AN IMAGE DIGITIZING SYSTEM FOR A SCANNING LASER ACOUSTIC MICROSCOPE

BY

STEVEN GERALD FOSTER

B.S., Illinois Institute of Technology, 1978

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1981

Urbana, Illinois

Acknowledgements

The author wishes to thank Professor W. D. O'Brien for suggesting the topic for this thesis, and for his advice and help during the work performed. Thanks are also due Dr. Ronald Johnston and Dave Duback for assistance in programming matter. Mrs. Wanda Elliott provided much assistance in the prepararation of this thesis.

A special note of thanks to my loving wife, Bessie, for without her constant support the thesis would not have been finished.

TABLE OF CONTENTS

CHAPTER	Contents	Page
1	Introduction	1
2	DATA AQUISITION SYSTEM THEORY OF OPERATION	5
	2.1 Introduction	5
	2.2 Amplifier Theory of Operation	7
	2.3 Digitiziter Specifications	13
	2.4 Signal Flow in the Digitizer	18
	2.5 Programming the Digitizer	23
	2.6 Digitizer Theory of Operation	26
3	PROGRAMMING SECTION	49
	3.1 Introduction	49
	3.2 A/D Value to LED Display on Control Panel	49
	3.3 Capture a Line	51
	3.4 Digitize Image	54
	3.5 Interference Line Enhancement	59
	3.6 Output Image to Printer	63
4	RESULTS	70
	4.1 Introduction	70
	4.2 Comparision of Different Data Extraction Techniques	70
	4.3 Histogram Equalization	73
	4.4 Effects From Varing Correlated Filter Length	78
5	IMPROVEMENTS	85
REFERENCES		86
Appendix A		87

CHAPTER 1

1.1 Introduction

In the early 1900's, X rays emerged as an important energy source in medical imaging in many areas such as in identifying bone fractures and locating metallic fragments. A half century later another form of energy is emerging - the acoustic wave (mostly at ultrasonic frequencies) and is having an important effect upon medical imaging. This new science can yield new and complementing information on tissues that are not easily imaged by other techniques. Two important ultrasonic properties of tissues for biological ultrasound are speed and attenuation both of which can yield unique information regarding the tissue. Both of these properties must be catalogued for a variety of normal and pathological tissues and to this end, this thesis was undertaken wherein automated procedures are made possible to obtain such data more accurately and with greater speed than has been possible.

The Bioacoustics Research Laboratory at the University of Illinios at Urbana-Champaign has a scanning laser acoustic microscope (SLAM) operating at an ultrasonic frequency of 100 MHz. The SLAM (Sonomicroscope 100, Sonoscan Inc, Bensenville, IL 60106) is used in part to study the fundamental mechanisms by which sound and tissues interact. Tissue is usually thinly sliced and placed on the microscope stage. A liquid medium (usually water or saline) is used as a coupling medium and a specially devised cover slip is placed on top of the tissue specimen. Now the operator

chooses one of the three operating modes; optical, acoustical, and interference mode.

The optical mode displays a 2mm x 3mm field of view of the specimen on a monitor to assist the operator in finding the desired section of tissue to work with. The image monitor is a standard black and white television picture. The specimen is scanned from above by a laser (in spatial and temporal synchronization with the monitor raster) while underneath a photo detector detects the varing amount of light getting through. The output of the photo detector is processed into a video signal and sent to the monitor.

In the acoustic image the dark areas correspond to high ultrasonic attenuation and light portions to regions of low attenuation. The acoustic mode uses the gold film on the cover slip (the side touching the specimen) to relect the laser to a different photo detector. The sound impinging on the gold film distorts the film causing the reflected laser to misalign with the photo detector resulting in modulation. Areas of low ultrasonic attenuation allow more acoustic energy through resulting in more distortion.

The video from the interference mode is produced similarly to the acoustic except the video from the the detector is mixed with a reference frequecy to detect phase changes in the detected video. The interference mode consists of 39 vertical lines which represent the spatial distribution of areas of changing velocity. These are difficult to resolve to a high degree of accuracy with the eye and are the subject of a later discussion (Chapter 4).

In water, the interference lines are straight since water is homogeneous. When a slice of liver is placed in the water, the interference lines will shift to the right at the water/liver border (indicating liver has a higher speed of sound) and when leaving the tissue entering the water they shift back. The interference lines in the liver are slightly bumpy due to the nonuniformity of the tissue (blood vessels, varying width of the slice, etc.). In skin, which is very heterogeneous the interference lines are ragged and in places dissapprear (O'Brien et al, 1981).

In order to extract meaningful ultrasonic information in an automated procedure, large amounts of data must be taken. To this end, the image is digitized and stored by the computer. The bulk of this thesis is devoted to the theory of operation of the system which digitizes the image from the acoustic microscope. Also included are preliminary steps which have been taken towards analysis of the image.

Digitized images from the SLAM are stored and manipulated by a 32-bit mini-computer (Perkin-Elmer Model 7/32, hereinafter referred to as the "7/32"). In addition to 384 kilobytes of static memory, two tape drives and a 10 megabyte disc unit serve to store programs as well as data obtained from the microscope. Programs are written in FORTRAN (Level VI); a useful feature of the FORTRAN compiler is the provision for the use of in-line assembly-language code. A significant decrease in execution times is achieved with this option for the sections of code which actually interact with the data-acquisition system, and for a

correlator-receiver which improves the signal-to-noise ratio of digitized data.

Soon after the digitizing system was developed the opportunity to apply part of the system to research on the detection of nodules in the larynx was required wherein audio signals were digitized. This research required continuous conversion of audio information at 110 kHz, i.e., digitize and store on magnetic tape speech lasting several minutes (perhaps 30 megabytes of data) in one playing of the recorded speech. One circuit card was completely redesigned and several modifications were made on other boards to reach this end. Also a complicated program was written to achieve 110 kHz transfer rate to the magnetic tape unit.

CHAPTER 2

DATA AQUISITION SYSTEM THEORY OF OPERATION

2.1 Introduction

Figure 2-1 shows a block diagram of the data acquisition system (DAS). The DAS consists of an r-f amplifier, a 30 MHz analog-to-digital (A/D) converter, and a high-speed buffer memory. Transfer of data from the A/D occurs first to the buffer (eight bits, 30 MHz); when a complete "line" of a television frame has been digitized, the buffer empties its contents to the 7/32 (approximately 50 kHz rate for two eight-bit bytes/transfer). To facilitate this rapid transfer of data between the buffer and the processor, the A/D and buffer are located near the 7/32. The acoustic microscope is located eighty meters from the 7/32.

The DATA AQUISITION SYSTEM is constructed of two separate devices; the rf amplifier and the digitizer. The AMPLIFIER serves the following purposes: (1) to drive the SLAM monitor and DIGITIZER simultaneously from one accessable video connector on the SLAM, (2) to maintain the video signal in the appropriate range for the A/D, (3) to convert the sync pulses of the SLAM to TTL compatible levels and (4) to give future ability to type characters on the monitor. The amplifier is local to the SLAM.

At the receiving end of the coaxial line, is the DIGITIZER whose job is to digitize the video or audio signal depending on the selected mode. In the video mode, the signal is digitized and stored in the fast memory buffer until the computer can access the digitized sample. In the

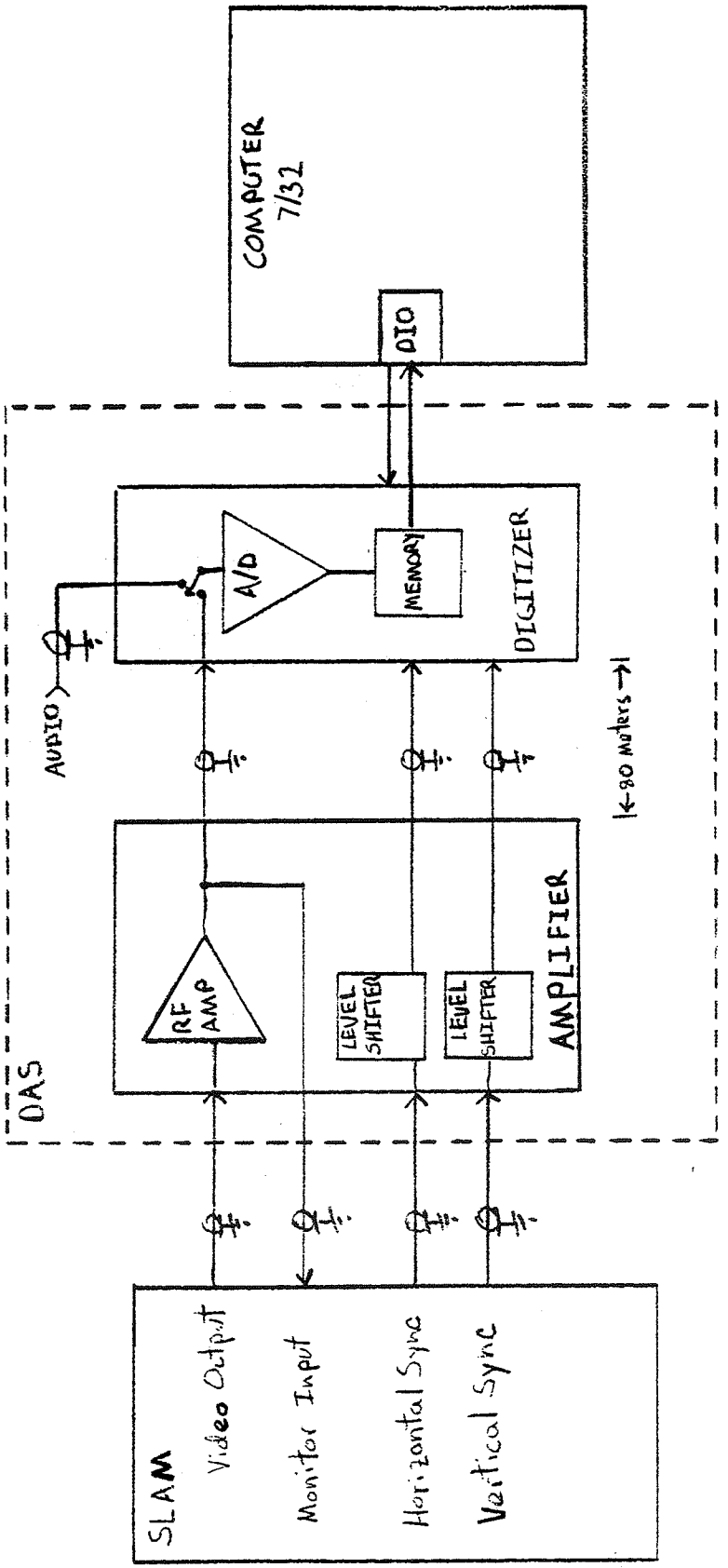


Figure 2-1: Block diagram of DAS.

audio mode, the four samples are gathered in such a way that the computer can read the digitized samples in a short burst allowing maximum speed for storage onto the magnetic tape and therefore semi-infinite storage. The DIGITIZER consist of a modular power supply, a high speed A/D (8 bits at 30 MHz), a fast buffer memory (1024 X 16, 35 ns access time), a programmable stop delay (up to 65536 clock cycles), a programmable video line detector, and controlling logic.

2.2 Amplifier Theory of Operation

The amplifier is powered by a customized power supply (figure 2-2) consisting of a 750 mA transformer with diodes and capacitors to yield a positive and negative 30 volt unregulated supply which, in turn feeds a preregulator employing both positive and negative integrated circuit voltage regulators to give an output voltage of +/-24 volts. The purpose of the preregulator is to reduce the three volt ripple on the unregulated supply to less than two millivolts. Various integrated regulators are connected to the output of the preregulator to their outputs.

The horizonal and vertical sync pulses coming from the SLAM are -3 volt / 0 volt levels and must be converted to TTL levels and drive the coaxial lines to the DIGITIZER. The horizonal and vertical sync signals are processed identically in the DAS and therefore only the horizontal signal is discussed. Referring to figure 2-3, the horizontal sync signal enters the amplifier at the point where three a resistor network translates the sync signal to TTL levels for the inverter which, in turn, drives both the 93 ohm coaxial

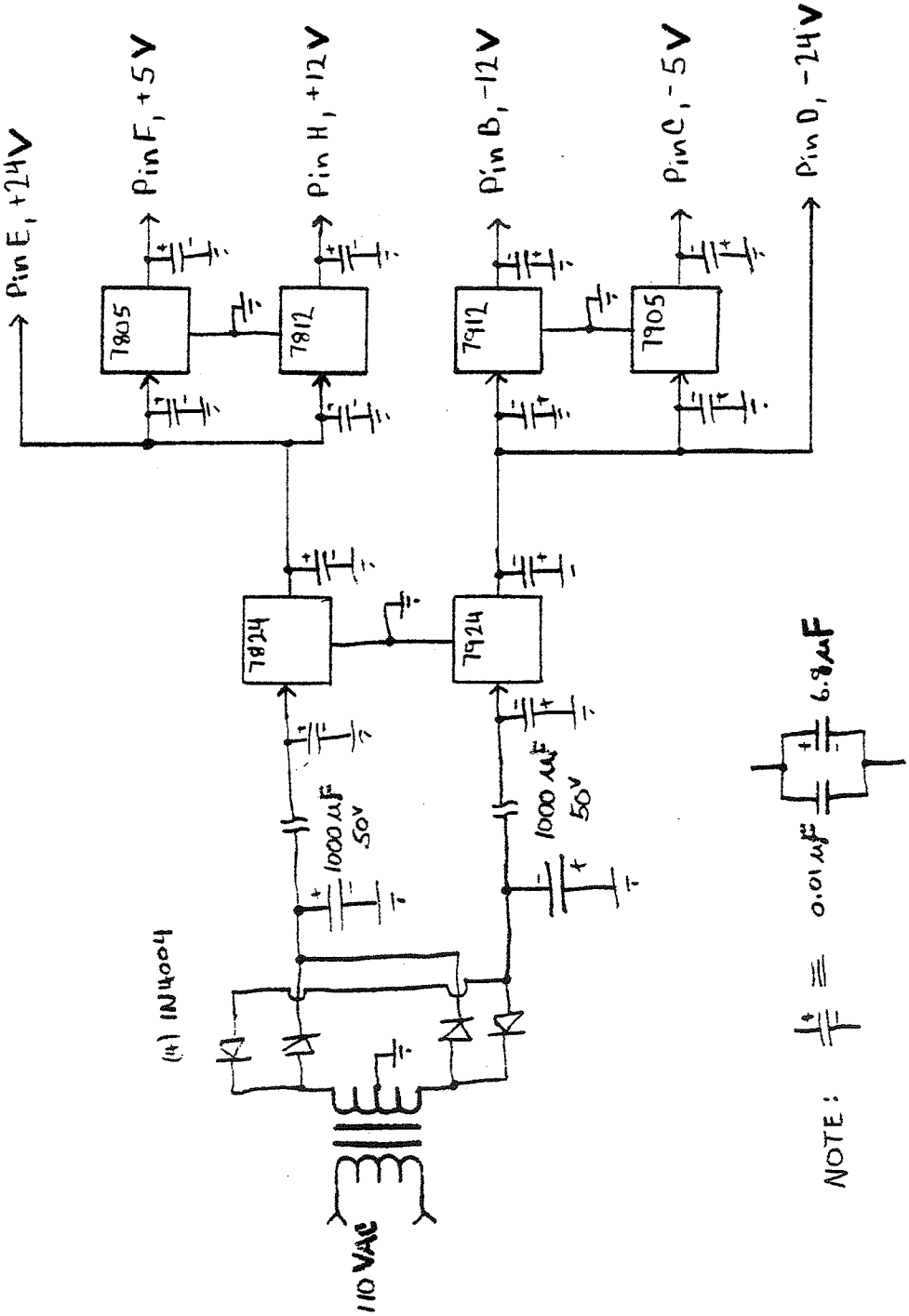
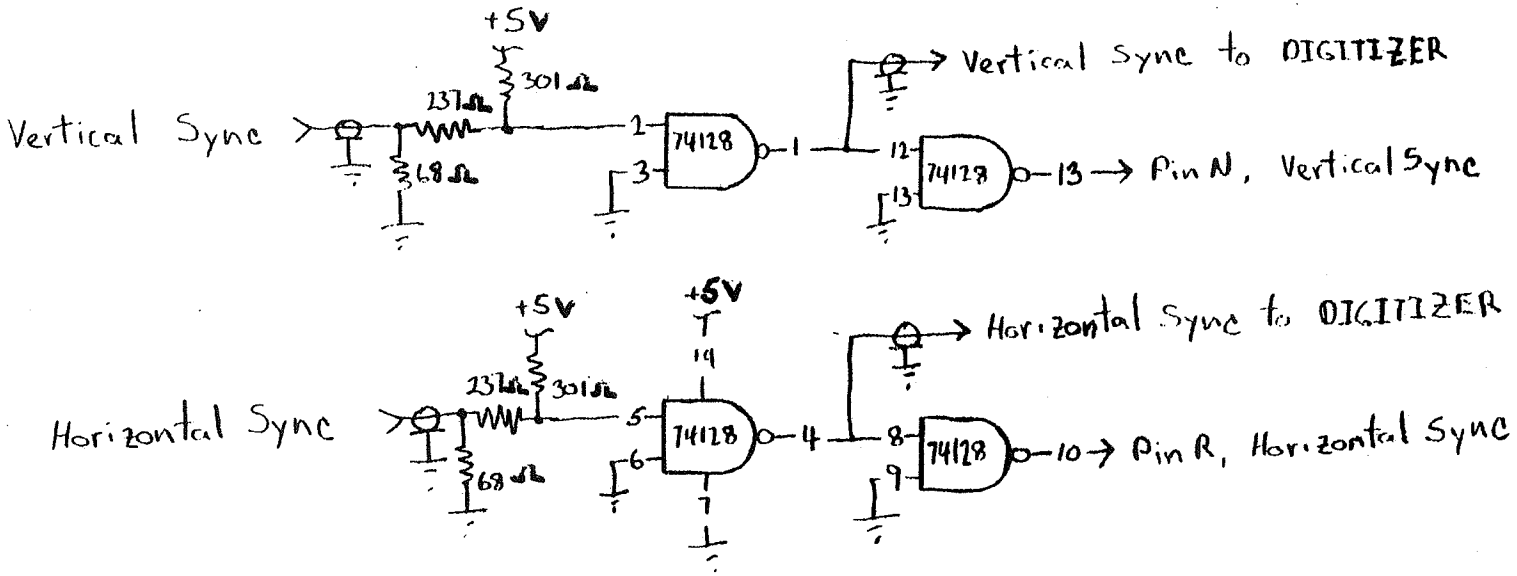


Figure 2-2: AMPLIFIER power supply.



Pin A, Ground

Pin F, +5V

Figure 2-3: Level shifter and driver.

line and an additional inverter. The coaxial line carries the sync pulse to the DIGITIZER. The output of the second inverter drives the character generator board.

Future expansion for characters generation is provided and therefore it is necessary to inject characters into the video. The letters are simply TTL level pulses that print spots of light if the pulse is high. TTL levels are not compatible with the video so the signal is translated down and attentuated. If TTL levels are placed on the video line the microscope's AGC circuit would be activated washing out the image, so it is necessary to keep the characters within video levels. This is the purpose of the character circuit shown in fig 2-4. The DG181A is a dual SPST analog switch which chooses either video or characters to pass depending on the state of the enables: pin 5 high selects the video mode, pin 10 the character mode, the two pins cannot be high together. The video is properly terminated in 75 ohms (at pin 2) before the switch, the load after the switch is not significant to affect matching. Signal identification of the characters to be written is applied at pin W and then go through an adjustable attentuator (500 ohm pot) to the switch. The transistor emitter voltage feeding the attentuator is adjustable by the 2K pot which is set for the desired offset for the characters.

The output of the two switches (S1 and S2 of DG181A) are tied together and go to both the microscope amplifier and the DIGITIZER amplifier. The microscope amplifier (figure 2-4) consists of two common emitter stages directly coupled to

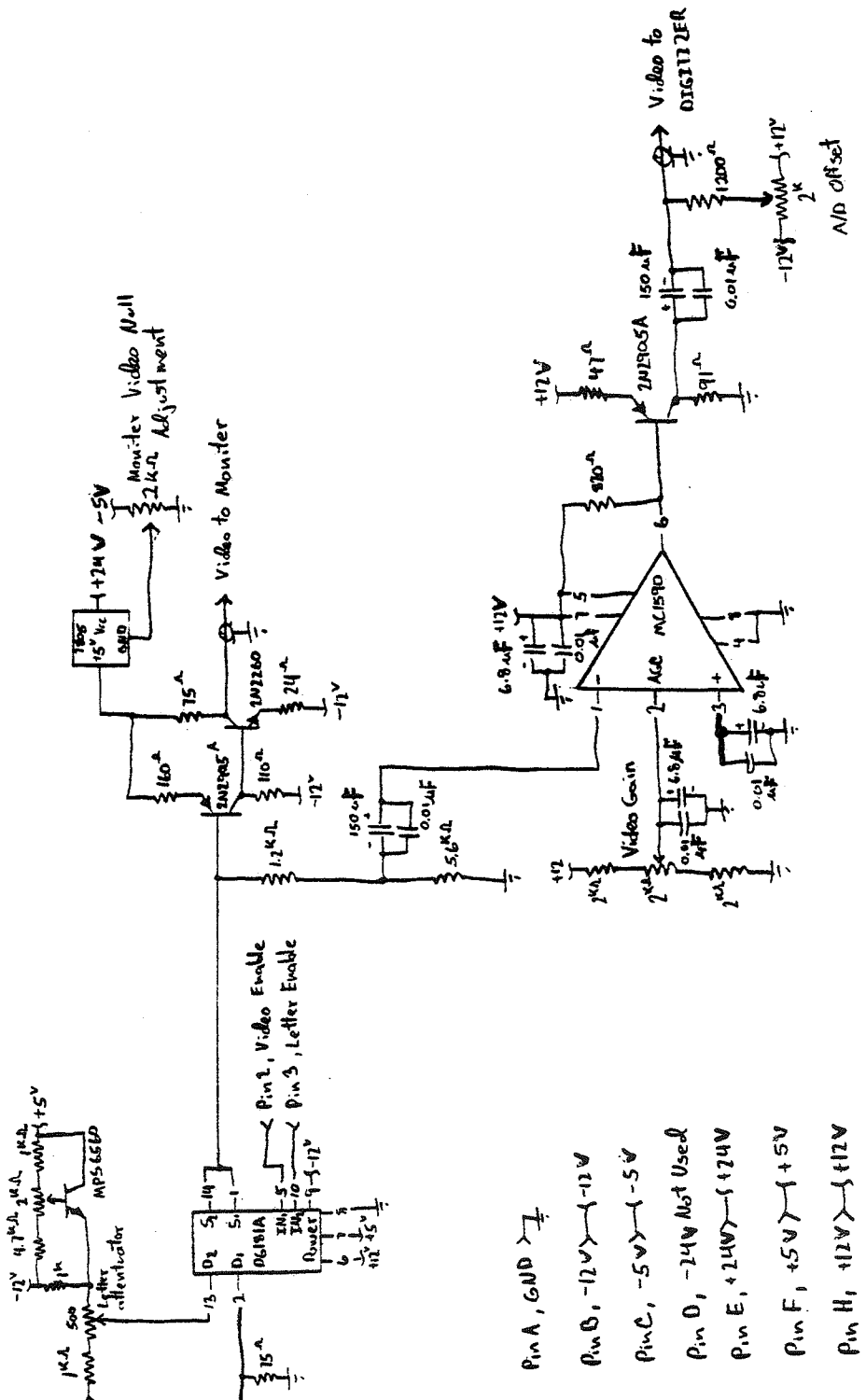


Figure 2-4: Video amplifier and character injection circiut.

provide enough gain so that the video entering the amplifier is identical to that leaving. This allows for the amplifier to be invisible to the SLAM. This amplifier's output impedance is 75 ohm to match the coaxial cable. The residual dc offset errors are nulled out by adjusting the positive supply with the 2 kiloohm pot.

The DIGITIZER amplifier drives the 93 ohm coaxial cable to the DIGITIZER and also have adjustable gain and offset. The heart of this circuit is the integrated video amplifier (MC1590). For input signal levels of less than 300 mV RMS, the chip has the capability for gain adjust. Since the video signal of a reasonable image is near this level, it is attenuated slightly by two resistors, the 1.2 kiloohm and the 5.6 kiloohm working in conjunction with the input impedance of the chip. No attempt is made to process the vertical and horizontal sync pulses in this stage. The circuit is capacitively coupled with a low frequency cuttoff of 1.5 Hz to allow the 60 Hz signals of the image to pass through (the lowest possible signal freguency is 60 Hz since the image is scanned at this rate). The gain adjust comes on the AGC pin (pin 2) of the chip from the simple pot circuit. The output of the MC1590 (pin 6) goes through a common emitter stage to acquire enough power to drive the 93 ohm line. Special care is taken to insure an output impedance of 93 ohm to match the coaxial cable to prevent reflections. The output circuitry is also capacitively coupled for circuit simplicity (direct coupling is possible but costly and tedious). In order to center the video signal at 0 volts an offset circuit is

provided. The offset adjustment is on the output after the capacitors, consisting of the 2 kiloohm pot and the 1200 ohm resistor.

2.3 Digitizer Specifications

The DIGITIZER is powered by two modular power supplies, a logical supply at 5 volts 3 amps, and an analog supply at +/-15 volts 800 mA (see figure 2-5).

The A/D is a printed circuit board (TRW TDC1007PCB) requiring external power (+/-15 volts and +5 volts) . The input video signal must be in the range +/-500 mV for digitization. The A/D operates from dc to 30 MHz; each clock yields a new 8-bit sample (fully parallel). The input impedance is 93 ohm. The format of the 8-bit output is $FF for -500 mV and $00 for +500 mV with decreasing digital value for increasing analog voltage, where the '$' represents hexadecimal. Overranges are represented as $FF and underranges as $00.

The memory buffer of the DIGITIZER store 2048 samples in a circular fashion at a sampling rate of up to 30 MHz. The memory is arranged as 1024x16 bits of fast bipolar RAM (access time 35 ns); each of the 1024 halfwords consists of two samples. This doubling up of samples effectively halves the access time allowing the memory to operate predictably at 30 MHz. The buffer is necessary because when digitizing at frequencies above 110 kHz the computer can not store the information in memory as fast as it is generated. The buffer holds the data so that the computer can read from it at a lower speed. The buffer size is programmable which allows

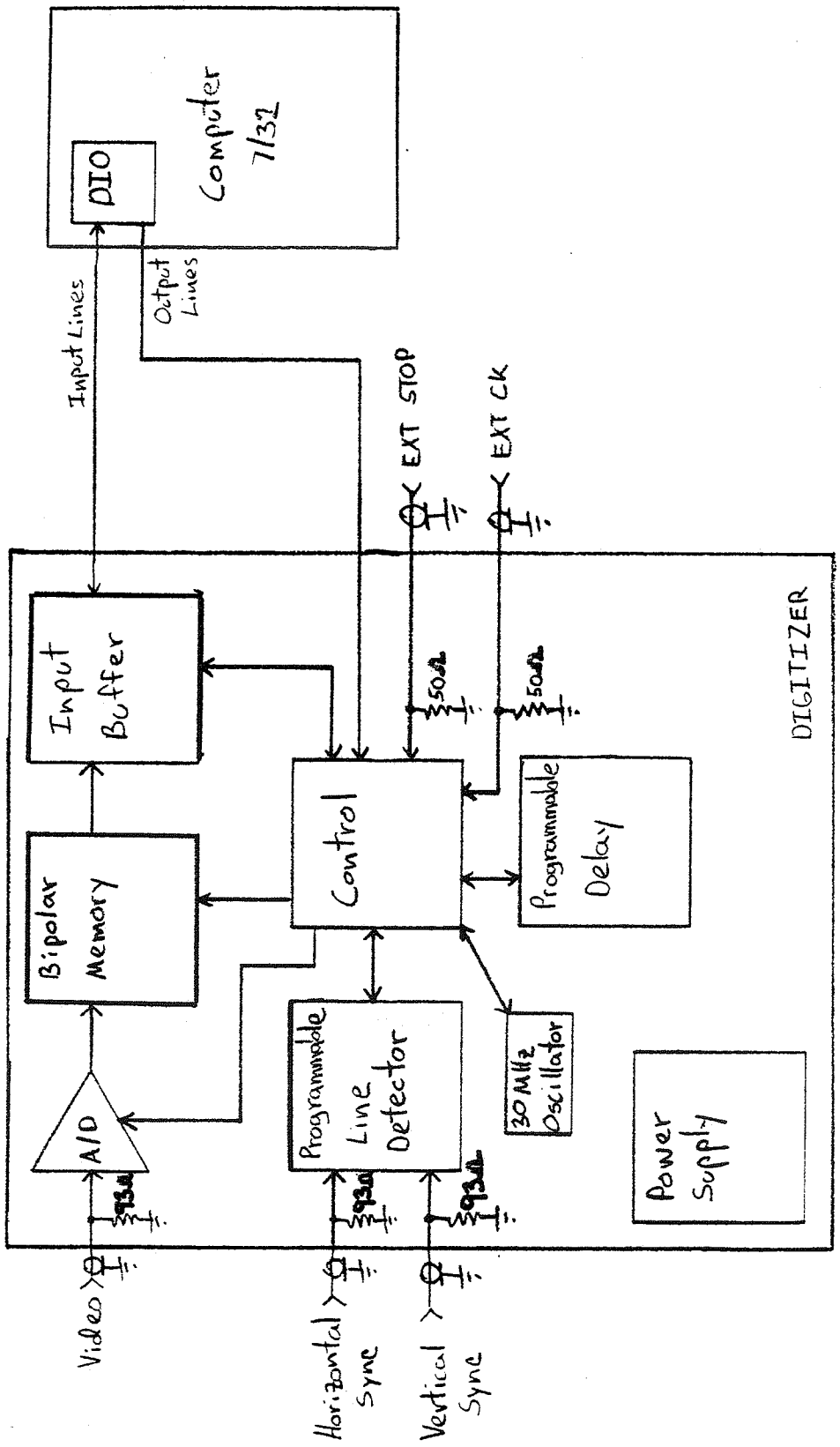


Figure 2-5: Block diagram of DIGITIZER.

the operator to read only the quantity of data desired instead of all 1024 halfwords which otherwise would have been the case with circular memory. The programmed sizes are: 18,34,66,130,258,514,1026,2048.

Digitization of the video is terminated when, depending on which mode is selected through programming, an external stop pulse is applied or the preprogrammed line of the monitor is detected. The DIGITIZER can anticipate the external stop up to the programmed memory size multiplied times the period of the sampling rate. The line detect mode can not anticipate (this feature is not needed for this application). Also the DAS can delay termination for up to 65536 periods of the sampling frequency. The delay uses two periods of the clock as a time base and is programmable from 1 to 32768. Hence the newest data and the oldest data are given respectively by

$$2*\text{Programmed Delay} - 6 + X(t)$$

$$2*\text{Programmed Delay} - \text{Programmed Memory Size} - 5 + X(t)$$

where X(t) is a random varable which is dependent upon the amount of information the operator has about the sychronization of the selected clock and the external stop pulse and to whether an even or an odd number of clock pulse have occurred (parity) since the last programmed reset. Figure 2-6 graphically represents the possible distributions of X(t). When the above formula yields a negative delay the digitizer is anticipating the stop pulse. With the line

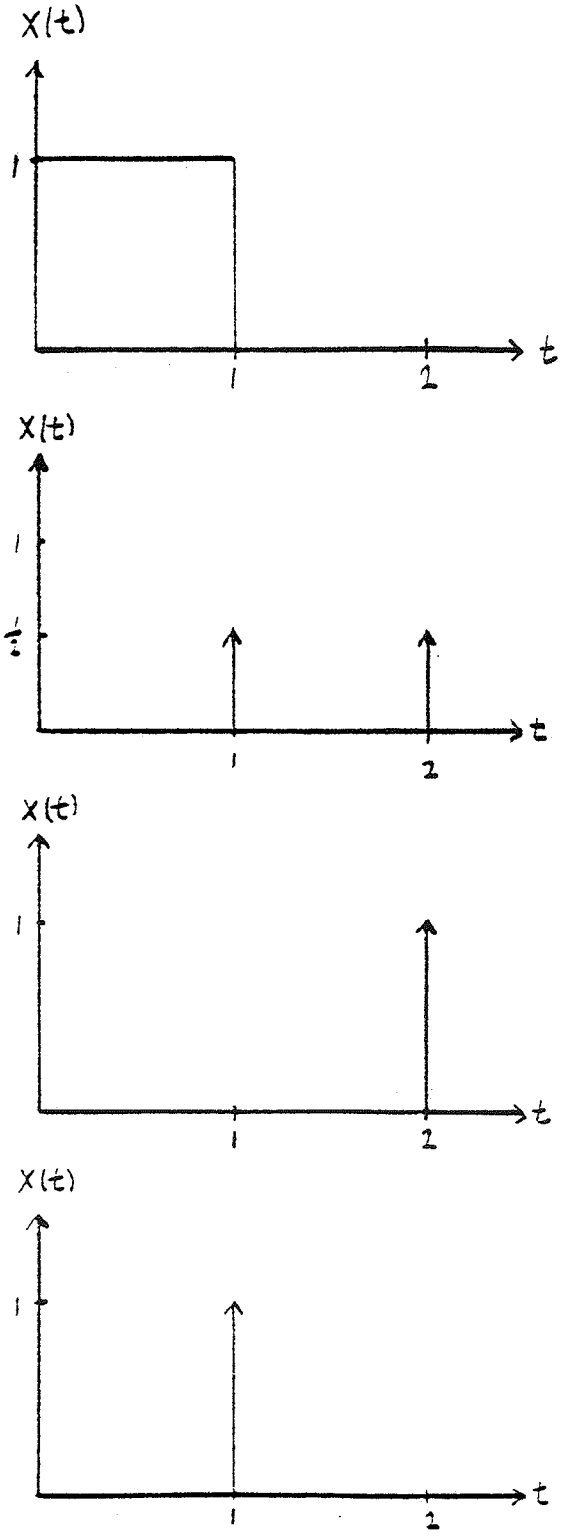
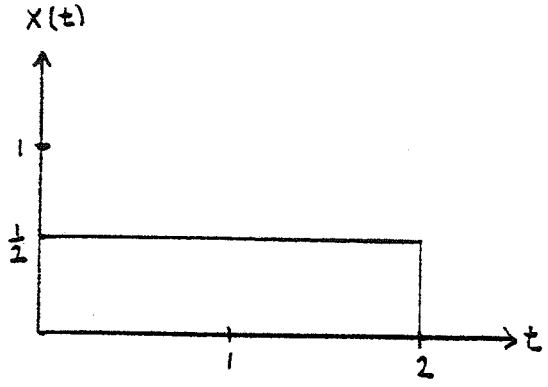
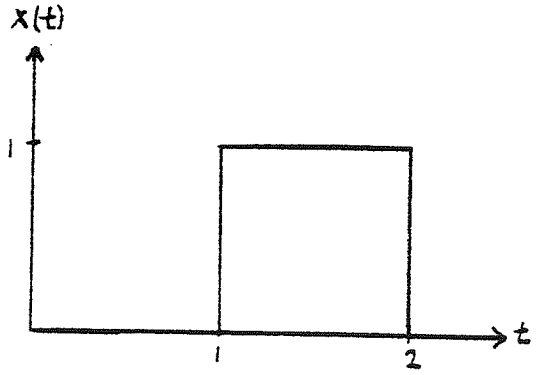


Figure 2-6A: Probability distribution of X(t).



No Synchronization
Unknown Parity



No Synchronization
Even Parity

Figure 2-6B: Probability distribution of X(t).

detect mode selected, synchronization of the 30 MHz clock with the horizonal is guaranteed and no clocks will be gated until the selected raster line occurs. For the line detect mode X(t) = Constant whose value is small and unimportant since trial and error is used to select the programmed delay size.

All of the video raster lines of the monitor except the first and second can be captured and digitized through programming. Detection of the programmed line terminates the digitization process in the same manner as the stop pulse mentioned earlier. To detect a desired line the LINE LATCH in the digitizer must be loaded by this formula (values in hexadecimal):

$802-LINE NUMBER

2.4 Signal Flow in the Digitizer

The signal flow in the DIGITIZER is shown in the block diagram of figure 2-7. Loading of all latches is done by the computer. The computer outputs a halfword with a strobe to the DIGITIZER which decodes the upper byte to select the latch to receive the lower byte. Discussion of the loading of the latches is deferred to section 2-6.

The video mode will be discussed first; the audio mode is relatively simple and is discussed later. Whether the termination of the digitization is through the external stop or the line detect mode the signal flow is the same after the STOP F/F. The setting of this F/F is as follows. A programmed reset first clears this F/F, digitization is

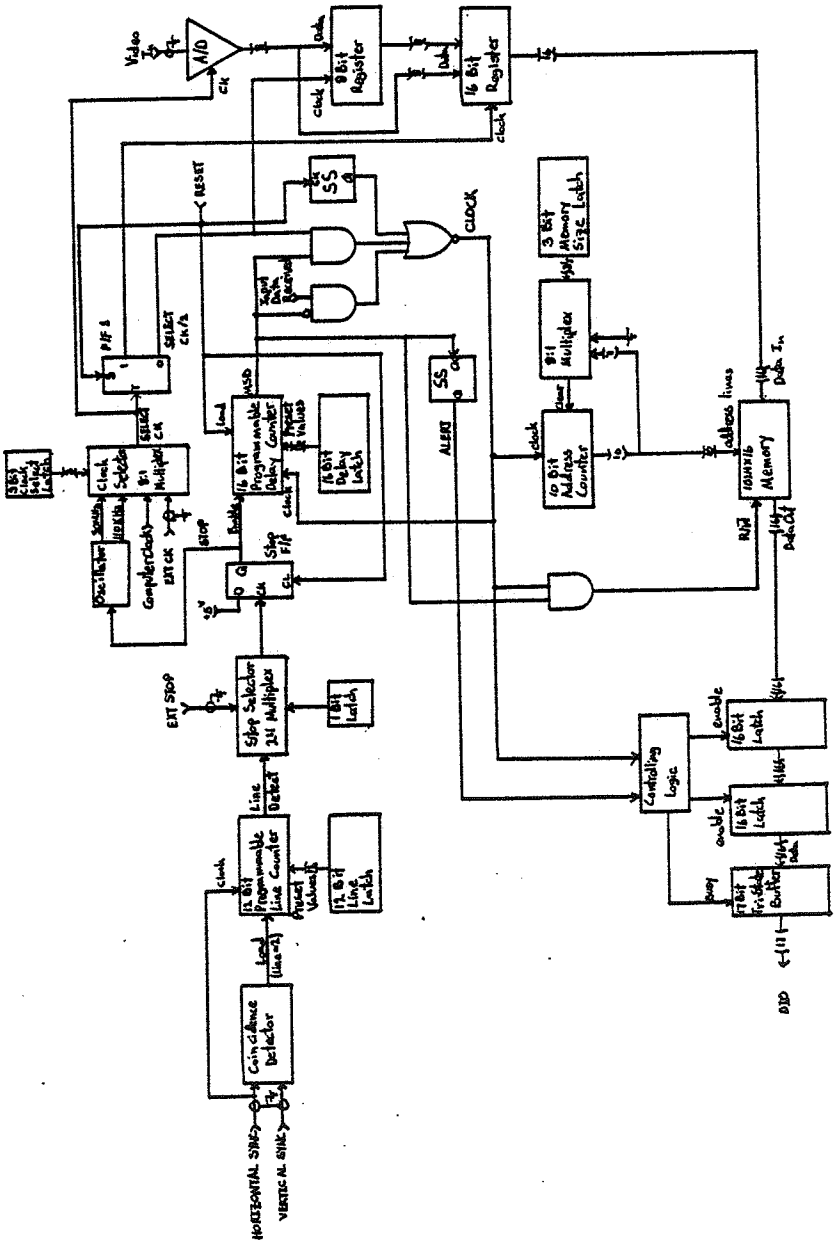


Figure 2-7: Signal flow diagram of the DIGITIZER.

continuous until the F/F is set and the delay has timed out. The EXT STOP signal or the LINE DETECT signal is select to set the STOP F/F through a 2:1 multiplex driven by a preprogrammed latch. If the EXT STOP mode is selected the signal is gated to the F/F and sets it. However the LINE DETECT mode is more difficult. From the microscope sync lines the DIGITIZER looks for the first line using the coincidence circuit. On the next line the 12-BIT PROGRAMMABLE LINE COUNTER is loaded. This counter then counts (up counts) lines until the most significant bit sets (MSB), which is then gated to the STOP F/F. Lines begin with a horizontal pulse, hence counting of lines is done by counting these sync pulses.

Following the setting of the STOP F/F one of the two enables to the DELAY COUNTER goes high, the other was initially high since the MSB is required to be set through programming before the digitization process is to begin. The counter is clocked by the SELECTED CLOCK / 2 which is gated through since the MSB was set. When the counter overflows the MSB goes low inhibiting further write pulse, terminating storage in memory. The falling transistion of the MSB triggers a single shot that alerts the computer that the DIGITIZER is finished. In addition to the above the STOP F/F enables the INPUT DATA RECIEVED pulse from the computer to increment the MEMORY ADDRESS COUNTER to access the stored data, while inhibiting the selected clock from scrambling the counter.

After the computer is alerted of termination it reads a

halfword from the MEMORY then echos back the INPUT DATA RECIEVED pulse which increments the MEMORY ADDRESS COUNTER allowing the next halfword to be read. This cycle is then repeated until all the desired data is read. For any halfword the lower byte is one clock cycle older than the upper byte. The lower byte of the first halfword read contains the oldest data in memory while the upper byte of the last halfword read is the most recent. Each additional read accesses more recent data.

The clock that drives the DIGITIZER during digitization is selected by programming the latches driving the 8:1 multiplex whose output is called SELECTED CLOCK. One of the four active inputs (the other four inputs may be used eventually) is select and routed to the A/D and a divide by two F/F generating the SELECTED CLOCK / 2 signal. When the MSB of the DELAY COUNTER is high (before termination and after RESET), the SELECTED CLOCK / 2 is gated to the ADDRESS COUNTER and the DELAY COUNTER and two latches on the output buffer.

Each time the A/D is clocked a byte of data is generated. Since the SELECTED CLOCK can be as high as 30 MHz (33 ns period), the MEMORY with an access time of 35 ns must slow the data down by some means for confident storage. This is achieved by building a halfword out of two sucessive samples hence doubling the time required for MEMORY to store data. The halfword is built by first clocking the data from the A/D into the 8-BIT REGESTER then on the next clock loading the 16-BIT REGISTER with the new byte from the A/D

and the old byte from the 8-BIT REGISTER. Every other clock of the SELECTED CLOCK (SELECTED CLOCK / 2) loads the 16-BIT REGISTER allowing the data in this regester to be active for 2 cycles (worst case 2*33 ns = 66 ns) which is long enough for the MEMORY to store the data accurately.

The size of MEMORY is programmed by loading the latches feeding the 8:1 multiplex connect to the most significant address lines. The programmed address line is feed back to the clear input of the ADDRESS COUNTER via the multiplex. When this line goes high, the clear is enabled and on the next clock to the counter the count is cleared. If the memory size is 2048 then a ground is selected and the counter is never cleared but just keeps counting modulus 1024.

The RESET signal starts the DIGITIZER by loading the DELAY COUNTER, setting the DIVIDE BY TWO F/F, and clearing the STOP F/F. This signal, generated by software, "arms the DIGITIZER" since after outputting this signal the next event is to look for the termination of the process.

During the video mode the halfword from MEMORY slips through the latches feeding the TRI-STATE output buffers. The latches are enabled under program control by OUTPUT ENABLE signal. The data then slips through the continously enabled latches, gating data from MEMORY to the computer through two sets of latches and the TRI-STATE output buffer. During the video mode the latches are transparent and the data falls through to the output buffer. The buffer is TRI-STATE so as to not interfer with other units using the same input lines to the computer. The buffer is made active

with the OUTPUT ENABLE signal which is programmed and latched. The ALERT pulse goes only to the buffer and then buffered to the computer. In the audio mode the latches serve as two halfword register to enable the computer to read twice without checking for the ALERT pulse the second time hence saving some critical overhead in the software.

In the audio mode several blocks of the DIGITIZER are inactive (STOP F/F, DELAY COUNTER, and MEMORY). The SELECT CLOCK is usually the internal 110.308 kHz clock, although any clock lower than this will work just as well. The internal 110.308 kHz clock is the fastest data rate the magnetic tapes can accept. Data from the A/D follows the previously mentioned course to MEMORY. But once in MEMORY the path continues to the output pins of MEMORY, that is the data passes through MEMORY and is not stored as in the video case. The data feed the pair of latches on the OUTPUT INTERFACE which allows storage of two halfwords. The SELECT CLOCK / 2 is divided by two, this allows an ALERT pulse to be generated half as often as the video case so the two halfwords per ALERT pulse can be read. This clock and its complement steer the halfwords to the correct latch.

2.5 Programming the Digitizer

This section covers the programming aspects. The computer outputs a halfword (from here on represented as four hexadecimal integers) to the units on the output lines; since there are other units operating on the same line, some method of determining which unit is to receive the data is necessary. This problem is solved by setting aside the four

MSBs (bits 0-3) as the "unit number", which allows for sixteen possible units to be active at one time. The unit looks at the unit number of the halfword and the data is received if they match. The DIGITIZER's number is $F, hence all data going to it has the form $FXXX where X is any hexadecimal integer. The next three MSB (bits 4-6) select the latch the data is to be sent. Bit 7 is not used by the DIGITIZER and henceforth taken as a logic 1. The remaining bits (8-15) are the data to be loaded into the latches.

TABLE 2-1

INSTRUCTION TABLE

$F1XX - Load upper byte of DELAY LATCH with XX

$F3XX - load lower byte of DELAY LATCH wlth XX

$F5XX - Computer generated clock, XX not used

$F7XX - Not used

$F9XY - MEMORY size select, X not used, Y is as follows

Y	0	1	2	3	4	5	6	7
Size	18	34	66	130	258	514	1026	2048

$FBXY - Load four MSBs of LINE LATCH with Y, X not used

$FDXX - Load lower byte of LINE LATCH with XX

$FFXX - Bit 15 If high, LINE DETECT STOP else EXT STOP

Bit 12-14 Control SELECT CLOCK

100 - 30 MHz clock

101 - EXT CLOCK

110 - 110.308 kHz clock

111 - Computer clock (F5XX)

Bit 11 If high, generates RESET pulse

Bit 10 If high the OUTPUT BUFFER is active

Bit 9 If high, AUDIO mode, otherwise VIDEO

Bit 11 of $FFXX requires additional discussion. When the operator has loaded all the latches (except $FFXX types) and is ready to digitize all that is necessary is to output the $FFXX instruction with the Bit 11 high with the other bits in their selected states. When the DIGITIZER finishes, no further digitization is possible until the $FFXX is

repeated with Bit 11 high which allows the operator to read the data without the problem of future data writing over the unread data. The digitization process can be continued by outputting just this one instruction. Sample programs will be given in Chapter 4.

2.6 Digitizer Theory of Operation

The following section describes in detail the circuitry of the DIGITIZER. There are six boards in the DIGITIZER labeled A1-A6 from left to right looking from the front. Component designation is done by prefixing its location with the board number e.g. pin 3 of integrated circiut 2 (IC2) of board 4 is labeled A4IC2-3 (see figure 2-8). Integrated circuit location is determined by looking at the component side in the upper left hand corner and counting to the right one space at a time regardless of whether it is filled by an IC. Reaching the right after counting 6 spaces, drop down one row and continue the process until IC24. Four additional ICs can be squeezed on the board and for their numbering see the figure. In Appendix A the wiring connections between the cards are given.

Loading of the latches (latches hold permanently the received instructions sent by the computer) is first discussed. The computer output lines consist of 17 lines, 16 of which are data lines as covered in the instruction section and 1 line is a strobe occurring when a new halfword is sent. The data is latched by a digital I/O port (call DIO) contained within the computer staying valid until it is written over by the next halfword outputed under programmable

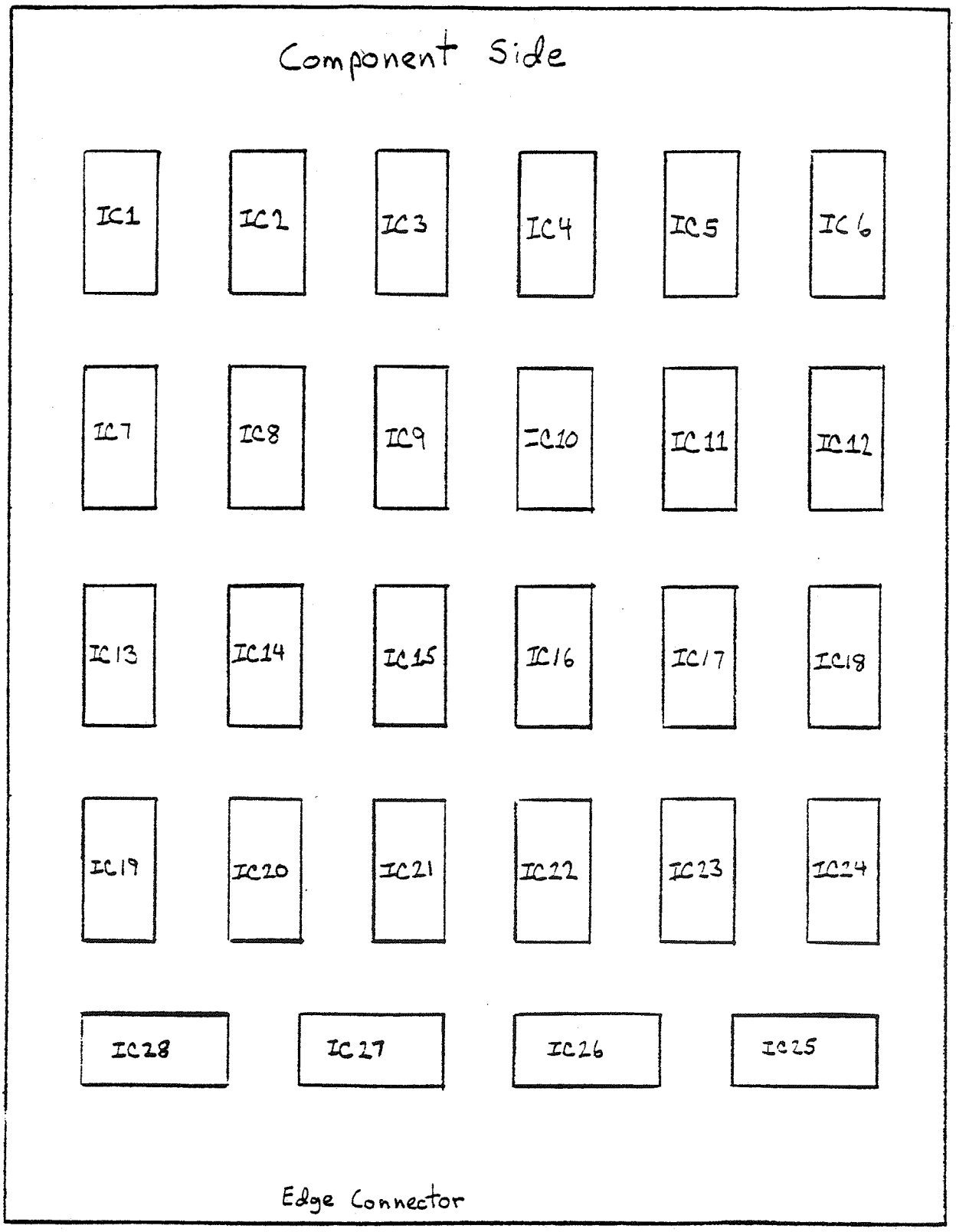


Figure 2-8: Component location on circiut boards.

control. The data from the DIO is received complemented relative to the instruction table. The strobe (OUTPUT DATA AVAILABLE) however is not latched in the DIO but rather is sent as a 4 us pulse in coincidence with the latching of the new halfword.

Data from the DIO output is sent to the edge connector of A3 via a ribbon cable, this board decodes the data and routes it throughout the digitizer (see figure 2-9). Bits 0-3 of the ouput data contains the unit number and are routed to A3IC23, a quad input NOR gate. If all bits are low ($FXXX sent) A3IC23-6 goes low enabling A3IC19 and the other enable of IC19 comes from two single shots (A3IC13) hooked in tandem. The single shots act together to delay the OUTPUT DATA AVAILABLE pulse to insure the data lines have settled down before latching them into the DIGITIZER. The OUTPUT DATA AVAILABLE pulse is wired from the connector to the Schmitt trigger input hex inverter A3IC22-11 which removes the ringing of this pulse. After the inverter the pulse triggers the single shots generating the other enable to A3IC19. A3IC19 is a 3:8 decoder where Bits 4-6 are decoded and the corresponding output line drops low. All eight of the lines pass through inverters (A3IC20,A3IC21) to make the logic compatible with the latches they drive. These lines are call STORE X lines where X is a number zero through eight. The 3:8 decoder stays active only 620 ns (pulse width of single shot) hence the eight lines are pulsed for the same duration. Bits 8-15 are connected to Schmitt input buffers (A3IC21-A3IC22) and then continue to the latches as data,

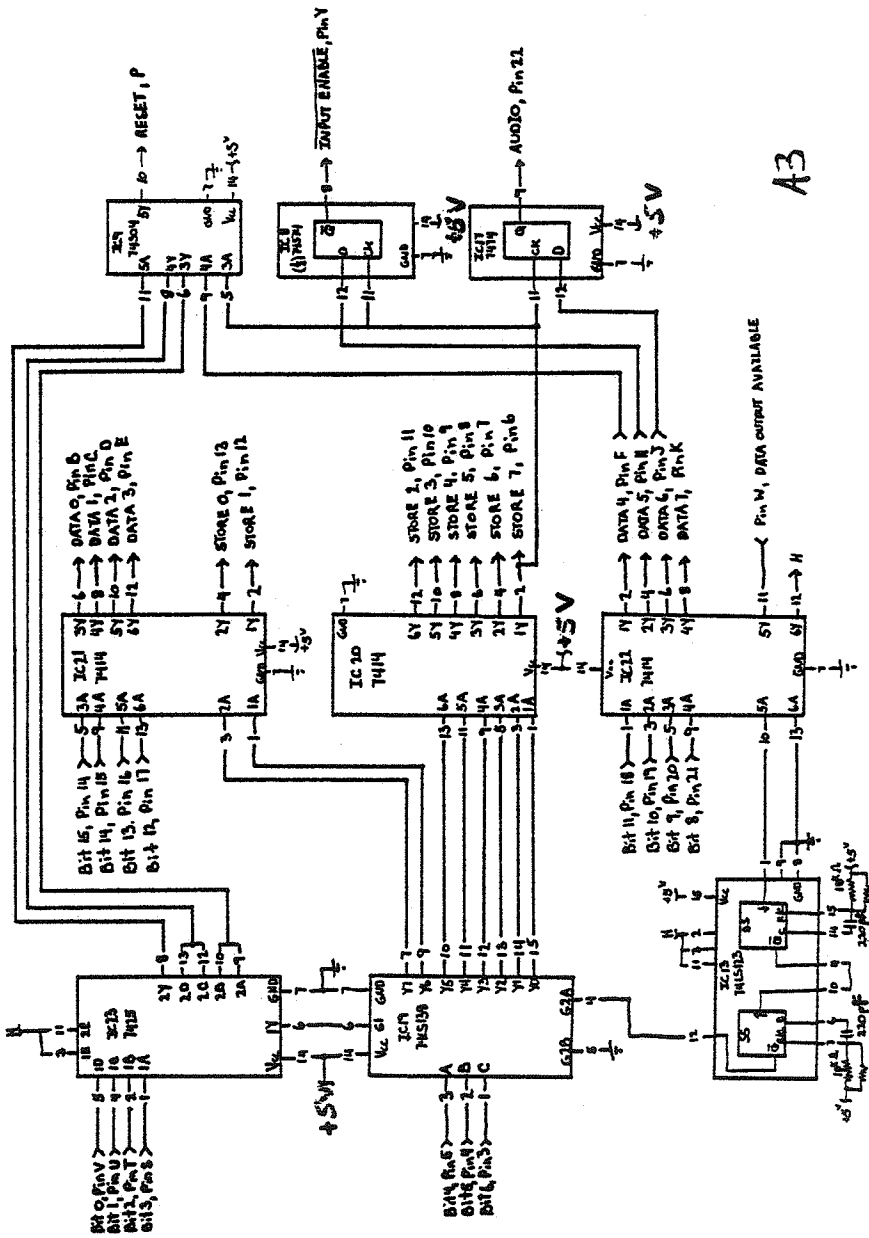


Figure 2-9: Computer output interface.

they are call DATA X where X is the same as above.

The latches which are loaded by instruction code are listed in table 2-2. They are 74LS75, the data comes in on pins 2,3,6,7 and latched when the enable pins 4 and 13 are high. The data then appears on pins 16,15,10,9 and their complement on pins 1,14,11,8. The Bits 10,9 (DATA 5,6) of $FFXX are stored in F/Fs instead of latches (since during designing some F/F were uncommitted it was decided to use them instead of adding additional latches). The F/F for $FFXX Bit 10 (DATA 5) is A3IC8, configured as a D-type F/F with the data as DATA 5 (pin 12) and the clock as STORE 7 (pin 11), the output (pin 8) enabling the OUTPUT BUFFER. As for $FFXX Bit 9 (DATA 6) the F/F is A3IC17 and is wired exactly the same as the one above, but drives the AUDIO MODE SELECT line. The RESET is generated from logical NAND of STORE 7 and DATA 4 using A3IC9 and A3IC23.

TABLE 2-2

LOCATION OF LATCHES BY FUNCTION

STORE 0 - A2IC9,A2IC10	$F1XX
STORE 1 - A2IC7,A2IC8	$F3XX
STORE 4 - A2IC15	$F9XX
STORE 5 - A3IC12	$FBXX
STORE 6 - A3IC10,A3IC11	$FDXX
STORE 7 - A3IC2	$FFXX

In the following paragraphs the generation of the SELECTED CLOCK is discussed with reference to figure 2-10.

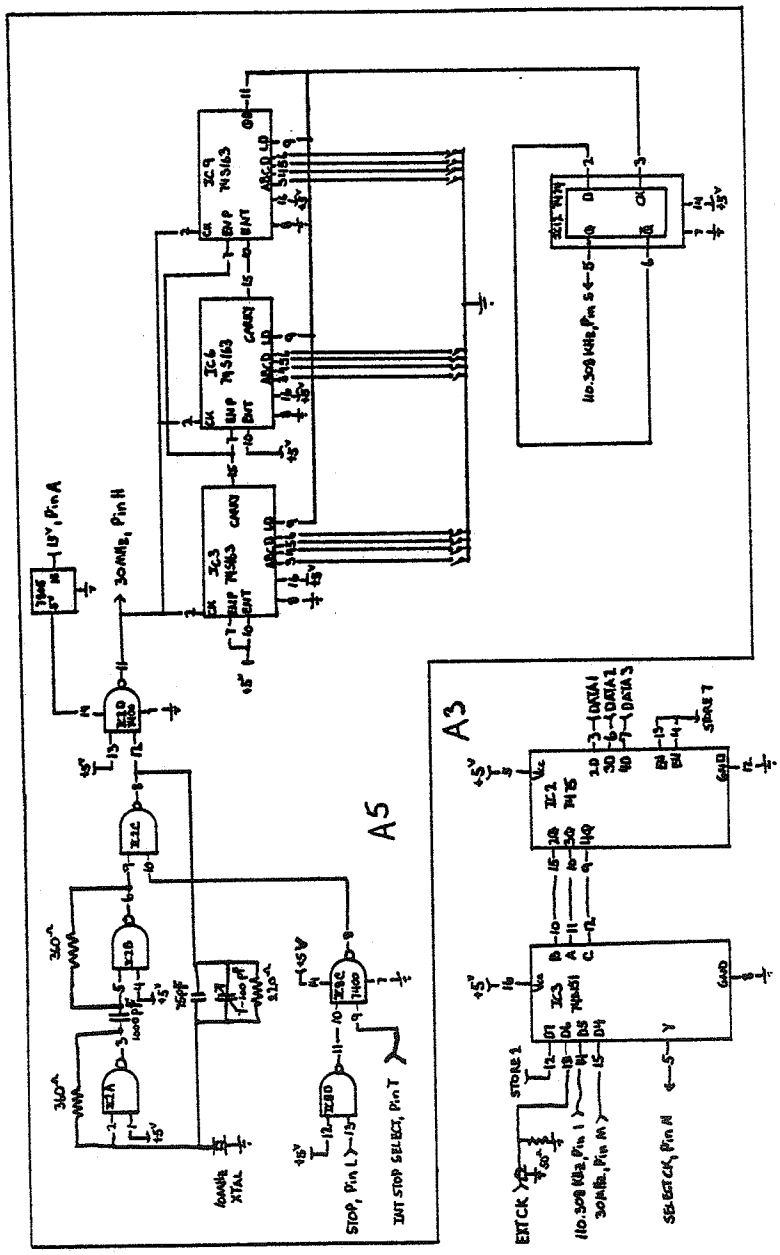


Figure 2-10: Clock selection circuitry.

Currently there are four possible clocks for this option: 30 MHz, 110.308 kHz, EXT CLOCK and computer generated clock STORE 2. The 30 MHz clock is always used when digitizing images and is available for other video purposes. If the operater prefers to supply ones own clock, this can done via the EXT CLOCK connector on the back panel. The 110.308 kHz clock is derived by dividing down the 30 MHz clock, and in addition to being the clock for the audio mode it is a handy clock for testing purposes. The computer generated clock is under total control of the operator in that every time $F5XX (STORE 2) is issued one clock is sent, this was used during the early days as a convenient clock but since never used however never disconnected.

The 30 MHz clock is generated on A5 with 10 MHz crystal operating in the third harmonic mode. The oscillator (A3IC2) is a simple crystal type except that the ring can be broken (no clocks) by bringing A5IC2-10 low. The purpose of this pin is to guarantee the oscillator starts oscillating in phase with the leading edge of the signal applied. This signal comes A5IC8-8 which performs this Boolean expression:

STOP + c(INT CLOCK SELECT)

The reason for this is to insure free running 30 MHz and 110.308 kHz clocks when internal stop mode is not selected and synchronized clocks with the START signal when this mode is selected. The START pulse comes from A3IC8-5 and as previously mentioned results when the programmed line is detected in the internal stop mode or an external stop is

applied in the external stop mode. The 110.308 kHz signal is generated by dividing down the 30 MHz clock with three 74S163 (A5IC3, A5IC6, A5IC9) programmed by sliding switches and the T-type F/F (A5IC12). A5IC12 squares the MSB of A5IC9 to yield better transmittion qualities. The operator is free two change the 110.308 kHz signal with the switches if desired. Both the 30 MHz and the 100.308 kHz signal are routed to A3IC3. In addition to these clocks the EXT CK and STORE 2 are also terminated at this IC with EXT CK through a 50 ohm load. This chip is a 8:1 multiplex, with pins 9-11 selecting one of the four above clocks to form SELECT CK (pin 5) . The programming pins are feed by the latch A3IC2. The SELECT CK is then routed to the A/D and to F/F 1 which divides the clock by two to generate SELECT CK / 2.

This section explains the circuitry that generates the STOP pulse This section with reference to figure 2-11 explains the circuitry that generates the STOP pulse. There exist two ways to begin the pulse depending on whether the external stop mode or the line detection mode is selected by program control ($FFXX Bit 15). The line dectection line is used only for digitizing images and works by loading an up counter with the two's complement of the desired line plus two on the occurance of the second line of the image and then waiting for the MSB to set thus issuing a pulse that sets the START F/F. In the external stop mode the STOP F/F is triggered by the EXT STOP signal.

The horizontal and vertical sync pulses enter the DIGITIZER through coaxial plugs on the back panel and are

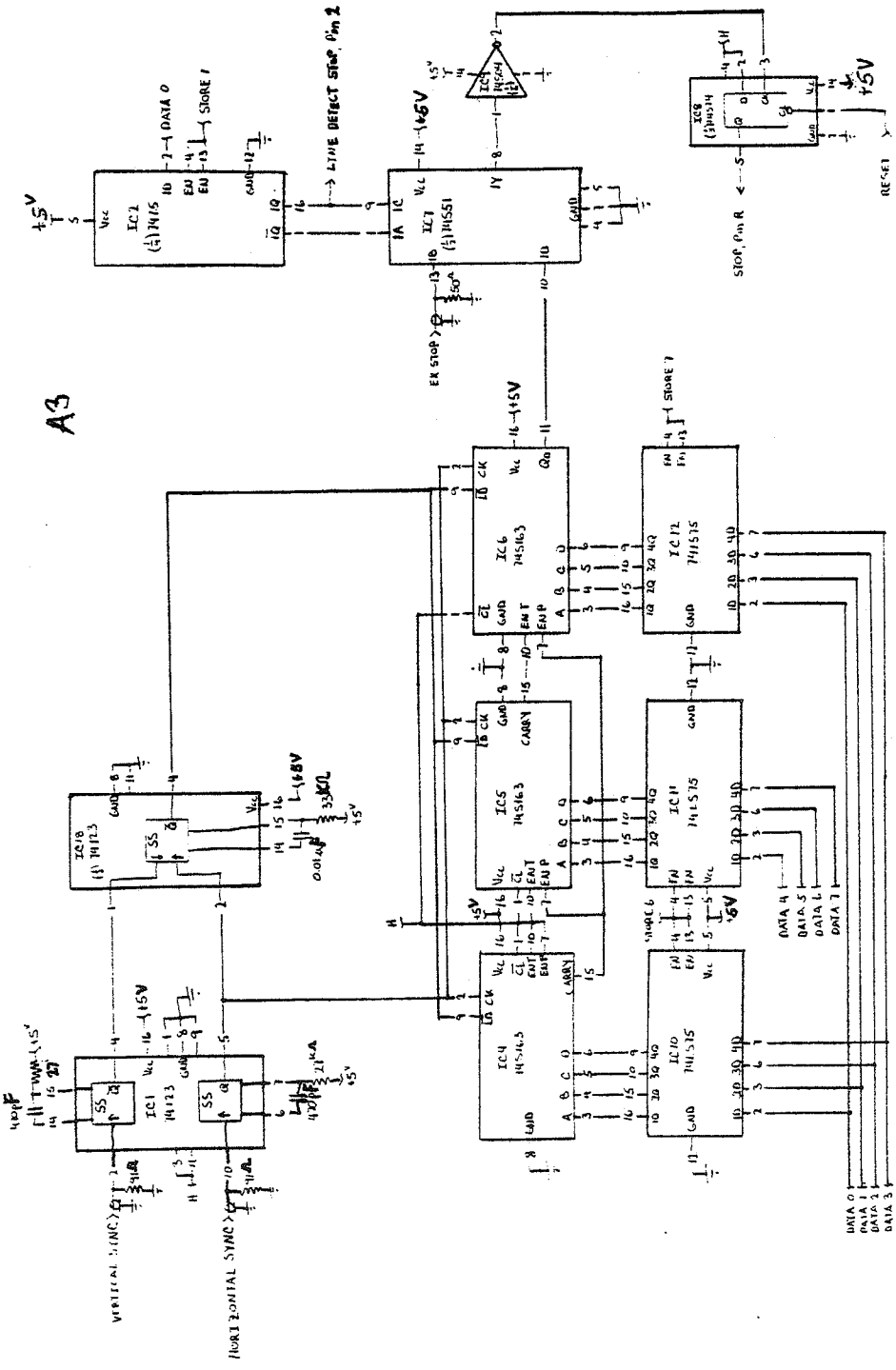


Figure 2-11: STOP signal generation circuitry.

terminated by 93 ohm loads at A3IC1-10,2 respectively. This IC is dual single shot (74123) triggering on the rising edge of the syncs for 4us. The single shot triggering on horizontal pulse is the line counter clock and an enable to an additional single shot (A3IC18). A3IC1 and A3IC18 make up the COINCIDENCE DETECTOR. The single shot triggering on the vertical sync pulse is the other enable to this single shot hence the chip is enable only when the two sync pulses have leading edges within 4us, i.e., the first line of the image. This single shot stays active for 90us driving the load input to the line counter. The line counter consist of three 74S163 (A3IC4-A3IC6) programmable four bit sychronous counters. The counters are loaded with the latched data in A3IC10-A3IC11 when the load line (pin 9) is low during which a rising edge has occured a the clock input (pin 2). This takes place on the second line of the image, the horizontal sync of the first line reaches the the clock input before the load can go low since it must travel through IC17 first hence the first line can not load the counter. However the single shot driving the load line will stay low for 90us which allows the next horizontal sync pulse that occurs 63.5us later to load the counter thus loading the counter on the second line. Pin 11 of A3IC6 is the MSD of the counter and is loaded low under program control, when the line count toggles this bit it triggers the START F/F if in the line detection mode.

The selection between EXT STOP or the line detect mode for the trigger to STOP F/F is made by the 2:1 multiplex

A3IC7 programmed by latch A3IC2. The trigger leaves A3IC7-8 and enters A3IC9-1, inverted and exits (pin 2) to set the STOP F/F (A3IC8-3). This F/F is reset before starting the digitization process by RESET under software control. Pin 5 of the STOP F/F is the orgin of the STOP signal.

The RESET signal from A3IC8-5 is wired to A2IC1-10 which is the first stage of the DELAY COUNTER incorporating four 74S163s (see figure 2-12). This is one of the two inputs to the digitizer, the other enable is A2IC1-7 coming from the MSB of this counter (A2IC4-11) which is initially set through program control. The counter is loaded with the data from latches A2IC7-A2IC10 during the reset pulse and begins counting when the STOP goes high and stops when the MSB goes low. The load line is connected directly to the RESET signal but the clock line is derived from a single shot triggering off RESET. The duration is the single shot is 400 ns to issue that the clock line has a positive transistion when the load line is low (the load line being active 620 ns). The clock begins its path by RESET triggering A2IC12-9 (the single shot) the pulse leaves pin 5 to A3IC17 which gates it through to the CLOCK line feeding the DELAY COUNTER amoung other things. The other inputs of A3IC17 are disabled, A3IC17-4 comes from A2IC22-8 which low since the F/F is set by RESET, A3IC17-10 is the INPUT DATA RECEIVED pulse which occures only when reading the data which is under program control and is kept low at this time.

The MSB of the DELAY COUNTER controls the gating of SELECT CLOCK / 2, INPUT DATA RECEIVED, the R/W line to

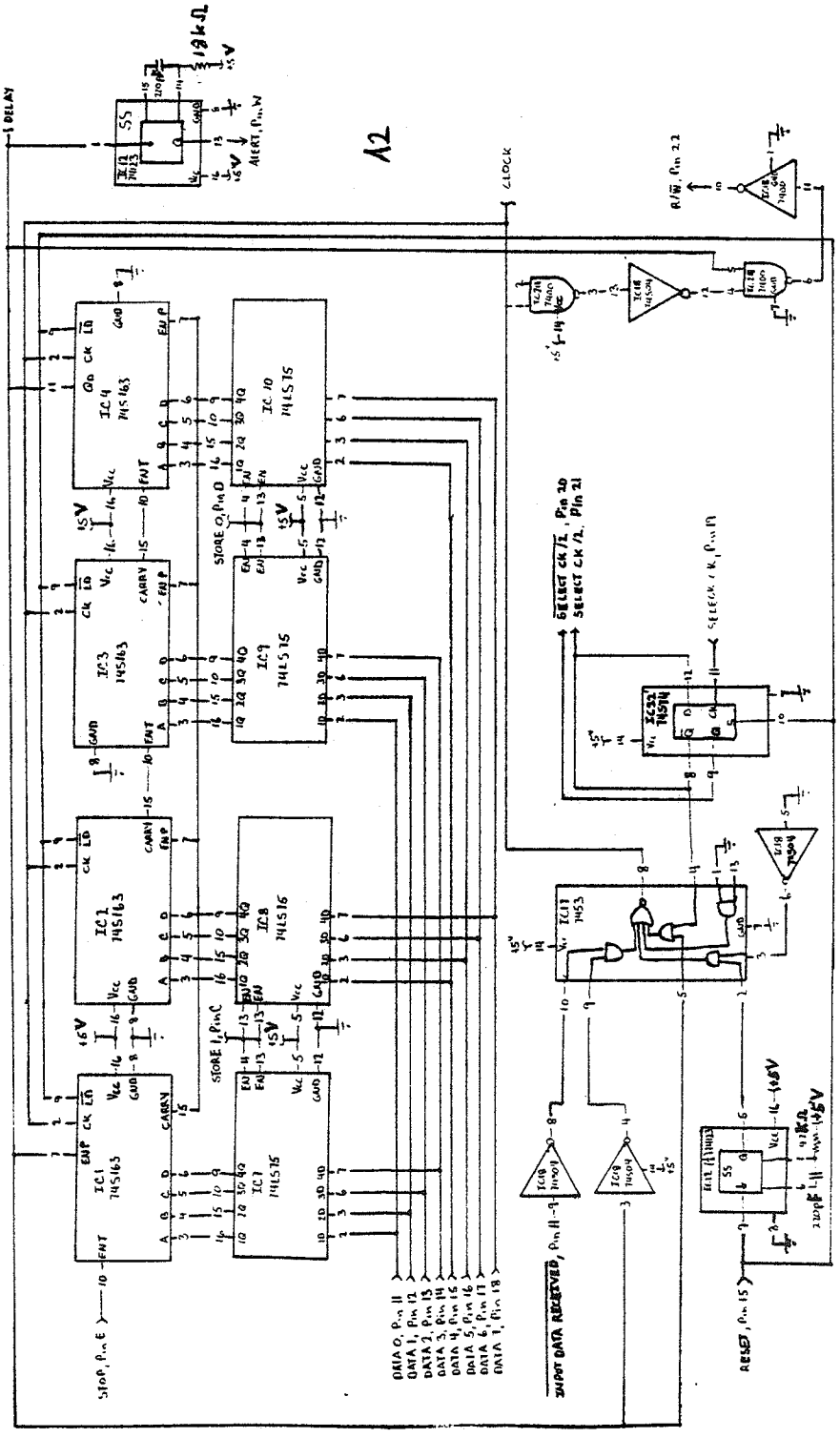


Figure 2-12: DELAY and SELECT CK/2 generation circuitry.

MEMORY, and the ALERT pulse. After RESET this bit is set (the programmer must insure to set this bit by loading F1XX Bit 7 high) placing a high on A2IC17-5 which gates SELECT CLOCK / 2 to the CLOCK line. When the DELAY COUNTER has overflowed this clock is stopped and the INPUT DATA RECEIVED clock (A2IC17-10) is enabled by the MSB being inverted by hex-inverter A2IC18 driving A2IC17-9. This clock is used to increment the MEMORY COUNTERS during the reading sequence under program control. The R/W line to MEMORY is the logical AND of the MSB with the the CLOCK, performed by A2IC18 and A2IC24. Also these two A2ICs delay and stretch the positive portion of the CLOCK input to A2IC24-1 to center the R/W pulse with the data being sent to MEMORY when operating at 30 MHz. The ALERT pulse is caused by the MSB triggering a single shot (A2IC12-1), this pulse passes through the INPUT BUFFER to notify the program of termination of the process.

The ADDRESS COUNTER (see figure 2-13) is driven by CLOCK and acts as a free running 10 bit counter addressing MEMORY. The counter is constructed with three 74S163 (A2IC19-A2IC21) 4 bit counters and a 74S151 8:1 multiplex (A2IC14). The ten lower bits of the counter are the ADDRESS lines to MEMORY with the upper seven address lines feeding the multiplexer. The remaining input to the multiplexer is a ground. The purpose is to select one address line to clear the counter on the next clock after the line goes high, hence devising a modular 2**N+1 counter where N is the address line feedback. The line is selected by latch A2IC15 which holds the data from instruction $F9XY (memory size select). When the ground

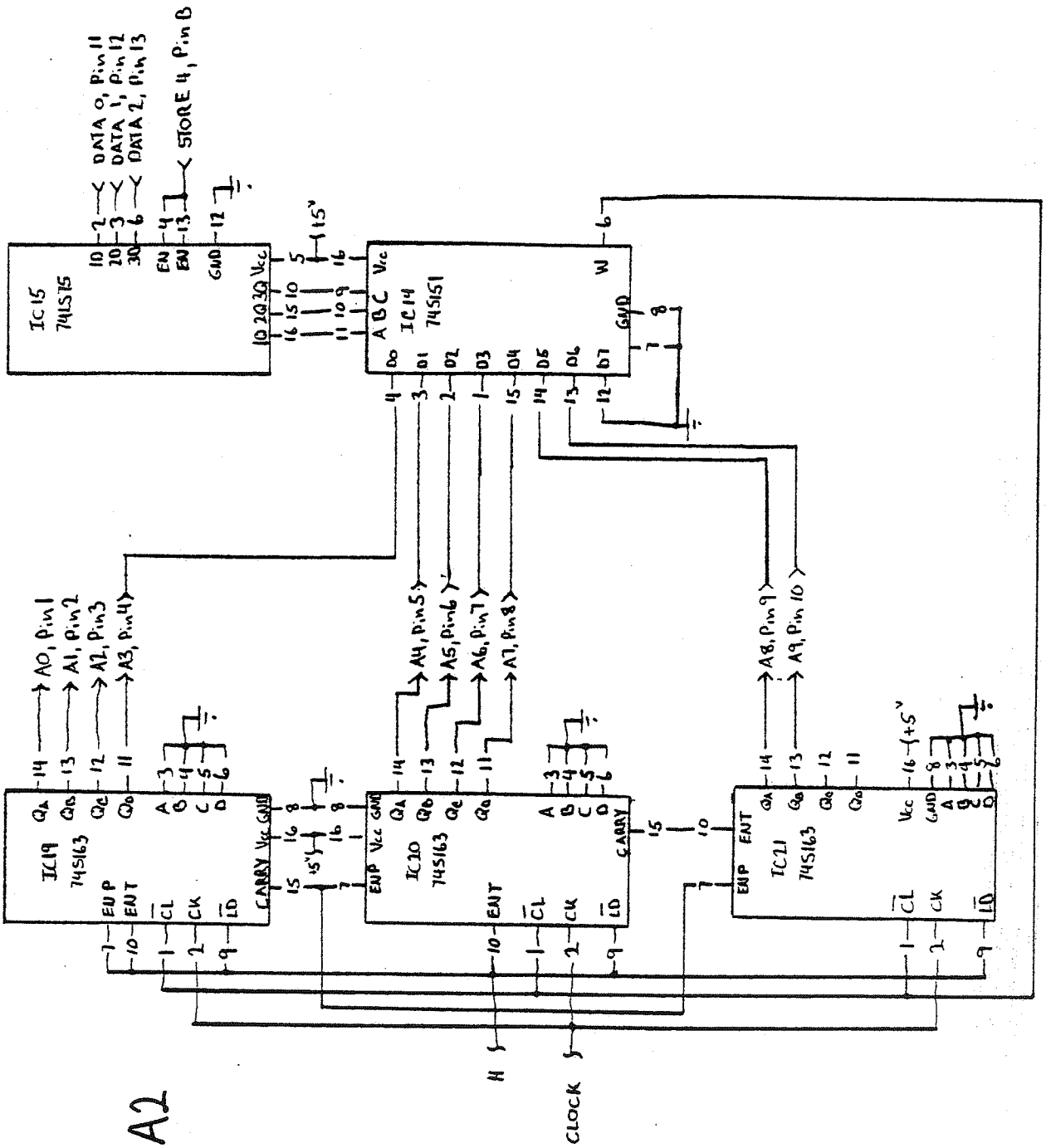


Figure 2-13: Address lines circiut.

is selected ($F907) the counter is not cleared allowing all 1024 halfwords to be used.

Formation of the halfword feeding MEMORY is now explained (see figure 2-14). The reason for combining two samples from the A/D is to double the time required for the memory write cycle since at 30 MHz the write cycle specs just barely fail for storage of one sample per cycle but pass for storage two samples at once every two cycles. A sample is generated with every positive transition of SELECT CK from the A/D, the A/D has a pipeline delay of one. The data then feeds two registers, an 8-Bit (A2IC18, A2IC24) and a 16-Bit (A2IC5, A2IC11, A2IC17, A2IC23) register. The IC are all 74S175 four bit clocked registers (positive edge). The 8-Bit register is clocked with the rising edge of SELECT CK / 2 storing the A/D data. Then the 16 Bit register is clocked with the rising edge of the complement of SELECT CK / 2 storing the data previously stored in the 8 Bit register and the next sample from the A/D. Hence building a 16 Bit halfword that supples memory with data. Since the clock is a result of SELECT CK divided down by two the memory must store a halfword every two periods (worst case is 30 MHz so data is valid for at least 66.6 ns) which guarantees storage since the memory chips are rated at memory access time of 35 ns.

MEMORY consist of 16 2125A bipolar memory chips. The chips are internally arranged as 1024 X 1 bits and externally as 1024 halfwords each chip handling one bit of the halfword. The address lines (A0-A9) are buffered first by hex-inverters

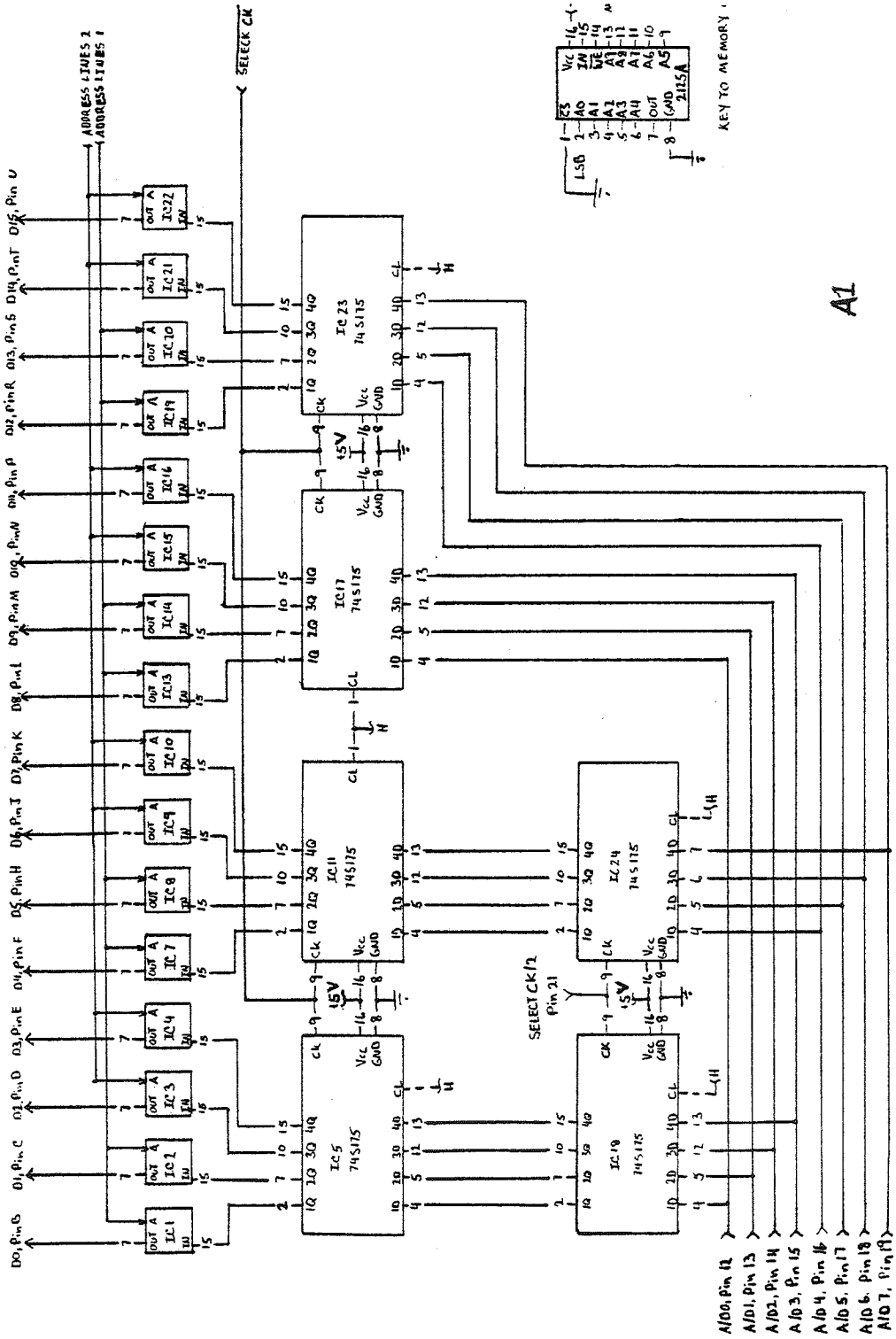


Figure 2-14A: Buffer memory.

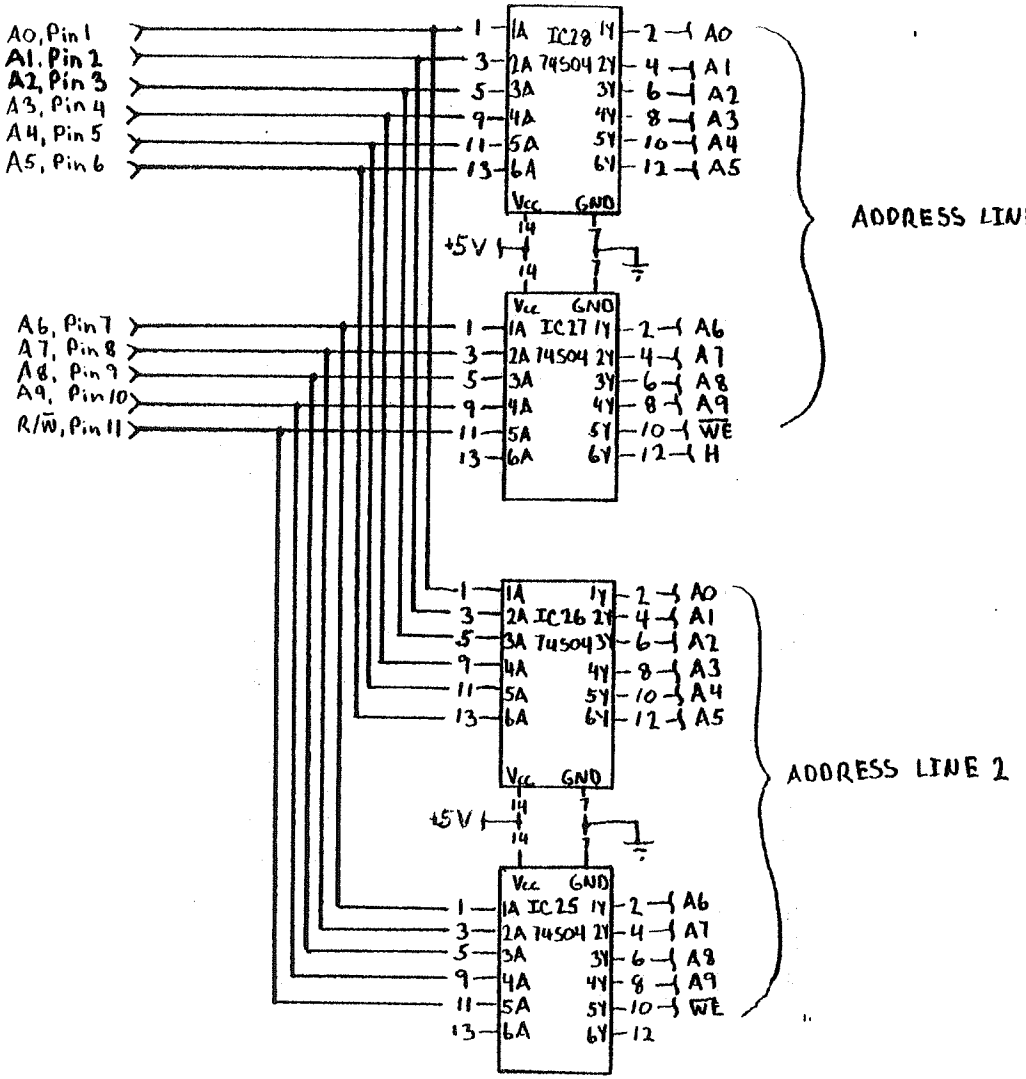


Figure 2-14B: Buffer memory.

A1IC25-A1IC28 to reduce the fannout of the ADDRESS COUNTER. The halfwords come in on pin 15 and exit on pin 7 (if the R/W line is high). When the R/W line is low the output is in the TRI-STATE inactive mode; this property is important so as to not overlook in the INPUT section. The output lines from MEMORY feed the INPUT card which is the subject of the next section.

The INPUT board (see figure 2-15) sends halfwords to the DIO with the ALERT pulse and receives the INPUT DATA RECEIVED pulse shapes it and outputs it to A2 (ADDRESS COUNTER). During the audio mode the INPUT card holds two halfwords to enable program to read two halfwords sequentially with one detection of the BUSY signal, hence eliminating one critical software detection loop, this is discussed in the programming section. In the video mode the latches are transparent (latch continuously enabled).

The INPUT DATA RECEIVED line comes from the DIO via ribbon cable and is connected to A6 by means of Scotch edge connectors. The signal is passed through a high pass R/C filter on the way to triggering single shot A6IC-6. The output (pin 4) is routed to A2 as c(INPUT DATA RECEIVED) which increments the ADDRESS COUNTER if DELAY is low (is over).

The BUSY triggers the DIO in a manner such that the software can detect it. This signal is generated by two means depending upon whether the audio or the video mode is selected. In the video mode (AUDIO is low), ALERT is gated through to the TRI-STATE buffers. In the audio mode (AUDIO

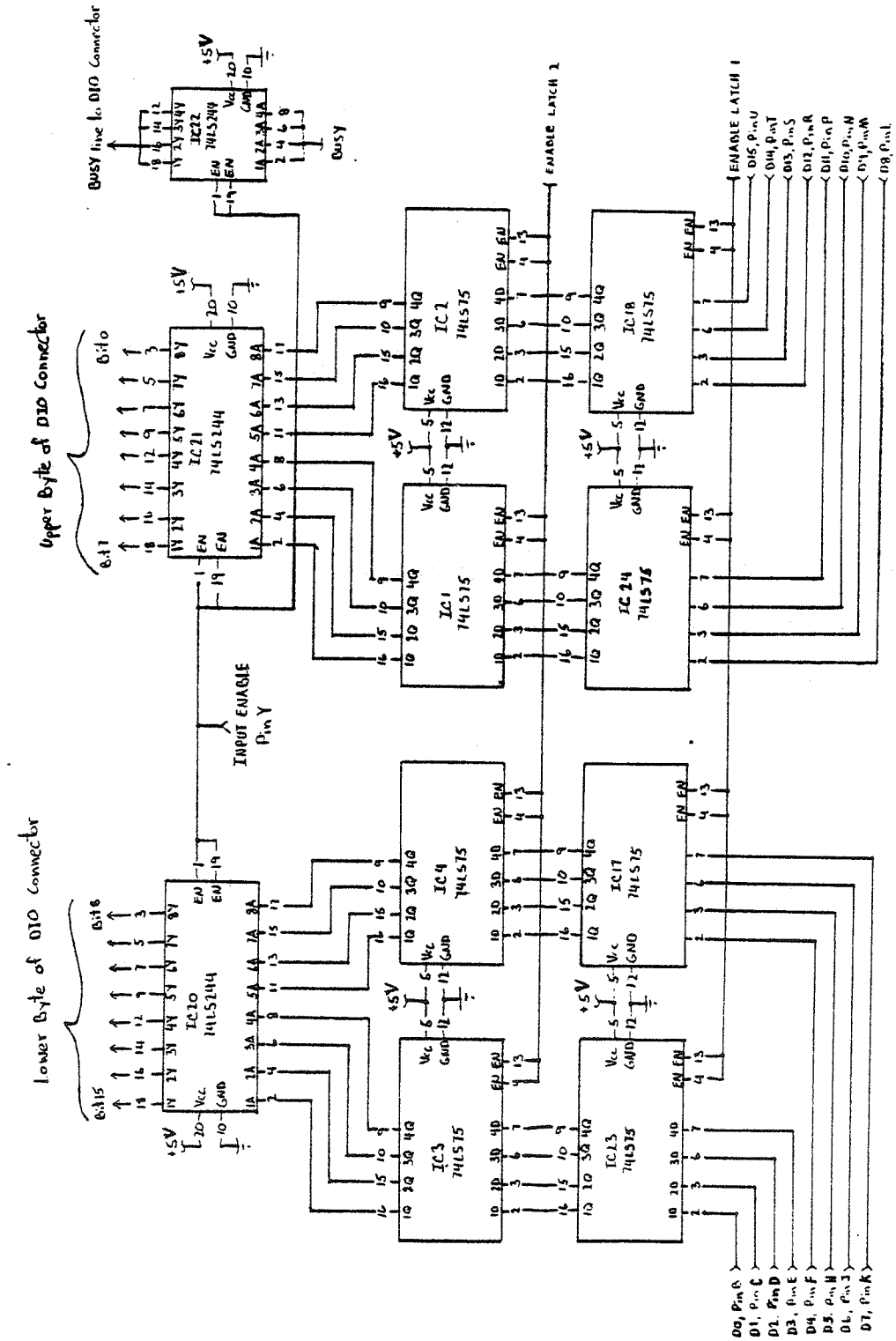


Figure 2-15A: INPUT board.

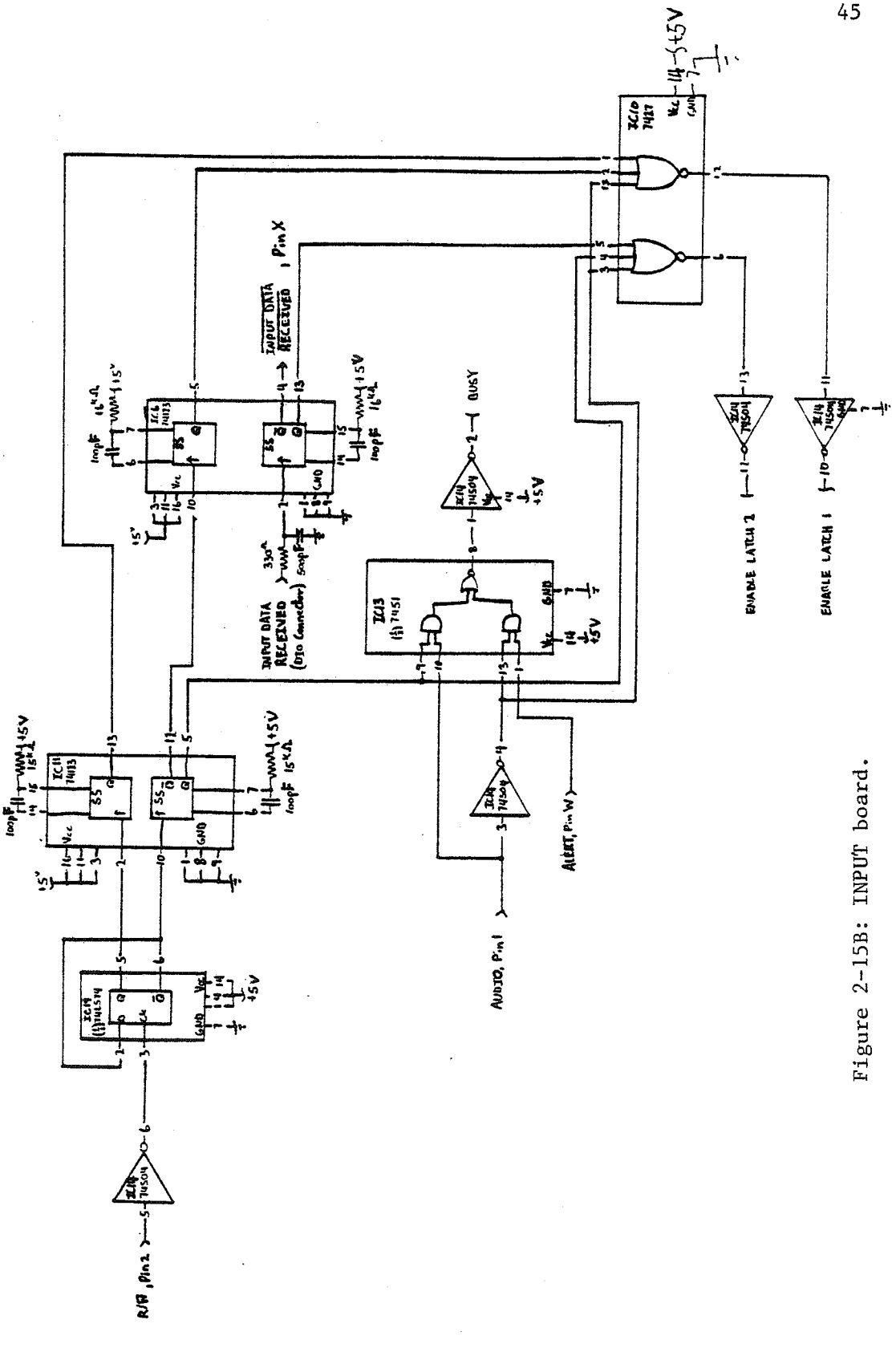


Figure 2-15B: INPUT board.

is high), the signal comes from A6IC11-5, a single shot that is triggered off of a F/F (A6IC19-6). The flipflop takes the CLOCK signal and divides it by two starting BUSY off of every other rising edge (a halfword is built with every rising edge hence the computer is notified every two halfwords). A6IC14 and A6IC13 form the selection circuit and send it to TRI-STATE buffer A6IC22.

The two 16-Bit latches are enabled differently depending on the status of AUDIO. If low, A6IC14-4 inverts it, driving A6IC10-6,12 low which are then inverted by A6IC14-10,12 enabling both latches together. However, in the audio mode (AUDIO high) the process is more involved. CLOCK enters the board at A6IC14-5, which inverts it (pin 6) and toggles the T-type F/F A6IC19-3. The F/F and three single shots (one on A6IC6, two on A6IC11) work together to load the latches with halfwords from MEMORY. Assume for a starting point that the F/F is cleared. The next falling edge of CLOCK will toggle the F/F so that the Q output (A6IC-5) will go high causing the single shot (A6IC11-2) to trigger. This pulse from pin 13 enters NOR gate A6IC10-1 and passes through (pin 12) to be inverted by A6IC14-10 which enables the first latch to store a halfword. The other inputs to the NOR gate are low, pin-13 being the complement of AUDIO and pin 2 the output of single shot A6IC6-5 which has terminated (the following discussion shall show this). On the next falling edge of CLOCK the F/F will toggle so that Q-complement goes high triggering single shot A6IC11-10. Pin 13 of this single shot goes high passing through A6IC10 and A6IC14 in a circuit similar to the one

above with the final inverter A6IC14-12 driving LATCH 2 which latches the data in LATCH 1. In addition to the above, the single shot drives (A6IC11-12) another single shot A6IC6-10. However this single shot triggers when the driving single shot terminates. The pulse the passes through the OR gate arrangement of A6IC10 and A6IC14 to enable LATCH 1 to store a halfword from MEMORY. The single shot pulse duration is about 1us and since the audio mode uses a clock with a period of about 18us no single shot is on simultaneously. As previously mentioned the BUSY is started with A6IC19-6 toggling high, the quickest the software can detect this is more than 5us so the INPUT DATA RECIEVED pulse will not come during the time the single shots are active hence will not interfering with the OR circuit. When the computer reads a halfword, it fires back the INPUT DATA RECEIVED pulse which, after the single shot A6IC6-13, passes through the OR gate configuration to load LATCH 2 with the data in LATCH 1 so the computer can read the next halfword. The INPUT DATA AVAILABLE pulse with the second read will load LATCH 2 again with data from LATCH 1 but these data are not read since the computer will now be waiting for the BUSY line to be pulsed. The BUSY pulse comes from the same pulse that loads LATCH 2 with good data so the computer will always read valid data if the BUSY is first detected. The purpose of loading LATCH 1 first instead of both latches first and thus saving a single shot is to keep the data in LATCH 2 valid as long as possible to give the software extra time to read the second halfword.

The output of the TRI-STATE MEMORY is active only during

the low portion of CLOCK; hence LATCH 1 is loaded during this time. This is why inverter A6IC14-5 is needed; it inverts CLOCK so that the INPUT BUFFER operates on a falling transistion.

The buffers are TRI-STATE and enabled by INPUT ENABLE ($FFXX Bit 10 set). They route the halfwords to the DIO via ribbon cable. The buffer handling BUSY is made of four stages in parallel to increase the driving power the pulse.

CHAPTER 3

PROGRAMMING SECTION

3.1 Introduction

This chapter will discuss five programs: 1) send A/D value to LED display, 2) capture a line of the image for examination, 3) digitize image, 4) enhance interference lines, and 5) output the image on the printer.

3.2 A/D Value To LED Display On Control Panel

This program averages 256 samples of the A/D and then outputs the value in hex on the 7-segment LEDs on the control panel of the value in hexadecimal to the 7-segment LEDs on the control panel of the computer. This aides the operator in adjusting the offset and null of the TRW A/D (see spec sheet for pot adjustment). The program is in figure 3-1.

The program selects the audio mode hence digitization is at 110.308 kHz lines 20-21. In this mode the program must read two halfwords (two bytes per halfword) for each time BUSY goes low in the DIO. The computer must then loop 64 times (40 in hexadecimal) detecting BUSY=0 once during each loop. Lines 22, 38, and 39 make this loop. The halfword read from the DIO consist of a lower byte made of a sample captured at time t and an upper byte captured at time t+1 where t has units of a clock period. The halfword must be split into the two bytes and added to the sum forming the average. This is done at Lines 26-39 for both halfwords. The after summing 256 samples the sum is divided by 256 and sent to the control panel LEDS by means of lines 40-49.

```
1           INTEGER*4 RGSTR(16)
2       $ASSM
3       DIOOUT    EQU   0              ASSIGN DIO OUTPUT TO RGSTR 0.
4       DIOIN     EQU   1              ASSIGN DIO INPUT TO RGSTR 1.
5       TEMP1     EQU   2              CREATE A SCRATCH RGSTR.
6       TEMP2     EQU   3              CREATE A SCRATCH RGSTR.
7       SUM       EQU   4              RGSTR FOR SUMMING A/D SAMPLES.
8       LOOPCT    EQU   5              LOOP COUNTER FOR READING A/D.
9       LED       EQU   6              ASSIGN CONTROL PANEL LED'S TO RGSTR 6.
10                STM   0,RGSTR        SAVE CPU RGSTR'S (TO RESTORE LATER).
11                LHI   DIOOUT,X'00A8' INT. DIOOUT RGSTR WITH DIO OUTPUT NUMBER.
12                LHI   DIOIN,X'00A9'  INT. DIOIN RGSTR WITH DIO INPUT NUMBER.
13                LIS   LED,1          INT. LED RGSTR WITH CONTROL PANEL NUMBER.
14                LHI   TEMP1,X'00C0'  LOAD TEMP1 WITH DIO COMMAND
15                OCR   DIOOUT,TEMP1   THAT DISABLES INTERUPTS FROM DIO ,
16                OCR   DIOIN,TEMP1    AND OUTPUT TO DIO I/O PORTS.
17                RHR   DIOIN,TEMP1    READ A HALFWORD FROM DIO TO SET BUSY.
18                LHI   TEMP1,X'F1FF'  SET MSB OF DELAY REGISTER
19                WHR   DIOOUT,TEMP1   AND OUTPUT IT.
20                LHI   TEMP1,X'FF7C'  SELECT AUDIO MODE, ENABLE TRI-STATE OUTPUT,
21                WHR   DIOOUT,TEMP1   SELECT 100 KHZ, LOAD DAU RGSTR, AND OUTPUT.
22      NEWSUM    LHI   LOOPCT,X'0040' LOAD LOOP COUNER WITH 64.
23                LIS   SUM,0          INITIALIZE SUM TO 0.
24      BUSY      SSR   DIOIN,TEMP1    CHECK IF BUSY=0,
25                BTBS  8,BUSY         LOOP UNTIL BUSY=0 (DATA AT DIO INPUT).
26                RHR   DIOIN,TEMP1    READ HALFWORD FROM DIO (2 SAMPLES).
27                EXBR  TEMP2,TEMP1    LOAD TEMP2 WITH UPPER BYTE OF HALFWORD.
28                NHI   TEMP1,X'00FF'  CLEAR SECOND SAMPLE FROM TEMP1.
29                NHI   TEMP2,X'00FF'  CLEAR FIRST SAMPLE TEMP2.
30                AR    SUM,TEMP1      ADD FIRST SAMPLE TO SUM.
31                AR    SUM,TEMP2      ADD SECOND SAMPLE TO SUM.
32                RHR   DIOIN,TEMP1    READ SECOND HALFWORD (AUDIO MODE
33                EXBR  TEMP2,TEMP1    REQUIRES TWO READS PER BUSY).
34                NHI   TEMP1,X'00FF'  PROCEDE AS ABOVE...
35                NHI   TEMP2,X'00FF'
36                AR    SUM,TEMP1
37                AR    SUM,TEMP2
38                SIS   LOOPCT,1       DECREMENT LOOP COUNTER.
39                BNZ   BUSY           BRANCH IF MORE SAMPLES ARE NEEDED.
40                SRLS  SUM,8          DIVIDE SUM BY 256 (NUMBER OF SAMPLES).
41                LHI   TEMP1,X'0040'  SET CONTROL PANEL TO SUCCESSIVE
42                OCR   LED,TEMP1      BYTE LOADING MODE.
43                WDR   LED,SUM        OUTPUT TO PANEL FIRST BYTE OF SUM
44                EXBR  SUM,SUM        MOVE SECOND BYTE OF SUM INTO FIRST BYTE
45                WDR   LED,SUM        AND OUTPUT IT.
46                EXHR  SUM,SUM        PLACE THIRD AND FOURTH BYTE INTO FIRST AND
47                WDR   LED,SUM        SECOND POSITION AND OUTPUT THIRD BYTE.
48                EXBR  SUM,SUM        PLACE FOURTH BYTE IN FIRST AND
49                WDR   LED,SUM        OUTPUT IT TO CONTROL PANEL.
50                B     NEWSUM         FORM AN INFITE LOOP THAT READS THE A/D.
51                LM    0,RGSTR        RESTORE THE ORIGINAL CPU REGISTERS.
52      $FORT
53            STOP
54            END
```

Figure 3-1: A/D to panel LED's program.

3.3 Capture a Line

The program in figure 3-2 averages any raster line (lines 1-525) any number of times up to 99999 and outputs the line to the printer as a graph of the voltage distribution with time running down the paper and voltage across the paper as it is printed out. The voltage scale is 1 volt for 256 dots across (234 mV per inch). In addition, the delay after the rising edge of the horizontal sync pulse and the number of points to be outputted to the printer are programable. The average, line number, delay, and number of points are varables entered by the operator. The structure of the image is given in table 3-1.

TABLE 3-1

IMAGE LINES

LINE	INFORMATION
1-22	NONE
22-262	VALID
263	HALF VALID
264-283	NONE
284	HALF VALID
285-525	VALID

There are 482 full lines in the image with valid information. The lines are interweaved starting with line 22 and its pair 285 at the top of the image continuing down until line 262 and its pair 525 at the bottom. Line 22 is the top line and line 525 is the bottom line.

```
1       $BATCH
2             INTEGER*4 RGSTR(16),DATA(2048),LIN(24)
3             WRITE(1,1)
4       1     FORMAT('ENTER LINE NUMBER (I3)')
5             READ(1,2)LINE
6       2     FORMAT(I3)
7             WRITE(1,3)
8       3     FORMAT('ENTER NUMBER OF AVERAGES (I5)')
9             READ(1,4)N
10      4     FORMAT(I5)
11            WRITE(1,5)
12      5     FORMAT('ENTER DELAY (I5)')
13            READ(1,4)M
14            WRITE(1,6)
15      6     FORMAT('ENTER NUMBER OF POINTS (I4)')
16            READ(1,7)NPOINT
17      7     FORMAT(I4)
18      $ASSM
19      DIOUT   EQU   0              RGSTR 0 WILL HOLD DIO OUTPUT ADDRESS.
20      DIOIN   EQU   1              RGSTR 1 WILL HOLD DIO INPUT ADDRESS.
21      TEMP1   EQU   2              TEMP1 IS SCRATCH AREA.
22      TEMP2   EQU   3              TEMP2 IS SCRATCH AREA.
23      LINCT   EQU   4              LINCT IS LOOP COUNTER FOR AVERAGES.
24      PIXEL   EQU   5              RGSTR'S 5,6,7 CONTROL THE NUMBER
25      *                            OF HALFWORDS (TWO SAMPLES) PER
26      *                            LINE TO READ. USED WITH BXLE.
27              STM   0,RGSTR        SAVE CPU RGSTR'S TO RESTORE LATER.
28              LHI   DIOUT,X'00A8'  LOAD DIOUT WITH DIO OUTPUT DEVICE NUMBER.
29              LHI   DIOIN,X'00A9'  LOAD DIOIN WITH DIO INPUT DEVICE NUMBER.
30              LHI   TEMP1,X'00C0'  LOAD COMMAND INSTRUCTION THAT
31              OCR   DIOUT,TEMP1    INHIBITS DIO INTERUPTS.  THEN
32              OCR   DIOIN,TEMP1    OUTPUT IT TO DIO (INPUT AND OUTPUT).
33              LHI   TEMP1,X'F907'  SELECT MEMORY SIZE = 2048 AND OUTPUT IT.
34              WHR   DIOUT,TEMP1
35              LHI   TEMP1,X'0802'  NEXT 9 LINES CONVERT 'LINE' INTO DAU CODE
36      *                            AND OUTPUT IT.
37              S     TEMP1,LINE     TEMP1=X'802'-LINE.
38              EXBR  TEMP2,TEMP1    UPPER BYTE OF TEMP1 TO LOWER BYTE OF TEMP2.
39              NHI   TEMP2,X'00FF'  DELETE UPPER BYTE RESIDUE.
40              OHI   TEMP2,X'F800'  ADD INSTRUCTION CODE FOR UPPER BYTE
41              WHR   DIOUT,TEMP2    OF LINE DETECT LATCH AND OUTPUT IT.
42              NHI   TEMP1,X'00FF'  DELETE UPPER BYTE RESIDUE OF X'0802'-LINE.
43              OHI   TEMP1,X'FD00'  ADD INSTRUCTION CODE FOR LOWER BYTE
44              WHR   DIOUT,TEMP1    OF LINE DETECT LATCH AND OUTPUT IT.
45              LI    TEMP1,Y'00010000'  NEXT 11 LINES OUTPUT DELAY TO DAU.
46              S     TEMP1,M        TEMP1=Y'10000'-DELAY
47              EXBR  TEMP2,TEMP1    LOAD TEMP2 WITH UPPER BYTE OF TEMP1.
48              NHI   TEMP2,X'00FF'  DELETE UPPER BYTE RESIDUE FROM EXBR.
49              OHI   TEMP2,X'F100'  ADD INSTRUCTION CODE FOR UPPER BTYE OF
50              WHR   DIOUT,TEMP2    DELAY LATCH AND OUTPUT IT.
51              NHI   TEMP1,X'00FF'  DELETE UPPER BYTE RESIDUE.
52              OHI   TEMP1,X'F300'  ADD INSTRUCTION CODE FOR LOWER BYTE OF
53              WHR   DIOUT,TEMP1    DELAY LATCH AND OUTPUT IT.
54              LHI   TEMP1,X'FF39'  SELECT 30 MHZ, LINE STOP, ENABLE I/O,
55              WHR   DIOUT,TEMP1    VIDEO MODE, AND LOAD ALL LATCHES.
56              RHR   DIOIN,TEMP1    SET BUSY FROM DIO INPUT.
57      BUSY1   SSR   DIOIN,TEMP1    WAIT ONE BUSY TO INSURE THAT NEXT
58              BTBS  8,BUSY1        BUSY IS THE RESULT OF PROGRAMMED LINE AND
59      *                            NOT RANDOM POWER UP STATES OR PREVIOUS
60      *                            PROGRAMMED STATES.
61              RHR   DIOIN,TEMP1    SET BUSY.
62              L     LINCT,N        LINCT=NUMBER OF AVERAGES.
63      NEWLIN  LHI   TEMP1,X'FF39'  INITIALIZE DAU AS ABOVE...
64              WHR   DIOUT,TEMP1
65      BUSY2   SSR   DIOIN,TEMP1    CHECK IF BUSY FROM DAU IS CLEAR.
66              BTBS  8,BUSY2        BRANCH IF BUSY=1
67              LIS   PIXEL,0        SET PIXEL OFFSET TO ZERO.
68              LIS   PIXEL+1,8      OFFSET INCREMENT =8 (EACH DATA POINT IS
69      *                            STORED AS INTEGER*4, TWO POINTS ARE CAPTURED
70      *                            PER 'RHR' INSTRUCTION HENCE THE OFFSET
71      *                            MUST BE INCREMENTED BY 2*4=8).
72              L     PIXEL+2,NPOINT END LOOP AT 4*(NPOINT-2), 4 IS A RESULT OF
73              SIS   PIXEL+2,2      DATA BEING INTEGER*4, -2 COMES FROM THE FACT
74              SLLS  PIXEL+2,2      THAT TWO PIXEL ARE READ PER LOOP.
75      NEWHW   RHR   DIOIN,TEMP1    READ A HALFWORD FROM DIO INPUT.
76              EXBR  TEMP2,TEMP1    LOAD SECOND SAMPLE OF HALFWORD INTO
77      *                            LOWER BYTE OF TEMP2.
78              NHI   TEMP1,X'00FF'  WINDOW OUT FIRST SAMPLE
79              NHI   TEMP2,X'00FF'  WINDOW OUT SECOND SAMPLE.
80              AM    TEMP1,DATA(PIXEL)  ADD FIRST SAMPLE TO MEMORY FORMING
```

Figure 3-2A: Program that captures any raster line.

```
81                AM     TEMP2,DATA+4(PIXEL) THE AVERAGE.  REPEAT FOR TEMP2
82          *                                 BUT MOVE OVER TO NEXT WORD LOCATION.
83                BXLE   PIXEL,NEWHW         IF PIXELS REMAIN THEN BRANCH.
84                SIS    LINCT,1             DECREMENT LINE COUNTER.
85                BNZ    NEWLIN              BRANCH UNTIL 'M' LINES READ.
86                LM     0,RGSTR             RESTORE CPU RGSTR'S
87          $FORT
88                WRITE(6,8)LINE,N,M,NPOINT
89          8     FORMAT('LINE = ',I3,3X,'AVERAGE = ',I5,3X,
90               1'DELAY = ',I5,3X,'POINT = ',I4/1HO)
91                DO 11 I=1,NPOINT
92                CALL CLRLIN(LIN)
93                IDOT=DATA(I)/N
94                IF(IDOT.EQ.IDOT1 .OR.  I.EQ.1) IDOT1=IDOT-1
95                J1=IDOT
96                J2=IDOT1-1
97                IF(IDOT.LT.IDOT1) GOTO 22
98                J1=IDOT1+1
99                J2=IDOT
100         22    CONTINUE
101               DO 23 J=J1,J2
102               CALL SETDOT(LIN,J)
103         23    CONTINUE
104               CALL OUTLIN(LIN)
105               IDOT1=IDOT
106         11    CONTINUE
107               WRITE(6,9)
108         9     FORMAT(1H1)
109               STOP
110               END
111         C*****************************************************
112         C   THIS SUBROUTINE PLOTS THE DOTS
113         C*****************************************************
114               SUBROUTINE PLTDOT(LINE,IDOT)
115               INTEGER LINE(24)
116               ENTRY CLRLIN
117               DO 1 I=1,24
118         1     LINE(I)=Y'40404040'
119               RETURN
120               ENTRY SETDOT
121               IWD=IDOT/24+1
122               IBIT=8*(MOD(IDOT,24)/6)+7-MOD(MOD(IDOT,24),6)
123               CALL BSET(LINE(IWD),IBIT)
124               RETURN
125               ENTRY OUTLIN
126               IX05=Y'05000000'
127               WRITE(6,2) (LINE(J),J=1,24),IX05
128         2     FORMAT(24A4,A2)
129               RETURN
130               END
131         $BEND
```

Figure 3-2B: Program that captures any raster line.

Lines 2-17 of the program enter the varables line number, number of averages, length of delay, and number of points to be outputted to printer. Typical numbers for length of delay and number of points are 1191 and 1578 respectively. These numbers will print out only the valid points of the line selected. Memory size of 2048 bytes is selected at lines 33 and 34. Lines 35-44 load the LINE LATCH of the DAS with $802-LINE so that the programmed raster line can be captured. Lines 45-53 load the DELAY COUNTER LATCHES with the programmed delay. Lines 54-61 wait for one BUSY=0 before reading the data so that the possibility of random states causing a false trigger is eliminated. Since the LINE COUNTER is loaded only at raster line 2, the new value will not be loaded until this time and the old value will be active. Line 62 initialize NEWLIN loop counter to the number of averages needed. Lines 63-66 wait for the next BUSY=0 from the DIO which signals that program that the DAS has data to transmit. Lines 67-83 read the samples from the DAS and add them to the DATA array in memory. Lines 84 and 85 make up the NEWLIN loop branch. Lines 88-109 connect the data points and output them to the printer. Lines 114-130 make up the subroutine that output dots to the computer.

3.4 Digitize Image

The program in figure 3-3 digitizes the entire image and stores it on tape. The operator can average the image from 1 to 64 times to improve noise performance; this is the only number the operator must enter. Instead of a simple raster line by raster line averaging scheme, it has been necessary

```
      INTEGER*2 DATA(100992),LINE,AVER,COUNT
      INTEGER*4 RGSTR(16),STATUS
      WRITE(1,1)
1     FORMAT('ENTER NUMBER OF AVERAGES (I2)')
      READ(1,2)AVER
2     FORMAT(I2)
      NBYTES=2*1578
      MEMPT=0
      COUNT=X'0483'
      DO 3 LINE=1,482
$ASSM
DIOUT   EQU   0                  POINTER FOR DIO OUTPUT IN RGSTR 0.
DIOIN   EQU   1                  POINTER FOR DIO INPUT IN RGSTR 1.
TEMP1   EQU   2                  RGSTR 2 FOR SCRATCH.
TEMP2   EQU   3                  RGSTR 3 FOR SCRATCH.
INITAL  EQU   4                  RGSTR'S 4,5,6 ARE USED AS LOOP RGSTR'S
*                                TO INITIALIZE RUNNING AVERAGE.
LINCT   EQU   7                  RGSTR'S 7,8,9 ARE USED WITH BXLE IN THE
*                                LOOP STORING 'AVER' NUMBER OF LINES.
RDLOOP  EQU   10                 RGSTR'S 10,11,12 ARE USED WITH BXLE IN THE
*                                LOOP FOR READING SAMPLES FROM THE DAU.
RTN1    EQU   13                 THIS RGSTR RESERVED FOR RETURN ADDRESS
*                                FOR SUBROUTINES ONE DEEP.
RTN2    EQU   14                 THIS RGSTR RESERVE FOR TWO DEEP.
LED     EQU   15                 HOLDS THE ADDRESS FOR CONTROL PANEL  LED'S.
NPOINT  EQU   1578
DELAY   EQU   Y'00010000'-1190
        STM   0,RGSTR            SAVE CPU RGSTR'S.
*
*****************************************************
*
* DECREMENT BCD COUNT AND OUTPUT TO CONTROL PANEL LED'S
        LHI   LED,X'0001'        ENTER CONTROL PANEL ADDRESS.
        LHI   TEMP1,X'0040'      PLACE PANEL IN SUCESSIVE BYTE
        OCR   LED,TEMP1          LOAD MODE.
        LH    TEMP1,COUNT        ENTER LINES YET TO OUTPUTPLUS 1.
        SIS   TEMP1,1            DECREMENT BY ONE
        LR    TEMP2,TEMP1        CHECK FOR A BORROW FROM TENS POSITION...
        NHI   TEMP2,X'000F'
        CLHI  TEMP2,X'000F'
        BNE   NOBOW
        SIS   TEMP1,6            PUT A 9 IN LEAST SIGNIFICANT DIDIT (LSD).
        LR    TEMP2,TEMP1        CHECK FOR A BORROW FROM HUNDREDS POSITION...
        NHI   TEMP2,X'00F0'
        CLHI  TEMP2,X'00F0'
        BNES  NOBOW
        SHI   TEMP1,X'0060'      PUT A 9 IN SECOND LSD.
NOBOW   STH   TEMP1,COUNT        STORE COUNT FOR NEXT LOOP PASS.
        WDR   LED,TEMP1          OUTPUT FIRST BYTE TO LED DISLAYS.
        EXBR  TEMP1,TEMP1        PUT SECOND BYTE INTO FIRST AND OUTPUT...
        WDR   LED,TEMP1
        EXHR  TEMP1,TEMP1        PUT THIRD BYTE INTO FIRST AND OUTPUT...
        WDR   LED,TEMP1
        EXBR  TEMP1,TEMP1        PUT FORTH BYTE IN FIRST AND OUTPUT...
        WDR   LED,TEMP1
*
*****************************************************
*
* DRIVER PROGRAM FOR DIGITIZING
        LHI   DIOUT,X'00A8'      ENTER DIO OUTPUT ADDRESS.
        LHI   DIOIN,X'00A9'      ENTER DIO INPUT ADDRESS.
        LHI   TEMP1,X'00C0'      INHIBIT DIO FROM INTERUPTING...
        OCR   DIOUT,TEMP1
        OCR   DIOIN,TEMP1
        LHI   TEMP1,DELAY        NEXT 7 LINES OUTPUT DELAY TO DELAY LATCHES.
        EXBR  TEMP2,TEMP1        UPPER BYTE OF TEMP1 TO LOWER BYTE OF TEMP2.
        NHI   TEMP2,X'00FF'      CLEAR RESIDUE FOR EXCHANGE.
        OHI   TEMP2,X'F100'      ADD INSTUCTION CODE TO LOAD UPPER BYTE OF
        WHR   DIOUT,TEMP2        DELAY REGISTER AND OUTPUT IT.
        NHI   TEMP1,X'00FF'      CLEAR UPPER BYTE.
        OHI   TEMP1,X'F300'      ADD INSTRUCTION CODE TO LOAD LOWER BYTE OF
        WHR   DIOUT,TEMP1        DELAY REGISTER AND OUTPUT IT.
        LHI   TEMP1,X'F907'      SELECT MEMORY SIZE = 2048...
        WHR   DIOUT,TEMP1
        LH    LINCT,LINE         LOAD THE LINE COUNTER RGSTR WITH LINE
        CHI   LINCT,1            FROM DO LOOP, BRANCH IF NOT 1...
        BNE   JUMP1
        BAL   RTN1,SBR1          SYNCHRONIZE COMPUTER WITH DAU.
        LIS   INITAL,1           START DIGITIZING WITH LINE 1 AND
        LIS   INITAL+1,1         THEN INCREMENT ENDING VALUE BY ONE
```

Figure 3-3A: Program for digitizing image.

```
81                LH     INITAL+2,AVER        UNTIL 'AVER' LINES ARE READ IN THE LOOP
82      *                                     AT A TIME.
83      LOOP1     LIS    LINCT,1              FIRST LINE IS 1.
84                LIS    LINCT+1,1            INCREMENT LINCT BY 1.
85                LR     LINCT+2,INITAL       ENDING LINE IS IN INITAL.
86                BAL    RTN1,SBR2            READ BLOCK OF LINES FROM LINCT TO
87      *                                     LINCT+2. EACH EXECUTION OF NEXT INSTRUCTION
88      *                                     WILL CAUSE BLOCK TO GROW BY ONE UNTIL IT
89      *                                     IS 'AVER' LINES WIDE.
90                BXLE   INITAL,LOOP1         DIGITIZE NEXT LINE.
91                B      EXIT                 FINISHED!
92      JUMP1     BAL    RTN1,SBR1            GET SYNCHRONIZED WITH DAU.
93                LIS    LINCT+1,1            LINE COUNTER INCREMENT=1
94                LR     LINCT+2,LINCT        LOAD LINCT+2 WITH LAST LINE IN BLOCK.
95                AH     LINCT+2,AVER         LINCT+2=LINCT + AVER - 1...
96                SIS    LINCT+2,1
97                CHI    LINCT+2,482          CHECK IF LAST LINE IN BLOCK EXCEEDS
98                BFFS   2,JUMP2              482, AND IF IT DOES, MAKE EQUAL TO 482.
99                LHI    LINCT+2,482
100     JUMP2     BAL    RTN1,SBR2            READ BLOCK OF LINES STATRING AT LINCT
101     *                                     AND ENDING AT LINCT+2.
102               B      EXIT                 FINISHED!
103     *
104     ***************************************************
105     *
106     * THIS SUBROUTINE ISSURES THAT NEXT BUSY=0 IS RESULT OF THE PROGRAMMED
107     * LINE BEING DETECTED AND NOT PREVIOUS RANDOM STATES.
108     SBR1      BAL    RTN2,SBR3            LOAD LINE LATCHES IN DAU WITH LINE IN LINCT.
109               RHR    DIOIN,TEMP1          BUSY=1
110               LHI    TEMP1,X'FF39'        SELECT; 30 MHZ, LINE DETECT MODE, LOAD
111               WHR    DIOUT,TEMP1          LATCHES; SEND RESET, ENABLE TRI-STATE BUFFER,
112     *                                     VIDEO MODE.
113     BUSY1     SSR    DIOIN,TEMP1          WAIT FOR BUSY=0...
114               BTBS   8,BUSY1
115               RHR    DIOIN,TEMP1          SET BUSY, NOW NEXT BUSY=0 WILL BE RESULT
116     *                                     OF PROGRAMMED LINE.
117               BR     RTN1                 RETURN FROM SUBROUTINE.
118     *
119     ***************************************************
120     *
121     * THIS SUBROUTINE DIGITIZES A BLOCK OF LINES STARTING
122     * AT THE LINE IN LINCT AND ENDING AT LINCT+2. THE
123     * LINES ARE ADDED TO THE RESPECTIVE MEMORY LOACTIONS
124     * IN THE DATA ARRAY.
125     SBR2      L      RDLOOP,MEMPT         POINT TO OLDEST DATA LOCATION IN DATA.
126               LIS    RDLOOP+1,4           DATA WILL BE INCREMENTED BY 4.
127     *                                     TWO SAMPLES, EACH INTEGER*2 EQUAL 4.
128               LR     RDLOOP+2,RDLOOP      ENDING LOCATION OF LINE BLOCK IS
129               AHI    RDLOOP+2,2*1578-4    FIRST LOCATION PLUS NUMBER OF BYTES IN A
130     *                                     LINE MINUS FOUR SINCE WE START AT OFFSET OF
131     *                                     ZERO AND READ SAMPLES (4 BYTE) AT A TIME.
132     LOOP3     LHI    TEMP1,X'FF39'        SELECT 30 MHZ, LINE DETECT MODE,
133               WHR    DIOUT,TEMP1          RESET, LOAD LATCHES, ENABLE TRI-STATE
134     *                                     AND SELECT VIDEO MODE.
135     BUSY2     SSR    DIOIN,TEMP1          WAIT FOR BUSY=0.
136               BTBS   8,BUSY2              WHEN BUSY=0 LINE HAS BEEN DIGITIZED.
137               AIS    LINCT,1              LOAD LINE DETECTOR WITH NEXT LINE NUMBER TO
138               BAL    RTN2,SBR3            INCREASE EFFICIENCY.
139               SIS    LINCT,1              RESTORE VALUE.
140     LOOP5     RHR    DIOIN,TEMP1          READ TWO SAMPLES FROM DAU.
141               EXBR   TEMP2,TEMP1          PUT SECOND BYTE OF TEMP1 INTO TEMP2.
142               NHI    TEMP1,X'00FF'        CLEAR UPPER BYTE.
143               NHI    TEMP2,X'00FF'        CLEAR UPPER BYTE.
144               AHM    TEMP1,DATA(RDLOOP)   ADD TO MEMORY FIRST SAMPLE.
145               AHM    TEMP2,DATA(RDLOOP)   ADD TO MEMORY SECOND SAMPLE.
146               BXLE   RDLOOP,LOOP5         INCREMENT MEMPT AND BRANCH IF MORE DATA.
147               CI     RDLOOP,2*64*1578     CHECK IF MEMPT MUST WRAP AROUND...
148               BNES   JUMP3
149               LIS    RDLOOP,0             WRAP AROUND TO 0.
150               LHI    RDLOOP+2,2*1578-4    POINT TO LAST POINT IN LINE.
151               B      JUMP4                BYPASS MEMPT INCREMENT.
152     JUMP3     AHI    RDLOOP+2,2*1578      MOVE ENDIND POINTER UP ONE LINE.
153     JUMP4     BXLE   LINCT,LOOP3          GRAB ANOTHER LINE IF NOT END OF BLOCK.
154               BR     RTN1                 RETURN FROM SUBROUTINE.
155     *
156     ***************************************************
157     *
158     * THIS SUBROUTINE CODES THE LINE NUMBER CONTAINED IN LINCT
159     * AND OUTPUTS IT TO DAU. LINES IN THIS PROGRAM ARE NUMBERED
160     * 1 THROUGH 482 INORDER OF PHYSICAL POSITION ON THE MONITER
```

Figure 3-3B: Program for digitizing image.

```
      * FROM TOP TO BOTTOM.
SBR3      LHI   TEMP1,X'06E6'        TEMP1=X'06E6'+X'0106'*MOD(LINCT,2)
          LR    TEMP2,LINCT         -INT(LINCT/2), ADD INSTRUCTION CODE TO
          AIS   TEMP2,1             OUTPUT TO UPPER BYTE OF LINE LATCH...
          SRLS  TEMP2,1
          BNCS  JUMP5
          AHI   TEMP1,X'0106'
JUMP5     SR    TEMP1,TEMP2
          EXBR  TEMP2,TEMP1         UPPER BYTE OF TEMP1 TO LOWER BYTE OF TEMP2.
          NHI   TEMP2,X'00FF'       CLEAR UPPER BYTE.
          OHI   TEMP2,X'FB00'       ADD INSTRUCTION CODE TO LOAD UPPER
          WHR   DIOUT,TEMP2         BYTE OF LINE LATCH.
          NHI   TEMP1,X'00FF'       CLEAR UPPER BYTE.
          OHI   TEMP1,X'FD00'       ADD INSTRUCTION CODE TO LOAD LOWER
          WHR   DIOUT,TEMP1         BYTE OF LINE LATCH.
          BR    RTN2                RETURN FROM SUBROUTINE.
EXIT      LM    0,RGSTR             RESTORE CPU RGSTR'S.
      $FORT
            J=MEMPT/2+1
            CALL SYSIO(RGSTR,Y'38',4,DATA(J),NBYTES,0)
            CALL IOERR(RGSTR,STATUS)
      $ASSM
      * CLEAR SPACE IN MEMORY HELD BY LAST LINE OUTPUTTED.
          LIS   1,0
          L     2,MEMPT
          LIS   3,2
          L     4,MEMPT
          AHI   4,2*1578-2
LOOP7     STH   1,DATA(2)
          BXLE  2,LOOP7
      $FORT
            MEMPT=MEMPT+NBYTES
            IF(MEMPT.EQ.2*64*1578) MEMPT=0
3     CONTINUE
      $ASSM
      * CLEAR CONTROL PANEL LEDS.
          STM   13,RGSTR
          LIS   15,0
          LIS   14,1
          LHI   13,X'0040'
          OCR   14,13
          WDR   14,15
          WDR   14,15
          WDR   14,15
          WDR   14,15
          LM    13,RGSTR
      $FORT
            STOP
            END
```

Figure 3-3C: Program for digitizing image.

to employ a more complicated approach in order to average out a slight horizontal drift problem of the SLAM. The reasons are discussed in more detail in Chapter 4. The program works with a block of adjacent raster lines (in physical position, not time of occurance) with the number of raster lines in the block being identical to the number of averages (n). Typically the block is organized with the top line of the block averaged n-1 times, the next raster line averaged n-2 times, the following raster line n-3 times, and so on, until the next from the bottom raster line of the block which is averaged once and the bottom raster line of the block is cleared (set equal to zero). The program then digitizes each line once more adding their values to the block. The top line of the block has now been averaged n times and is outputted to the magnetic tape unit. The block then moves down the image one raster line; the previous second raster line of the block from the top is shifted into the top position, the previous last raster line of the block shifts into the second to the last raster line of the block and zeros are shifted into the bottom raster line of the block. The image is stored on tape as 482 records, one record for each raster line with successive records corresponding to successive raster lines. Each record holds 1578 samples, each sample is in binary and two bytes long, the successive samples correspond to successive digitization in time. A record is 2*1578=3156 bytes long.

Lines 33-55 output the number of raster lines yet to be outputted to the LEDs on the control panel of the computer.

COUNT is a binary coded decimal number that keeps tract of the raster lines remaining. Lines 60-102 initialize the DAS and load the loop parameters that point to which line in the block is is being digitized. On the first pass the block must be setup, lines 78-91 handle this. The program at lines 86 and 100 call the subroutine that digitizes the lines of the block contained by the loop parameters.

Lines 108-117 make up the subroutine that wait for BUSY=0 before preceding (issue new programmed line is loaded into LINE COUNTER of the DAS). Lines 125-154 form the subroutine that digitizes the lines contained within the parameters of LINCT and LINCT+2 loop counters passed to it from the driver program. Lines 162-177 consist of the subroutine that load the LINE LATCH with the value contained in LINCT. Line 180 output to the magnetic tape the top line of the block. Lines 184-190 clear the bottom line of block. The lines in the block are not physically shifted but the pointer (MEMPT) is incremented by the number bytes necessary to point to the next line. If the pointer exceeds the bounds of the DATA array then it is wrapped around to zero (lines 147-151 and 192-194).

3.5 Interference Line Enhancement

One of the goals of this thesis is to store a interference pattern and enhance the interference lines from the SLAM interference image. This program (figure 3-4) enhances the interference lines. A correlated receiver is used to compare the waveform on the raster line with a stored replica of an ideal cross section of a interference line and

```
1       $BATCH
2             INTEGER*4 Y(1578),PBLK(5),LINE(33),RGSTR(7)
3             INTEGER*2 X(1578),H(33),POINT(48),DATAO(23136),COUNT,C1,C2
4             DATA(H(I),I=1,33)/874,952,1074,1201,1350,1501,
5            11652,1796,1928,2043,2144,2221,2285,2324,2357,
6            22362,2366,2324,2323,2276,2234,2166,2106,2017,
7            31936,1831,1736,1619,1517,1399,1300,1187,1099/
8             WRITE(1,1)
9       1     FORMAT('ENTER NUMBER OF DUPLICATIONS NEEDED (I2)')
10            READ(1,4)IDUP
11      4     FORMAT(I2)
12            WRITE(1,5)
13      5     FORMAT('ENTER NUMBER OF IMAGES STORED (I1)')
14            READ(1,6)NUM
15      6     FORMAT(I1)
16            WRITE(1,16)
17      16    FORMAT('ENTER LENGTH OF CORRELATION (I2)')
18            READ(1,17)LEN
19      17    FORMAT(I2)
20            C1=4*LEN
21            C2=2*1578-4*LEN-2
22            I1=17-LEN
23            I2=2*LEN+I1
24            I3=0
25            DO 18 I=I1,I2
26            I3=I3+1
27            X(I3)=X(I)
28            I3=I3+1
29      18    CONTINUE
30            DO 14 JH=1,NUM
31            COUNT=X'0483'
32            DO 3 I=1,482
33      $ASSM
34                  STM   13,RGSTR
35                  LIS   15,1
36                  LHI   14,X'0040'
37                  OCR   15,14
38                  LH    14,COUNT
39                  SIS   14,1
40                  LR    13,14
41                  NHI   13,X'000F'
42                  CLHI  13,X'000F'
43                  BNE   JUMP
44                  NHI   14,X'FFF0'
45                  OHI   14,X'0009'
46                  LR    13,14
47                  NHI   13,X'00F0'
48                  CLHI  13,X'00F0'
49                  BNES  JUMP
50                  NHI   14,X'FF0F'
51                  OHI   14,X'0090'
52      JUMP        STH   14,COUNT
53                  WDR   15,14
54                  EXBR  14,14
55                  WDR   15,14
56                  EXHR  14,14
57                  WDR   15,14
58                  EXBR  14,14
59                  WDR   15,14
60                  LM    13,RGSTR
61      $FORT
62            CALL SYSIO(PBLK,Y'58',3,X,3156,0)
63      $ASSM
64                  STM   10,RGSTR
65                  LH    15,C2
66                  LIS   14,2
67                  LIS   13,0
68      LOOP1       LH    12,C1
69                  LIS   11,0
70      LOOP2       LH    10,X(12,13)
71                  MH    10,H(12)
72                  AR    11,10
73                  SIS   12,2
74                  BTBS  7,LOOP2
75                  SLLS  13,1
76                  ST    11,Y(13)
77                  SRLS  13,1
78                  BXLE  13,LOOP1
79                  LM    10,RGSTR
80      $FORT
```

Figure 3-4A: Program for correlated receiver.

```
      DO 118 J=1,48
118   POINT(J)=0
      K=1
      J2=1578-2*LEN-2
      DO 2 J=3,J2
      IF(Y(J-1).LE.Y(J-2))GOTO 2
      IF(Y(J+1).LE.Y(J+2))GOTO 2
      IF(Y(J).LE.Y(J-1))GOTO 2
      IF(Y(J).LE.Y(J+1))GOTO 2
      POINT(K)=J
      K=K+1
2     CONTINUE
      DO 8 J=1,47
      K=48*(I-1)+J
      DATAO(K)=POINT(J)
8     CONTINUE
      DATAO(48*I)=48*I-47
      CALL SYSIO(PBLK,Y'38',4,POINT,96,0)
3     CONTINUE
$ASSM
          STM    13,RGSTR
          LIS    15,0
          LIS    14,1
          LHI    13,X'0040'
          OCR    14,13
          WDR    14,15
          WDR    14,15
          WDR    14,15
          WDR    14,15
          LM     13,RGSTR
$FORT
      DO 7 L=1,IDUP
      DO 9 I=1,482
      CALL CLEARL(LINE)
      DO 10 J=1,47
      K=48*(I-1)+J
      IDOT=DATAO(K)/3
      IF(IDOT.NE.0) CALL SETDOT(LINE,IDOT)
10    CONTINUE
      CALL OUTLIN(LINE)
9     CONTINUE
      WRITE(6,11)
11    FORMAT(1H1)
      DO 12 I=1,1578
      CALL CLEARL(LINE)
      DO 13 J=1,482
      K=48*J
      K1=DATAO(K)
      IDOT=DATAO(K1)
      IF(IDOT.NE.I) GOTO13
      DATAO(K)=DATAO(K)+1
      J1=483-J
      CALL SETDOT(LINE,J1)
13    CONTINUE
      CALL OUTLIN(LINE)
12    CONTINUE
      WRITE(6,11)
      DO 15 I=1,23136,48
      DATAO(I+47)=I
15    CONTINUE
7     CONTINUE
14    CONTINUE
      STOP
      END
      SUBROUTINE PLTDOT(LINE,IDOT)
      INTEGER LINE(32)
      ENTRY CLEARL
      DO 1 I=1,32
1     LINE(I)=Y'40404040'
      RETURN
      ENTRY SETDOT
      IWD=IDOT/24+1
      IBIT=8*(MOD(IDOT,24)/6)+7-MOD(MOD(IDOT,24),6)
      CALL BSET(LINE(IWD),IBIT)
      RETURN
      ENTRY OUTLIN
      IX05=Y'05000000'
      WRITE(6,2) (LINE(J),J=1,32),IX05
2     FORMAT(32A4,A2)
      RETURN
      END
$BEND
```

Figure 3-4B: Program for correlated receiver.

where relative maxima occur in the correlated result the interference line is detected (H. L. Van Trees, 1968). Array DATA0 holds the detected interference points for all 482 lines. The first 48 terms of the array are for the first image line, the next 48 terms for the second and so on. All images have less than 48 interference lines (typically 38). The last term of the 48 terms reserved for a image line is a pointer used in the program. This program prints out every dot stored in DATA0 to give the operator a hard copy of the enhanced image. Two sizes are printed, one condensed bearing a close similarity to the image on the monitor and the other the true resolution of the DIGITIZER.

Lines 4-7 contain the data for the stored replica and are loaded into array H. Lines 8-11 input the number of times each image will be printed out on the printer. Lines 12-15 input the number of images on the tape to be correlated. This allows the operator to store multiple images during a days work then before leaving enter the number of stored images and have the computer work overnight without further operator intervention. Lines 16-19 allow the length of the correlated receiver to be selected. Lines 20-29 initialize the H array with the new length and sets up some constants. Line 30 is the DO loop for the number of stored images. Lines 33-61 output to the control panel LED's the number of raster lines remaining in the current image to be digitized. Line 62 reads a digitized raster line from the magnetic tape and stores it in the X array. Lines 63-80 perform the correlation of the X array with the H array. The

result is stored in the Y array. Care is taken to line up the H array with the X array to issure all multiplications are on valid X array data. Notice the Y array is INTEGER*4; hence memory locations are on word boundries. Lines 81-82 clear the POINT array which is a scratch pad array for the peak detection (interference line location). Lines 83-92 find the relative maxima of the correlated raster (Y array), a maximum being three points that increase and then the third point with the next two decreasing. The maximum is stored in the POINT array as location of occurrance, with successive maxima being loaded successively. Line 93-97 load the DATA0 array with the new detected interference lines with the last member of the POINT array being loaded with the location of the first member of POINT array in the DATA0 array. Line 98 outputs the POINT array on to tape for a permanent record. Lines 100-111 clear the panel display. Line 112 is the loop controlling the number of duplications. Lines 113-121 output the condensed image (horizontal axis of image divided by three). Lines 124-140 output the image rotated by 90 degrees (paper to short for 1578 points but not for 482 points). Also term 48 of each line has been altered and is restored at lines 138-140. Line 141 is the bottom of the duplicate loop. Line 142 is the bottom of the image loop. Lines 145-161 make up the subroutine for the printer.

3.6 Output Image To Printer

The program in figure 3-5 creates a replica on the printer of the image stored on tape. The printer has two gray levels, print a dot or don't print a dot. To achieve a

```
      INTEGER*4 RGSTR(16),LINE(24),SEED,SUM,PBLK(5)
      INTEGER*4 HIS(256),HIS1(256),MAP(256)
      INTEGER*2 DATA(2048),AVER
      ICAR=Y'05000000'
      SEED=1
      WRITE(1,1)
1     FORMAT('ENTER AVERAGE  (I2)')
      READ(1,2)AVER
2     FORMAT(I2)
      AVER=AVER*3
      NBYTES=2*1578
      DO 3 I=1,482
      CALL SYSIO(PBLK,Y'58',2,DATA,NBYTES,0)
      CALL IOERR(PBLK,J)
$ASSM
*************************************************
* THIS ASSM PROGRAM PREPARES LINE(ARRAY) TO OUTPUT
* A LINE OF DOTS REPRESENTIVE OF THE DIGITIZED LINE
* STORED IN DATA(ARRAY). A DOT WILL BE PRINTED AT A
* DOT LOCATION DEPENDING ON IF THE AVERAGE OF THE
* THREE PIXELS (MAKING THE LOCATION) EXCEED A
* RANDOMLY GENERATED NUMBER WITH THE SAME RANGE
* THIS PROCESS IS CALLED DITHERING.
* IN ADDITION THE GRAY LEVEL DISTRIBUTION IS FOUND.
        STM   0,RGSTR
        BAL   0,CLRLIN              CLEAR LINE ARRAY.
        LIS   1,0                   NEXT THREE INITIALIZE POINT LOOP.
        LIS   2,6
        LHI   3,2*1578-6
        LIS   4,1                   NEXT 2 INTIALIZE OFTENLY USED CONSTANTS.
        LH    5,AVER
LOOP1   LH    6,DATA(1)             NEXT 4 FORM THE AVERAGE FOR POINT.
        AH    6,DATA+2(1)
        AH    6,DATA+4(1)
        DHR   6,5
        BAL   0,RANDOM              FIND AND PUT RANDOM NUMBER IN RGSTR 15.
        CR    7,15                  IF AVERAGE<RANROM DON?T PRINT...
        BMS   JUMP1
        LR    13,1                  SET DOT AT CORRESPONDING TO THREE PIXELS...
        LIS   14,2*3
        DHR   13,14
        BAL   0,SETDOT
JUMP1   SLLS  7,2                   INCREMENT GRAY LEVEL HISTOGRAM
        AM    4,HIS(7)              FOR THIS POINT.
        BXLE  1,LOOP1               LOOP UNTIL WHOLE LINE IS'PREPARED.
        LM    0,RGSTR
$FORT
      WRITE(3,4)(LINE(J),J=1,24),ICAR
4     FORMAT(24A4,A2)
3     CONTINUE
      WRITE(3,7)
7     FORMAT(1H1)
      MAX=HIS(1)
      DO 6 I=2,256
      MAX=MAXO(MAX,HIS(I))
6     CONTINUE
      DO 5 I=1,256
$ASSM
        STM   0,RGSTR               CLEAR LINE(ARRAY)
        BAL   0,CLRLIN
        LM    0,RGSTR
$FORT
      WRITE(3,4)(LINE(J),J=1,24),ICAR
$ASSM
        STM   0,RGSTR               OUTPUT NORMALIZED HISTOGRAM.
        L     1,I
        SLLS  1,2
        LIS   2,0
        L     3,HIS-4(1)
        SLLS  3,8
        D     2,MAX
LOOP2   LR    14,3
        BAL   0,SETDOT
        SIS   3,1
        BNMS  LOOP2
        LM    0,RGSTR
$FORT
      WRITE(3,4)(LINE(J),J=1,24),ICAR
5     CONTINUE
      WRITE(3,7)
```

Figure 3-5A: Output image and histogram equalized image program.

```
      DO 8 I=1,482
      CALL SYSIO(PBLK,Y'A0',2,0,1,0)
      CALL IOERR(PBLK,J)
      IF(I.EQ.1) WRITE(1,101)J
101   FORMAT(Z8)
8     CONTINUE
      SUM=HIS(1)
      DO 9 I=2,256
      J=SUM*255*3/(482*1578)+1
      MAP(I)=J
      HIS1(J)=HIS1(J)+HIS(I)
      SUM=SUM+HIS(I)
9     CONTINUE
      DO 10 I=1,482
      CALL SYSIO(PBLK,Y'58',2,DATA,NBYTES,0)
      CALL IOERR(PBLK,J)
$ASSM
          STM   0,RGSTR             PREPARE TO OUTPUT A LINE OF HISTOGRAM
          BAL   0,CLRLIN            EQUALIZED IMAGE.
          LIS   1,0
          LIS   2,6
          LHI   3,2*1578-6
          LH    5,AVER
LOOP3     LH    6,DATA(1)
          AH    6,DATA+2(1)
          AH    6,DATA+4(1)
          DHR   6,5
          SLLS  7,2
          L     7,MAP(7)
          BAL   0,RANDOM
          CR    7,15
          BMS   JUMP2
          LR    13,1
          LIS   14,2*3
          DHR   13,14
          BAL   0,SETDOT
JUMP2     BXLE  1,LOOP3
          LM    0,RGSTR
$FORT
      WRITE(3,4)(LINE(J),J=1,24),ICAR
10    CONTINUE
      WRITE(3,7)
      MAX=HIS1(1)
      DO 11 I=2,256
      MAX=MAXO(MAX,HIS1(I))
11    CONTINUE
      DO 12 I=1,256
$ASSM
          STM   0,RGSTR
          BAL   0,CLRLIN
          LM    0,RGSTR
$FORT
      WRITE(3,4)(LINE(J),J=1,24),ICAR
$ASSM
          STM   0,RGSTR             OUTPUT NEW HISTOGRAM.
          L     I,I
          SLLS  1,2
          LIS   2,0
          L     3,HIS1-4(1)
          SLLS  3,8
          D     2,MAX
LOOP4     LR    14,3
          BAL   0,SETDOT
          SIS   3,1
          BNMS  LOOP4
          LM    0,RGSTR
$FORT
      WRITE(3,4)(LINE(J),J=1,24),ICAR
12    CONTINUE
      STOP
$ASSM
RANDOM    LI    15,65539            PUT A RANDOM NUMBER (-1<RN<256)
          L     14,SEED             IN RGSTR 15...
          MR    14,14
          ST    15,SEED
          SRL   15,24
          BR    0
SETDOT    CI    14,0                CHECK -1<RGSTR 14<24*24 OTHERWISE SET TO
          BNCS  JMP1                CORRESPONDING BOUND...
          LIS   14,0
```

Figure 3-5B: Output image and histogram equalized image program.

```
161     JMP1    CI    14,24*24
162             BCS   JMP2
163             LHI   14,24*24-1
164     JMP2    LHI   15,24              CONVERT RGSTR 14 INTO DOT CODE FOR
165             DHR   14,15              PRINTER AND SET CORRESPONDING BIT OF WORD...
166             LR    13,15
167             SLLS  13,2
168             LIS   15,6
169             DHR   14,15
170             SLLS  15,3
171             AIS   15,7
172             SR    15,14
173             SBT   15,LINE(13)
174             BR    0
175     CLRLIN  LI    15,Y'40404040'     CLEAR LINE ARRAY.
176             LHI   14,4*24
177     LOP1    ST    15,LINE-4(14)
178             SIS   14,4
179             BNZS  LOP1
180             BR    0
181     *FORT
182           END
```

Figure 3-5C: Output image and histogram equalized image program.

greater range of gray levels a process called dithering is used. The dithering process takes a pixel (value between 0 and 255 inclusive), compares it with a random number over the same range and prints a point if the pixel value exceeds the random number. On the average a patch of area with the same gray level will have the same number of points set to give the patch the appearance from a distance of that gray level. Three consecutive pixels are combined into one dot. Each line of the image is printed out as one line on the printer. Hence the image on the printer is 526 dots across (horizontally) and 482 dots down (vertically). In addition this program does a histogram equalization on the image and prints it with the original image. Histogram equalization is applying a mapping process that maps gray levels of the original image into new gray levels that flatten the histogram of the original image. The histogram of both images is printed out.

In the program array HIS and HIS1 are the storage space for the histogram of the orignal and equalized images respectively. The MAP array holds the look up table for the mapping of the gray levels of the original into the new gray levels for the equalized image. Lines 6-9 input the number of times the image is averaged. The pixels on tape have not been divided by the number of averages taken. In this program it is necessary to recover the true average and this is the purpose of entering this number. Line 13 reads the data from the tape. Lines 24-36 divide the pixels of the line read from tape by the average, compare them with a

ramdom number generated by the branch to RANDOM, and increment the member of HIS that's counting the frequency of occurance of gray levels in the original image. One dot of the printer is represented by three pixels that are averaged together. Line 47 prints out one line of the image to the printer. Lines 51-79 output the normalized histogram of the original image to the printer. Lines 81-85 backrecord the tape to the beginning of the image.

Lines 86-93 form the map for the histogram equalization process. SUM is the accumulative sum of the frequency of occurance of gray levels starting at 0 gray level (HIS(1)) ending at the current gray level 'I' of the original image. The gray level is mapped to a new gray level based on the fractional number of occurances the have taken place at and before this gray level. For example if the accumulated number of occurances of a gray level equal one fourth the number of pixels, then this gray level will be assigned to one forth of 256 or 64.

Lines 94-122 read a line from the image, map the gray level into the histogram equalized level, then set the coresponding point of the line if the level exceeds the random number. Lines 124-150 normalize the histogram of the histogram equalized image and output it to the computer.

The assemble subroutines are called several times through the program and hence as subroutines save programing space. The first subroutine (Lines 153-159) is the random number generator. It works by product overflow. The second subroutine (Lines 160-179) sets a bit in the LINE array

representive of the dot in the image line to be printed (see printer manual). The last subroutine (lines 179-184) clears LINE array so that a new image line can be built.

CHAPTER 4

RESULTS

4.1 Introduction

This chapter discusses the results of histogram equalization of an image and the effect on averaging and length of correlation on the signal to noise ratio.

4.2 Comparision of Different Data Extraction Techniques

The effect on resolution and accuracy of the interference lines were examined by three different data extraction schemes and will be discussed (see figure 4-1). Only one interference line from each digitization method is shown. The interference line of each image is that of water with the LAG push-button activated (every raster line of the SLAM monitor represents the same laser scan line through the specimen. Under the LAG condition, the interference lines should be vertical and perfectly straight since the same raster line of the specimen is used for all the raster lines of the image. The interference line on the left of figure 4-1 is obtained by averaging the same raster line 64 times then proceeding to the next line, averaging it 64 times and so on. The bumps in the lines are due to the SLAM drifting horizontally. The perturbation of the drift is +/- 2 sample points of the 30 MHz A/D (+/- 67 ns) which is equivalent to +/- 5% of the normalized interference line shift (see Goss and O'Brien, 1979). Since the perturbation takes place over approximately 10 seconds, as will be shown, it is not perceptable to the eye or short exposures of film and

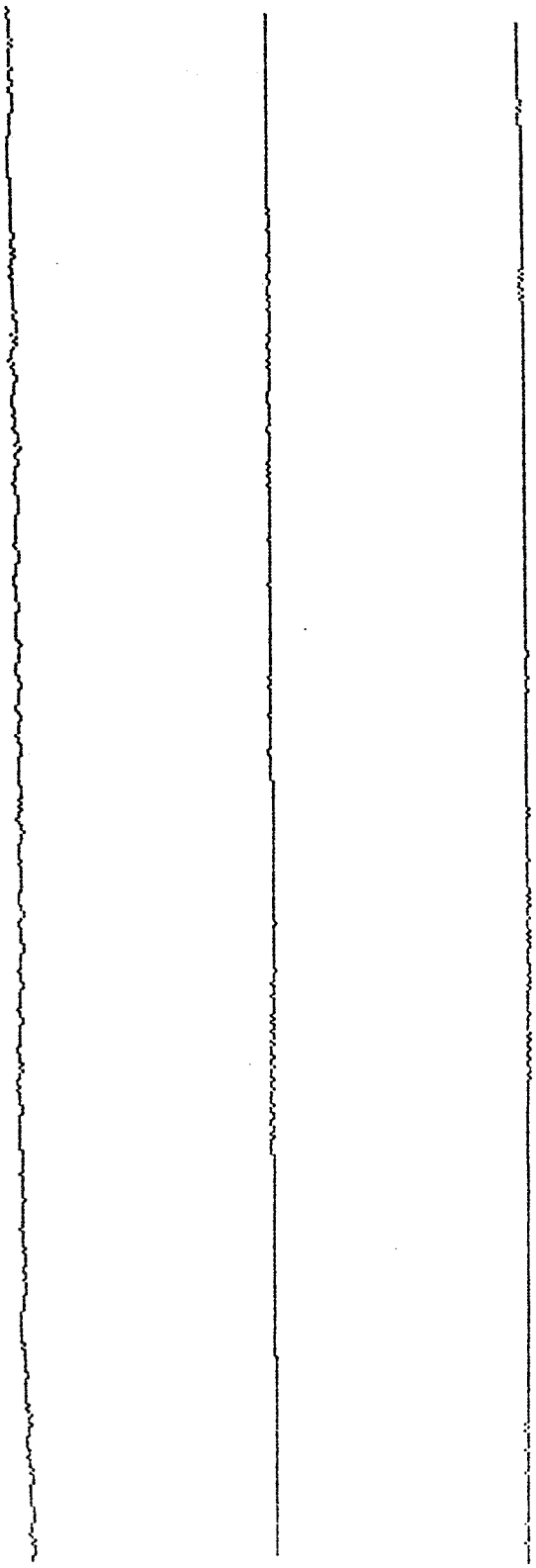


Figure 4-1: Comparision of different digitization methods.

therefore does not represent a serious problem. However, the elimination of this perturbation would greatly increase the potential accuracy for velocity determinations. But since there was a random component of this drift, the problem can be practically addressed in software as discussed below. The drift also appears to exhibit a periodic component at about 5 lines per cycle; hence an approximate period of 10 seconds.

The middle interference line in figure 4-1 is obtained by dividing the image into 7 blocks of 60 raster lines per block and a bottom block which has 62 raster lines. Each block is averaged as a whole; that is, first the top raster line of the block is digitized, then the next raster line and so on until the bottom line of the block is digitized. Then the top line of the block is again captured and the next and so until each line has been averaged the desired number of times. For a average number of 60 each line within the block will be digitized every 2 seconds hence allowing samples to be taken over many cycles of the drift. This method mostly eliminates the drift within a block . However the interference lines of bordering blocks do not always match up as the middle interference line in figure 4-1 shows. The reason for this mismatch is the amplitude of the drift is somewhat random and may have drifted more in one block than the bordering block. The randomness of the interference pattern can be seen in the line on the left which is effectively a graph of the drift of te SLAM. Ideally it would be desirable for the whole image to be in the computer's memory so that the image could be averaged over

and over on an image by image basis instead of a block by block basis but the memory requirements are massive: 1.5 megabytes of memory necessary when only 384 kilobytes are available.

The interference line on the right side of figure 4-1 resulted from a running average approach. In this approach, a block of raster lines within the image is operated on at once. The number of raster lines in the block is the same as the desired number of averages. The block of lines are arrange in order of position in the image and not in occurance of time. The first line in the block will have been averaged n-1 times (where n is the desired number of averages) the second line n-2 and so on until the last line of the block which is cleared. The program then digitizes one block of lines outputting the top line which will have been digitized n times. The block then moves down the image one line so that the previous second line, now becomes the first line, the third line the second, and so on, until the present last line is cleared. The block will continue down the block until raster line 525 is captured. Hence at n=64 the lines are averaged about every 2 seconds with adjacent lines being averaged over nearly the same time period so that adjacent lines are will connected.

4.3 Histogram Equalization

The effect of histogram equalization is demonstrated in figure 4-2. The image consist of three sheets of plastic forming steps in thickness from the top of the image with only water to the bottom with 3 sheets of plastic with the

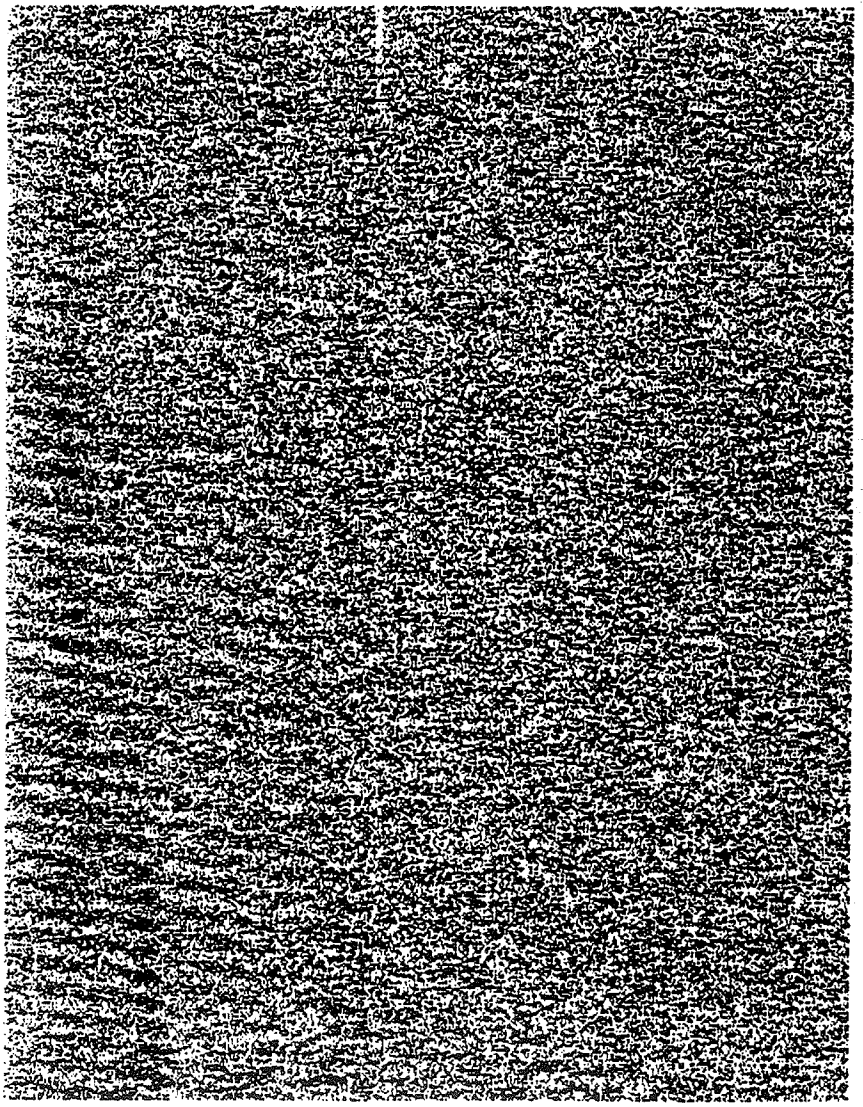


Figure 4-2A: Original image.

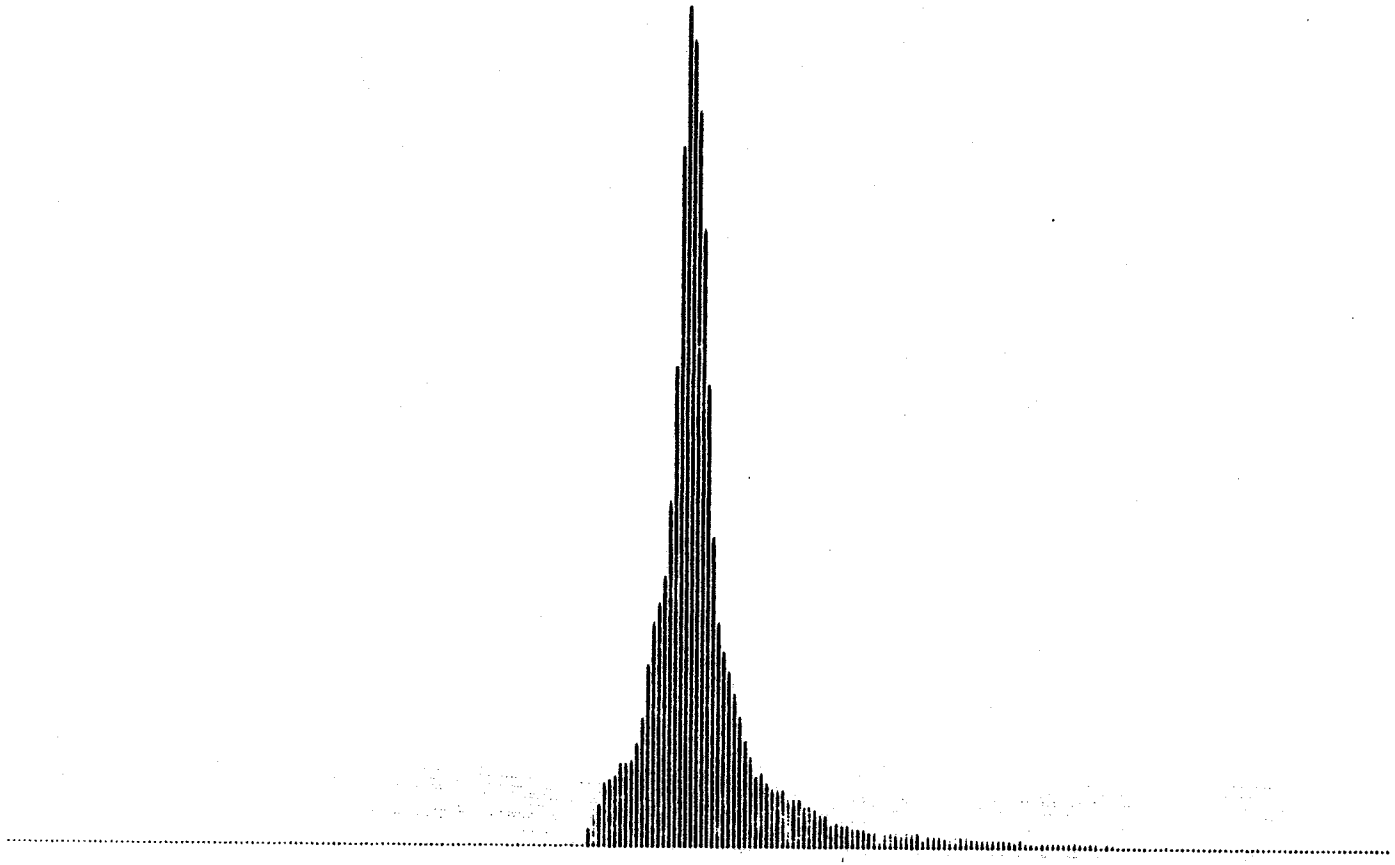


Figure 4-2B: Histogram of gray levels in original image.

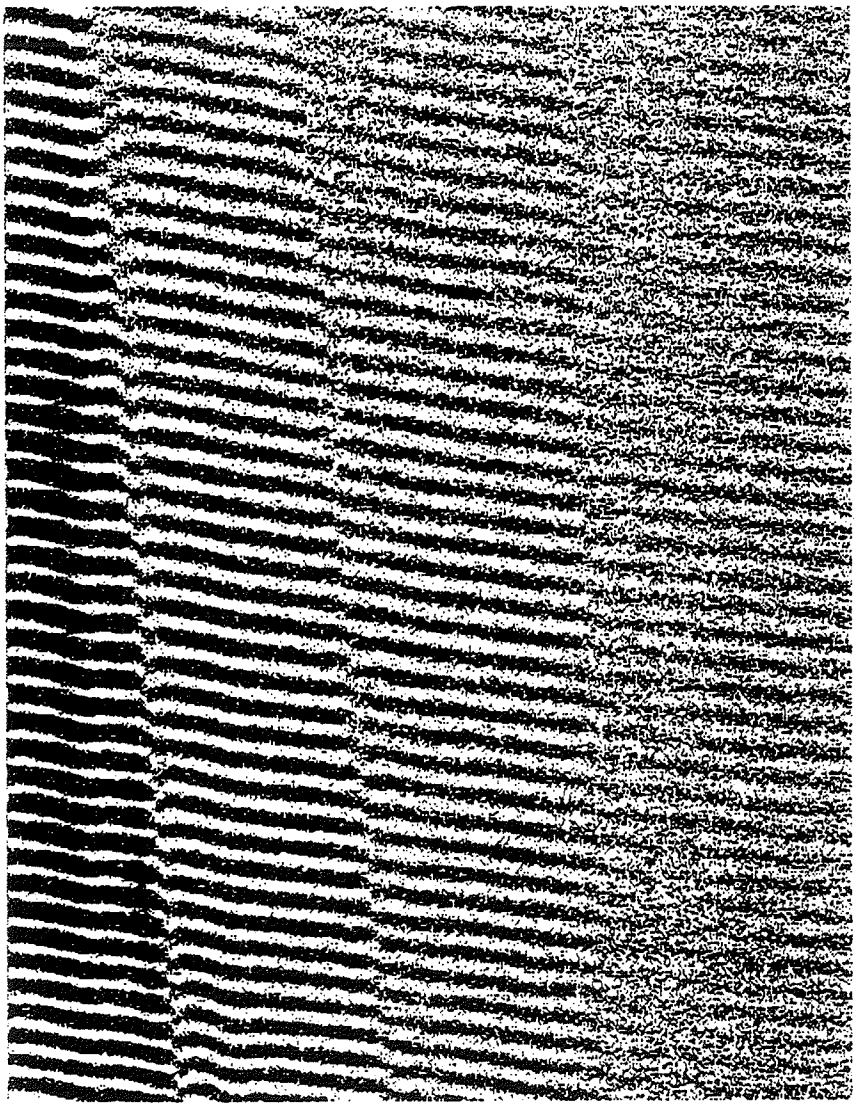


Figure 4-2C: Histogram equalized image.

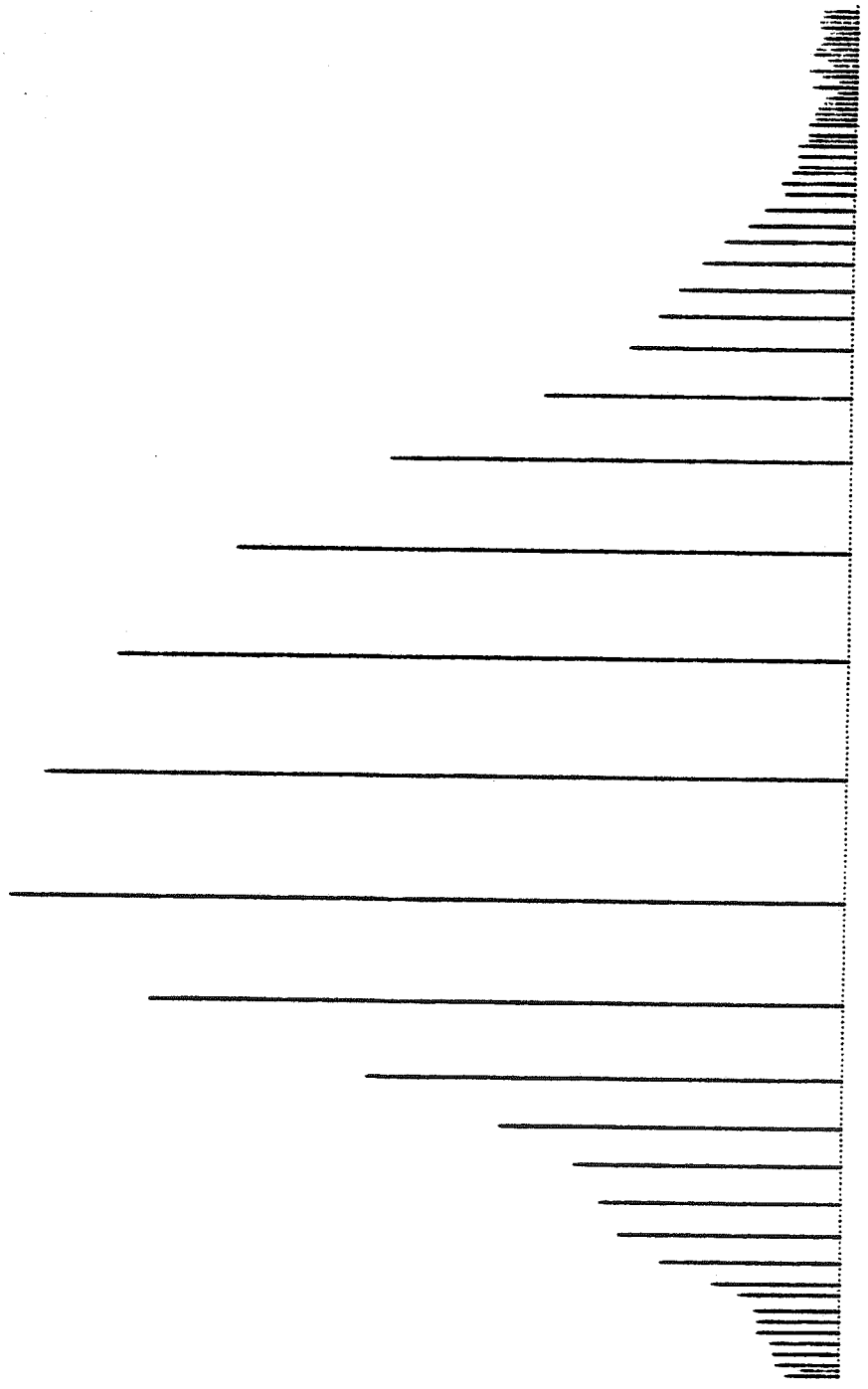


Figure 4-2D: Histogram of gray levels in histogram equalized image.

SLAM in the interference mode. The histograms accompany their corresponding image. The histogram of the original image is clustered in the middle of the gray levels where as the equalized histogram is more uniformly spread out. The quality of the histogram equalized image is clearly superior to the original image. The different levels of the plastics stand out in the equalized image but are hardly discernable in the original image.

The noise reduction due to averaging the image in figure 4-2 is shown best by the clarity of the interference lines extracted by the correlated receiver. The four interference lines from left to right identified by figure 4-3 are the result of averaging the image of figure 4-2 1, 4, 16, and 64 times respectively. The rule of thumb when averaging images is a double in the number of averages will result in a 3 dB increase in signal to noise ratio. Hence, from 1 average to 64 averages an 18 dB signal to noise ratio increase will take place. As the plastics become thicker, the signal becomes weaker and the SNR decreases, as examplified by the interference lines of figure 4-3 which become less connected as one procedes down the image.

4.4 Effects From Varing Correlated Filter Length

The correlated receiver has a variable length for the stored replica. This paragraph along with figures 4-4A, 4-4B, 4-4C, and 4-4D shall discuss the effects on the accuracy of detecting the center of the interference lines by varying this parameter. The longer the filter length the better the SNR; also as a trade-off the program will take

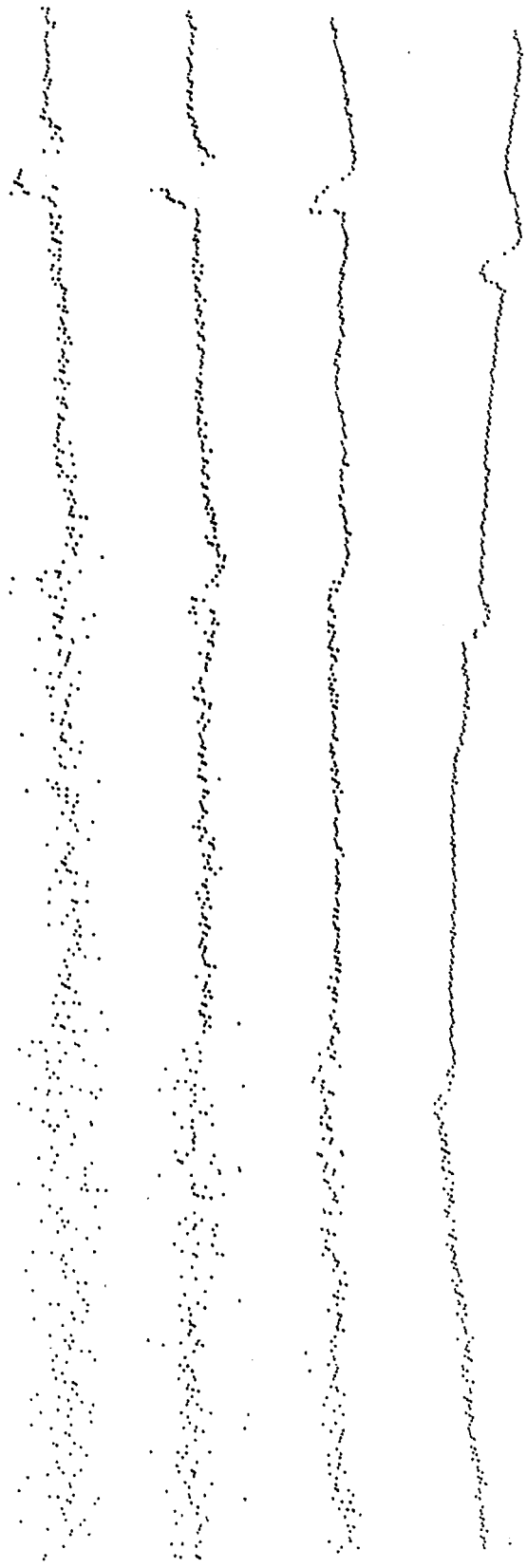


Figure 4-3: Effects of averaging on interference lines,
from left to right the average is 1, 4, 16, and 64.

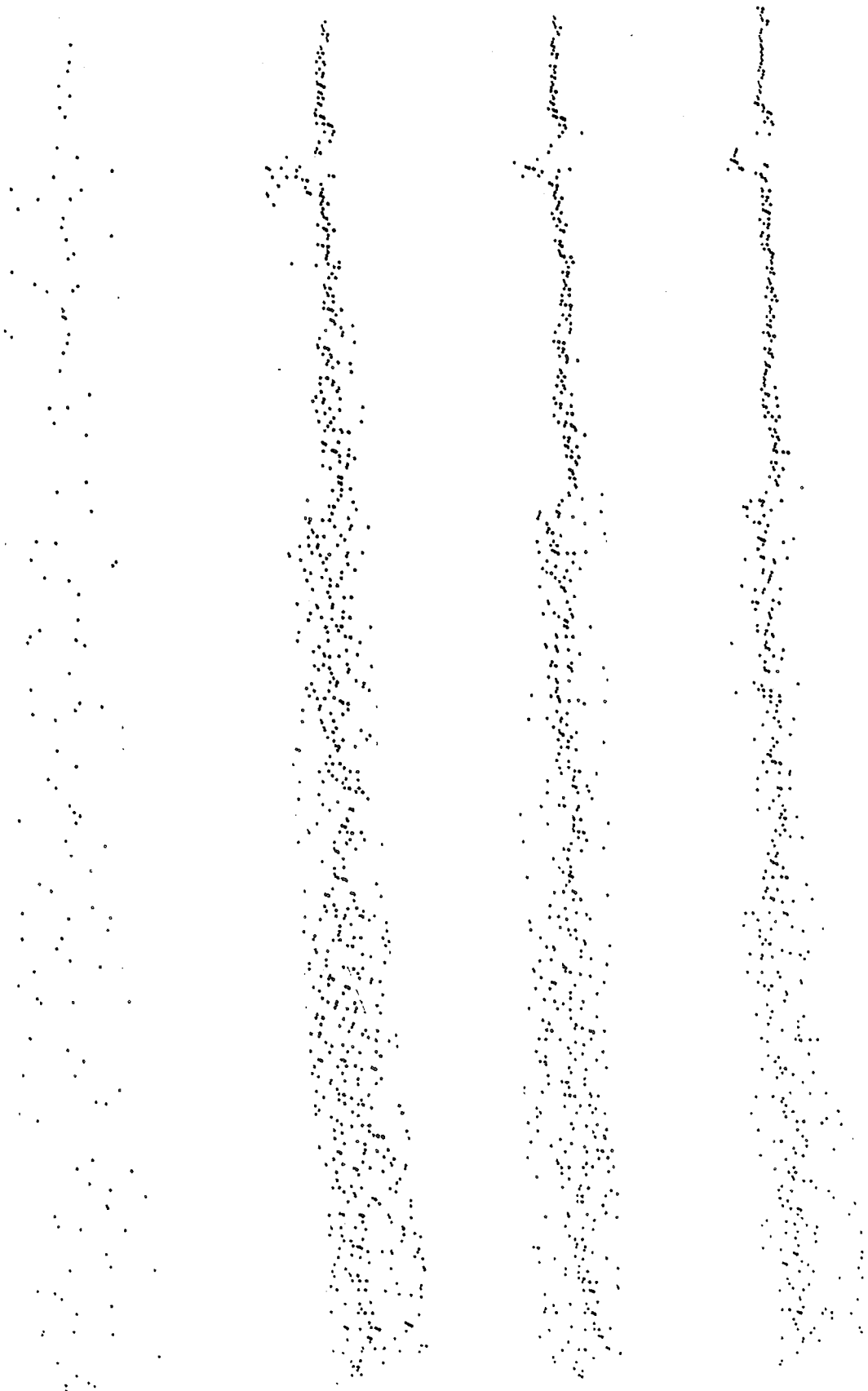


Figure 4-4A: Averaged 1 time, correlated 1, 4, 8, 16 times.

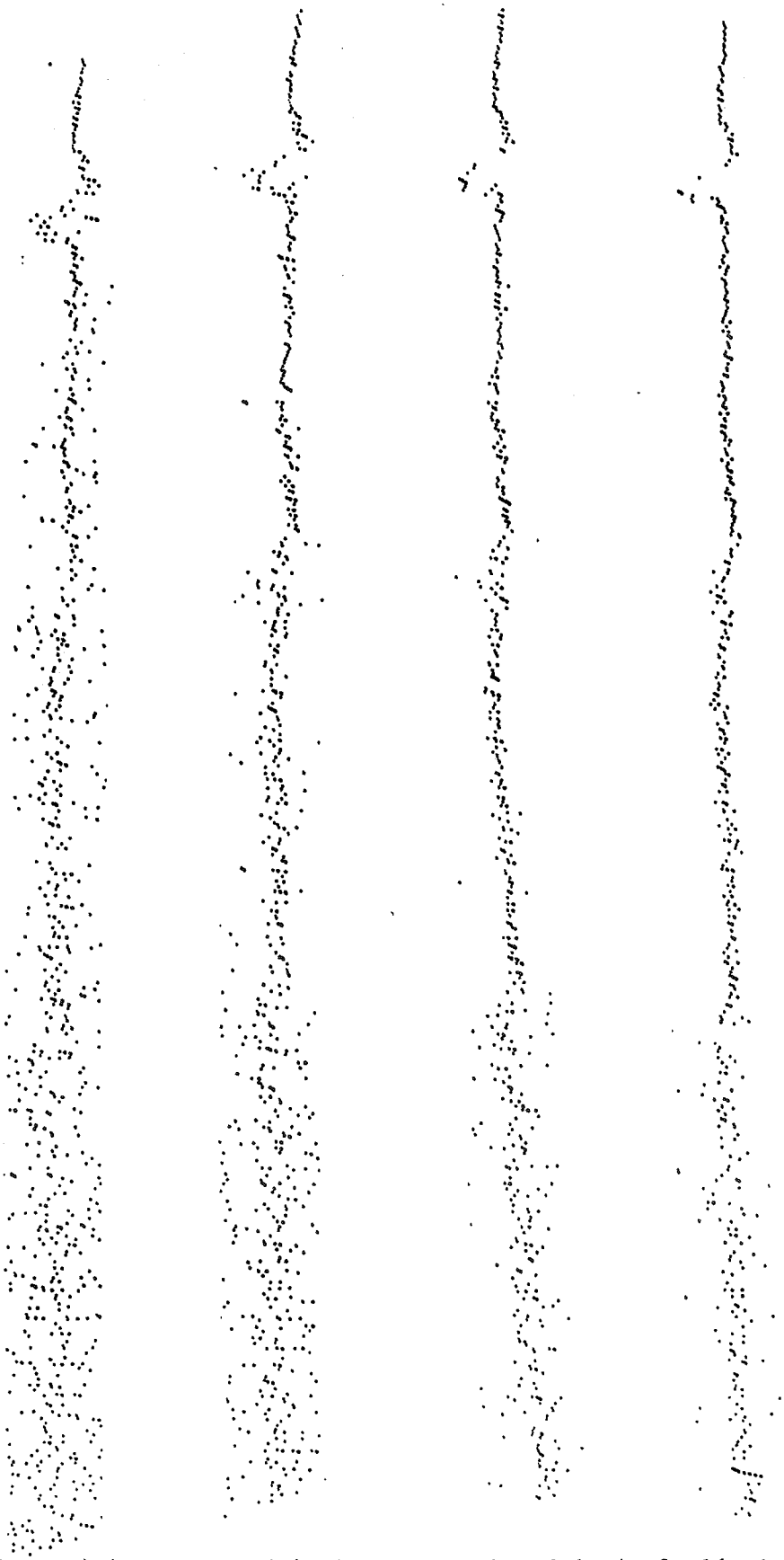


Figure 4-4B: Averaged 4 times, correlated 1, 4, 8, 16 times.

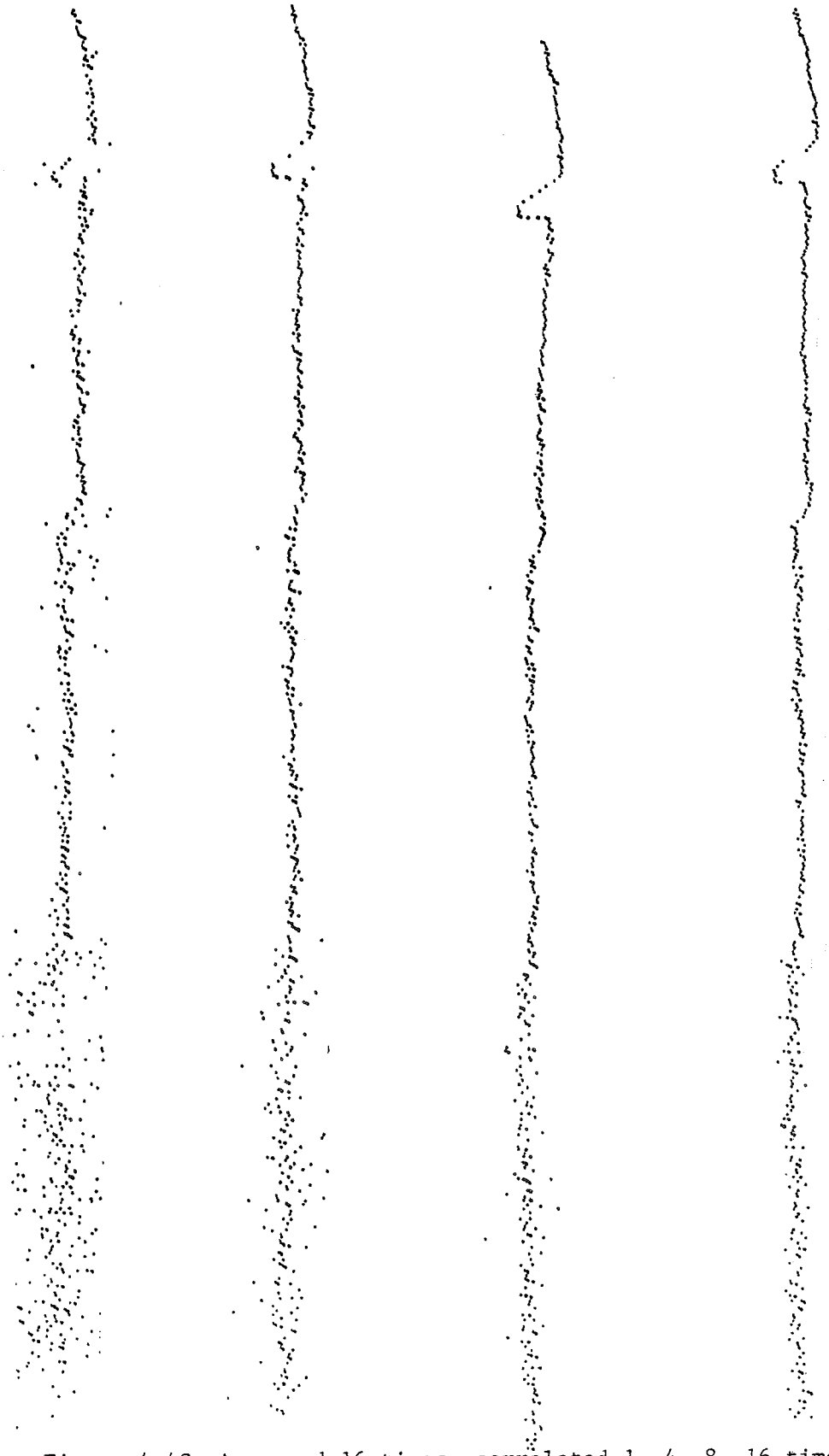


Figure 4-4C: Averaged 16 times, correlated 1, 4, 8, 16 times.

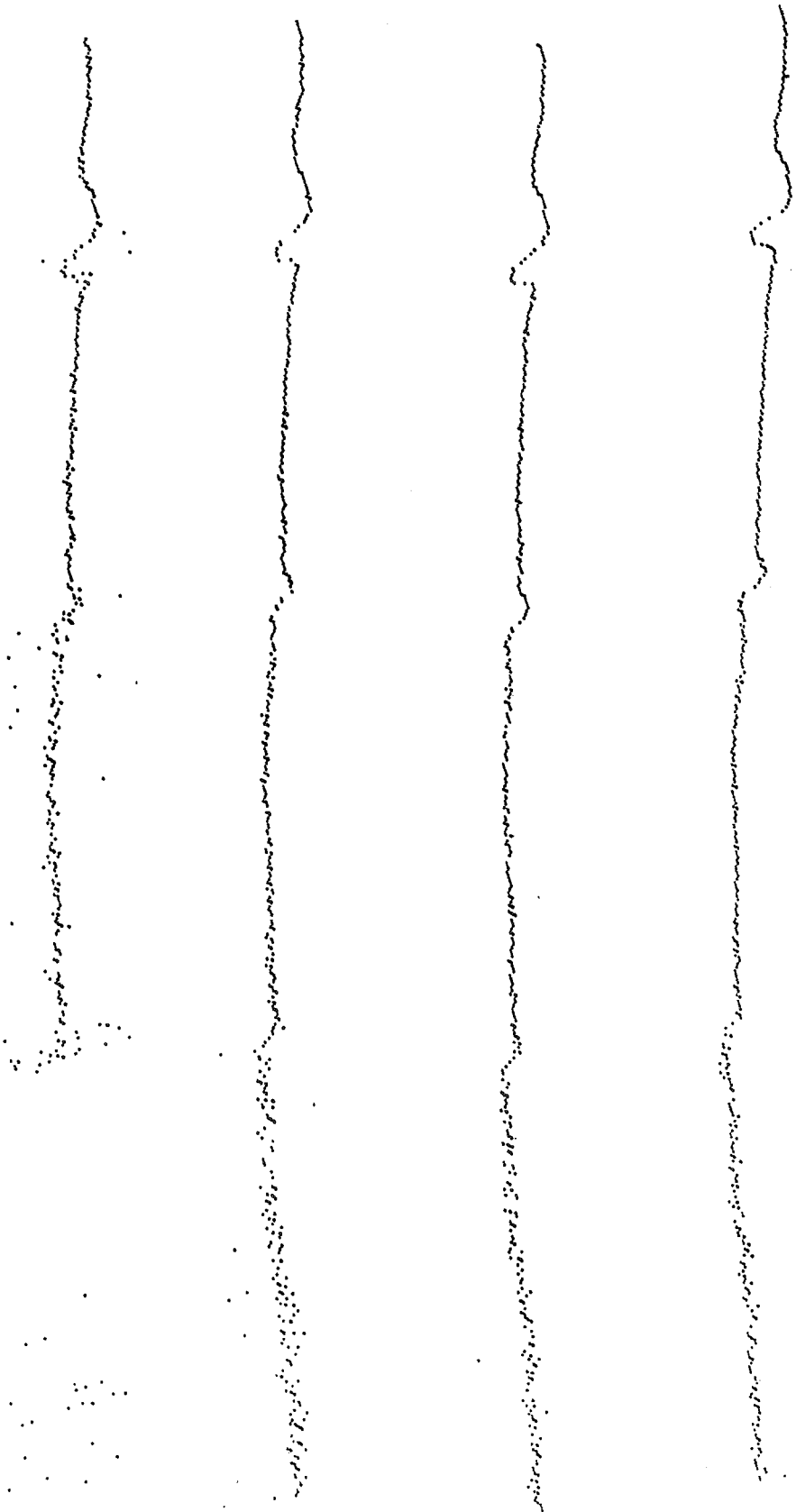


Figure 4-4D: Averaged 64 times, correlated 1, 4, 8, 16 times.

longer to output the interference patterns . In figure 4-4 there are 16 interference lines in four groups of four. In each group the image (see figure 4-2) has been averaged the same number of times using the running average method. The first group (figure 4-4A) is averaged once, the second group (figure 4-4B) is 4 times, the third group (figure 4-4C) is 16 times, and the fourth group (figure 4-4D) is 64 times. In each group the width of the filter is the variable; from left to right the length of the filter is 1, 4, 8, and 16. The results strongly suggest that the length of the filter and averaging improve the ability to detect the center of the interference lines.

CHAPTER 5

IMPROVEMENTS

The drift from the SLAM must by eliminated by the manufacturer. The drift still gives some randomness to the interference lines when digitizing at 30 MHz, although the drift is not a problem for other research not involving the DAS. Elimination of this would help the image quality.

The 30 MHz crystal clock should incorporate a phase locked loop circiut. Starting the crystal oscillating has still some randomness with it. The phase locked loop could eliminate this.

More buffer memory would allow more lines to be captured at once, and make hardware averaging cost effective (instead of using software for averaging). This would drastically cut down on the time waiting for an image to be digitized. For example, the best memory size would be 1.5 megabytes which could hold the image averaged up to 256 times; this means 64 averages would require 64/30 seconds, whereas, currently it takes 18 minutes (a 482 fold increase in speed). The expanded buffer memory would also allow for the digitization and examination of moving structures in the image.

REFERENCES

S. A. Goss, W. D. O'Brien, Jr. (1979). "Direct ultrasound velocity measurements of mammalian collagen threads," J. Acoust. Soc. Am. 65, 507-511.

W. D. O'Brien, Jr., J. Olerud, K. K. Shung, J. M. Reid (1981). "Quantitative acoustical assessment of wound maturation with acoustic microscopy," J. Acoust. Soc. Am. 69, 575-579.

H. L. Van Trees, Detection, Estimation, and Modulation Theory Part 1, Wiley, New York, 1968, page 277.

Appendix A

Wiring Map

PIN	NAME	CONNECTION
1	A0	A2-1
2	A1	A2-2
3	A2	A2-3
4	A3	A2-4
5	A4	A2-5
6	A5	A2-6
7	A6	A2-7
8	A7	A2-8
9	A8	A2-9
10	A9	A2-10
11	R/W	A2 22
12	A/D0	A4-P
13	A/D1	A4-Z
14	A/D2	A4-Y
15	A/D3	A4-X
16	A/D4	A4-V
17	A/D5	A4-U
18	A/D6	A4-T
19	A/D7	A4-S
20	c(SELECT CK/2)	A2-20
21	SELECT CK/2	A2-20
22		
A	+5 VOLTS	POWER SUPPLY
B	D0	A6-B
C	D1	A6-C
D	D2	A6-D
E	D3	A6-E
F	D4	A6-F
H	D5	A6-H
J	D6	A6-J
K	D7	A6-K
L	D8	A6-L
M	D9	A6-M
N	D10	A6-N
P	D11	A6-P
R	D12	A6-R
S	D13	A6-S
T	D14	A6-T
U	D15	A6-U
V		
W		
X		
Y		
Z	GND	POWER SUPPLY

Figure A-1: Wiring map for circiut board A1.

PIN	NAME	CONNECTION
1	A0	A1-1
2	A1	A1-2
3	A2	A1-3
4	A3	A1-4
5	A4	A1-5
6	A5	A1-6
7	A6	A1-7
8	A7	A1-8
9	A8	A1-9
10	A9	A1-10
11	DATA0	A3-B
12	DATA1	A3-C
13	DATA2	A3-D
14	DATA3	A3-E
15	DATA4	A3-F
16	DATA5	A3-H
17	DATA6	A3-J
18	DATA7	A3-K
19	SELECT CK	A3-N
20	c(SELECT CK/2)	A1-20
21	SELECT CK/2	A1-21
22	R/c(W)	A1-11,A6-2
A	+5 VOLTS	POWER SUPPLY
B	STORE 4	A3-9
C	STORE 1	A3-12
D	STORE 0	A3-13
E	STOP	A3-R
F		
H	c(INPUT DATA RECEIVED)	A6-X
J		
K		
L		
M		
N		
P	RESET	A3-P
R		
S		
T		
U		
V		
W	ALERT	A6-W
X		
Y		
Z	GND	POWER SUPPLY

Figure A-2: Wiring map for circiut board A2.

PIN	NAME	CONNECTION
1	110 KHZ	A5-S
2	LINE DETECT STOP	A5-T
3	BIT 6	DIO OUTPUT CONNECTION...
4	BIT 5	
5	BIT 4	
6	STORE 7	NOT CONNECTED...
7	STORE 6	
8	STORE 5	
9	STORE 4	A2-B
10	STORE 3	NOT CONNECTED...
11	STORE 2	
12	STORE 1	A2-C
13	STORE 0	A2-D
14	BIT 15	DIO OUTPUT CONNECTION...
15	BIT 14	
16	BIT 13	
17	BIT 12	
18	BIT 11	
19	BIT 10	
20	BIT 9	
21	BIT 8	
22	AUDIO/c(VIDEO)	A6-1
A	+5 VOLTS	POWER SUPPLY
B	DATA0	A2-11
C	DATA1	A2-12
D	DATA2	A2-13
E	DATA3	A2-14
F	DATA4	A2-15
H	DATA5	A2-16
J	DATA6	A2-17
K	DATA7	A2-18
L		
M	30 MHZ	A5-H
N	SELECT CK	A2-19,A4-N
P	RESET	A2-P
R	STOP	A2-E,A5-L
S	BIT 3	DIO OUTPUT CONNECTOR
T	BIT 2	
U	BIT 1	
V	BIT 0	
W	DATA OUTPUT AVAILABLE	
X		
Y	c(INPUT ENABLE)	A6-Y
Z	GND	POWER SUPPLY

Figure A-3: Wiring map for circiut board A3.

PIN	NAME	CONNECTION
1		
2	ANALOG GND	POWER SUPPLY
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22	GND	POWER SUPPLY
A		
B		
C		
D		
E	-15 VOLTS	POWER SUPPLY
F		
H	+15 VOLTS	POWER SUPPLY
J		
K		
L		
M	+5 VOLTS	POWER SUPPLY
N	SELECT CK	A3-N
P	A/D0	A1-12
R	+5 VOLTS	
S	A/D7	A1-19
T	A/D6	A1-18
U	A/D5	A1-17
V	A/D4	A1-16
W	+5 VOLTS	POWER SUPPLY
X	A/D3	A1-15
Y	A/D2	A1-14
Z	A/D1	A1-13

Figure A-4: Wiring map for circiut board A4.

PIN	NAME	CONNECTION
1	NOT CONNECTED...	
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
A	+15 VOLTS	POWER SUPPLY
B	+5 VOLTS	POWER SUPPLY
C		
D		
E		
F		
H	30 MHZ	A3-M
J		
K		
L	STOP	A3-R
M		
N		
P	GND	
R		
S	110 KHZ	A3-1
T	LINE DETECT STOP	A3-2
U		
V		
W		
X	-15 VOLTS	
Y		
Z	ANALOG GND	POWER SUPPLY

Figure A-5: Wiring map for circiut board A5.