
1340 TC=SQR(MI)
1350 IFAX<BXTHENA1=AX:AX=BX:BX=A1:TL=TL-90*FNS(TL-90)
1370 PRINT" FCA";INT(VT);INT(PT);INT(AX);INT(BX);INT(TL);INT(TC*10)/10
1380 REM END OF FCA
1390 Q1=COS(PT/P):Q2=SIN(PT/P)
1400 GOTO1850
1410 REM..................... ATAN(Y/X),0-360 ETC.....................
1420 IFX<>0THEN1440
1430 A=90*(2-FNS(Y)):GOTO1450
1440 A=ATN(Y/X)*P+90*(2-FNS(Y)*(1+FNS(X)))
1450 A1=A/P:AA=A:IFAA>180THENAA=AA-360
1460 RETURN
1470 REM..................... WEED FOR PMFCA .....................
1480 FORK=1TO3:I=M(K)
1490 IFI=0THENPRINT" NLP";K:IR=4:IX=1:RETURN
1520 FORJ=1TOI:MX=MX(K,J)
1530 S0=SQR(C(K,MX-1)):S1=SQR(R(K,J)):S2=SQR(C(K,MX+1))
1540 MR=(S0+S2)/2-S1:MI=(S2-S0)/2:X=-MI/MR/2
1550 S0=MR*X*X+MI*X+S1
1560 T(K,J)=(X+MX-17)*TS-K*CY
1570 R(K,J)=SIN(PZ*S0)
1600 NEXTJ
1610 IFI=2THENIFR(K,1)/R(K,2)>1.8THENI=I-1
1630 M(K)=I:NEXTK
1640 REM HERE M(K)=1OR2
1650 IS=M(1)+M(2)+M(3)-3
1660 ONIS+1GOTO1670,1710,1780,1780
1670 S1=0:S2=0:FORK=1TO3:S1=S1+T(K,1):S2=S2+ABS(T(K,1)):NEXTK
1675 IFS2=0THEN1825
1680 NT=INT(ABS(S1)/S2*100)*2-(S1<0)
1690 IFABS(S1)/S2>.3THENIR=4:IX=3:RETURN
1700 GOTO1810
1710 FORK=1TO3:IFM(K)=2THENK1=K
1720 NEXTK:IR=K1
1730 K2=3-INT(K1/3):K3=2-INT(K1/2):S1=T(K2,1)+T(K3,1)
1740 S2=ABS(T(K2,1))+ABS(T(K3,1)):S0=S2+ABS(T(K1,1)):IFS0=0THEN1825
1745 S=S1+T(K1,1):NT=INT(ABS(S)/S0*100)*2-(S<0)
1750 FORK=1TO2:S=T(K1,K):IFS2+ABS(S)=0THEN1825
1752 IFABS(S1+S)/(S2+ABS(S))<.2THEN1770
1755 NEXT
1760 IR=4:IX=4:RETURN
1770 T(K1,1)=T(K1,K):R(K1,1)=R(K1,K):GOTO1810
1780 NT=ABS(T(1,1))+ABS(T(2,1))+ABS(T(3,1)):IFNT=0THEN1825
1785 S=T(1,1)+T(2,1)+T(3,1):NT=ABS(S)/NT
1790 IFNT>.1THENIR=4:NT=INT(NT*100)*2-(S<0):RETURN
1800 IR=5:NT=INT(NT*100)*2-(S<0)
1810 T1=T(1,1):T2=T(2,1):T3=T(3,1):R1=R(1,1):R2=R(2,1):R3=R(3,1)
1820 RETURN
1825 PRINT"DIV-BY-0,SKIP HT":FL=1: NE=NE+1:POKE10-1,255:IR=4:RETURN
```

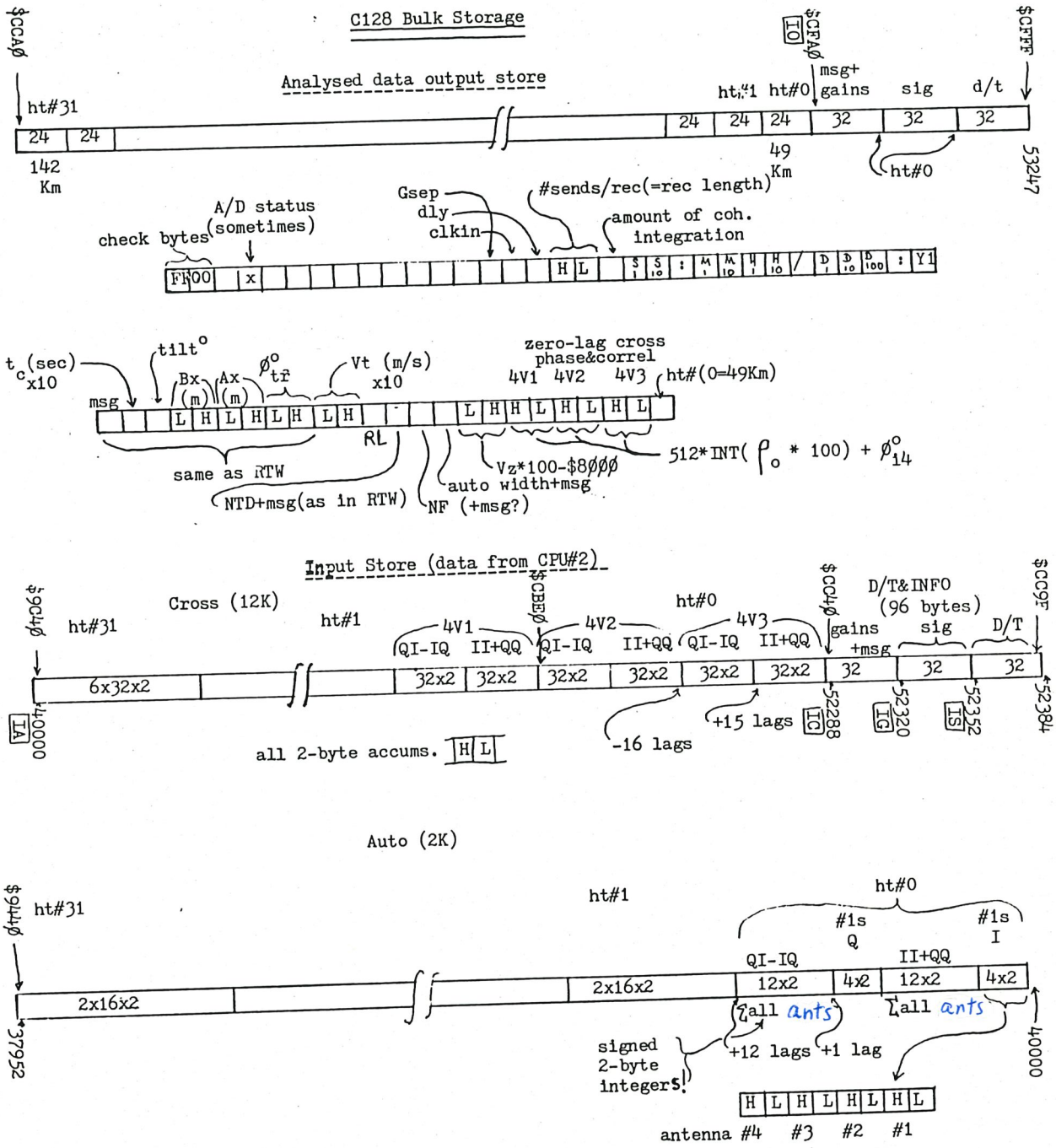
↑ V2 !! PROGRAM WAS OK (PRINTER DROPPED THE CHARACTER ?!?)

V18J - cont'd

```
1830 REM-.....O/P.....
1840 PRINT" ERR";NT;RL;IS;IR;IX:GOTO2000
1850 VT=VT*10:IFVT>TVTHENVT=TV
1860 K=IO-14:X%=INT(VT/TT):POKEK,X%
1870 X%=INT(VT-X%*TT):POKEK-1,X%
1880 X%=INT(PT/TT):POKEK-2,X%:X%=INT(PT-X%*TT):POKEK-3,X%
1890 IFAX>TVTHENAX=TV
1900 X%=INT(AX/TT):POKEK-4,X%
1910 X%=INT(AX-X%*TT):POKEK-5,X%
1920 IFBX>TVTHENBX=TV
1930 X%=INT(BX/TT):POKEK-6,X%
1940 X%=INT(BX-X%*TT):POKEK-7,X%
1950 X%=INT(TL):POKEK-8,X%:X%=TC*10
1960 IFX%>255THENX%=255
1970 POKEK-9,X%
1980 I%=(IH-3)/3:IF(I%-1)*(6-I%)<0THEN2000
1990 ME(I%,1)=ME(I%,1)+VT*Q1/10:ME(I%,2)=ME(I%,2)+VT*Q2/10:NM(I%)=NM(I%)+1
2000 ER%=IS+IR*4+IX*32:POKEIO-24,ER%
2010 K=IO-12:POKEK,NT:POKEK-1,RL
2020 IC=IC-384:IA=IA-64:IO=IO-24:NEXTIH
2030 PRINT"ACCUM DIV-BY-0="NE:GOTO140
2040 REM-.....PRINT 9 KM HRLY MEANS.....
2045 SYS37696
2050 IFHR%<>8THEN2100
2060 PRINT"  DAYHR!  64-70KM  !  73-79KM  !  82-88KM  !";
2070 PRINT"  91-97KM  ! 100-106KM ! 109-115KM !"
2080 PRINT" COHRTW! #   N   E ! #   N   E ! #   N   E !";
2090 PRINT" #   N   E ! #   N   E ! #   N   E !"
2100 DH=LD%*100+LH%:IFDH>9999THENPRINTSTR$(DH);:GOTO2130
2110 IFDH>999THENPRINT" ";STR$(DH);:GOTO2130
2120 PRINT"  ";STR$(DH);
2130 FORK=1TO6:KK=(K-1)*3+1:MZ%(KK)=NM(K)
2140 IFNM(K)>0THEN2160
2150 MZ%(KK+1)=0:MZ%(KK+2)=0:GOTO2210
2160 FORL=1TO2
2170 I%=ME(K,L)/NM(K):IFI%<-99THENI%=-99
2180 IFI%>99THENI%=99
2190 MZ%(KK+L)=I%
2200 NEXTL
2210 NM(K)=0:ME(K,1)=0:ME(K,2)=0
2220 NEXTK
2230 FORK=1TO18
2240 IFMZ%(K)<-9THEN2280
2250 IFMZ%(K)<0THEN2270
2260 IFMZ%(K)>9THEN2280
2270 PRINT"  ";STR$(MZ%(K));:GOTO2300
2280 PRINT" ";STR$(MZ%(K));:GOTO2300
2290 PRINTSTR$(MZ%(K));
2300 NEXTK
2310 PRINT:SYS37738
2320 LD%=DY%:LH%=HR%:RETURN
2330 REM-USR ADDR,MEM LMTS,MACH SUBR.
2340 POKE4626,63:POKE4627,147:REM-MEM LMTS ($933F)
2350 POKE4633,0:POKE4634,11:REM-USR ADDR($B00)
2354 REM-PUT IN CHECK BYTES
2355 POKE52352,255:POKE52353,0:RESTORE2380
2360 FORJ=2816TO2977:READX%:POKEJ,X%:NEXTJ
2370 RETURN
```

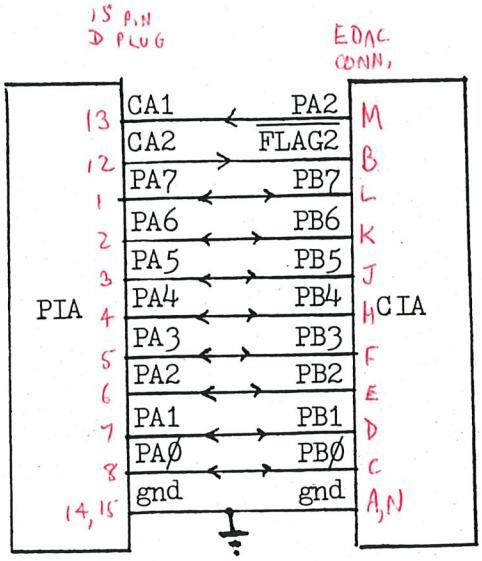
V18.5 cont'd

```
2380 DATA 120,173,0,255,72,169,62,141,0,255
2390 DATA 160,96,185,63,204,153,159,207,136,208
2400 DATA 247,169,4,141,2,221,169,255,141,3,221
2405 DATA169,4,141,0,221
2410 DATA 165,160,72,165,161,72,173,13,221
2420 DATA 169,207,133,161,169,160,133,160,162,4
2430 DATA 160,95,177,160,141,1,221,169,0,141
2440 DATA 0,221,169,4,141,0,221,169,0,145,160,173
2450 DATA 13,221,41,16,240,249,136,192,255,208
2460 DATA 225,198,161,202,208,220,169,64,133,160
2470 DATA 169,204,133,161,160,1,234,234,136,208
2480 DATA 251,234,169,0,141,3,221,162,57,160
2490 DATA 95,173,13,221,41,16,240,249,173,1
2500 DATA 221,145,160,169,0,141,0,221,169,4
2510 DATA 141,0,221,136,192,255,208,229,198,161
2520 DATA 202,208,224,104,133,161,104,133,160,104
2530 DATA 141,0,255,88,96
3500 REM-POKE PTR ROUT
3510 RESTORE3600:FORI=37696TO37830:READX:POKEI,X:NEXTI
3520 FORI=2978TO3002:READX:POKEI,X:NEXTI:RETURN
3600 DATA 120,174,0,255,169,62,141,0,255,169,162,141,38,3,169,11,141,39,3,173
3620 DATA3,223,41,251,141,3,223,169,255,141,2,223,169,44,141,3,223,142,0,255
3640 DATA88,96,169,121,141,38,3,169,239,141,39,3,96,56,72,233,1,208,252,104
3660 DATA233,1,208,246,96,140,61,148,104,141,62,148,104,141,63,148,41,127,201,32
3680 DATA176,31,201,13,240,4,169,32,208,23,173,2,223,169,13,141,2,223,160,12
3685 DATA 44,3,223
3690 DATA234,234,32,117,147,136,208,245,240,8,141,2,223,169,32,32,117,147,173,63
3695 DATA148,72,173,62,148,72,172,61,148,76,177,11
3700 REM-LOW MEM SECTION OF PTR ROUT
3710 DATA8,120,72,173,0,255,72,169,62,141,0,255,76,129,147,104,141,0,255,88,104
3720 DATA40,76,121,239

READY.
```


C128 Bulk Storage
$CCAØ
$CFAØ
IO
$CFFF
Analysed data output store
ht#31
ht#1
ht#0
msg+ gains
sig
d/t
24 24 24 24 24 32 32 32
142 Km
49 Km
53247
ht#0
A/D status (sometimes)
check bytes
Gsep
dly
clkin
#sends/rec(=rec length)
amount of coh. integration
FF00
x
H L
Y1
tc (sec) x10
tilt°
Bx (m)
Ax (m)
Øtf°
Vt (m/s) x10
zero-lag cross phase&correl
4V1 4V2 4V3
ht#(0=49Km)
msg
L H L H L H L H
RL
same as RTW
NTD+msg(as in RTW)
NF (+msg?)
auto width+msg
Vz*100-$8ØØØ
512*INT(ρo * 100) + Ø°14
Input Store (data from CPU#2)
$99CØ
$CBEØ
$CC4Ø
$CC9F
Cross (12K)
ht#31
ht#1
ht#0
4V1
4V2
4V3
QI-IQ II+QQ QI-IQ II+QQ QI-IQ II+QQ
D/T&INFO (96 bytes)
gains +msg
sig
D/T
6x32x2
32x2 32x2 32x2 32x2 32x2 32x2 32 32 32
40000
IA
52288
IQ
52320
IG
52352
IS
52384
-16 lags
+15 lags
all 2-byte accums. H L
Auto (2K)
$944Ø
ht#31
ht#1
ht#0
#1s Q
#1s I
QI-IQ
II+QQ
2x16x2
2x16x2
12x2 4x2 12x2 4x2
all ants
all ants
37952
40000
signed 2-byte integers!
+12 lags
+1 lag
H L H L H L H L
antenna #4 #3 #2 #1

```
. 00B00  78        SEI              disable interrupts        COMM LINK TO CPU#2
. 00B01  AD 00 FF  LDA $FF00  }---- save old bank structure
. 00B04  48        PHA        }
. 00B05  A9 3E     LDA #$3E   }---- set new bank structure(so can see port)
. 00B07  8D 00 FF  STA $FF00  }
. 00B0A  A0 60     LDY #$60   }
. 00B0C  B9 3F CC  LDA $CC3F,Y}---- move D/T,Sig,Gains from last input store
. 00B0F  99 9F CF  STA $CF9F,Y}        to present output store
. 00B12  88        DEY        }
. 00B13  D0 F7     BNE $0B0C  }
. 00B15  A9 04     LDA #$04   ] set PA2=output (handshake line)
. 00B17  8D 02 DD  STA $DD02  ]
. 00B1A  A9 FF     LDA #$FF   ] set PB all bits output
. 00B1C  8D 03 DD  STA $DD03  ]
. 00B1F  A9 04     LDA #$04   ] set  PA2=high
. 00B21  8D 00 DD  STA $DD00  ]
. 00B24  A5 A0     LDA $A0    }   save some zero page mem on stack
. 00B26  48        PHA        }    so can use as scratch here
. 00B27  A5 A1     LDA $A1    }
. 00B29  48        PHA        }
. 00B2A  AD 0D DD  LDA $DD0D     clear interrupt flag (if any)
. 00B2D  A9 CF     LDA #$CF
. 00B2F  85 A1     STA $A1       set up indirect addr. for output store
. 00B31  A9 A0     LDA #$A0
. 00B33  85 A0     STA $A0
. 00B35  A2 04     LDX #$04      # pages(or part pages) to send
. 00B37  A0 5F     LDY #$5F                                  [sending CFFF-CCA0]
. 00B39  B1 A0     LDA ($A0),Y   get data (analysed ),
. 00B3B  8D 01 DD  STA $DD01     store in port,
. 00B3E  A9 00     LDA #$00   }  and toggle
. 00B40  8D 00 DD  STA $DD00   }  handshake          ("data sent")
. 00B43  A9 04     LDA #$04   }   line.
. 00B45  8D 00 DD  STA $DD00   }
. 00B48  A9 00     LDA #$00   }  Put 0 back in mem (i,e. clears analysed data store)
. 00B4A  91 A0     STA ($A0),Y}    (this is faster than doing it in basic)
. 00B4C  AD 0D DD  LDA $DD0D  }
. 00B4F  29 10     AND #$10   }---- wait for handshake ("data taken")
. 00B51  F0 F9     BEQ $0B4C  }
. 00B53  88        DEY              next byte
. 00B54  C0 FF     CPY #$FF
. 00B56  D0 E1     BNE $0B39
. 00B58  C6 A1     DEC $A1          next page
. 00B5A  CA        DEX
. 00B5B  D0 DC     BNE $0B39 ------- end of data send, now set up for receive
. 00B5D  A9 40     LDA #$40
. 00B5F  85 A0     STA $A0          set indirect mem addr. (1st data goes into$CC9F, then
. 00B61  A9 CC     LDA #$CC                                  $CC9E,   etc....)
. 00B63  85 A1     STA $A1
. 00B65  A0 01     LDY #$01   }
. 00B67  EA        NOP        }  optional delay before turning the port around
. 00B68  EA        NOP        }
. 00B69  88        DEY        }
. 00B6A  D0 FB     BNE $0B67  }
. 00B6C  EA        NOP
. 00B6D  A9 00     LDA #$00   }---- set PB all bits input  ("port B"of CIA)
. 00B6F  8D 03 DD  STA $DD03  }
. 00B72  A2 39     LDX #$39         want to receive $3860 bytes
. 00B74  A0 5F     LDY #$5F
. 00B76  AD 0D DD  LDA $DD0D  }
. 00B79  29 10     AND #$10   }---- wait for handshake ("data ready")
. 00B7B  F0 F9     BEQ $0B76  }
. 00B7D  AD 01 DD  LDA $DD01     read port (PB)
. 00B80  91 A0     STA ($A0),Y   save data
. 00B82  A9 00     LDA #$00   }
. 00B84  8D 00 DD  STA $DD00   }---- send "data taken" handshake
. 00B87  A9 04     LDA #$04   }
. 00B89  8D 00 DD  STA $DD00   }
. 00B8C  88        DEY              next byte
. 00B8D  C0 FF     CPY #$FF
. 00B8F  D0 E5     BNE $0B76
. 00B91  C6 A1     DEC $A1
. 00B93  CA        DEX              next page
. 00B94  D0 E0     BNE $0B76
. 00B96  68        PLA        }
. 00B97  85 A1     STA $A1    }---- restore zero page memory
. 00B99  68        PLA        }
. 00B9A  85 A0     STA $A0    }
. 00B9C  68        PLA        }---- restore bank structure
. 00B9D  8D 00 FF  STA $FF00  }
. 00BA0  58        CLI              enable interrputs
. 00BA1  60        RTS              and quit.
```

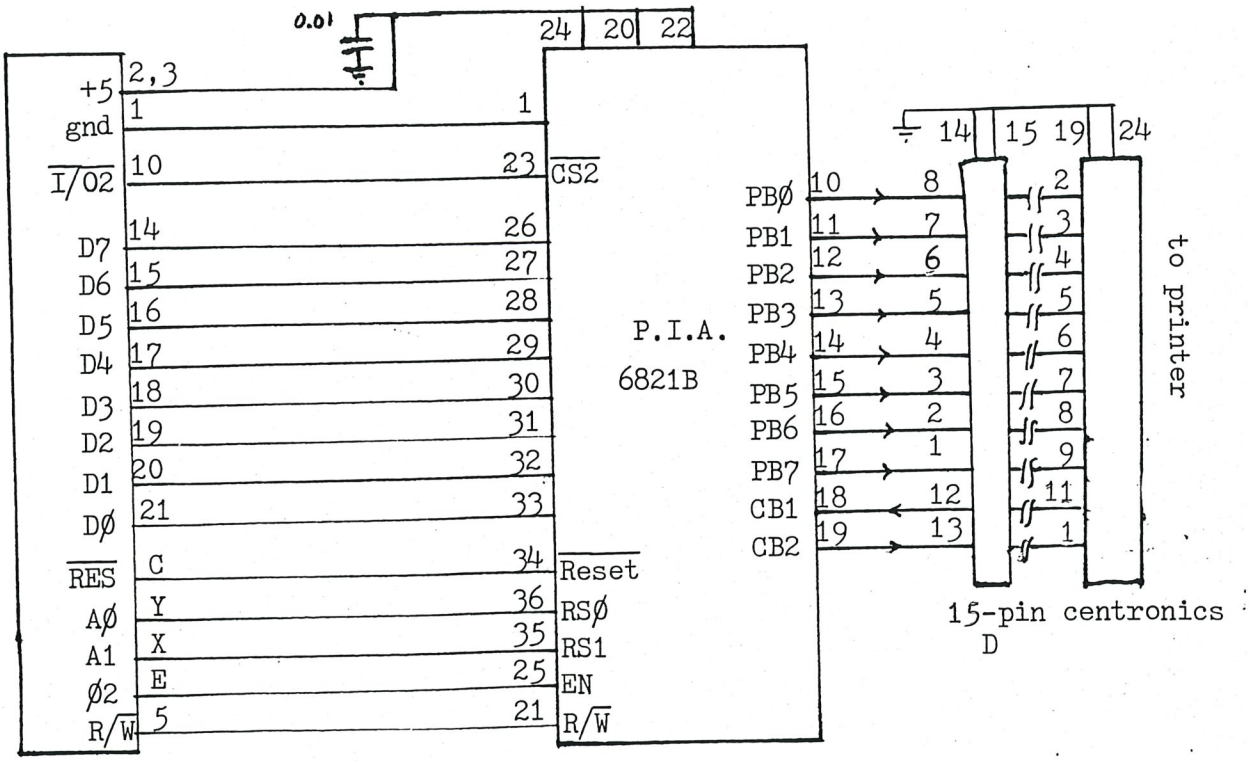
```
. 00BA2  08        PHP     save processor status        PATCH TO PRINTER ROUTINE
. 00BA3  78        SEI
. 00BA4  48        PHA       save print.char.
. 00BA5  AD 00 FF  LDA $FF00
. 00BA8  48        PHA       save bank struct.
. 00BA9  A9 3E     LDA #$3E
. 00BAB  8D 00 FF  STA $FF00  change bank struct.
. 00BAE  4C B1 93  JMP $9381  go to printer output routine
. 00BB1  68        PLA       restore old bank struct.      RETURN FROM PRINTER ROUTINE
. 00BB2  8D 00 FF  STA $FF00
. 00BB5  58        CLI       en_able interr.(?need)
. 00BB6  68        PLA       get back print character
. 00BB7  28        PLP       restore orig. processor status(incl. irq status!)
. 00BB8  4C 79 EF  JMP $EF79  and go to normal output routine

                  ← PROGRAM IN UPPER MEMORY →
. 09340  78        SEI       disable interrupts           SELECT PRINTER OUTPUT
. 09341  AE 00 FF  LDX $FF00  save old bank structure        sys37696
. 09344  A9 3E     LDA #$3E  use new bank structure
. 09346  8D 00 FF  STA $FF00
. 09349  A9 A2     LDA #$A2  change char. output vector to point at my routine
. 0934B  8D 26 03  STA $0326
. 0934E  A9 0B     LDA #$0B
. 09350  8D 27 03  STA $0327
. 09353  AD 03 DF  LDA $DF03  select DDR on PIA
. 09356  29 FB     AND #$FB
. 09358  8D 03 DF  STA $DF03
. 0935B  A9 FF     LDA #$FF  all bits o/p
. 0935D  8D 02 DF  STA $DF02
. 09360  A9 2C     LDA #$2C  select port, auto strobe(CB2) on write to port
. 09362  8D 03 DF  STA $DF03
. 09365  8E 00 FF  STX $FF00  restore old bank structure
. 09368  58        CLI
. 09369  60        RTS       and quit.
. 0936A  A9 79     LDA #$79                    DESELECT PRINTER OUTPUT
. 0936C  8D 26 03  STA $0326  restore char. output vector      sys37738
. 0936F  A9 EF     LDA #$EF
. 09371  8D 27 03  STA $0327    to its normal location
. 09374  60        RTS
. 09375  38        SEC                          DELAY
. 09376  48        PHA
. 09377  E9 01     SBC #$01      (same as 'wait' from Apple monitor)
. 09379  D0 FC     BNE $9377       delay depends on initial value of A
. 0937B  68        PLA             and clock speed. A=0,clk=1MHz gives
. 0937C  E9 01     SBC #$01        0.16 sec delay.
. 0937E  D0 F6     BNE $9376
. 09380  60        RTS
. 09381  8C 3D 94  STY $943D    need Y here,so store old   SEND CHARACTER TO PRINTER
. 09384  68        PLA          save old bank struct.
. 09385  8D 3E 94  STA $943E
. 09388  68        PLA          save print char.
. 09389  8D 3F 94  STA $943F
. 0938C  29 7F     AND #$7F     strip top bit of char.
. 0938E  C9 20     CMP #$20     is it a 'control' character?
. 09390  B0 1F     BCS $93B1    no,so go to print
. 09392  C9 0D     CMP #$0D     yes; is it a return?
. 09394  F0 04     BEQ $939A    yes,so do return
. 09396  A9 20     LDA #$20     no,so just write a space
. 09398  D0 17     BNE $93B1
. 0939A  AD 02 DF  LDA $DF02    clear rdy flag
. 0939D  A9 0D     LDA #$0D     send return char. to port
. 0939F  8D 02 DF  STA $DF02
. 093A2  A0 0C     LDY #$0C     wait max of 12 'delays' for printer rdy, then back to pro
. 093A4  2C 03 DF  BIT $DF03
. 093A7  EA        NOP          (was branch,but rdy flag not working,so do full wait
. 093A8  EA        NOP                for now)
. 093A9  20 75 93  JSR $9375    do delay loop
. 093AC  88        DEY
. 093AD  D0 F5     BNE $93A4
. 093AF  F0 08     BEQ $93B9    forced (ptr not rdy,so forget about it!)
. 093B1  8D 02 DF  STA $DF02    send char to port
. 093B4  A9 20     LDA #$20     character delay (30ms at 1MHz, or 15ms at 2MHz)
. 093B6  20 75 93  JSR $9375
. 093B9  AD 3F 94  LDA $943F    restore the stack to its original condition
. 093BC  48        PHA
. 093BD  AD 3E 94  LDA $943E
. 093C0  48        PHA
. 093C1  AC 3D 94  LDY $943D      and restore Y
. 093C4  4C B1 0B  JMP $0BB1      and jump back through low memory to basic
```



Apple interface card CPU#2

C128 User Port

Communication link connections between the Apple (CPU#2) and C128 (CPU#3)



C128 cartridge port:2x22 pin, 0.1" centres

C128 Printer Interface

SECTION V.

SECTION V.

Section V.: Conversion to 3-antenna array +odds&ends [page 119-170]

General information (W) [page 119]
List of mods required (L) [page 123]
Listing of modifed sections of program (L) [page 125]
Sample of use of "Load and Go" feature (1) [page 139]
Raw data port specs (W) [page 141]
Raw data o/p description (F) [Page 143]
Single card CPU:CPU#1 memory allocation (F) [Page 144]
Collected upgrades to CNSR & Tromso: May'93 (Pg. 161-170)

½

Program changes required for COHRTW 3-antenna Rx array

This note lists all the changes required to modify the 4 antenna programs to operate in 3-antenna (equilateral Rx array) mode. No hardware changes are required. "Note X.X" is later used to label program segment listings which follow. The areas of change are marked with a vertical bar on the left hand side of the listing. Note that in the CPU#1 listing, the port designations (C7xx) are all as for an Apple with FØØØ-FFFF EPROM. The single board computer ("SBC") port addresses are 4Øxx.

CPU#1-EPROM (changes to version CPU#1BK28)

Note 1.1: Accumulate mean mags($F76O-): the sum of magnitudes must now be divided by 24 (3 Rx - times- 8 points) instead of 32. Very hard to div-by-3, so do approx div-by-3. The whole routine will be replaced.

Note 1.2: Bit-check($FOEO-): must make Rx#4 look good when there is no Rx#4. Also must shift bit-check byte in preparation for display by CPU#2 (easier than changing CPU#2 program - because local lack of space). These just require several mods at end of routine.
[Note: may also want to indicate error in some way if using 3-ant EPROM CPU#1 with 4-ant CPU#2 program, and vice-versa]

Note 1.3: Get magnitudes and bit amps ($F240-): The only change required here is the antenna counter (2246:LDA#03).

Note 1.4: The raw data input routines($F3DA- and $F450- also send the raw data out to an external port. This is not used by the COHRTW system, but may be tapped by another system if interferometer analysis or Rx calibrations are desired. To keep the amount of output consistent with 3 antennas, receiver#4 will not be read or sent out in 3 Rx systems, but the DEY must be kept to so the memory indexing is the same. Fill deleted area with NOPs with BNE and BPL at the beginning to skip over the NOPs (faster).

Note 1.5: Whenever the program is modified , the checksum [2 bytes in FFDE(L) and FFDF(H)] must be recalculated; otherwise the ROM test will fail - although the program won't stop.

Note 1.6: Put $B1 in F8D5 (this makes the gains/info data sent out to an external system up to 192 bytes; the same as the raw data "buffer length" in the 3-ant system. [The 4-antenna "buffer length" is 256 bytes]

Note 1.7: Push from 22EO-to-end-of-routine down 2 bytes, inserting 2 DEYs (at 22EO,22E1). Also use LDX#3 and LDY#6 (same routine, a little after DEYs) instead of LDX#1 and LDY#8.

Note 1.8: Put I.D. (for raw data o/p) in F8C5: $DO for Park, $CC for London, $C3 for Calgary, $D8 for portable. NB may use $4C for Calgary, to distinguish between Park coherent system (which is also designated "C" sometimes).

A lot of other changes could be made to reduce the computation

time, but we have enough anyway. Is there a way to I.D. the EPROM: i.e. send back I.D. with status? or is there this getting too fancy?

CPU#2 (changes to TOTALCPU#2V'-D/N)

Note 2.1: Display bit check status($E20-): The only change required is E33:LDY#06

Note 2.2: Complex correlation ($E70-): This has been completely re-written, and fits in with plenty of space to spare!

Note 2.3: Auto correlation($1670-): The only change is 1670:LDA#3 (amazingly enough!). Note that the RX#4 spots in the Auto accums for the #1-matches at zero lag are still there. C128 will have to dodge them.

Note 2.4: Display worst offsets ($1B40-): the only change is 1B90:LDY#03

Note 2.5: Display I/Q or Q/I ...($1C40-): Just require 1C48:LDA#03

Note 2.6: Put site I.D. in C84 (same I.D. as in CPU#1) by LDA#____ , STA BC8A.

As with CPU#1, other changes can be made, but the program is fast enough as it is. (execution time will increase by approx 8%, because of new cross-correl routine, but it already goes 8 times as fast as needed.)

Note 2.7: clear the reference to antenn#4 from the screen display: Put $A0 in 1DEE-F1 and 1DF4

A lot more mods will be required to operate whatever data storage device we develope/buy. $2000.5EAF can be used to stack records before dumping if necessary.

CPU#3 (mods required to COHRTW-PMFCA-V18J')

Note 3.1: Line 60: change antenna spacing (m), and pair (geographic, deg E of N) directions in order 2=>1, 3=>2, 1=>3

Note 3.2: Line 287,288: put "IF I < 4"

Note 3.3: line 300: delete "+AR(4)" and "+AI(4)" and change denominators to ".../3"

Note 3.4: line 330: put .... N=3* ....

Note 3.5: line 510-560: must be re-written

****** LISTING OF MODIFIED (for 3-ants) AREAS *********

(CPU#1, CPU#2, CPU#3)

NOTE 1.2

```
2126-    F0 01        BEQ   $2129
2128-    38           SEC
2129-    66 E0        ROR   $E0
212B-    CA           DEX
212C-    D0 EF        BNE   $211D
212E-    A5 E0        LDA   $E0      <-- bit check status
2130- *  0A           ASL            shift #4 off end so it doesn't contribute
2131-    0A           ASL               (also means you don't have to do this
2132-    85 E0        STA   $E0         in CPU#2)   <-- used to check if bit-check fini.(i.e. ok)
2134-    85 FF        STA   $FF      <-- status going back to CPU#2
2136- *  60           RTS
2137-    FF           ???
```

NOTE 1.3

```
2240-    A0 FF        LDY   #$FF
2242-    A9 20        LDA   #$20
2244-    85 9F        STA   $9F
2246- *  A9 03        LDA   #$03     3-antennas
2248-    85 9E        STA   $9E
224A-    A5 9F        LDA   $9F
224C-    AA           TAX
224D-    B5 BF        LDA   $BF.X
224F-    0A           ASL
2250-    0A           ASL
2251-    0A           ASL
2252-    85 97        STA   $97
2254-    98           TYA
2255-    29 07        AND   #$07
2257-    05 97        ORA   $97
2259-    AA           TAX
225A-    B1 A9        LDA   ($A9).Y
225C-    84 9A        STY   $9A
225E-    38           SEC
225F-    FD 00 08     SBC   $0800.X
2262-    B0 04        BCS   $2268
2264-    49 FF        EOR   #$FF
2266-    69 01        ADC   #$01
2268-    86 9D        STX   $9D
226A-    A6 9A        LDX   $9A
226C-    7E 00 02     ROR   $0200.X
226F-    A6 9F        LDX   $9F
2271-    E4 E2        CPX   $E2
2273-    D0 01        BNE   $2276
2275-    48           PHA
2276-    AA           TAX
2277-    BD 00 FE     LDA   $FE00.X
227A-    85 9C        STA   $9C
227C-    BD 00 FD     LDA   $FD00.X
227F-    85 9B        STA   $9B
2281-    88           DEY
2282-    A6 9D        LDX   $9D
2284-    CA           DEX
2285-    B1 A9        LDA   ($A9).Y
2287-    84 9A        STY   $9A
2289-    38           SEC
228A-    FD 00 08     SBC   $0800.X
228D-    B0 04        BCS   $2293
228F-    49 FF        EOR   #$FF
2291-    69 01        ADC   #$01
2293-    A6 9A        LDX   $9A
2295-    7E 00 02     ROR   $0200.X
```

```
2298-    A6 9F       LDX   $9F
229A-    E4 E2       CPX   $E2
229C-    D0 01       BNE   $229F
229E-    48          PHA             I & Q of selected height pushed on
229F-    AA          TAX               stack, to later get Σ I/ Σ Q  stuff
22A0-    18          CLC                   therefor there are 6 bytes on the stack
22A1-    BD 00 FD    LDA   $FD00.X       after
22A4-    65 9B       ADC   $9B
22A6-    85 9B       STA   $9B
22A8-    BD 00 FE    LDA   $FE00.X
22AB-    65 9C       ADC   $9C
22AD-    85 9C       STA   $9C
22AF-    C9 80       CMP   #$80
22B1-    90 0C       BCC   $22BF
22B3-    A9 FF       LDA   #$FF
22B5-    D0 10       BNE   $22C7
22B7-    EA          NOP
22B8-    EA          NOP
22B9-    EA          NOP
22BA-    EA          NOP
22BB-    EA          NOP
22BC-    EA          NOP
22BD-    EA          NOP
22BE-    EA          NOP
22BF-    18          CLC
22C0-    26 9B       ROL   $9B
22C2-    26 9C       ROL   $9C
22C4-    20 A0 F7    JSR   $F7A0
22C7-    EA          NOP
22C8-    A6 9F       LDX   $9F
22CA-    18          CLC
22CB-    7D DF 08    ADC   $08DF.X
22CE-    9D DF 08    STA   $08DF.X
22D1-    90 03       BCC   $22D6
22D3-    FE BF 08    INC   $08BF.X
22D6-    A4 9A       LDY   $9A
22D8-    88          DEY
22D9-    C6 9E       DEC   $9E       next antenna.
22DB-    F0 03       BEQ   $22E0     last antenna.
22DD-    4C 4A F2    JMP   $F24A     not last,so loop.
22E0-*   88          DEY             last of ant#1-3, so skip over indexing for #4
22E1-*   88          DEY
22E2-    C6 9F       DEC   $9F       Next height
22E4-    F0 03       BEQ   $22E9
22E6-    4C 46 F2    JMP   $F246
22E9-*   A2 03       LDX   #$03      1st byte off stack is Q3, so  set X to point at accum.
22EB-*   A0 06       LDY   #$06      6 bytes on stack :Q3,I3,Q2,I2,Q1,I1 (at sel. height).
22ED-    68          PLA               get Σ(I-LTM)i ,Σ(Q-LTM)i    (running sums)
22EE-    7D 6F 09    ADC   $096F,X   L
22F1-    9D 6F 09    STA   $096F,X
22F4-    90 08       BCC   $22FE
22F6-    FE 67 09    INC   $0967,X   H1
22F9-    D0 03       BNE   $22FE
22FB-    FE 5F 09    INC   $095F,X   H2
22FE-    E8          INX
22FF-    88          DEY
2300-    D0 EB       BNE   $22ED
2302-*   60          RTS
2303-    FF          ???
```

NOTE 1.7

NOTE 1.4

```
23DA-   A0 00       LDY   #$00
23DC-   A9 00       LDA   #$00
23DE-   91 AB       STA   ($AB).Y
23E0-   88          DEY
23E1-   D0 FB       BNE   $23DE
23E3-   A0 FF       LDY   #$FF
23E5-   AD 02 C7    LDA   $C702
23E8-   91 AD       STA   ($AD).Y
23EA-   99 00 03    STA   $0300.Y
23ED-   8D 1A C7    STA   $C71A
23F0-   88          DEY
23F1-   AD 00 C7    LDA   $C700
23F4-   91 AD       STA   ($AD).Y
23F6-   99 00 03    STA   $0300.Y
23F9-   8D 1A C7    STA   $C71A
23FC-   88          DEY
23FD-   AD 06 C7    LDA   $C706
2400-   91 AD       STA   ($AD).Y
2402-   99 00 03    STA   $0300.Y
2405-   8D 1A C7    STA   $C71A
2408-   88          DEY
2409-   AD 04 C7    LDA   $C704
240C-   91 AD       STA   ($AD).Y
240E-   99 00 03    STA   $0300.Y
2411-   8D 1A C7    STA   $C71A
2414-   88          DEY
2415-   AD 0A C7    LDA   $C70A
2418-   91 AD       STA   ($AD).Y
241A-   99 00 03    STA   $0300.Y
241D-   8D 1A C7    STA   $C71A
2420-   88          DEY
2421-   AD 08 C7    LDA   $C708
2424-   91 AD       STA   ($AD).Y
2426-   99 00 13    STA   $1300.Y
2429-   8D 1A C7    STA   $C71A
242C-   88          DEY
242D-   D0 09       BNE   $2438
242F-   F0 07       BEQ   $2438
2431-   EA          NOP
2432-   EA          NOP                 Antenna#4 I
2433-   EA          NOP
2434-   EA          NOP          don't read or save or send to raw data port
2435-   EA          NOP
2436-   EA          NOP
2437-   EA          NOP
2438-   88          DEY
2439-   D0 09       BNE   $2444
243B-   F0 07       BEQ   $2444 but keep memory storage the same as 4-antennas
243D-   EA          NOP
243E-   EA          NOP
243F-   EA          NOP
2440-   EA          NOP                 Antenna#4 Q
2441-   EA          NOP
2442-   EA          NOP                   ditto
2443-   EA          NOP
2444-   88          DEY
2445-   C0 FF       CPY   #$FF
2447-   F0 03       BEQ   $244C
2449-   4C E5 23    JMP   $23E5
244C-   AD 18 C7    LDA   $C718
244F-   60          RTS

2450-   18          CLC
2451-   A0 FF       LDY   #$FF
2453-   AD 02 C7    LDA   $C702
2456-   8D 1A C7    STA   $C71A
2459-   71 AD       ADC   ($AD).Y
245B-   91 AD       STA   ($AD).Y
245D-   90 06       BCC   $2465
245F-   A9 00       LDA   #$00
2461-   71 AB       ADC   ($AB).Y
2463-   91 AB       STA   ($AB).Y
2465-   88          DEY
```

```
2466-   AD 00 C7    LDA   $C700
2469-   8D 1A C7    STA   $C71A
246C-   71 AD       ADC   ($AD).Y
246E-   91 AD       STA   ($AD).Y
2470-   90 06       BCC   $2478
2472-   A9 00       LDA   #$00
2474-   71 AB       ADC   ($AB).Y
2476-   91 AB       STA   ($AB).Y
2478-   88          DEY
2479-   AD 06 C7    LDA   $C706
247C-   8D 1A C7    STA   $C71A
247F-   71 AD       ADC   ($AD).Y
2481-   91 AD       STA   ($AD).Y
2483-   90 06       BCC   $248B
2485-   A9 00       LDA   #$00
2487-   71 AB       ADC   ($AB).Y
2489-   91 AB       STA   ($AB).Y
248B-   88          DEY
248C-   AD 04 C7    LDA   $C704
248F-   8D 1A C7    STA   $C71A
2492-   71 AD       ADC   ($AD).Y
2494-   91 AD       STA   ($AD).Y
2496-   90 06       BCC   $249E
2498-   A9 00       LDA   #$00
249A-   71 AB       ADC   ($AB).Y
249C-   91 AB       STA   ($AB).Y
249E-   88          DEY
249F-   AD 0A C7    LDA   $C70A
24A2-   8D 1A C7    STA   $C71A
24A5-   71 AD       ADC   ($AD).Y
24A7-   91 AD       STA   ($AD).Y
24A9-   90 06       BCC   $24B1
24AB-   A9 00       LDA   #$00
24AD-   71 AB       ADC   ($AB).Y
24AF-   91 AB       STA   ($AB).Y
24B1-   88          DEY
24B2-   AD 08 C7    LDA   $C708
24B5-   8D 1A C7    STA   $C71A
24B8-   71 AD       ADC   ($AD).Y
24BA-   91 AD       STA   ($AD).Y
24BC-   90 06       BCC   $24C4
24BE-   A9 00       LDA   #$00
24C0-   71 AB       ADC   ($AB).Y
24C2-   91 AB       STA   ($AB).Y
24C4-   88          DEY                    "discard"antenna#4 I,Q
24C5-*  D0 10       BNE   $24D7      quicker to branch than to run
24C7-*  F0 0E       BEQ   $24D7       through NOPs
24C9-|  EA          NOP
24CA-|  EA          NOP
24CB-|  EA          NOP
24CC-|  EA          NOP
24CD-|  EA          NOP
24CE-|  EA          NOP
24CF-|  EA          NOP
24D0-|  EA          NOP
24D1-|  EA          NOP
24D2-|  EA          NOP
24D3-|  EA          NOP
24D4-|  EA          NOP
24D5-|  EA          NOP
24D6-|  EA          NOP
24D7-   88          DEY              but keep 4-antenna memory indexing
24D8-|  D0 10       BNE   $24EA      quicker to branch ...
24DA-|  F0 0E       BEQ   $24EA
24DC-|  EA          NOP
24DD-|  EA          NOP
24DE-|  EA          NOP
24DF-|  EA          NOP
24E0-|  EA          NOP
24E1-|  EA          NOP
24E2-|  EA          NOP
24E3-|  EA          NOP
24E4-|  EA          NOP
24E5-|  EA          NOP
24E6-|  EA          NOP
24E7-|  EA          NOP
24E8-|  EA          NOP
24E9-|  EA          NOP
24EA-   88          DEY
24EB-   C0 FF       CPY   #$FF
24ED-   F0 03       BEQ   $24F2
24EF-   4C 53 F4    JMP   $F453
24F2-   AD 18 C7    LDA   $C718
24F5-   60          RTS
24F6-   FF          ???
```

NOTE 1.1

```
2760-    A2 20       LDX   #$20
2762-  * BC DF 08     LDY   $08DF,X  L
2765-  | B9 00 FC     LDA   $FC00,Y   H→L
2768-  | BC BF 08     LDY   $08BF,X  H
276B-  | 19 00 FB     ORA   $FB00,Y   L→H
276E-  | 85 9F        STA   $9F      L
2770-  | B9 00 FC     LDA   $FC00,Y  H   ÷16
2773-  | 4A           LSR                ÷2 (and save carry)
2774-  | 66 9F        ROR   $9F
2776-  | A4 9F        LDY   $9F
2778-  | B9 00 FC     LDA   $FC00,Y  H→L
277B-  | 65 9F        ADC   $9F          + 1/16
277D-  | 85 9F        STA   $9F
277F-  | 4A           LSR
2780-  | 4A           LSR
2781-  | 65 9F        ADC   $9F          + 1/4 ( )
2783-  | 7D 3F 09     ADC   $093F,X
2786-  | 9D 3F 09     STA   $093F,X
2789-  | 90 03        BCC   $278E
278B-  | FE 1F 09     INC   $091F,X
278E-  | D0 03        BNE   $2793    clear accums
2790-  | FE FF 08     INC   $08FF,X
2793-  | A9 00        LDA   #$00
2795-  | 9D DF 08     STA   $08DF,X
2798-  | 9D BF 08     STA   $08BF,X
279B-  | CA           DEX
279C-  | D0 C4        BNE   $2762
279E-  * 60          RTS
```

Accumulate mean magnitudes

same as the 4-antenna version but must div-by-24 instead of 32!

Method is to div-by-32 and mult.-by-1.333

$$== \frac{sum}{32}\left(1+\frac{1}{16}\right)\left(1+\frac{1}{4}\right)$$

$$= 1.328*\frac{sum}{32} \quad \text{(close enough)}$$

NOTE 1.8, 1.6

```
28C4-  * A9 D8       LDA   #$D8
28C6-    A2 00       LDX   #$00
28C8-    F0 06       BEQ   $28D0
28CA-    8A          TXA
28CB-    8D 1A C7    STA   $C71A
28CE-    49 FF       EOR   #$FF
28D0-    8D 1A C7    STA   $C71A
28D3-    CA          DEX
28D4-  * E0 B1       CPX   #$B1
28D6-    D0 F2       BNE   $28CA
28D8-    AD 18 C7    LDA   $C718
28DB-    A9 00       LDA   #$00
28DD-    EA          NOP
28DE-    EA          NOP
28DF-    EA          NOP
28E0-    38          SEC
28E1-    48          PHA
28E2-    E9 01       SBC   #$01
28E4-    D0 FC       BNE   $28E2
28E6-    68          PLA
```

I.D. "X" to raw data port

and then fill in the rest of the 192 ($CØ) bytes with test data

(sets # bytes "padding" to gain and info o/p)

NOTE 2.6

```
0C6B-    A9 00       LDA   #$00
0C6D-    8D FF 5E    STA   $5EFF
0C70-    A9 34       LDA   #$34
0C72-    8D 03 C5    STA   $C503
0C75-    A9 3C       LDA   #$3C
0C77-    8D 03 C5    STA   $C503
0C7A-    A9 00       LDA   #$00
0C7C-    8D 81 BC    STA   $BC81
0C7F-    A9 FF       LDA   #$FF
0C81-    8D 80 BC    STA   $BC80
0C84- *  A9 D8       LDA   #$D8
0C86-    8D 8A BC    STA   $BC8A
0C89-    60          RTS
0C8A-    FF          ???
0C8B-    FF          ???
0C8C-    FF          ???
```

CMD to CPU#1 (ØØ = normal operation of COHRTW)

Site I.D. (here = "X" for portable)
Calgary will be "C" ($C3)
London will be "L" ($CC)
Park will be "P" ($DØ) or null ($ØØ)

NOTE 2.1

```
0E20-    AD FF 5F    LDA   $5FFF
0E23-    D0 0C       BNE   $0E31
0E25- *  A0 06       LDY   #$06
0E27-    B9 69 1E    LDA   $1E69.Y
0E2A-    99 0D 07    STA   $070D.Y
0E2D-    88          DEY
0E2E-    D0 F7       BNE   $0E27
0E30-    60          RTS
0E31-    85 C7       STA   $C7
0E33- *  A0 06       LDY   #$06
0E35-    06 C7       ASL   $C7
0E37-    A9 7F       LDA   #$7F
0E39-    B0 02       BCS   $0E3D
0E3B-    A9 D1       LDA   #$D1
0E3D-    99 0D 07    STA   $070D.Y
0E40-    88          DEY
0E41-    06 C7       ASL   $C7
0E43-    A9 7F       LDA   #$7F
0E45-    B0 02       BCS   $0E49
0E47-    A9 C9       LDA   #$C9
```

display 6 bits of bit-check status
(has already been shifted by #1 so msb=#3Q)
CPU

NOTE 2.2

```
0E70-    A9 03       LDA   #$03
0E72-    85 AE       STA   $AE
0E74-    A5 C1       LDA   $C1
0E76-    85 AC       STA   $AC
0E78-    A5 C2       LDA   $C2
0E7A-    85 AD       STA   $AD
0E7C-    A0 40       LDY   #$40
0E7E-    A2 0A       LDX   #$0A
0E80-    B5 7F       LDA   $7F.X
0E82-    95 9F       STA   $9F.X
0E84-    B5 75       LDA   $75.X
0E86-    95 DF       STA   $DF.X
0E88-    CA          DEX
0E89-    D0 F5       BNE   $0E80
0E8B-    A9 FF       LDA   #$FF
0E8D-    85 B4       STA   $B4
```

3-pairs
Xcor counter

Complex Xcorrel(85msec)

completely re-written/re-organized
for 3-antennas

base address "4V3 Re" = 2V1 in this 3-antenna version

lag counter

Must do all lags for each pair separately

I2 to rolling store

Q2 " " "

zero-page antenna stores are rotated to get diff. pairs

flag -ve lags

```
0E8F-  18           CLC
0E90-  A9 00        LDA   #$00
0E92-  85 B2        STA   $B2
0E94-  A5 A7        LDA   $A7
0E96-  25 9D        AND   $9D
0E98-  AA           TAX
0E99-  BD 00 63     LDA   $6300.X
0E9C-  65 B2        ADC   $B2
0E9E-  85 B2        STA   $B2
0EA0-  A5 A6        LDA   $A6
0EA2-  25 9C        AND   $9C                    I pos'n1 vs I pos'n2
0EA4-  AA           TAX
0EA5-  BD 00 63     LDA   $6300.X
0EA8-  65 B2        ADC   $B2
0EAA-  85 B2        STA   $B2
0EAC-  A5 A5        LDA   $A5
0EAE-  25 9B        AND   $9B
0EB0-  AA           TAX
0EB1-  BD 00 63     LDA   $6300.X
0EB4-  65 B2        ADC   $B2
0EB6-  85 B2        STA   $B2
0EB8-  A5 A4        LDA   $A4
0EBA-  25 9A        AND   $9A
0EBC-  AA           TAX
0EBD-  BD 00 63     LDA   $6300.X
0EC0-  65 B2        ADC   $B2
0EC2-  85 B2        STA   $B2
0EC4-  A5 A3        LDA   $A3
0EC6-  25 99        AND   $99
0EC8-  AA           TAX
0EC9-  BD 00 63     LDA   $6300.X
0ECC-  65 B2        ADC   $B2
0ECE-  85 B2        STA   $B2
0ED0-  A5 A2        LDA   $A2
0ED2-  25 98        AND   $98
0ED4-  AA           TAX
0ED5-  BD 00 63     LDA   $6300.X
0ED8-  65 B2        ADC   $B2
0EDA-  85 B2        STA   $B2
0EDC-  A5 A1        LDA   $A1
0EDE-  25 97        AND   $97
0EE0-  AA           TAX
0EE1-  BD 00 63     LDA   $6300.X
0EE4-  65 B2        ADC   $B2
0EE6-  85 B2        STA   $B2
0EE8-  A5 A0        LDA   $A0
0EEA-  25 96        AND   $96
0EEC-  AA           TAX
0EED-  BD 00 63     LDA   $6300.X
0EF0-  65 B2        ADC   $B2
0EF2-  85 B2        STA   $B2 ---------------------
0EF4-  A5 E7        LDA   $E7
0EF6-  25 93        AND   $93
0EF8-  AA           TAX
0EF9-  BD 00 63     LDA   $6300.X
0EFC-  65 B2        ADC   $B2
0EFE-  85 B2        STA   $B2
0F00-  A5 E6        LDA   $E6
0F02-  25 92        AND   $92                    Q posn1 vs Q posn2
0F04-  AA           TAX
0F05-  BD 00 63     LDA   $6300.X
0F08-  65 B2        ADC   $B2
0F0A-  85 B2        STA   $B2
0F0C-  A5 E5        LDA   $E5
0F0E-  25 91        AND   $91
0F10-  AA           TAX
0F11-  BD 00 63     LDA   $6300.X
0F14-  65 B2        ADC   $B2
0F16-  85 B2        STA   $B2
0F18-  A5 E4        LDA   $E4 <-- Q2
0F1A-  25 90        AND   $90 <-- Q1
0F1C-  AA           TAX
0F1D-  BD 00 63     LDA   $6300.X
0F20-  65 B2        ADC   $B2
0F22-  85 B2        STA   $B2
0F24-  A5 E3        LDA   $E3
0F26-  25 8F        AND   $8F
0F28-  AA           TAX
0F29-  BD 00 63     LDA   $6300.X
0F2C-  65 B2        ADC   $B2
0F2E-  85 B2        STA   $B2
```

```
0F30- | A5 E2       LDA   $E2
0F32- | 25 8E       AND   $8E
0F34- | AA          TAX
0F35- | BD 00 63    LDA   $6300.X
0F38- | 65 B2       ADC   $B2
0F3A- | 85 B2       STA   $B2
0F3C- | A5 E1       LDA   $E1
0F3E- | 25 8D       AND   $8D
0F40- | AA          TAX
0F41- | BD 00 63    LDA   $6300.X
0F44- | 65 B2       ADC   $B2
0F46- | 85 B2       STA   $B2
0F48- | A5 E0       LDA   $E0
0F4A- | 25 8C       AND   $8C
0F4C- | AA          TAX
0F4D- | BD 00 63    LDA   $6300.X
0F50- | 65 B2       ADC   $B2
0F52- | 85 B2       STA   $B2
0F54- | 24 B4       BIT   $B4        -ve lags?
0F56- | 30 30       BMI   $0F88      YES
0F58- | A5 9F       LDA   $9F
0F5A- | 25 95       AND   $95                  zero or +ve lags, so
0F5C- | AA          TAX
0F5D- | BD 00 63    LDA   $6300.X
0F60- | 65 B2       ADC   $B2
0F62- | 85 B2       STA   $B2        2 more Ivs I
0F64- | A5 9E       LDA   $9E
0F66- | 25 94       AND   $94
0F68- | AA          TAX
0F69- | BD 00 63    LDA   $6300.X
0F6C- | 65 B2       ADC   $B2
0F6E- | 85 B2       STA   $B2
0F70- | A5 DF       LDA   $DF
0F72- | 25 8B       AND   $8B
0F74- | AA          TAX              and 2 more Q vs Q
0F75- | BD 00 63    LDA   $6300.X
0F78- | 65 B2       ADC   $B2
0F7A- | 85 B2       STA   $B2
0F7C- | A5 DE       LDA   $DE
0F7E- | 25 8A       AND   $8A
0F80- | AA          TAX
0F81- | BD 00 63    LDA   $6300.X
0F84- | 65 B2       ADC   $B2
0F86- | 85 B2       STA   $B2
0F88- | 71 AC       ADC   ($AC).Y
0F8A- | 91 AC       STA   ($AC).Y
0F8C- | 90 08       BCC   $0F96
0F8E- | 88          DEY
0F8F- | A9 00       LDA   #$00
0F91- | 71 AC       ADC   ($AC).Y    accumulate II+QQ
0F93- | 91 AC       STA   ($AC).Y
0F95- | C8          INY              --back to H-byte of accum
0F96- | A5 AC       LDA   $AC
0F98- | 38          SEC
0F99- | E9 40       SBC   #$40       Now point to Im "4v3"
0F9B- | 85 AC       STA   $AC
0F9D- | B0 02       BCS   $0FA1
0F9F- | C6 AD       DEC   $AD
0FA1- | 18          CLC
0FA2- | A9 00       LDA   #$00
0FA4- | 85 B2       STA   $B2     +ve sum of QI
0FA6- | 85 B7       STA   $B7     sum of -ve IQ
0FA8- | A5 E7       LDA   $E7     Qposn2
0FAA- | 25 9D       AND   $9D     Iposn1
0FAC- | AA          TAX
0FAD- | BD 00 63    LDA   $6300.X                QI-IQ
0FB0- | 65 B2       ADC   $B2
0FB2- | 85 B2       STA   $B2
0FB4- | A5 E6       LDA   $E6
0FB6- | 25 9C       AND   $9C
0FB8- | AA          TAX
0FB9- | BD 00 63    LDA   $6300.X
0FBC- | 65 B2       ADC   $B2
0FBE- | 85 B2       STA   $B2
0FC0- | A5 E5       LDA   $E5
0FC2- | 25 9B       AND   $9B
0FC4- | AA          TAX
0FC5- | BD 00 63    LDA   $6300.X
0FC8- | 65 B2       ADC   $B2
0FCA- | 85 B2       STA   $B2
0FCC- | A5 E4       LDA   $E4
0FCE- | 25 9A       AND   $9A
0FD0- | AA          TAX
0FD1- | BD 00 63    LDA   $6300.X
0FD4- | 65 B2       ADC   $B2
0FD6- | 85 B2       STA   $B2
```

```
0FD8-  A5 E3      LDA   $E3
0FDA-  25 99      AND   $99
0FDC-  AA         TAX
0FDD-  BD 00 63   LDA   $6300,X
0FE0-  65 B2      ADC   $B2
0FE2-  85 B2      STA   $B2
0FE4-  A5 E2      LDA   $E2
0FE6-  25 98      AND   $98
0FE8-  AA         TAX
0FE9-  BD 00 63   LDA   $6300,X
0FEC-  65 B2      ADC   $B2
0FEE-  85 B2      STA   $B2
0FF0-  A5 E1      LDA   $E1
0FF2-  25 97      AND   $97
0FF4-  AA         TAX
0FF5-  BD 00 63   LDA   $6300,X
0FF8-  65 B2      ADC   $B2
0FFA-  85 B2      STA   $B2
0FFC-  A5 E0      LDA   $E0
0FFE-  25 96      AND   $96
1000-  AA         TAX
1001-  BD 00 63   LDA   $6300,X
1004-  65 B2      ADC   $B2
1006-  85 B2      STA   $B2
1008-  A5 A7      LDA   $A7     Iposn2
100A-  25 93      AND   $93     Qposn1        nb."posn1"  means from the antenna #1 store
100C-  AA         TAX                           since the ant.data are rotated, this
100D-  BD 00 63   LDA   $6300,X                 doesn't mean ant#1 always
1010-  65 B7      ADC   $B7
1012-  85 B7      STA   $B7
1014-  A5 A6      LDA   $A6
1016-  25 92      AND   $92
1018-  AA         TAX
1019-  BD 00 63   LDA   $6300,X
101C-  65 B7      ADC   $B7
101E-  85 B7      STA   $B7
1020-  A5 A5      LDA   $A5
1022-  25 91      AND   $91
1024-  AA         TAX
1025-  BD 00 63   LDA   $6300,X
1028-  65 B7      ADC   $B7
102A-  85 B7      STA   $B7
102C-  A5 A4      LDA   $A4
102E-  25 90      AND   $90
1030-  AA         TAX
1031-  BD 00 63   LDA   $6300,X
1034-  65 B7      ADC   $B7
1036-  85 B7      STA   $B7
1038-  A5 A3      LDA   $A3
103A-  25 8F      AND   $8F
103C-  AA         TAX
103D-  BD 00 63   LDA   $6300,X
1040-  65 B7      ADC   $B7
1042-  85 B7      STA   $B7
1044-  A5 A2      LDA   $A2
1046-  25 8E      AND   $8E
1048-  AA         TAX
1049-  BD 00 63   LDA   $6300,X
104C-  65 B7      ADC   $B7
104E-  85 B7      STA   $B7
1050-  A5 A1      LDA   $A1
1052-  25 8D      AND   $8D
1054-  AA         TAX
1055-  BD 00 63   LDA   $6300,X
1058-  65 B7      ADC   $B7
105A-  85 B7      STA   $B7
105C-  A5 A0      LDA   $A0
105E-  25 8C      AND   $8C
1060-  AA         TAX
1061-  BD 00 63   LDA   $6300,X
1064-  65 B7      ADC   $B7
1066-  85 B7      STA   $B7
1068-  24 B4      BIT   $B4        -ve lags?
106A-  30 30      BMI   $109C    YES
106C-  A5 DF      LDA   $DF            + lags, so
106E-  25 95      AND   $95
1070-  AA         TAX
1071-  BD 00 63   LDA   $6300,X
1074-  65 B2      ADC   $B2          2 more Qposn2 vs I posn1
1076-  85 B2      STA   $B2
```

```
1078-  A5 DE       LDA   $DE
107A-  25 94       AND   $94
107C-  AA          TAX
107D-  BD 00 63    LDA   $6300,X
1080-  65 B2       ADC   $B2
1082-  85 B2       STA   $B2
1084-  A5 9F       LDA   $9F
1086-  25 8B       AND   $8B
1088-  AA          TAX
1089-  BD 00 63    LDA   $6300,X
108C-  65 B7       ADC   $B7          and 2 more I posn2 vs Q posn1
108E-  85 B7       STA   $B7
1090-  A5 9E       LDA   $9E
1092-  25 8A       AND   $8A
1094-  AA          TAX
1095-  BD 00 63    LDA   $6300,X
1098-  65 B7       ADC   $B7
109A-  85 B7       STA   $B7
109C-  38          SEC
109D-  B1 AC       LDA   ($AC),Y  L byte of Im accum.
109F-  E5 B7       SBC   $B7
10A1-  B0 10       BCS   $10B3        -I2Q1 + Q2I1
10A3-  65 B2       ADC   $B2
10A5-  91 AC       STA   ($AC),Y
10A7-  B0 19       BCS   $10C2
10A9-  88          DEY               point to H-byte of accum
10AA-  B1 AC       LDA   ($AC),Y
10AC-  E9 00       SBC   #$00
10AE-  91 AC       STA   ($AC),Y
10B0-  C8          INY               back to L byte of accum
10B1-  D0 0F       BNE   $10C2       forced branch
10B3-  18          CLC
10B4-  65 B2       ADC   $B2
10B6-  91 AC       STA   ($AC),Y  L
10B8-  90 08       BCC   $10C2
10BA-  88          DEY
10BB-  A9 00       LDA   #$00
10BD-  71 AC       ADC   ($AC),Y  H
10BF-  91 AC       STA   ($AC),Y
10C1-  C8          INY               back to L
10C2-  24 B4       BIT   $B4         -ve lags?
10C4-  10 4A       BPL   $1110       No
10C6-  18          CLC
10C7-  26 A9       ROL   $A9
10C9-  26 A8       ROL   $A8           -ve lags,so roll 10 bytes :
10CB-  26 A7       ROL   $A7
10CD-  26 A6       ROL   $A6
10CF-  26 A5       ROL   $A5              shift I sequence by 1 bit
10D1-  26 A4       ROL   $A4
10D3-  26 A3       ROL   $A3
10D5-  26 A2       ROL   $A2
10D7-  26 A1       ROL   $A1
10D9-  26 A0       ROL   $A0
10DB-  18          CLC
10DC-  26 E9       ROL   $E9
10DE-  26 E8       ROL   $E8
10E0-  26 E7       ROL   $E7
10E2-  26 E6       ROL   $E6
10E4-  26 E5       ROL   $E5
10E6-  26 E4       ROL   $E4              shift Q sequence by 1 bit
10E8-  26 E3       ROL   $E3
10EA-  26 E2       ROL   $E2
10EC-  26 E1       ROL   $E1
10EE-  26 E0       ROL   $E0
10F0-  88          DEY
10F1-  88          DEY
10F2-  C0 20       CPY   #$20        next lag = 0-lag ??
10F4-  D0 0C       BNE   $1102
10F6-  A9 00       LDA   #$00     Yes,so mark that next lag is 0 or +ve
10F8-  85 B4       STA   $B4
10FA-  85 9E       STA   $9E
10FC-  85 9F       STA   $9F
10FE-  85 DE       STA   $DE
1100-  85 DF       STA   $DF
1102-  A5 AC       LDA   $AC          move accum. pointer back to "4V3 Re"
1104-  18          CLC
1105-  69 40       ADC   #$40
1107-  85 AC       STA   $AC
1109-  90 02       BCC   $110D
110B-  E6 AD       INC   $AD
110D-  4C 8F 0E    JMP   $0E8F    ----next lag
```

```
1110-  88          DEY             +ve or zero lags
1111-  88          DEY
1112-  F0 38       BEQ   $114C      end of one cross-correlation
1114-  18          CLC
1115-  26 A7       ROL   $A7
1117-  26 A6       ROL   $A6
1119-  26 A5       ROL   $A5          shift I sequence  by 1 bit
111B-  26 A4       ROL   $A4
111D-  26 A3       ROL   $A3
111F-  26 A2       ROL   $A2
1121-  26 A1       ROL   $A1            +ve lags
1123-  26 A0       ROL   $A0
1125-  26 9F       ROL   $9F
1127-  26 9E       ROL   $9E
1129-  18          CLC
112A-  26 E7       ROL   $E7
112C-  26 E6       ROL   $E6
112E-  26 E5       ROL   $E5          shift Q sequence by 1 bit
1130-  26 E4       ROL   $E4
1132-  26 E3       ROL   $E3
1134-  26 E2       ROL   $E2
1136-  26 E1       ROL   $E1
1138-  26 E0       ROL   $E0
113A-  26 DF       ROL   $DF
113C-  26 DE       ROL   $DE
113E-  A5 AC       LDA   $AC        go back to "Re of 2V1" for next lag
1140-  18          CLC
1141-  69 40       ADC   #$40
1143-  85 AC       STA   $AC
1145-  90 02       BCC   $1149
1147-  E6 AD       INC   $AD
1149-  4C 8F 0E    JMP   $0E8F
114C-  A5 AC       LDA   $AC        move accum pointer down to "3V2 Re"
114E-  38          SEC
114F-  E9 40       SBC   #$40
1151-  85 AC       STA   $AC
1153-  B0 02       BCS   $1157
1155-  C6 AD       DEC   $AD
1157-  A2 14       LDX   #$14
1159-  B5 89       LDA   $89.X
115B-  48          PHA                rotate antenna zero page bit-amp stores
115C-  B5 75       LDA   $75.X        so can get next ant-pair with same program
115E-  95 89       STA   $89.X
1160-  B5 61       LDA   $61.X
1162-  95 75       STA   $75.X
1164-  68          PLA
1165-  95 61       STA   $61.X
1167-  CA          DEX
1168-  D0 EF       BNE   $1159
116A-  C6 AE       DEC   $AE        next Xcor
116C-  F0 03       BEQ   $1171
116E-  4C 7C 0E    JMP   $0E7C
1171-  60          RTS
1172-  FF          ???
1173-  FF          ???
1174-  FF          ???
1175-  FF          ???
1176-  FF          ???
1177-  FF          ???
1178-  FF          ???
1179-  FF          ???
117A-  FF          ???
117B-  FF          ???
117C-  FF          ???
117D-  FF          ???
117E-  FF          ???
117F-  FF          ???
1180-  85 B2       STA   $B2        NOTE:
1182-  A5 E0       LDA   $E0          the rest of the old Xcor program has been
1184-  25 78       AND   $78            left where it sits.... so space $1180- $15AE is free!
1186-  AA          TAX                          .
1187-  BD 00 63    LDA   $6300.X                .
118A-  65 B2       ADC   $B2                    .
118C-  85 B2       STA   $B2                    .
118E-  24 B4       BIT   $B4                    .
1190-  30 30       BMI   $11C2                  .
1192-  A5 9F       LDA   $9F
```

NOTE 2.3

```
1670-* A9 03      LDA   #$03        3-antennas          Auto Correl (< 46ms)
1672-   85 B4     STA   $B4
1674-   A9 89     LDA   #$89
1676-   85 AA     STA   $AA
1678-   A2 08     LDX   #$08
167A-   B5 95     LDA   $95.X
167C-   95 9F     STA   $9F.X
167E-   B5 8B     LDA   $8B.X
1680-   95 DF     STA   $DF.X
1682-   CA        DEX
1683-   D0 F5     BNE   $167A
1685-   A9 00     LDA   #$00
1687-   85 9E     STA   $9E
1689-   85 9F     STA   $9F
168B-   85 DE     STA   $DE
168D-   85 DF     STA   $DF
168F-   A0 18     LDY   #$18
1691-   A5 C3     LDA   $C3
1693-   85 AC     STA   $AC
1695-   A5 C4     LDA   $C4
```

NOTE 2.4

```
1B85-   EA        NOP
1B86-   A9 B6     LDA   #$B6
1B88-   85 A0     STA   $A0
1B8A-   A9 05     LDA   #$05
1B8C-   85 A1     STA   $A1
1B8E-   A0 08     LDY   #$08
1B90-* A9 03      LDA   #$03        antenna counter (3-antennas)
1B92-   85 A3     STA   $A3
1B94-   B9 D7 60  LDA   $60D7.Y
1B97-   48        PHA
1B98-   88        DEY
1B99-   B9 D7 60  LDA   $60D7.Y
1B9C-   48        PHA
1B9D-   88        DEY
1B9E-   84 A2     STY   $A2
1BA0-   A0 05     LDY   #$05
1BA2-   A9 02     LDA   #$02
1BA4-   85 A4     STA   $A4
1BA6-   EA        NOP
1BA7-   EA        NOP
```

NOTE 2.5

```
1C40-   A9 AD     LDA   #$AD
1C42-   85 A0     STA   $A0
1C44-   A9 05     LDA   #$05
1C46-   85 A1     STA   $A1
1C48-* A9 03      LDA   #$03    Rx (screen line) counter : 3-antennas
1C4A-   85 A2     STA   $A2
1C4C-   A0 07     LDY   #$07
1C4E-   A9 A0     LDA   #$A0
1C50-   91 A0     STA   ($A0).Y
1C52-   88        DEY
1C53-   10 FB     BPL   $1C50
1C55-   A0 07     LDY   #$07
1C57-   A6 A2     LDX   $A2
1C59-   BD 99 60  LDA   $6099.X
1C5C-   10 0C     BPL   $1C6A
1C5E-   A0 02     LDY   #$02
1C60-   C9 64     CMP   #$64
1C62-   D0 06     BNE   $1C6A
1C64-   A9 B1     LDA   #$B1
1C66-   91 A0     STA   ($A0).Y
```

part of fixed screen display

```
1DE0- BA 00 A8 06 A3 B3 A1 00
1DE8- B3 06 A1 00 28 07 A0 A0
1DF0- A0 00 33 07 A0 00 3F 07
1DF8- C3 D0 D5 A3 B2 BA 00 54
```

put blanks instead of "#4!"

and instead of "!"

Mods to C128PMFCA-V18J' to get 3-antenna version: C128PMFCA-3ANTS-V18J'

CM, Feb22'90

Title line (line 10) plus the following:

spacing and 2=>1, 3=>2, 1=>3 direction for main array (Park) test

```
60 D=270:P=57.29578:P1= 0/P:P2=240/P:P3=120/P:PZ=1.5708:TT=256:TU=TT*TT
```

```
287 IFI<4THENIFAR(I)=NNORX=NNTHENPRINT" CONST SIG":POKEIG,PEEK(IG)+48:GOTO2000
288 IFI<4THENIFAR(I)=0ORX=0THENPRINT" CONST SIG":POKEIG,PEEK(IG)+48:GOTO2000
```

```
300 SR=(AR(1)+AR(2)+AR(3))/NN/3:SI=(AI(1)+AI(2)+AI(3))/NN/3
```

330 FORI=5TO10:N=3*(NN-I+4): .....

get combined pair means and variences

```
510 MR=AR(1)/NN:MI=AI(1)/NN:VR=MR*(1-MR)+MI*(1-MI)
530 FORI=1TO3:K=I+1:IFK>3THENK=1
540 X=AR(K)/NN:Y=AI(K)/NN:S0=X*(1-X)+Y*(1-Y)
550 MN(1,I)=X*MR+Y*MI:MN(2,I)=X*MI-Y*MR:V(I)=SQR(S0*VR)
560 VR=S0:MR=X:MI=Y:NEXTI
```

!!

reverse signs!!!

indicate site I.D. on printer output

sense of x-prods in CPU#2

$I_2I_1+Q_2Q_1$	$Q_2I_1-I_2Q_1$
$I_3I_2+Q_3Q_2$	$Q_3I_2-I_3Q_2$
$I_1I_3+Q_1Q_3$	$Q_1I_3-I_1Q_3$
Re	Im

```
2080 PRINT" COHRTW! #  N  E ! #  N  E ! #  N  E !";
```

Robsrt
Sylvan
Tromsö

****** SAMPLE PROGRAM TO SHOW LOAD&GO USAGE ******

```
2000-   A9 3C       LDA   #$3C     }  NMI control = high
2002-   8D 03 C5    STA   $C503    }
2005-   A9 38       LDA   #$38        select DDR on port
2007-   8D 01 C5    STA   $C501
200A-   A9 FF       LDA   #$FF        all output
200C-   8D 00 C5    STA   $C500
200F-   A9 3C       LDA   #$3C     }  handshake (CA2) = high
2011-   8D 01 C5    STA   $C501    }
2014-   A9 34       LDA   #$34     ]  L send NMI   to CPU#1
2016-   8D 03 C5    STA   $C503    ]
2019-   A9 3C       LDA   #$3C     ]  H
201B-   8D 03 C5    STA   $C503    ]
201E-   EA          NOP
201F-   EA          NOP
2020-   EA          NOP
2021-   EA          NOP
2022-   EA          NOP
2023-   EA          NOP
2024-   EA          NOP
2025-   EA          NOP
2026-   EA          NOP
2027-   EA          NOP
2028-   A9 00       LDA   #$00     ]
202A-   85 A0       STA   $A0      ]     initial address of "data" to send to CPU#1
202C-   A9 21       LDA   #$21     ]
202E-   85 A1       STA   $A1      ]
2030-   A0 04       LDY   #$04        "data" has length of program embedded in it,so
2032-   B1 A0       LDA   ($A0).Y     dig it out (here assumed to be less than 1 page!)
2034-   18          CLC
2035-   69 30       ADC   #$30        + header length = total length to send
2037-   AA          TAX               Use X as counter
2038-   EA          NOP
2039-   EA          NOP
203A-   EA          NOP
203B-   EA          NOP
203C-   A0 00       LDY   #$00
203E-   AD 01 C5    LDA   $C501    ]  wait for "ready to receive" signal from CPU#1
2041-   10 FB       BPL   $203E    ]
2043-   AD 00 C5    LDA   $C500       clear handshake signal
2046-   B1 A0       LDA   ($A0).Y     get data
2048-   8D 00 C5    STA   $C500       and send it,
204B-   A9 34       LDA   #$34     ]  with handshake
204D-   8D 01 C5    STA   $C501    ]
2050-   A9 3C       LDA   #$3C     ]
2052-   8D 01 C5    STA   $C501    ]
2055-   C8          INY               next data address
2056-   CA          DEX               next byte
2057-   D0 E5       BNE   $203E
2059-   4C 69 FF    JMP   $FF69       jump to monitor
205C-   FF          ???               CMD to CPU#1 (non-zero = "load program and go")
                                                   (zero = normal COHRTW operation)
2000- A9 3C 8D 03 C5 A9 38 8D         start addr. for program in CPU#1 ($2ØØ!)
*2100.2128
                 H  L  H  L           length of program(not including header!) = 34 bytes=$22 bytes
2100- 01 02 00 00 22 00 00 00                 HEADER DATA  ( 48 bytes)
2108- 00 00 00 00 00 00 00 00
2110- 00 00 00 00 00 00 00 00
2118- 00 00 00 00 00 00 00 00
2120- 00 00 00 00 00 00 00 00
2128- 00 00 00 00 00 00 00 FF         comm link check bytes
*2130L

2130-   A9 01       LDA   #$01           program (here stored in CPU#1 starting at $2ØØ)
2132-   85 02       STA   $02                    (because of param. in header)
2134-   20 00 FA    JSR   $FA00       approx 1 sec delay
2137-   A5 02       LDA   $02
2139-   20 40 F5    JSR   $F540       LED on                Program turns on lsb LED,then off;
213C-   20 00 FA    JSR   $FA00       delay                 then 2 lsb LEDs, then off, etc...
213F-   A5 02       LDA   $02
2141-   20 50 F5    JSR   $F550       LED off
2144-   A5 02       LDA   $02            .....should have CLC before roll, then 1 led on
2146-   2A          ROL                        at a time.
2147-   90 03       BCC   $214C
2149-   A9 01       LDA   #$01
214B-   18          CLC
214C-   85 02       STA   $02
214E-   D0 E4       BNE   $2134          program length:$213Ø-2151 = $22 bytes
2150-   F0 E2       BEQ   $2134
2152-   FF          ???
```

CPU#2Program to test "load and Go" feature of CPU#1BK25'' EPROM

3 or 4 ants!!

****

Raw data port specs

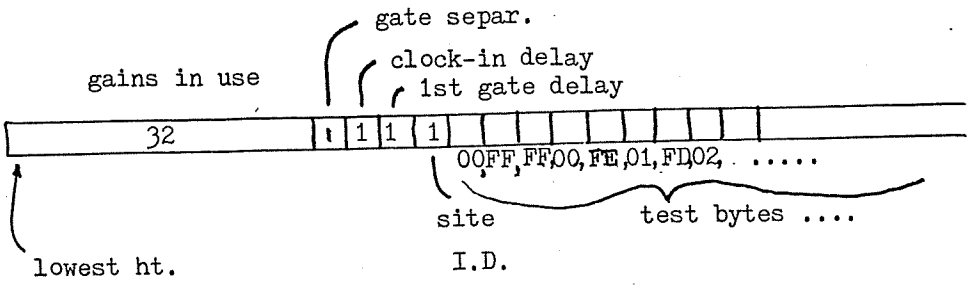
******* 1st DATA SENT TO EXTERNAL "RAW DATA" PORT **********

PROBLEMS

CLASS

NAME

DATE



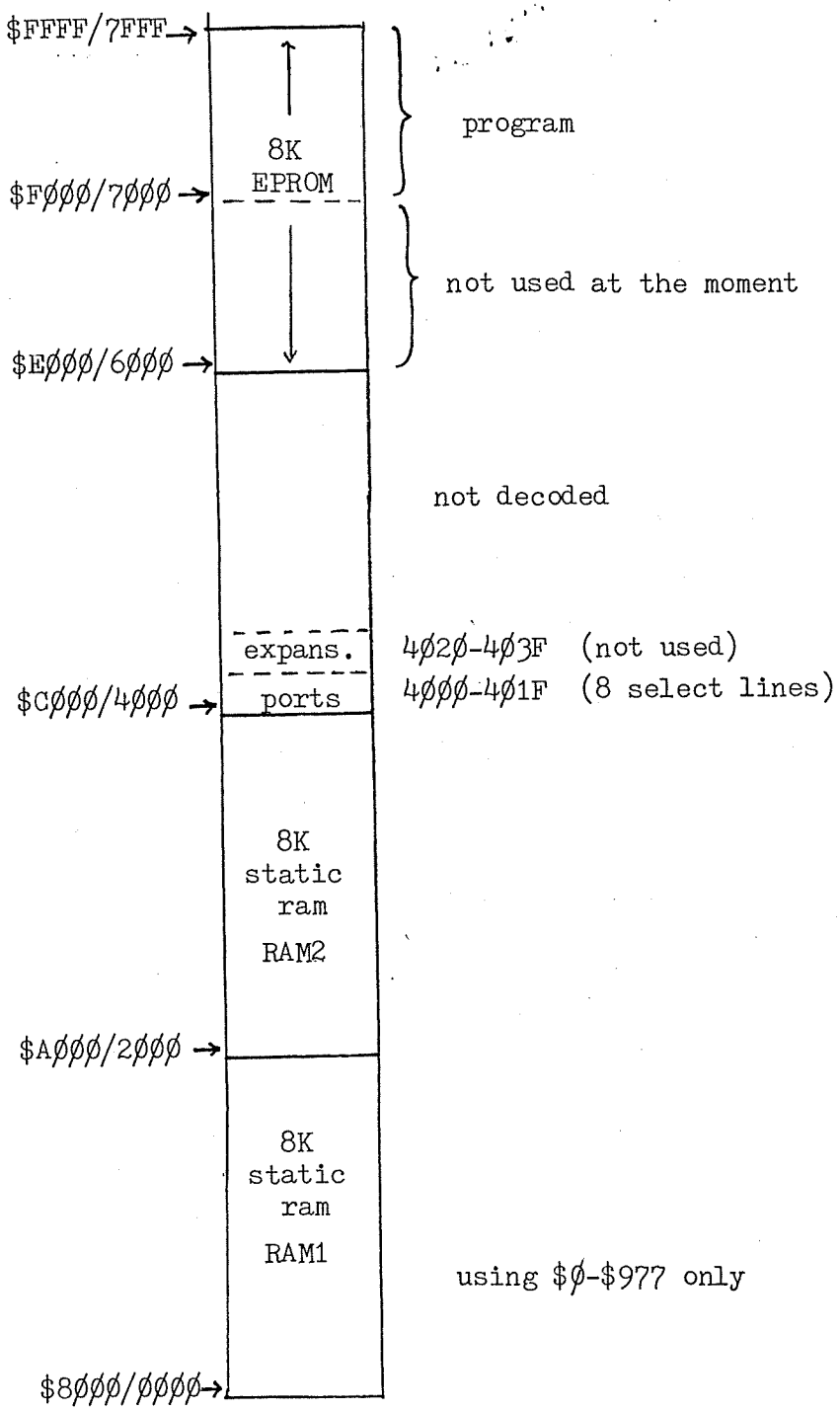
First "buffer" sent to raw data port: length=256 for 4 antennas system

: length=192 for 3 antennas system

PROBLEMS

NAME | DATE

Memory allocation for single card computer (CPU#1)



gap: page 145-160

(left for future expansion !)

MAY 12 1993

CNSR and Tromsø Upgrades

General:

The following notes are not intended to document changes down to the last byte, just to describe new features (beyond the Jan 24,1992 version of the yellow report)- e.g. system tests, extended ranges for system parameters, etc. The most advanced programs (May 9'93) are CPU#1BK29T'(2000.2FFF), TOTCPU#2W-3ACNSR6T(900.203F), and ROBSART/SYLVAN L. C128- V18J'. As in a previous section, program segments are labelled with the "note numbers" given below.

Rx Interface(CPU#1):

Note 1.1: We can now do any amount of integration - a full divide routine was added (the previous program only did powers of two). Now we can vary Δt at will, by changing the amount of integration (along with the record length- which **still** must be divisible by 8!- viz the "number of sends" is specified, each send has 8 bit-amps)

Note 1.2: CPU#1 now saves the signals for each Rx for the "selected height" in the final data store going to CPU#2.[This is displayed on the monitor by CPU#2 to give a rough estimate of different antenna sensitivites - so they can be equalized with attenuators.]

CPU#2 (Apple):

Note 2.0:There was (and **is!**) a bug in all CPU#2 programs to date (including version 5R,5S Robsart and Sylvan L.). This bug **doesn't** cause any error if the record length is "$40 sends", so as long as we don't change the record length, the CNSR programs are "OK". Tromso has been fixed because we want to be able to vary the amount of integration/record length at will. In fact probably we will run Tromso at length=$40 and 24 pulse integration (50 Hz prf), Δt=0.48s, because the new open wire antennas are farther apart (164 m) than the loops were (124m).[With the loops we had 3x6 pulse integration @50Hz and Δt=0.36s]

Note 2.1: display signals for each Rx at selected height (which is automatically selected **in** CPU#1)

Note 2.2: Break out of stuck C128 comm. link. If the Apple finds the C128 not ready, then it sends a continuous string of $80's until the C128 "disappears" into BASIC (detected by the delay in C128 response). The C128 tests location $9440 (high byte of Σ(QI-IQ) at lag=12); if it finds $80, then it knows it is being broken out - the only other way to get $80 is with an extremely LONG record. See Note 3.1 below for its response.

Note 2.3: display CPU#1 check bytes on #1 or #3 comm link error (this was a quick diagnostic addition, but left in anyway)

Note 2.4: Save signals and "selected height" in D/T store (going to C128 and Tape) --- why not; we have room, and it gives an extra diagnostic for the Tromso system.

Note 2.5: save the seconds, $S_{10}*16+S_1$, of the last time in the record in the date/time store (goes onto tape). This is intended to monitor the synch rate at Tromso, since we are taking our synch from the Digi-sonde and just **assuming** it is 50Hz.

Note 2.6: A simple memory test has been added. The bottom 16K is already checked by the program checksum (also zero page is the most important page in the Apple, so if it isn't working, maybe nothing will work!); the remaining two "banks" are $4000-7FFF and $8000-BFFF (which each consist of a row of 8 16Kx1 4116's). Actually these memories are "tested" by the C128 program, since errors could cause BASIC errors - which will appear on the printer. Anyway, a test was added which loads and compares froma set of memory addresses spanning $4000-$BFFF, and if it finds a change, displays the chip "ROW" and bit # of the **first** bad bit (#0=left-most 4116/9016 .. different makers have different names!). The "TOP" row is the one (parallel to and) furthest from the keyboard. The "MID" row is the middle row ...!

Note 2.7: Tape drive variations: The CNSR sites (Robsart/Sylvan L.) use DDC Pertec Bandstor 9-track 1600 bpi drives - these have reliable backspace and re-

read functions, and can also be put on line whether or not the tape drive interface has been initialized. The COHRTW and Tromso have 800 bpi drives; programming for Tromso is shown (i.e. no backspace or re-read). The tape drive interface is initialized at the start of CPU#2 program by saving a return address and jumping to the tape drive EPROM (this effort is necessary because the EPROM doesn't have a specific init. routine setup for external use).

Note 2.8: Some parameters may be different for Tromso - e.g. the amount of coherent integration and the "record length" (in terms of number of "records" received by CPU#2 from CPU#1, not necessarily in time), because the antenna spacing and Tx rate is different. The listing is not **final** for Tromso, it just shows where the parameters are.

C128 (CPU#3):

Note 3.1: An error trap was added - so any error sends the C128 to a "subroutine" which turns on the printer, prints out the type of error and line number it ocurred in. Two errors are **created** on purpose, in line 145 and line146. An error in line 145 indicates that it has found $80 in $9440- so the Apple was breaking it out, not sending data. An error in line 146 means at least one of the $00,$FF comm link check bytes is wrong. After printing the message, a ~ 2 min time wasting loop is performed, and then it goes back to look for new data. [Note: if the Apple finds the C128 not ready anytime, it breaks it out, but also resets its "1st record" counter - so that the next C128 output is discarded, as if the system were just started.]

Note 3.2: A "#rec/day" printer o/p (at UT=08hr, just after the titles) is added so that comm link troubles (which finally fix themselves) can be monitored - usually a comm link error (CPU#2<->#1 at least) will hang the system for at least 1 record, so that instead of 288 per day, there will 287 for example.

NOTE 1.1

```
2140-  A0 00     LDY  #$00
2142-  B1 A7     LDA  ($A7),Y
2144-  85 93     STA  $93      H
2146-  B1 A9     LDA  ($A9),Y
2148-  85 94     STA  $94      L
214A-  A5 E6     LDA  $E6      =n
214C-  85 91     STA  $91
214E-  A9 00     LDA  #$00
2150-  91 A7     STA  ($A7),Y  clear H-byte for next integr.
2152-  85 90     STA  $90
2154-  85 92     STA  $92
2156-  A2 08     LDX  #$08     start (H,L) ÷ n
2158-  18        CLC
2159-  66 91     ROR  $91
215B-  66 92     ROR  $92
215D-  A5 94     LDA  $94
215F-  38        SEC
2160-  E5 92     SBC  $92
2162-  85 95     STA  $95
2164-  A5 93     LDA  $93
2166-  E5 91     SBC  $91
2168-  90 06     BCC  $2170
216A-  85 93     STA  $93
216C-  A5 95     LDA  $95
216E-  85 94     STA  $94
2170-  26 90     ROL  $90
2172-  CA        DEX
2173-  D0 E3     BNE  $2158
2175-  A5 90     LDA  $90      'result
2177-  91 A9     STA  ($A9),Y  kept in 'L' byte
2179-  88        DEY           next avg.
217A-  D0 C6     BNE  $2142
217C-  60        RTS
217D-  FF        ???
217E-  FF        ???
217F-  FF        ???
```

Get mean signals

here use full divide, so any amount of integration (n=$E6) can be used)

(THIS IS A "COPY" of THE DIVIDE AT 27E$, WHICH IS USED from 'ANALYSIS'.)

93	94
91	0

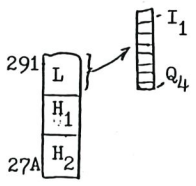
shift and subtract

NOTE 1.2

```
2334-  10 0B        BPL  $2341
2336-  20 30 F6     JSR  $F630    go to save Σ|I| Σ|Q| in final data store to #2
2339-  20 30 F8     JSR  $F830                (to get "signal" .vs. Rx)
233C-  4C 71 F3     JMP  $F371
233F-  EA           NOP
2340-  EA           NOP
2341-  A5 A3        LDA  $A3
2343-  F0 09        BEQ  $234E
2345-  20 DA F3     JSR  $F3DA
2348-  A9 00        LDA  #$00
234A-  85 A3        STA  $A3
234C-  F0 03        BEQ  $2351
234E-  20 50 F4     JSR  $F450
2351-  C6 A6        DEC  $A6
2353-  D0 1C        BNE  $2371
2355-  A5 A5        LDA  $A5
2357-  F0 0B        BEQ  $2364
2359-  A5 FE        LDA  $FE
235B-  09 01        ORA  #$01
235D-  85 FE        STA  $FE
*2630L
```

& Q/I, I/Q calculation

```
2630-  A0 18        LDY  #$18
2632-  B9 5F 09     LDA  $095F,Y
2635-  99 79 02     STA  $0279,Y  save in final param store
2638-  88           DEY           going to CPU#2
2639-  D0 F7        BNE  $2632
263B-  EA           NOP
263C-  EA           NOP
263D-  EA           NOP
263E-  EA           NOP
263F-  EA           NOP
2640-  A9 04        LDA  #$04
2642-  85 EC        STA  $EC
2644-  A0 08        LDY  #$08
2646-  B9 5F 09     LDA  $095F,Y
2649-  88           DEY
264A-  D9 5F 09     CMP  $095F,Y
```

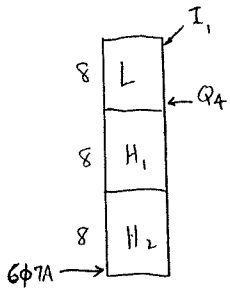


I/Q, Q/I

```
*1AB0LLL   NOTE 2.0

1AB0-   A5 C6       LDA   $C6                    FINAL "DATA" ENTRY
1AB2-   18         CLC
1AB3-   69 01       ADC   #$01
1AB5-   C9 20       CMP   #$20
1AB7-   D0 02       BNE   $1ABB
1AB9-   A9 00       LDA   #$00
1ABB-   4A          LSR
1ABC-   4A          LSR
1ABD-   4A          LSR
1ABE-   F0 08       BEQ   $1AC8
1AC0-   0A          ASL
1AC1-   0A          ASL
1AC2-   0A          ASL
1AC3-   38          SEC                    NOTE: This is a corrected version (MAY93!!)
1AC4-   E9 03       SBC   #$03                    ( PARK,ROBSART,SYLVAN L.
1AC6-   B0 02       BCS   $1ACA                  have not been upgraded, because
1AC8-   A9 1D       LDA   #$1D                   the bug does not affect the result when
1ACA-   85 CA       STA   $CA                    the # sends is $40.)
1ACC-   A9 20       LDA   #$20
1ACE-   85 C7       STA   $C7
1AD0-   A9 84       LDA   #$84
1AD2-   85 C8       STA   $C8
1AD4-   A9 0B       LDA   #$0B
1AD6-   85 CB       STA   $CB
1AD8-   A2 20       LDX   #$20
1ADA-   BD FF A7    LDA   $A7FF,X

1B14-   AD 04 03    LDA   $0304
*1B30L  NOTE 2.1
1B30-   20 40 1C    JSR   $1C40   displ. I/Q;Q/I        Display Final Param.
1B33-   20 40 1B    JSR   $1B40   displ. worst LTM's
1B36-   20 80 11    JSR   $1180      display sigs vs. rx,&mem test
1B39-   60         RTS
1B3A-   FF         ???

NOTE 2.2

1A10-   AD B1 C0    LDA   $C0B1
1A13-   30 07       BMI   $1A1C   C128 is ready
1A15-   EA          NOP           not ready!
1A16-   20 D0 19    JSR   $19D0   so "break it out"
1A19-   4C 00 1C    JMP   $1C00   and C128 NOT READY msg.
1A1C-   A9 28       LDA   #$28
1A1E-   8D B1 C0    STA   $C0B1
1A21-   A9 00       LDA   #$00

19D0-   A9 80       LDA   #$80                    Break out C128
19D2-   8D B0 C0    STA   $C0B0
19D5-   AD B0 C0    LDA   $C0B0
19D8-   A9 34       LDA   #$34          Send string  of $80 until C128
19DA-   8D B1 C0    STA   $C0B1         "dissappears" in to BASIC. The C128
19DD-   A9 3C       LDA   #$3C          will detect the $80 where it should
19DF-   8D B1 C0    STA   $C0B1         never occur for "normal" length records,
19E2-   A0 FF       LDY   #$FF          and go to an error handling loop
19E4-   AD B1 C0    LDA   $C0B1
19E7-   30 EC       BMI   $19D5   if no res-   (2m wait).
19E9-   88          DEY           ponse in $FF
19EA-   D0 F8       BNE   $19E4   cycles,then..
19EC-   A9 01       LDA   #$01       Mark "1st record" (so next C128 data is discarded)
19EE-   85 BA       STA   $BA
19F0-   60          RTS
```

NOTE 2.1

```
1180-  A0 00     LDY  #$00        }                    Display Signals at
1182-  B9 00 12  LDA  $1200,Y     }                    Selected height
1185-  99 50 07  STA  $0750,Y     }
1188-  C8        INY              }  SIG:RX#1bbb#2bbb#3bbb
1189-  C0 17     CPY  #$17        }
118B-  D0 F5     BNE  $1182       }
118D-  A2 08     LDX  #$08        }
118F-  A0 03     LDY  #$03        }  ÷ all Σ I1, Σ Q1  by 8
1191-  18        CLC              }
1192-  7E 79 60  ROR  $6079,X     }
1195-  7E 81 60  ROR  $6081,X     }
1198-  7E 89 60  ROR  $6089,X     }
119B-  88        DEY              }
119C-  D0 F3     BNE  $1191       }
119E-  CA        DEX              }
119F-  D0 EE     BNE  $118F       }
11A1-  20 20 12  JSR  $1220       div-by"#sends"
11A4-  EA        NOP
11A5-  A2 00     LDX  #$00        screen pointer
11A7-  A0 03     LDY  #$03        signal index
11A9-  18        CLC              -(dont need)
11AA-  B9 82 BC  LDA  $BC82,Y     sig.
11AD-  C9 64     CMP  #$64        100
11AF-  90 08     BCC  $11B9
11B1-  FE 56 07  INC  $0756,X     100's screen digit
11B4-  38        SEC
11B5-  E9 64     SBC  #$64
11B7-  B0 F4     BCS  $11AD
11B9-  E8        INX
11BA-  C9 0A     CMP  #$0A        10's digit
11BC-  90 08     BCC  $11C6
11BE-  FE 56 07  INC  $0756,X
11C1-  38        SEC
11C2-  E9 0A     SBC  #$0A
11C4-  B0 F4     BCS  $11BA
11C6-  E8        INX
11C7-  18        CLC
11C8-  7D 56 07  ADC  $0756,X
11CB-  9D 56 07  STA  $0756,X
11CE-  E8        INX
11CF-  E8        INX
11D0-  E8        INX
11D1-  E8        INX
11D2-  E8        INX
11D3-  88        DEY
11D4-  D0 D3     BNE  $11A9
11D6-  A2 00     LDX  #$00
11D8-  A9 03     LDA  #$03
11DA-  48        PHA
11DB-  A0 03     LDY  #$03
11DD-  18        CLC
11DE-  BD 56 07  LDA  $0756,X
11E1-  B0 04     BCS  $11E7
11E3-  C9 A1     CMP  #$A1
11E5-  90 02     BCC  $11E9
11E7-  09 B0     ORA  #$B0
11E9-  9D 56 07  STA  $0756,X
11EC-  EA        NOP
11ED-  E8        INX
11EE-  88        DEY
11EF-  D0 ED     BNE  $11DE
11F1-  E8        INX
11F2-  E8        INX
11F3-  E8        INX
11F4-  E8        INX
11F5-  EA        NOP
11F6-  68        PLA
11F7-  38        SEC
11F8-  E9 01     SBC  #$01
11FA-  D0 DE     BNE  $11DA
11FC-  60        RTS
```



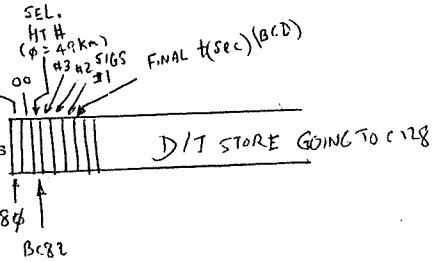

```
1200- D3 C9 C7 A3 B1 BD A0 A0     "SIG#1=ɓɓɓɓ#2=ɓɓɓɓ#3=ɓɓɓɓ"
1208- A0 A0 A3 B2 BD A0 A0 A0
1210- A0 A3 B3 BD A0 A0 A0 A0
1218- A3 B4 BD A0 A0 A0 FF FF
*1ED0LLL
```

NOTE 2.1 cont'd

```
1220-  A2 08     LDX  ##$08   I,Q comp.
1222-  A0 03     LDY  ##$03   4-ant#
1224-  BD 81 60  LDA  $6081,X  Σ I_H
1227-  85 F3     STA  $F3
1229-  BD 89 60  LDA  $6089,X  Σ I_L
122C-  CA        DEX
122D-  7D 89 60  ADC  $6089,X  Σ Q_L
1230-  85 F4     STA  $F4
1232-  A5 F3     LDA  $F3
1234-  7D 81 6Q  ADC  $6081,X  Σ Q_H
1237-  90 02     BCC  $123B
1239-  A9 FF     LDA  ##$FF   overfl.,so
123B-  85 F3     STA  $F3     clip
123D-  AD 91 BC  LDA  $BC91   ÷ by #sends
1240-  85 F1     STA  $F1
1242-  8A        TXA
1243-  48        PHA
1244-  20 60 12  JSR  $1260
1247-  A5 F0     LDA  $F0
1249-  99 82 BC  STA  $BC82,Y save RX#1 sig,#2,#3 in D/T store
124C-  68        PLA          going to C128
124D-  AA        TAX
124E-  CA        DEX
124F-  88        DEY
1250-  D0 D2     BNE  $1224
1252-  A9 20     LDA  ##$20   also save ht# (Ø==49Km)
1254-  38        SEC          used for the sig vs. rx stuff
1255-  ED 9E 60  SBC  $609E
1258-  8D 82 BC  STA  $BC82   in D/T store
125B-  20 8D 12  JSR  $128D
125E-  60        RTS
```

Get mean (I+Q) at 3 antennas

(don't have H_2 byte because of pre-divide)




```
1260-  A9 00     LDA  ##$00
1262-  85 F0     STA  $F0
1264-  85 F2     STA  $F2
1266-  A2 08     LDX  ##$08
1268-  18        CLC
1269-  66 F1     ROR  $F1
126B-  66 F2     ROR  $F2
126D-  A5 F4     LDA  $F4
126F-  38        SEC
1270-  E5 F2     SBC  $F2
1272-  85 F5     STA  $F5
1274-  A5 F3     LDA  $F3
1276-  E5 F1     SBC  $F1
1278-  90 06     BCC  $1280
127A-  85 F3     STA  $F3
127C-  A5 F5     LDA  $F5
127E-  85 F4     STA  $F4
1280-  26 F0     ROL  $F0
1282-  CA        DEX
1283-  D0 E3     BNE  $1268
1285-  60        RTS
```

DIVIDE

$$\frac{F3(H), F4(L)}{F1} = F\emptyset$$

(remainder in F4)

NOTE 2.5

```
1310-  AD 05 03  LDA  $0305  S_10
1313-  0A        ASL
1314-  0A        ASL
1315-  0A        ASL
1316-  0A        ASL
1317-  0D 04 03  ORA  $0304  S_1
131A-  8D 86 BC  STA  $BC86   save in D/T store
131D-  60        RTS
131E-  00        BRK
131F-  63        ???
1320-  65 B2     ADC  $B2     ..... old stuff.....
1322-  85 B2     STA  $B2
1324-  A5 A5     LDA  $A5
1326-  25 9B     AND  $9B
1328-  AA        TAX
```

SAVE FINAL SECONDS OF CLOCK

NOTE 2.6

```
12BD-   20 10 13    JSR   $1310      save final clk secs(bcd)      MEM TEST
1290-   A2 80       LDX   #$80
1292-   A9 00       LDA   #$00
1294-   85 F0       STA   $F0
1296-   A9 40       LDA   #$40
1298-   85 F1       STA   $F1
129A-   A0 FF       LDY   #$FF
129C-   B1 F0       LDA   ($F0),Y
129E-   51 F0       EOR   ($F0),Y
12A0-   D0 16       BNE   $12B8
12A2-   98          TYA
12A3-   0A          ASL
12A4-   90 03       BCC   $12A9
12A6-   A8          TAY
12A7-   B0 F3       BCS   $129C
12A9-   A5 F0       LDA   $F0
12AB-   18          CLC
12AC-   69 FF       ADC   #$FF
12AE-   85 F0       STA   $F0
12B0-   90 02       BCC   $12B4
12B2-   E6 F1       INC   $F1
12B4-   CA          DEX
12B5-   D0 E3       BNE   $129A
12B7-   60          RTS
12B8-   48          PHA
12B9-   A0 17       LDY   #$17
12BB-   B9 E7 12    LDA   $12E7,Y
12BE-   99 D4 06    STA   $06D4,Y
12C1-   88          DEY
12C2-   D0 F7       BNE   $12BB
12C4-   06 F1       ASL   $F1
12C6-   90 0B       BCC   $12D3
12C8-   A0 03       LDY   #$03
12CA-   B9 FE 12    LDA   $12FE,Y
12CD-   99 E2 06    STA   $06E2,Y
12D0-   88          DEY
12D1-   D0 F7       BNE   $12CA
12D3-   A0 08       LDY   #$08
12D5-   68          PLA
12D6-   88          DEY
12D7-   0A          ASL
12D8-   90 FC       BCC   $12D6
12DA-   98          TYA
12DB-   09 70       ORA   #$70
12DD-   8D EB 06    STA   $06EB
12E0-   60          RTS
```

NOTE 2.5

Test various bytes 4000-BFFF by load and compare.

$4000-7FFF = "mid" row of 4116's

$8000-BFFF = "top" row (furthest from keybd)

base addr.=4000+(n-1)*FF

index=FF,FE,FC,F8,FØ,EØ,CØ,8Ø,Ø

[This test is just a test - it is not based on any "inside" knowledge of DRAMS]

MEM FAULT FOUND!!

display"MEM FAULT:ROW=MID,BIT= " (flashing)

look at high addr. byte to get chip row
ok, it is the mid row

replace "MID" with "TOP" (but the MID row could still have a fault)

display"the" bad bit# (Ø=leftmost chip in ROW)

NOTE:: only the 1st bad bit found is displayed

```
12E8- 4D 45 4D 60 46 41 55 4C
12F0- 54 7A 52 4F 57 7D 4D 49    "MEM FAULT:ROW=MID,BIT=Ø"
12F8- 44 6C 42 49 54 7D A0 54
1300- 4F 50 FF FF FF
*1200.121F
```

NOTE 2.3

```
2000-   AD F9 5F    LDA   $5FF9
2003-   29 F0       AND   #$F0
2005-   4A          LSR
2006-   4A          LSR
2007-   4A          LSR
2008-   4A          LSR
2009-   09 B0       ORA   #$B0
200B-   8D E0 07    STA   $07E0 screen addr.
200E-   AD F9 5F    LDA   $5FF9
2011-   29 0F       AND   #$0F
2013-   09 B0       ORA   #$B0
2015-   8D E1 07    STA   $07E1
2018-   AD F8 5F    LDA   $5FF8
201B-   29 F0       AND   #$F0
201D-   4A          LSR
201E-   4A          LSR
201F-   4A          LSR
2020-   4A          LSR
2021-   09 B0       ORA   #$B0
2023-   8D E3 07    STA   $07E3 screen addr(.last line)
2026-   AD F8 5F    LDA   $5FF8
2029-   29 0F       AND   #$0F
202B-   09 B0       ORA   #$B0
202D-   8D E4 07    STA   $07E4
2030-   4C 40 09    JMP   $0940
```

Display Comm#1 check bytes

This is a temporary diagnostic for comm#1 errors, but note: if comm#3 error, still displays comm#1 check bytes!!

NOTE 2.7

```
0976-   A9 00       LDA   #$00
0978-   9D CF 5E    STA   $5ECF,X
097B-   CA          DEX
097C-   D0 FA       BNE   $0978
097E-   85 C5       STA   $C5
0980-   85 B6       STA   $B6
0982-   A9 01       LDA   #$01
0984-   85 BA       STA   $BA
0986-   A9 91       LDA   #$91
0988-   85 F9       STA   $F9
098A-   A9 09       LDA   #$09
098C-   85 FA       STA   $FA
098E-   20 21 C7    JSR   $C721
0991-   EA          NOP
0992-   A0 20       LDY   #$20
```

Init. tape drive interface, so it can be put on line (Tromsø)

```
0E12-   EA          NOP
0E13-   EA          NOP
0E14-   EA          NOP
0E15-   EA          NOP
0E16-   EA          NOP
0E17-   A9 17       LDA   #$17        Turn on Tx (for BUZZ during tape dump)
0E19-   2D 02 C5    AND   $C502
0E1C-   8D 02 C5    STA   $C502
0E1F-   60          RTS
```

Tape Dump (Tromsø)

```
1970-   A9 10       LDA   #$10
1972-   85 B6       STA   $B6
1974-   20 07 1F    JSR   $1F07       dummy, now
1977-   20 12 0E    JSR   $0E12       Tx on
197A-   EA          NOP
197B-   A9 00       LDA   #$00
197D-   85 19       STA   $19
197F-   A9 40       LDA   #$40
1981-   85 1A       STA   $1A
1983-   A9 C0       LDA   #$C0
1985-   85 1B       STA   $1B
1987-   A9 06       LDA   #$06
1989-   85 1C       STA   $1C
198B-   A9 00       LDA   #$00
198D-   85 1D       STA   $1D
198F-   A9 82       LDA   #$82        12.5 ips, 9tr (Tromsø)
1991-   85 1E       STA   $1E
1993-   20 0C C7    JSR   $C70C       dump to tape
1996-   A5 1F       LDA   $1F         status after dump (if ok,no msg)
1998-   D0 1C       BNE   $19B6
199A-   4C C2 19    JMP   $19C2       Status ok, skip re-read because Tromso drive
199D-   EA          NOP                 maybe not reliable backspace
199E-   20 06 C7    JSR   $C706
19A1-   A9 C0       LDA   #$C0
19A3-   85 19       STA   $19
19A5-   A9 46       LDA   #$46
19A7-   85 1A       STA   $1A
19A9-   20 00 C7    JSR   $C700
19AC-   A9 00       LDA   #$00
19AE-   85 B6       STA   $B6
19B0-   A5 1F       LDA   $1F
19B2-   29 BF       AND   #$BF
19B4-   F0 09       BEQ   $19BF
19B6-   20 18 1C    JSR   $1C18       set "problems" msg.
19B9-   20 20 1F    JSR   $1F20       incr."tape err"counter on screen
19BC-   18          CLC
19BD-   90 03       BCC   $19C2
19BF-   20 E0 1B    JSR   $1BE0         clear msg
19C2-   AD 02 C5    LDA   $C502
19C5-   09 08       ORA   #$08
19C7-   8D 02 C5    STA   $C502
19CA-   60          RTS
```

```
1F07-   60          RTS   skip EOT test for Tromsø
1F08-   2B          ???
1F09-   C7          ???
1F0A-   29 20       AND   #$20
1F0C-   F0 11       BEQ   $1F1F
1F0E-   A9 1F       LDA   #$1F
1F10-   48          PHA
1F11-   A9 1C       LDA   #$1C
1F13-   48          PHA
1F14-   18          CLC
1F15-   A9 1E       LDA   #$1E
1F17-   48          PHA
1F18-   A9 C2       LDA   #$C2
1F1A-   4C 21 1C    JMP   $1C21
1F1D-   68          PLA
1F1E-   68          PLA
1F1F-   60          RTS
```

NOTE 2.8

```
0BE0-   78          SEI
0BE1-   D8          CLD
0BE2-   20 80 0B    JSR   $0B80
0BE5-   A9 7D       LDA   #$7D
0BE7-   8D 8F BC    STA   $BC8F
0BEA-   8D F8 5E    STA   $5EF8
0BED-   A9 7F       LDA   #$7F
0BEF-   8D 8E BC,   STA   $BC8E
0BF2-   8D F9 5E    STA   $5EF9
0BF5-   A9 08       LDA   #$08
0BF7-   8D 8D BC    STA   $BC8D
0BFA-   8D FA 5E    STA   $5EFA
0BFD-   20 D0 1E    JSR   $1ED0
0C00-   EA          NOP
0C01-   EA          NOP
0C02-   EA          NOP
0C03-   EA          NOP
0C04-   EA          NOP
0C05-   A9 55       LDA   #$55      #"sends"
0C07-   8D 91 BC    STA   $BC91
0C0A-   8D F6 5E    STA   $5EF6
0C0D-   85 BD       STA   $BD
0C0F-   A9 00       LDA   #$00
0C11-   8D 90 BC    STA   $BC90
```

SET RECORD LENGTH

This is liable to be changed if the amount of integration is changed while the Tx rate is constant.

(will probably be 40 for Tromsø)

```
0C81-   8D 80 BC    STA   $BC80
0C84-   A9 D4       LDA   #$D4      "T" for Tromso
0C86-   8D 8A BC    STA   $BC8A
0C89-   60          RTS
```

```
1ED0-   20 F8 19    JSR   $19F8
1ED3-   AD 59 C0    LDA   $C059     — PARK ONLY (changes synch rate)
1ED6-   A9 04       LDA   #$04
1ED8-   48          PHA
1ED9-   AD 0A 03    LDA   $030A     H10
1EDC-   0A          ASL
1EDD-   0A          ASL
1EDE-   0A          ASL
1EDF-   0A          ASL
1EE0-   0D 09 03    ORA   $0309     H1
1EE3-   C9 00       CMP   #$00
1EE5-   90 16       BCC   $1EFD     TEST WHETHER HR BETWEEN "0" & "24" ! ∴ always is
1EE7-   C9 24       CMP   #$24
1EE9-   B0 12       BCS   $1EFD
1EEB-   AD 58 C0    LDA   $C058     — PARK ONLY (changes synch rate)
1EEE-   20 17 0E    JSR   $0E17     - TX ON
1EF1-   A0 05       LDY   #$05
1EF3-   20 A8 FC    JSR   $FCA8     } time wasting loop
1EF6-   88          DEY
1EF7-   D0 FA       BNE   $1EF3     }
1EF9-   68          PLA
1EFA-   A9 18       LDA   #$18      24 pulse integration
1EFC-   48          PHA
1EFD-   68          PLA
1EFE-   8D 92 BC    STA   $BC92
1F01-   8D F7 5E    STA   $5EF7
1F04-   60          RTS
```

RECORD INITIALIZE

(if hour between "0" & "24")

C128 NOTES & IMPORTANT LOCATIONS:

```
10 REM-C128 COH-RTW PMFCA, JAN24,1992,VERSION 18J'-3A ROBSART
20 BANK0:FAST:DEFFNS(T)=SGN(T<0)-SGN(T>=0):REM-NB  C128 LOG. TRUE=-1 !!!
25 TT=0:I=0:J=0:IA=0:IC=0:X=0:Y=0:MR=0:MI=0:S0=0:S1=0:S2=0:N=0:K=0:NN=0
30 DIMC(3,32),M(3),MX(3,2),T(3,2),R(3,2),D%(16),MN(2,3),V(3)
40 DIMMZ%(18),ME(6,2),NM(6),AR(10),AI(10)
50 REM-ANT PAIR DIR 2=>1,3=>2,1=>3
60 D=202.5:P=57.29578:P1=210/P:P2=-30/P:P3=90/P:PZ=1.5708:TT=256:TU=TT*TT
70 TV=TU-1:TW=TU/2-1
80 D2=D*D:X1=D*COS(P1):X2=D*SIN(P1):X3=D*COS(P2)
90 X4=D*SIN(P2):X5=D*COS(P3):X6=D*SIN(P3)
100 U1=COS(2*P1):U2=COS(2*P2):U3=COS(2*P3)
110 V1=SIN(2*P1):V2=SIN(2*P2):V3=SIN(2*P3)
120 TS=.533:CC=.5:LX=.1:CY=0.00:FC=100*135/360/2/TS
130 GOSUB2330:REM-LOAD MACH LANG,SET MEM LMTS
135 GOSUB3500:REM-POKE PTR PROG
137 TRAP4000:REM-ADDR FOR ERROR HANDL
140 REM-.....START PROCESSING.....
144 PRINT"WAITING FOR DATA....":X=USR(X)
145 IFPEEK(37952)=128THENX=1/0
146 IFPEEK(52353)<>0ORPEEK(52352)<255THENX=1/0
150 TI$="000000":IC=52288:IA=40000:IO=53152:IG=52320:IV=52352:NR=NR+1
160 REM-READ D/T
```

TU=TT-1; TW=TU/2-1 — ANTENNA PAIR SPACING (m)

P1=210/P:P2=-30/P:P3=90/P — Antenna pair directions (° E of Geog. N)

TS=.533 — Δt

NOTE 3.1

NOTE 3.2

```
1990 ME(I%,1)=ME(I%,1)+VT*Q1/10:ME(I%,2)=ME(I%,2)+VT*Q2/10:NM(I%)=NM(I%)+1
2000 ER%=IS+IR*4+IX*32:POKEIO-24,ER%
2010 K=IO-12:POKEK,NT:POKEK-1,RL
2020 IC=IC-384:IA=IA-64:IO=IO-24:NEXTIH
2030 PRINT"ACCUM DIV-BY-0=":NE:GOTO140
2040 REM-.....PRINT 9 KM HRLY MEANS.....
2045 SYS37696
2050 IFHR%<>8THEN2100
2060 PRINT"  DAYHR!  64-70KM  !  73-79KM  !  82-88KM  !";
2070 PRINT"  91-97KM  ! 100-106KM ! 109-115KM !"
2080 PRINT" ROBSRT ! #   N   E ! #   N   E ! #   N   E !";
2090 PRINT"  #   N   E ! #   N   E ! #   N   E !"
2100 DH=LD%*100+LH%:IFDH>9999THENPRINTSTR$(DH);:GOTO2130
2110 IFDH>999THENPRINT" ";STR$(DH);:GOTO2130
2120 PRINT"  ";STR$(DH);
2130 FORK=1TO6:KK=(K-1)*3+1:MZ%(KK)=NM(K)
2140 IFNM(K)>0THEN2160
```

"Transp"

```
2270 PRINT"  ";STR$(MZ%(K));:GOTO2300
2280 PRINT" ";STR$(MZ%(K));:GOTO2300
2290 PRINTSTR$(MZ%(K));
2300 NEXTK
2310 PRINT:SYS37738
2320 LD%=DY%:LH%=HR%:RETURN
```

2305 IF HR% = 8 THEN PRINT:PRINT"#REC'D DAY=";NR;:NR=0

NOTE 3.2

```
4000 REM-ERROR HANDLING
4010 SYS37696:PRINTDY%" "HR%" "MN%"!"ERR$(ER);EL;" DROP  REC"
4030 SYS37738:TI$="000000":FORK=52384TO53247:POKEK,0:NEXTK
4050 IFTI<7200THEN4050:REM-WAIT 2 MIN
4055 POKE52352,255:POKE52353,0
4060 RESUME 144
```

NOTE 3.1

SECTION VI.

SECTION VI.

Section V.: Pertinent extracts from 1979 RTW report. [page 219-243]

Includes derivation of the required correction to correlation values for the use of bit amplitudes, and derivation of the PMFCA analysis. This system used signal MAGNITUDES (i.e. non-coherent receiver), but most of the material applies directly to the COHRTW system. Also a 6502 machine language reference chart is appended.

APPENDIX B

Theory of binary correlation

1. Calculation of correlation coefficient from number of 1-matches
2. Correction to binary correlation (Gaussian amplitude distribution)
3. Effect of overloading on correction function (Gaussian)
4. Test of correction function on Rayleigh distributed data
5. Further comments on the binary method

APPENDIX C

Pet program details

1. Description of Pet analysis
2. Correction to time delays for antenna cycling
3. Weighted least squares fit to time delays
4. Poor Man's Full Correlation Analysis (PMFCA)

6502 Machine language chart

******* Reprinted from 1979 RTW System Report ********

APPENDIX B

Theory of binary correlation

1. Calculation of correlation coefficient from number of 1-matches
2. Correction to binary correlation (Gaussian amplitude distribution)
3. Effect of overloading on correction function (Gaussian)
4. Test of correction function on Rayleigh distributed data
5. Further comments on the binary method

1. Calculation of correlation from # 1-matches

The correlation coefficient is defined by:

$$\rho = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\sqrt{(\overline{x^2} - \bar{x}^2)(\overline{y^2} - \bar{y}^2)}}$$

where x and y are elements of the same (auto correlation) or different (cross correlation) fading sequences, and there may be some lag between x and y measurements.

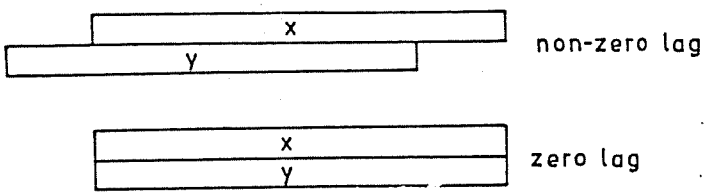


Fig. 1

If the elements x and y are binary (i.e. 0 or 1), then $\sum xy$ is given directly by the results of the 'AND' operation; and so $\overline{xy}$ = (# 1-matches)/(N-ℓ), where N is the total number of samples in a sequence and ℓ is the lag. Also $\bar{x} = \sum x/N$, where $\sum x$ is the number of ones in the sequence ; and $\overline{x^2} = \sum x^2/N$, where $\sum x^2$ is again the number of ones (since x can only be 0 or 1) . The number of 1's could be counted in each sequence separately, however, since the fading should be statistically the same at each antenna, the sum of the #1-matches from the zero lag auto correlation calculations can be used to estimate the 'average' number of 1's in a sequence. For various reasons this sum has only be calculated over the three outer antennas (#1,2,3). Since the amplitude distribution is usually skewed towards low amplitudes (Rayleigh type), the number of 1's is usually $< \frac{1}{2}N$. Thus, if a_o is the number of 1-matches ($\sum xy$), and a_1 is the number of 1's in any sequence ($\sum x$), the correlation coefficient is:

$$\rho' = \frac{\frac{a_o}{N-\ell} - \frac{a_1^2}{N^2}}{\frac{a_1}{N} - \frac{a_1^2}{N^2}}$$

This is the correlation between binary sequences, and must be corrected as explained below, to estimate the correlation which would have been obtained with the original sequences.

2. Correction to binary correlation

As will be shown below, the expectation value of correlation between two amplitude sequences which have been converted to binary with respect to their means is lower than that which would have been found between the original sequences. Thus some sort of correction is required. The derivation of this correction function closely follows that of Weinreb(1963). A basic assumption is that the amplitudes have an approximately Gaussian (i.e. Normal) distribution. This is not generally true in practice - usually the distribution is closer to a Rayleigh - however, it makes the problem solvable. The binary correlation, as employed in the present analysis, counts the number of times that both X and Y amplitudes are greater than their means. The probability of this event will be derived next.

Assume that X and Y are taken from a Normal distribution with zero mean and unit s.d. Then the joint probability density function for X and Y is (e.g. Hald, 1952):

$$f(x,y) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left\{-\frac{1}{2(1-\rho^2)}(x^2 - 2\rho xy + y^2)\right\}$$

where ρ is the correlation coefficient between X and Y. Thus the probability of finding X between x and x+dx and Y between y and y+dy simultaneously is $f(x,y)dxdy$. The probability of finding X and Y both greater than their means is then:

$$P(X>0,Y>0) = \int_{x=0}^{\infty}\int_{y=0}^{\infty} f(x,y)dydx$$

If polar coordinates are used ($x=r\cos\theta$, $y=r\sin\theta$) the integral is reduced to

$$P(X>0,\ Y>0) = \frac{1}{2\pi\sqrt{1-\rho^2}} \int_{\theta=0}^{\frac{\pi}{2}} \int_{r=0}^{\infty} \exp\left\{ \frac{-r^2(1-\rho \sin 2\theta)}{2(1-\rho^2)} \right\} r\ dr\ d\theta$$

A further change of variable $\xi = r^2$, $d\xi = 2r dr$, and integration of the inner integral leaves:

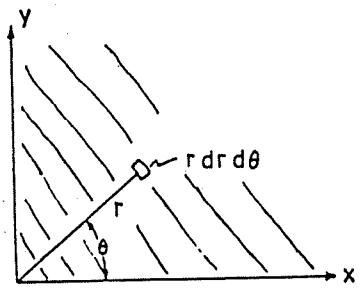


Fig. 1

$$P(X>0,\ Y>0) = \frac{\sqrt{1-\rho^2}}{2\pi} \int_{\theta=0}^{\frac{\pi}{2}} \frac{1}{(1-\rho \sin 2\theta)}\, d\theta$$

Then let $\alpha = 2\theta$:

$$P(X>0,\ Y>0) = \frac{\sqrt{1-\rho^2}}{2} \int_{\alpha=0}^{\frac{\pi}{2}} \frac{1}{1-\rho \sin\alpha}\, d\alpha$$

$$= \frac{1}{\pi} \tan^{-1}\left\{ \frac{\tan\frac{\alpha}{2} - \rho}{\sqrt{1-\rho^2}} \right\} \Bigg|_{\alpha=0}^{\frac{\pi}{2}}$$

(This last integral is taken from math tables). The identity

$$\tan^{-1}(a) - \tan^{-1}(b) = \tan^{-1}\left(\frac{a-b}{1-ab} \right)$$

is used to obtain:

$$P(X>0,\ Y>0) = \frac{1}{\pi} \tan^{-1}\sqrt{\frac{1+\rho}{1-\rho}}$$

Now $P(X>0, Y>0)$ must be related to the correlation of the binary sequences X' and Y'. These are defined by:

$X' = 1$ if $X>0$ $\qquad$ $Y' = 1$ if $Y>0$

$X' = 0$ if $X\leq 0$ $\qquad$ $Y' = 0$ if $Y\leq 0$

$Y'=$	$X'=0$	$X'=1$	tot.
0	a	c	$\frac{1}{2}$
1	b	d	$\frac{1}{2}$
tot.	$\frac{1}{2}$	$\frac{1}{2}$	

Fig. 2

Figure 2 shows the joint probability function of X', Y'. Here $d = P(X>0, Y>0)$. Since a Gaussian distribution has been assumed, the marginal probabilities are $P(X>0)=P(X\leq 0)=P(Y>0)=P(Y\leq 0)=\frac{1}{2}$. (It is easy to show that $a=d$ and $b=c$). The cross correlation between X' and Y' is then:

$$\rho' = \frac{\overline{X'\,Y'} - \overline{X'}\ \overline{Y'}}{\sqrt{\overline{X'^2} - \overline{X'}^2}\sqrt{\overline{Y'^2} - \overline{Y'}^2}} = \frac{d - (\frac{1}{2})^2}{\sqrt{(\frac{1}{2}) - (\frac{1}{2})^2}\sqrt{(\frac{1}{2}) - (\frac{1}{2})^2}} = 4d-1$$

Thus:

$$\rho' = \frac{4}{\pi} \tan^{-1}\sqrt{\frac{1+\rho}{1-\rho}} \quad - 1$$

This can be re-arranged to give

$$\rho = \sin \frac{\pi}{2} \rho'$$

This then is the correction function which must be applied to the result of correlating the binary representations of sequences in order to estimate the actual correlation between the original sequences.

3. Effect of overloading on the correlation correction

It is of interest to inquire whether receiver overloading will affect the correction function. This may also indicate whether less major receiver non-linearity will affect the results.

The receiver output for an input signal causing overloading will be assumed to be the maximum output value. It can be seen (Fig. 1) that $\bar{A}$, the mean amplitude, is less than the actual mean of the Gaussian. Thus the binary sequence will have more ones than zeroes, and thus a mean $> \frac{1}{2}$. Alternatively non-linearity of the type shown in Fig.2 will skew the distribution towards low amplitudes (assuming no overloading occurs) resulting in a mean $< \frac{1}{2}$. Such skewing occurs naturally in a Rayleigh distribution, and a later section will try to answer the present question in this case.

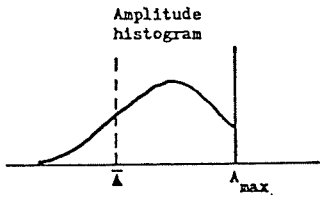


Fig. 1

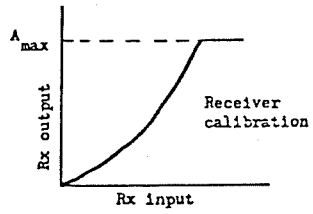


Fig. 2

Here, the procedure is to choose a value, ϵ, and find $P(X>\epsilon, Y>\epsilon)$ for various values of ρ (the actual correlation). ϵ is negative in the case of overloading. Fig. 3 shows contours of $f(x,y)$ for ρ = +ve, 0, and -ve; and the region (shaded) which must be integrated over

to find $P(X>\epsilon, Y>\epsilon)$.

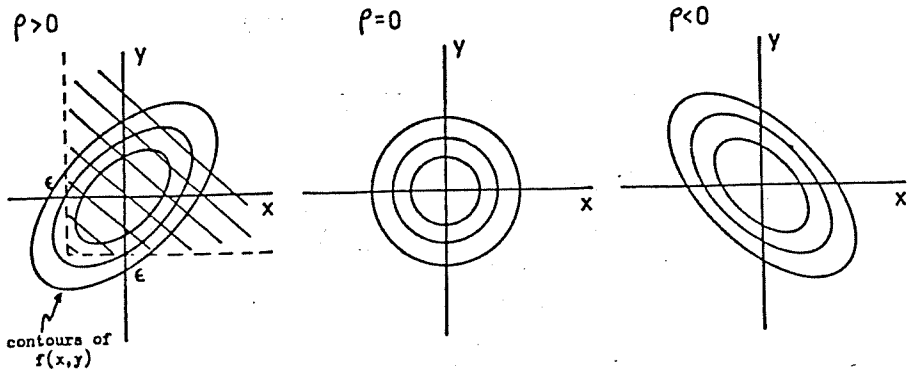


Fig. 3

It is assumed that the Gaussians (for X and Y) have zero mean and unit s.d. as before (there is no loss of generality in this assumption). Thus the mean of the binary sequences, m, is the area of the normal curve (from statistical tables) above ϵ (see Fig. 4), and the binary correlation is:

$$\rho' = \frac{P(X>\epsilon, Y>\epsilon) - m^2}{m(1-m)}$$

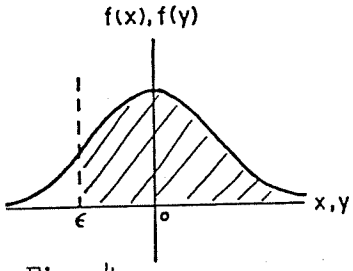


Fig. 4

where: $P = \int_{x=\epsilon}^{\infty}\int_{y=\epsilon}^{\infty} f(x,y)dydx$. Unfortunately when $\epsilon \neq 0$ the integral does not reduce to a simple solution, and has been performed numerically here (The upper limits used were x=y=8, so the P values will be slightly low). Values of ϵ between ±1 were used; however, instead of quoting ϵ, m is given since it is directly available in the experimental situation. Fig. 5 shows a plot of ρ vs. $\sin\frac{\pi}{2}\rho'$ for +ve ρ and several values of m. The difference between the ideal and overloaded data

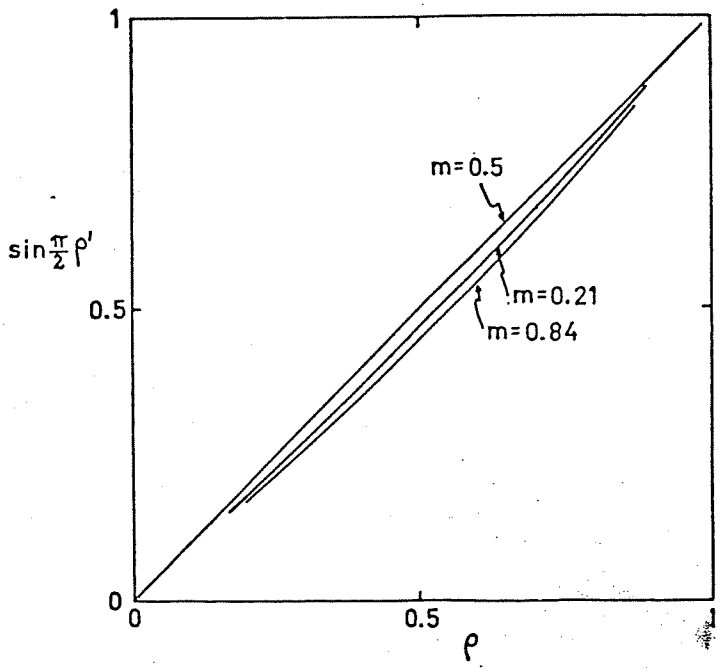


Fig. 5

(m=0.84) is seen to be almost negligible, and not worth correcting for.

In practice, however, overloading may have serious effects on the correlation if, as is done here, the sequences are assumed to be statistically equivalent (i.e. equal m). This problem will be discussed in Section 5.

4. Test with a Rayleigh distribution

In order to see whether the correction function works when the data are Rayleigh distributed, pairs of sequences X and Y were constructed as follows:

$$X_i = \sqrt{(a_i + K\, e_i)^2 + (b_i + K\, f_i)^2}$$

$$Y_i = \sqrt{(c_i + K\, e_i)^2 + (d_i + K\, f_i)^2}$$

where a_i, b_i, etc. are independent random values taken from a Gaussian distribution. This definition results in a non-zero expected correlation between X and Y while leaving them Rayleigh distributed. The expectation value of correlation depends on K, which has been chosen to be 2 for this test. 200 pairs of sequences (100 points each) were set up and correlated before and after conversion to binary (with respect to the mean). Histograms of the ratio of the actual to corrected binary correlations , and the actual correlations are shown in Fig. 1. Although there is some spread in the corrected values, there is no bias away from a ratio of 1. Thus the

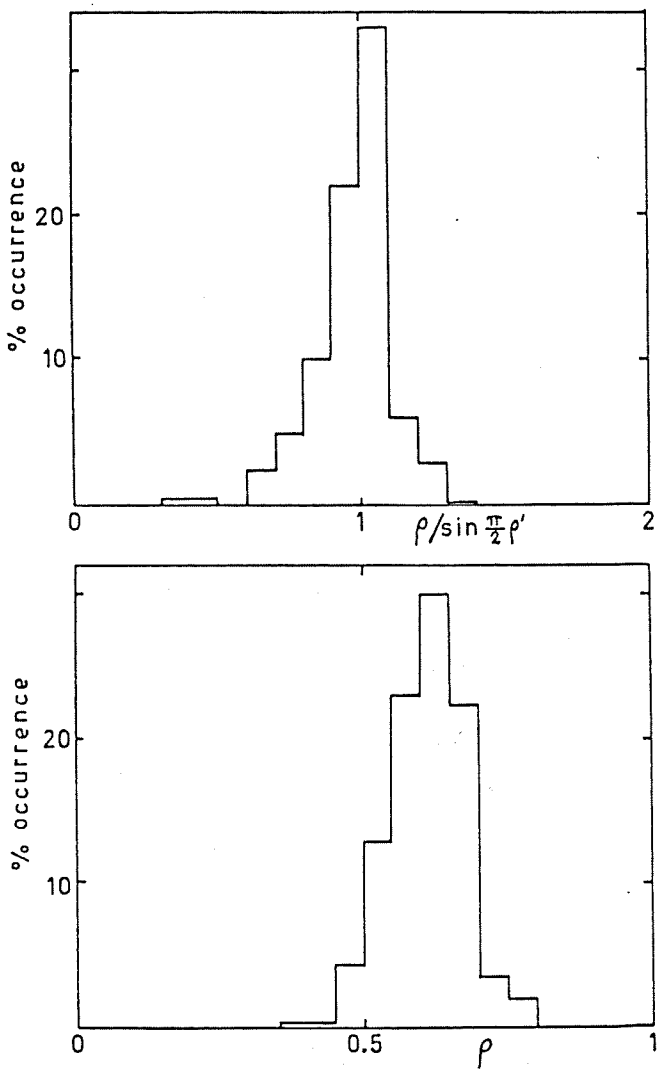


Fig. 1

same correction function should work for Rayleigh distributed data as well.

5. Further comments on binary correlation

a) The binary correlation, ρ' , has been defined as:

$$\rho'_{ij} = \frac{P_{ij} - m^2}{m(1-m)}$$

where m is the 'average' mean of the binary sequences over all antennas (in practice just #1,2,3). This is not a strictly correct definition; it should be

$$\rho'_{ij} = \frac{P_{ij} - m_i m_j}{\sqrt{m_i(1-m_i)m_j(1-m_j)}}$$

where ρ'_{ij} is the correlation between binary sequences from antenna i and j; however, if ρ' is not very sensitive to changes in m, the first definition should be sufficient since all antennas should see similar amplitude distributions (and therefore similar values of m). A plot of ρ' (and $\rho = \sin \frac{\pi}{2}\rho'$) versus P and m is shown in Fig. 1.

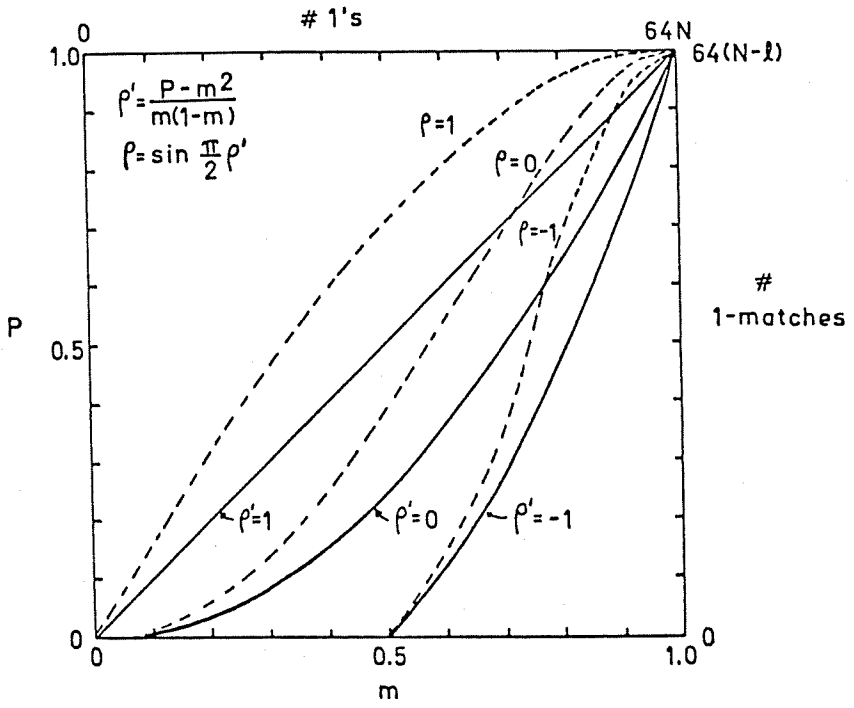


Fig. 1

Note that P cannot be > m since the # 1-matches cannot be greater than the number of 1's in any sequence; also, if $m > 0.5$, P must be $> 2m^2 - m$, because if both sequences are more than half 1's, there must be some 1-matches.

For reasonably Gaussian data (or any data with a symmetrical distribution) $m \cong 0.5$, and ρ' is relatively insensitive to small errors in m. However, if there is serious overloading, m will generally increase and Fig. 1 shows that ρ' becomes more sensitive to errors in m (and to errors in P for that matter, although P is a strictly correct value and will only have errors in a statistical sense). If, in addition, the amplitude distributions at different antennas are not strictly the same then the small error caused by the use of m rather than m_i, m_j may move the whole correlation curve up or down (with only small changes in shape) depending on whether m is less or greater than $\sqrt{m_i m_j}$. This shifting is seen quite often in badly overloaded data (Fig. 2) although a direct connection to differences in m_i, m_j etc. has not been proven as yet.

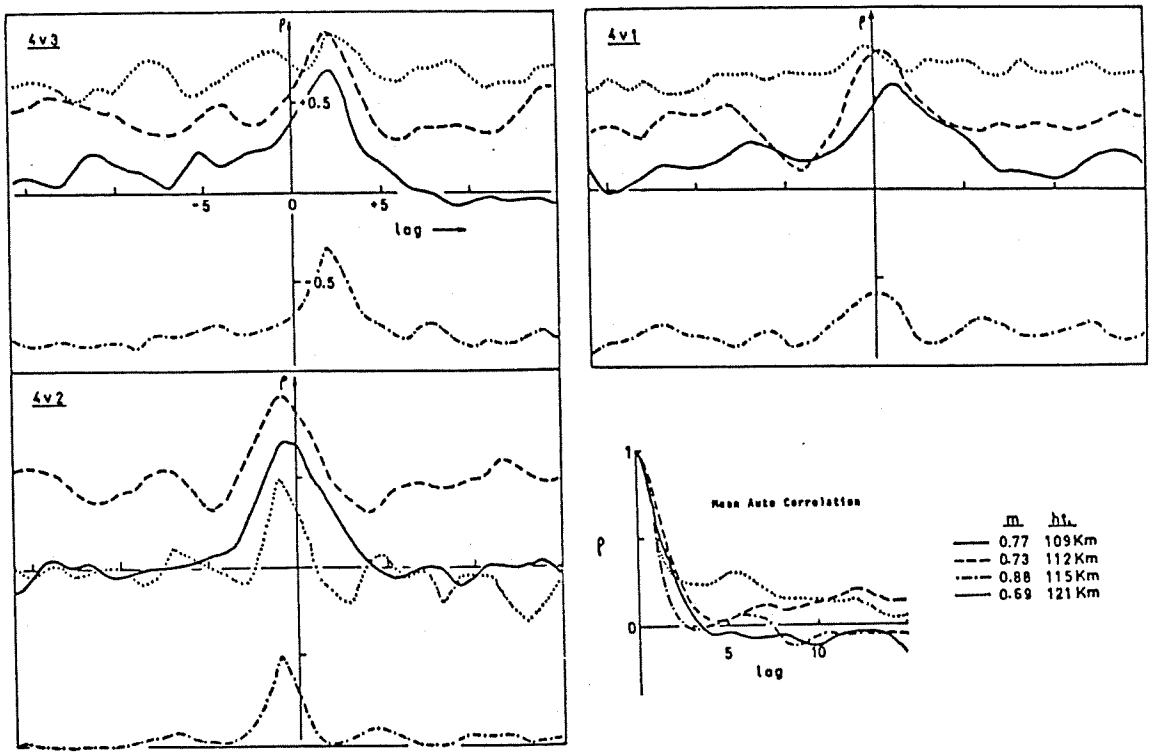


Fig. 2

In a computor simulation of the binary correlation process on real data which used separate sequence means, this problem didn't occur at all. On a separate point, if m_i and m_j are different but m is used to find ρ', then values > 1 or < -1 are possible; however, the correction through the sin function will reduce the limits to ± 1.

Other information which can be deduced from Fig. 1 is the expected resolution in ρ' (or ρ); how likely are exactly equal adjacent values of ρ (see the discussion in Appendix C, Section 1, pg.69 on locating peaks in cross correlations). The number of 1-matches is given by $P(N-\ell)$, and the number of 1's by mN (where N is the number of points in a sequence and ℓ is the lag - since $\ell << N$, it may be ignored). For example if N=512 (8 blocks/record) and m=0.5, there are 256 possible values of correlation which gives an average resolution of~0.008 in ρ'. If m=0.8 (i.e. bad overloading), there are 154 values, giving an average resolution of 0.14.

The conclusion is that overloading (or any other receiver non-linearity which causes amplitude distributions to be skewed towards high values) is quite serious, and should be avoided if at all possible. On the other hand skewing the distribution in the other direction (i.e. low m) is quite tolerable, and perhaps desirable, since ρ' becomes less sensitive to m here; and it has already been shown that the $\sin \frac{\pi}{2}\rho'$ correction works on at least one distribution of this type (i.e. Rayleigh).

If, as has been postulated, the sensitivity of ρ' to m is the cause of the trouble shown in Fig. 2, then a solution which tolerates overloading would be to use individual means m_i instead of the average mean m in the calculation of ρ'. From the point of view of the Apple program, the auto correlation routine (ACOR) would have to be modified to operate on all four antennas, and the zero lag values would have to be accumulated separately for each antenna. This appears to be feasible from an execution time stand-

point and would not require major modifications to the program.

b) It has been noticed by simulating the binary correlation on raw data from the previous wind system, that the correlation correction seems to work well below $\sim$ 80 Km, but results in an average reduction in peak values of $\sim$ 1.5-2 below the normal (non-binary calculation) correlation peaks above $\sim$ 80 Km. Presumably this has something to do with a change in amplitude distribution - possibly related to the effect of meteor trails at these heights. Investigation proceeds...

***** Reprinted from 1979 RTW System Report ********

APPENDIX C

Pet program details

1. Description of Pet analysis
2. Correction to time delays for antenna cycling
3. Weighted least squares fit to time delays
4. Poor man's full correlation analysis (PMFCA)
5. Machine language subroutine
6. Input/output memory locations
7. Listing of BASIC program

1. Pet analysis

The input to the Pet computer is the number of 1-matches at each of 32 lags (-16 to +15) for the three cross correlations (c_{ij}; i=1,32, j=1,3) and the sum (over three antennas) of the number of 1-matches for 16 lags (0 to 15) for the auto correlation (A_i; i=0,15). The conversion of these values to correct correlation values has been described in Appendix B (pg.222) and the relevant equations are repeated below.

$$\rho = \sin \frac{\pi}{2} \rho'$$

where:
$$\rho' = \frac{\frac{c_{ij}}{N-\ell} - m^2}{m(1-m)} \qquad \text{(cross correlation)}$$

or:
$$\rho' = \frac{\frac{a_i}{N-\ell} - m^2}{m(1-m)} \qquad \text{(auto correlation)}$$

where $a_i=A_i/3$, $m=a_o/N$, N= # points in each fading sequence (i.e. N = number of blocks per record times 64), and ℓ is the absolute value of the lag. Here ρ' is the actual correlation value for the binary sequences and ρ is the corrected value. In the Pet analysis usually ρ' is used in place of ρ until an accurate value is required, since time is at a premium.

The program (RTW Report pg. 79) falls into nine sections, which are described below.

a) Initialization

D is the antenna spacing (assumed equal for all antenna pairs); P1,P2,P3 are the directions of the antenna pairs 4v3 (in the sense 4 to 3 since t_{43} is being used), 4v2, and 4v1. TS is the sample spacing at any antenna (= the lag step), CC is the rejection limit on the auto correlation (see (c)); LX is the rejection limit for peaks in the cross correlation, and CY is used in the cycling correction (see section 2, pg.236). Since the Pet is a BASIC machine, the machine language subroutine used to communicate with the Apple must be loaded

from BASIC. It is placed in Tape Buffer #2, and a listing is given in Section 5

b) Data input

The locations of the incoming and outgoing data are shown in Section 6 ( ). The data (which apply to a single height) are read from these locations and put in the appropriate matrices.

c) Fast fading test and definition of one point in the mean auto correlation

ρ is calculated for increasing lag (1 to 5) until it is less than CC. If $\rho < CC$ at the first lag, the record is rejected for fast fading (i.e. noise). If $\rho \not< CC$, its value at the last tested lag is saved for future use in the PMFCA . Otherwise linear interpolation is used to find the exact time for ρ to fall to CC. Under the Gaussian assumption this point defines the auto correlation function.

d) Limited lag

This section tries to define a more limited lag (< 16) for the peak search in the cross correlations as described by Meek (1978). Successful definition requires that all cross correlations at zero lag be greater than zero. This is checked by examining just the numerator of ρ' . At present the nominal zero lag values (not corrected for cycling) are used, otherwise interpolation to find the correlation value at true zero lag would be required.

The main purpose of this attempt is to avoid spurious peaks in the cross correlations, not to save time.

e) Peak location

A search is made for the two greatest local maxima in each cross correlation, and their index positions saved. Because of the binary nature of the correlated sequences, the possibility of exactly equal adjacent values at a peak cannot be ignored, especially if the number of blocks/record is small. The present analysis identifies the first such value, from -ve lag, as the peak. Fig. 1 illustrates

two undesirable situations which could occur. The present resolution (8 blocks/record) is about 0.008 (see Appendix B, Section 5, pg. 63 for further comments) and so exactly equal values are not very likely.

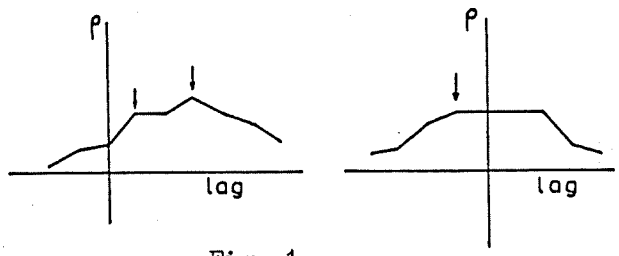


Fig. 1

f) Peak selection

Here accurate values of ρ'_{max} and t_{max} for each peak are found by a parabolic fit around the index positions found in e). Then ρ'_{max} is corrected to get ρ_{max} and the t_{max} are corrected for cycling (see section 2, pg. 71). If there is a resulting peak value less than LX it is discarded. If after this process there is a sequence without peaks, the record is rejected. The number of peaks is further reduced by comparing primary and secondary (if any) peaks in each sequence. If $\rho_{pri}/\rho_{sec} > 1.8$, then ρ_{sec} is discarded.

At this point there are only one or two peaks left in each sequence. If there is only one significant peak left in each cross correlation sequence, the NTD must be less than 0.3 for the calculation to continue; if two significant peaks in one cross correlation and one in each of the others, then one of the two combinations of peaks must have an NTD < 0.2 or the data are rejected; if two significant peaks in two or more cross correlations, then the calculation continues only if the NTD of the greatest peaks is less than 0.1 . If the data have not been rejected, the result is three time delays and three maximum correlation values.

g) Least squares fit to times

Since seven measured values are available to determine six FCA parameters, there is one degree of freedom. The derivation of the PMFCA (Section 4, pg.73) has assumed that the NTD=0 (as it must be if the assumptions for FCA are

satisfied). To make NTD=0 a weighted least squares fit (see Section 3, pg.72) of an apparent velocity vector to the time delays is made, which is then translated back into time delays. Since the peaks were chosen on the basis of low NTD, the resulting time delays will be shifted only slightly from the originals. The original ρ_{max} values are retained.

h) PMFCA

The PMFCA method is based on the Gaussian correlation assumption (e.g. Fedor,1967). Under this assumption it can be shown that the widths of the auto and all cross correlation peaks (e.g. at half the peak value) are equal (Meek,1978). Thus the width of the auto correlation theoretically defines all the widths - and is used for this purpose since it is better defined under experimental conditions than the widths of the cross correlation peaks. The remaining unknowns, for a complete definition of the correlation function (and thus the pattern parameters, velocity etc.),are the magnitudes of the three cross correlation peaks, and two of the time delays (the third being defined by NTD=0). The solution for the FCA parameters in terms of these measured values is derived in Section 4 (pg.73).

i) Output

Here the output **variables** are put into the 16 specified memory locations which the machine language routine sends back to the Apple. If there is no solution, the bytes reserved for FCA parameters are zeroed.

2. Cycling correction

Since all antennas are not sampled simultaneously, "0 lag" is only a nominal term in the Apple program, and a correction to the time lag (e.g. for maximum cross correlation) is required. Fig. 1 (below) shows the transfer of a particular correlation value to the Pet. The antenna cycling is in the order #1,2,3,4 at a rate of 7.5 Hz (.133 sec between antennas). Now suppose there is

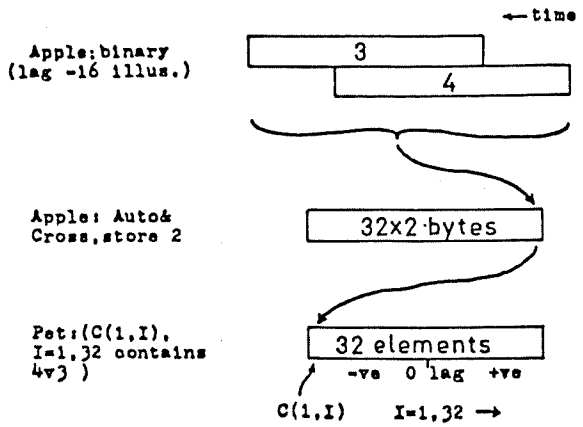


Fig. 1

a peak in the 4v3 cross correlation exactly at zero lag (nominal). Since antenna #3 is sampled 0.133 sec before #4, this means that the 'event' must have occurred in #3 and moved to #4 in 0.133 sec (although nominally they occur simultaneously)- i.e. the pattern moves from #3 to #4. Thus if t'_{34} (the time taken for the pattern to move from #3 to #4) is the nominal value, the corrected value is:

$$t_{34} = t'_{34} + 0.133 \text{ sec} ,$$

similarly: $$t_{24} = t'_{24} + 2\text{x}0.133 \text{ sec},$$

and: $$t_{14} = t'_{14} + 3\text{x}0.133 \text{ sec} .$$

But, by the definition of negative lag in the sketch, we are actually measuring t'_{43} etc. (not t'_{34}) - since a peak at negative lag indicates that the 'event' happened in #3 before #4. Therefore, since $t'_{43} = - t'_{34}$, the required correction is:

$$t_{43} = t'_{43} - 0.133 \text{ sec} \quad \text{etc.}$$

3. Weighted least squares fit to times

If the antenna pairs are defined by (d_i, ψ_i), where the time delay t_i (given by the position of the maximum in the cross correlation) is taken to be positive if it indicates pattern motion in the direction ψ_i; then the delays for a particular apparent velocity (V_a, ϕ_a) are given by:

$$t_i = \frac{d_i}{V_a} \cos(\psi_i - \phi_a)$$

$$= \frac{\cos \phi_a}{V_a}(d_i \cos \psi_i) + \frac{\sin \phi_a}{V_a} (d_i \sin \psi_i) .$$

Let

$$\beta = \begin{bmatrix} \frac{\cos\phi_a}{V_a} \\ \frac{\sin\phi_a}{V_a} \end{bmatrix}, \quad Y = \begin{bmatrix} t_1 \\ \cdot \\ \cdot \\ \cdot \\ \circ \\ t_n \end{bmatrix}, \quad X = \begin{bmatrix} d_1\cos\psi_1 & d_1\sin\psi_1 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ d_n\cos\psi_n & d_n\sin\psi_n \end{bmatrix}.$$

and let ρ_i be the peak values of cross correlation (at delay t_i). (Usually there will be only n=3 pairs of antennas involved). Then the solution for a least squares fit of apparent velocity to time delays is (e.g. Hoel et al.,1971 Ch. 4):

$$\beta = (X^T X)^{-1} X^T Y$$

This solution minimizes the squared error in the time delays, which is

$$\Sigma e^2 = (X\beta - Y)^T (X\beta - Y).$$

If X and Y are modified according to:

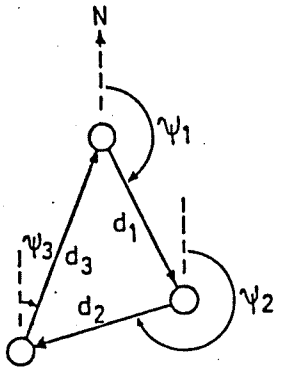
$$x_{ij} = x_{ij}\rho_i$$

$$y_i = y_i\rho_i$$

before solving for β, the squared error in time delays is effectively weighted by ρ_i^2. Thus the final apparent velocity vector will agree best with the time delay corresponding to the highest cross correlation value, and this delay will be modified the least when the apparent velocity is translated back into time delays.

4. Poor man's FCA

This section derives a set of equations for determining FCA parameters from just the positions, t_{max_i}, and the values of the maxima, ρ_{max_i}, in the three cross correlations, and one point, (t_a, ρ_a), from the mean auto correlation. The assumptions are that the correlation function is Gaussian, and the NTD=0.



Antenna drray

Fig. 1

Fig. 1 describes the general antenna arrangement. The direction given to each pair defines the sign of t_{max_i}; i.e. a pattern moving in the direction of the arrow gives a positive delay.

The auto correlation is given by:

$$\rho(0,0,t) = \exp\left(-\tfrac{1}{2}\left[V^2\left(\frac{\cos^2(\phi-\theta)}{a^2} + \frac{\sin^2(\phi-\theta)}{b^2}\right) + \frac{1}{c^2}\right]t^2\right) \quad \ldots(1)$$

Let

$$Q = V^2\left(\frac{\cos^2(\phi-\theta)}{a^2} + \frac{\sin^2(\phi-\theta)}{b^2}\right) + \frac{1}{c^2} \quad . \quad \ldots(2)$$

Then $$\rho(0,0,t) = \exp\left(-\tfrac{1}{2}Qt^2\right), \quad \ldots(3)$$

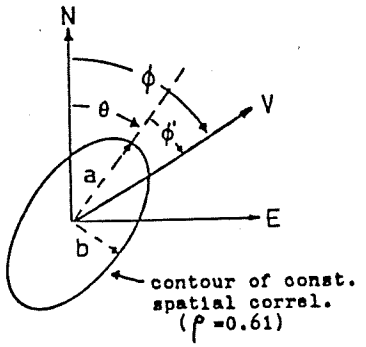
and Q can be found from one point on the auto correlation:

$$Q = \frac{-2\ln \rho_a}{t_a^2} \quad . \quad \ldots(4)$$

The cross correlations are given by:

$$\rho(d_i,\psi_i,t) = \exp\left[-\tfrac{1}{2}\left(\frac{(d_i\cos(\psi_i-\theta)-V\cos(\phi-\theta)t)^2}{a^2} + \frac{(d_i\sin(\psi_i-\theta)-V\sin(\phi-\theta)t)^2}{b^2} + \frac{t^2}{c^2}\right)\right] \quad \ldots(5)$$

where the antenna pair and direction are defined by d_i and ψ_i as shown in Fig. 1. The other parameters are the axes of the characteristic ellipse(a,b), the characteristic time (c), the true velocity (V), the tilt angle (θ) of the ellipse, and the drift direction



FCA parameters

Fig. 2

(ϕ). These are shown in Fig. 2.

Then for the i^{th} antenna pair, t_{max_i} is found by setting the derivative of (5) equal to zero.

$$t_{max_i} = \frac{Vd_i}{Q}\left(\frac{\cos(\phi-\theta)\cos(\psi_i-\theta)}{a^2} + \frac{\sin(\phi-\theta)\sin(\psi_i-\theta)}{b^2}\right) \qquad ...(6)$$

This is substituted into (4) to get ρ_{max_i}:

$$-2\ln\rho_{max_i} = d_i^2\left(\frac{\cos^2(\psi_i-\theta)}{a^2} + \frac{\sin^2(\psi_i-\theta)}{b^2}\right) - Qt^2_{max_i} \qquad ...(7)$$

Now define

$$m_i = \frac{-2\ln\rho_{max_i} + Qt^2_{max_i}}{d_i^2}$$

$$= \frac{1}{r^2b^2}\left(\cos^2(\psi_i-\theta) + \sin^2(\psi_i-\theta)r^2\right) \qquad ...(8)$$

where $r = a/b$, and let $m_{ij} = m_i - m_j$.

Then the solution for the pattern parameters is:

$$\left.\begin{aligned} \tan(2\theta) &= \frac{m_{32}\cos 2\psi_1 + m_{13}\cos 2\psi_2 + m_{21}\cos 2\psi_3}{-(m_{32}\sin 2\psi_1 + m_{13}\sin 2\psi_2 + m_{21}\sin 2\psi_3)} \\ \frac{r^2}{1-r^2} &= \frac{m_1\cos^2(\psi_2-\theta) - m_2\cos^2(\psi_1-\theta)}{m_{21}} \\ b^2 &= \frac{\frac{3}{2}\left[\frac{1}{r^2}+1\right]}{m_1+m_2+m_3}, \qquad a^2 = b^2r^2 \end{aligned}\right\} \qquad ...(9)$$

The drift velocity is found from equation (6) for i=1,2.

$$\tan(\phi-\theta) = \frac{t_1d_2\cos(\psi_2-\theta) - t_2d_1\cos(\psi_1-\theta)}{t_2d_1\sin(\psi_1-\theta) - t_1d_2\sin(\psi_2-\theta)}\left(\frac{b^2}{a^2}\right) \qquad ...(10)$$

$$V = \frac{Qt_1}{d_1\left[\frac{\cos(\phi-\theta)\cos(\psi_1-\theta)}{a^2} + \frac{\sin(\phi-\theta)\sin(\psi_1-\theta)}{b^2}\right]} \qquad \ldots(11)$$

where $t_i = t_{max_i}$. Then c is found from equation (1):

$$c = \sqrt{\frac{1}{Q - V^2\left[\frac{\cos^2(\phi-\theta)}{a^2} + \frac{\sin^2(\phi-\theta)}{b^2}\right]}} \qquad \ldots(12)$$

If a^2, b^2, or c^2 is negative, the FCA solution must be rejected as physically impossible.

Table 1 gives an example for testing the programmed solution. Note that the quoted V (drift) is the pattern velocity (from eqn. 11) divided by 2.

Table 1.

antenna parameters: $d_1=d_2=d_3=156$m; $\psi_1=210^\circ$, $\psi_2=90^\circ$, $\psi_3=330^\circ$

input data	solution
$t_a=1.066$ sec	$Q = 0.53$
$\rho_a=0.74$	$a = 97.4$m
$t_{max_i}=-2.05$s, 2.84s, -.79s	$b = 56.0$m
$\rho_{max_i}=0.17, 0.25, 0.31$	$\theta = 158^\circ$
	$V = 16.6$m/s (drift)
	$\phi = 90^\circ$
	$c = 2.2$s

6502 CPU

INSTRUCTION SET – OP CODES, Execution Time, Memory Requirements

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED			(IND,X)			(IND),Y			Z,PAGE,X			ABS,X			ABS,Y			RELATIVE			INDIRECT			Z,PAGE,Y			CONDITION CODES					
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
ADC	A+M+C → A (4) (1)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										✓	✓	✓	–	–	✓
AND	A∧M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										✓	✓	–	–	–	–
ASL	C ← 7 … 0 ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													✓	✓	✓	–	–	–
BCC	BRANCH ON C=0 (2)																															90	2	2							–	–	–	–	–	–
BCS	BRANCH ON C=1 (2)																															B0	2	2							–	–	–	–	–	–
BEQ	BRANCH ON Z=1 (2)																															F0	2	2							–	–	–	–	–	–
BIT	A∧M				2C	4	3	24	3	2																															M7	✓	–	–	–	M6
BMI	BRANCH ON N=1 (2)																															30	2	2							–	–	–	–	–	–
BNE	BRANCH ON Z=0 (2)																															D0	2	2							–	–	–	–	–	–
BPL	BRANCH ON N=0 (2)																															10	2	2							–	–	–	–	–	–
BRK	(See Fig 1)													00	7	1																									–	–	–	1	–	–
BVC	BRANCH ON V=0 (2)																															50	2	2							–	–	–	–	–	–
BVS	BRANCH ON V=1 (2)																															70	2	2							–	–	–	–	–	–
CLC	0 → C													18	2	1																									–	–	0	–	–	–
CLD	0 → D													D8	2	1																									–	–	–	–	0	–
CLI	0 → I													58	2	1																									–	–	–	0	–	–
CLV	0 → V													B8	2	1																									–	–	–	–	–	0
CMP	A–M (1)	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										✓	✓	✓	–	–	–
CPX	X–M	E0	2	2	EC	4	3	E4	3	2																															✓	✓	✓	–	–	–
CPY	Y–M	C0	2	2	CC	4	3	C4	3	2																															✓	✓	✓	–	–	–
DEC	M–1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													✓	✓	–	–	–	–
DEX	X–1 → X													CA																											✓	✓	–	–	–	–
DEY	Y–1 → Y													88																											✓	✓	–	–	–	–
EOR	A ⊻ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										✓	✓	–	–	–	–
INC	M+1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													✓	✓	–	–	–	–
INX	X+1 → X													E8	2	1																									✓	✓	–	–	–	–
INY	Y+1 → Y													C8	2	1																									✓	✓	–	–	–	–
JMP	JUMP TO NEW LOC.				4C	3	3																												6C	5	3				–	–	–	–	–	–
JSR	(See Fig 2) JUMP SUB				20	6	3																																		–	–	–	–	–	–
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										✓	✓	–	–	–	–

CMP: C=1 if M≤A
C=0 if M>A

MNEMONIC	OPERATION	IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM.			IMPLIED			(IND,X)			(IND),Y			Z,PAGE,X			ABS,X			ABS,Y			RELATIVE			INDIRECT			Z,PAGE,Y			CONDITION CODES					
		OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3										B6	4	2	✓	✓	–	–	–	–
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3												✓	✓	–	–	–	–
LSR	0→ 7 6 5 4 3 2 1 0 →C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3												0	✓	✓	–	–	–
NOP	NO OPERATION													EA	2	1																								–	–	–	–	–	–
ORA	AVM → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										✓	✓	–	–	–	–
PHA	A→ Ms S-1→S													48	3	1																								–	–	–	–	–	–
PHP	P→ Ms S-1→S													08	3	1																								–	–	–	–	–	–
PLA	S+1→S Ms→A													68	4	1																								✓	✓	–	–	–	–
PLP	S+1→S Ms→P													28	4	1																								(RESTORED)					
ROL	7 6 5 4 3 2 1 0 C				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3												✓	✓	✓	–	–	–
ROR	C 7 6 5 4 3 2 1 0				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3												✓	✓	✓	–	–	–
RTI	(See Fig. 1) RTRN INT.													40	6	1																								(RESTORED)					
RTS	(See Fig. 2) RTRN SUB													60	6	1																								–	–	–	–	–	–
SBC	A-M-$\bar{C}$ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3									✓	✓	(3)	–	–	✓
SEC	1 → C													38	2	1																								–	–	1	–	–	–
SED	1 → D													F8	2	1																								–	–	–	–	1	–
SEI	1 → I													78	2	1																								–	–	–	1	–	–
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3									–	–	–			
STX	X → M				8E	4	3	86	3	2																														96	4	2	–	–	–	–	–	–
STY	Y → M				8C	4	3	84	3	2													94	4	2																–	–	–	–	–	–
TAX	A → X													AA	2	1																								✓	✓	–	–	–	–
TAY	A → Y													A8	2	1																								✓	✓	–			–
TSX	S → X													BA	2	1																								✓	✓	–		–	
TXA	X → A													8A	2	1																								✓	✓		–		–
TXS	X → S													9A	2	1																								–	–				
TYA	Y → A													98	2	1																								✓	✓	–	–	–	–

(1) ADD 1 TO "N" IF PAGE BOUNDRY IS CROSSED

(2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE.
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE.

(3) CARRY NOT = BORROW.

(4) IF IN DECIMAL MODE Z FLAG IS INVALID.
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT.

X INDEX . X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
Ms MEMORY PER STACK POINTER

• ADD
– SUBTRACT
∧ AND
∨ OR
⊻ EXCLUSIVE OR
✓ MODIFIED
– NOT MODIFIED
M_7 MEMORY BIT 7
M_6 MEMORY BIT 6

N NO. CYCLES
NO. BYTES