5-1-1981

A digital image processing system for slow scan television

James Schueckler

A Digital Image Processing System for Slow Scan Television

by

James R. Schueckler

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
in
Electrical Engineering

Approved by: Prof. Roger Rol Morta
(Thesis Advisor)

Edward R. Sakm

Name Illegible

Henry E. Rhody
(Department Head)

Department of Electrical Engineering
College of Engineering
Rochester Institute of Technology
Rochester, New York
May, 1981

I, _____ hereby grant permission to the Wallace Memorial Library, of R.I.T., to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Dedicated to

God

and

my wonderful family

Acknowledgments

I wish to thank my advisor, Dr. Roger Morton, for his technical and editorial help, and Dr. George Steber, WB9LVI, for his articles which first inspired my interest in digital image processing. Many friends have provided ideas and consultation, especially Mike Clayton, Ed Granger, Fred Metildi, Don Oinen, Phil Pearce, Steve Sasson, and Chuck Wise. Many thanks to Ruth Sisson for her patient typing and to my family for their patience and support.

Abstract

This paper describes a low cost, but powerful, digital image processing system. Although general purpose, it was designed for experimentation with amateur radio slow scan television (SSTV). The project involved conceptualizing, designing, building, programming, and operating the complete integrated system.

The image format is the SSTV standard: 128 picture elements (pixels) per line, 128 lines, 4 bits per pixel. The IEEE Standard 696 (S-100) microcomputer bus was used for system interconnect to provide flexibility and expandability. Several microcomputer boards were purchased, and two image interface boards were designed and built by the author.

One image interface board provides for transmitting and receiving images at SSTV rates. The other board can accept a single frame image from a standard TV camera, store it in memory, and continuously output an image from memory to a video monitor.

Any image in memory can be analyzed or modified, pixel by pixel, by programs running on the system's Z80 microprocessor. Simple keyboard commands can initiate many digital image processing programs that were written for the system, including the powerful point processing and two dimensional convolution functions.

TABLE OF CONTENTS

List of Figures

List of Symbols

AM - amplitude modulation

ASCII - American Standard Code for Information Interchange

CCTV - closed circuit television

CPU - central processing unit

CRT - cathode ray tube

DMA - direct memory access

FM - frequency modulation

ham - a proper word for "amateur radio operator"

Hz - Hertz, one cycle per second

I/O - input/output

KHz - kilohertz

MOS - metal oxide semiconductor

MHz - megahertz

ms - millisecond

ns - nanosecond

pixel - picture element

PROM - programmable read only memory

RAM - random access memory

RF - radio frequency

SSB - single sideband

SSTV - slow scan television

TV - television

us - microsecond

CHAPTER 1. INTRODUCTION

The purpose of this study is to combine several fields of endeavor into a working electronic system. This chapter will briefly define those fields and how they are combined together.

Section 1A. Digital Image Processing

Digital image processing can be defined as the use of numerical methods to analyze, modify, or generate images. An image must first be sampled in two dimensions and quantized. Optical reflectance or transmittance values from the image are converted into a two dimensioned array of numbers to represent the image. Each element of the array specifies the "brightness" of a specific spot in the image. This array can be analyzed or modified by computer programs for almost any desired purpose. Of specific interest are those programs based on digital signal processing theory which perform filtering of an image in the spatial frequency domain.

The modified image data can then be sent to a CRT or other image output device for direct viewing or to create a hard copy. Original or processed images can also be stored digitally as any other computer data, for later processing, comparison, or display.

Section 1B. Amateur Radio Slow Scan Television

FCC regulation 97.65(d)[1] allows amateur radio slow scan television (SSTV) to use AM or FM television with a maximum bandwidth of 3KHz. Hams have been almost completely free to experiment with modulation modes, sweep timing, and other variables.

The present unregulated, but accepted, SSTV signal standard provides compatibility between the older analog, and newer digital, image storage methods. The standard is: FM, a 1:1 aspect ratio, 128 horizontal scan lines at 1/15 second per line, non interlaced: providing a frame time of 8.5 seconds. Additional standards for digital systems are 128 picture elements (pixels) per line, 4 bits per pixel, providing 16 gray levels. This voice-bandwidth SSTV signal may be sent over voice grade telephone lines, recorded on inexpensive audio cassette tape recorders, and especially, used on the high frequency amateur radio bands to provide world wide image communication. Many hams have exchanged images with others in more than 100 countries.

Section 1C. System Integration

This study combines digital image processing and slow scan TV into a single electronic system. A low cost, S-100 bus computer is the foundation. Interface boards were designed to input images from a standard TV camera and output to a standard video monitor as well as the SSTV video input and output. Several digital image processing programs were implemented to make a complete integrated system.

Design of this system was separated into these tasks:

1. Choice of the system specifications and resultant architecture of the Digital Image Processing System for SSTV.
2. Design of the image interface boards; one for standard (fast) video, one for SSTV.
3. Implementation of digital image processing algorithms as programs that are simple to use.
4. Preparation of the documentation.

CHAPTER 2. HISTORICAL REVIEW

Three fields of endeavor are combined in this study. This chapter presents a brief history of those fields.

Section 2A. Brief History of Digital Image Processing

Digital image processing began as digital computers were pressed into scientific applications. One of the first papers in this field was by Kirsch,[2] et al., in 1957. In 1959, California Institute of Technology's Jet Propulsion Laboratory began their famous space image research, and in 1962, the University of Southern California undertook digital image processing on a more general theme.

Space exploration, military applications, industrial robotics, medical imaging, and many other fields soon provided both the needs and the funding to develop digital image processing hardware and algorithms. Many other schools and corporations developed digital image processing systems. Most of these systems were designed for medium to high image resolution (256 X 256 to 2048 X 2048) and had to be used with a mainframe computer or at least a large minicomputer. The systems were, therefore, quite expensive, with costs starting around $40,000.

One of the first efforts to make a low cost digital image processing system was published by M. Nagao,[3] et al., in 1974, describing a minicomputer based system with a resolution of 320 x 250 x 6.

Today, digital image processing systems can be purchased from more than 20 vendors with a wide variety of image resolution and processing capability options. Many have special purpose hardware to perform convolutions in 1/30 second. Prices for these systems range from $5,000 to $500,000.

Section 2B. History of SSTV

In 1958, Copthorne McDonald[4] sought a way to transmit a television image using less than 3KHz bandwidth on amateur radio (ham) frequencies. He designed an amplitude modulated (AM) system that required 8 seconds to transmit one frame of an image at reduced spatial resolution. He started a new mode of amateur communication, slow scan television (SSTV). In two published articles, he described a flying spot slide scanner and a receiving monitor that used a radar CRT with a very long persistence phosphor.

Soon many other hams duplicated his system and designed additional components. Within a few years, FM was chosen as the standard mode, a vidicon slow scan camera was designed, and two companies were selling completely assembled equipment. Many hundreds of hams were exchanging SSTV images all over the world, in slide-show fashion.

One serious drawback was the frame time of 8 seconds. By the time the bottom of an image was received, the top portion of the image would have faded almost completely away, even if viewed in a darkened room. The convenience of a bright, stationary image was needed.

In 1973, Robot Research, Incorporated of San Diego introduced its model 300 scan converter. It used a "lithocon" image storage tube which can be described as a combined storage CRT and vidicon. A charge image can be written onto the silicon target at fast or slow rates, and then repeatedly scanned out at either rate. Such an image can be

stored for several minutes with little detectable degradation, producing a bright, stationary image on a standard television monitor.

Two events in 1975 started digital SSTV. Dr. George Steber[5] published a series of articles describing a scan converter that stored an image digitally as a 128x128x4 array in 64 shift register chips. Almost at the same time, Robot Research introduced their model 400 scan converter which used 16 dynamic memory chips (4096x1 each), using the same image format. Since then, several hams have built Dr. Steber's scan converter and more than 10,000 hams have purchased the Robot 400 unit for $695.

These scan converters can store a digital image, but have no provisions to interface to a computer. Since this study was started, Clayton Abrams[6] published an article on a scan converter to computer interface. His system could not "grab" from a fast video source and could not simultaneously display and process an image. John Vandenburg of Mount Hope, Ontario, Canada, has designed a similar, unpublished system.

Section 2C. History of Hobby Computing

Small, low cost computers became a reality in 1972 with the announcement of the Intel 8008, an 8 bit microprocessor. In 1974 the Intel 8080 and Motorola 6800 microprocessors, as well as a variety of low cost semiconductor memory devices, allowed hundreds of persons to own their own "home-brew" computers. Within the next few years, many companies were formed to sell hardware, software, or systems to hobbyists. Today it is possible to purchase a complete system capable of running Fortran or BASIC languages for less than $1,000.00.

This study combines amateur radio slow scan television, digital image processing, and personal computing into a powerful, but low cost, system.

CHAPTER 3. SYSTEM SPECIFICATIONS

The most important factor in the design of any electronic system is the careful selection of system specifications. Based on the intended purpose, the present state of the art, and the cost of components on the "hobby" market, the following criteria have been chosen as goals and specifications of the Digital Image Processing system for SSTV. They are listed below, and then covered in more detail separately.

1. SSTV image format (128 x 128 x 4)
2. Video output timing to EIA specification RS-170[7]
3. RS-170 video input, single field
4. Standard, but programmable SSTV interface
5. Simultaneous display while processing
6. Simultaneous display while SSTV input/output
7. S-100 microcomputer bus
8. Z-80 microprocessor
9. Minimum bus time for display
10. Interactive software
11. Hardware and software expansion capabilities

1. SSTV IMAGE FORMAT (128 x 128 x 4)

The format of the stored image is 128 pixels per line, 128 lines, with each pixel represented by a 4 bit number, allowing 16 possible levels: black, white, and 14 gray shades. While this may seem to be rather poor resolution, it is the standard for ham radio SSTV. It would serve no purpose to have higher resolution than other SSTV equipment the system will be exchanging images with. The aspect ratio of standard SSTV is 1:1, which produces a square image. When an SSTV image is displayed on a CRT with a 3:4 aspect ratio, there must be borders on the left and right sides of the image to prevent geometric distortion.

2. VIDEO OUTPUT TIMING TO EIA SPECIFICATION RS-170

RS-170 specifies image format and timing used by standard monochrome (black and white) television, broadcast or closed circuit, in the U.S. A low cost video monitor, or a slightly modified TV set, could be used as an image output device. The video output signal could also go to a video tape recorder or be transmitted via fast scan amateur television on frequencies above 432MHz.

3. RS-170 VIDEO INPUT, SINGLE FIELD

The RS-170 input allows the use of an inexpensive closed circuit TV (CCTV) camera. The single field grab capability provides for minimizing motion blur and allows the capture of a "snap shot" of a moving image.

4. STANDARD, BUT PROGRAMMABLE SSTV INTERFACE

A programmable SSTV interface allows both the use of the present SSTV standard and experimentation with SSTV modulation techniques in a quantitative fashion, as more thoroughly explained later.

5. SIMULTANEOUS DISPLAY WHILE PROCESSING

Being able to observe images while they are being processed is one of the most valuable features of an interactive digital image processing system. Watching an algorithm as it processes various portions of an image is a valuable learning experience. Debugging is aided by being able to observe a new program as it runs (or doesn't).

6. SIMULTANEOUS DISPLAY WHILE SSTV INPUT/OUTPUT

Simultaneous display while receiving SSTV allows fine adjustments to be made for contrast and brightness, or adjustments made to the receiver while the image is being received.

7. S-100 MICROCOMPUTER BUS

More than 100 companies make boards for the IEEE S-100 microcomputer bus, allowing a wide range of low cost options, as bare boards, kits, or completely assembled.

8. Z-80 MICROPROCESSOR

The Zilog Z-80 appears to be the best 8 bit MOS microprocessor for digital image processing. Its block memory, block I/O, and half-byte instructions make it ideal. Many of these special instructions were used to make the programs faster.

9. MINIMUM BUS TIME FOR DISPLAY

The image display uses direct memory access (DMA), which requires use of the system bus, but must leave sufficient bus time for the CPU to do processing, to allow fast program execution.

10. INTERACTIVE SOFTWARE

This allows many simple image processing functions to be executed at the press of a few keys, with immediately observable results.

11. HARDWARE AND SOFTWARE EXPANSION CAPABILITIES

To be truly flexible as a research tool, the system hardware and software must be easily expanded. The bus oriented architecture and structured programming make this possible.

The next chapters describe how these specifications were met. First the architecture of the complete integrated system is presented, then the design of the two image interface boards. Finally, the power and flexibility of the system is revealed in the last chapter, a description of some of the image processing programs that have been implemented.

CHAPTER 4. SYSTEM ARCHITECTURE

A block diagram of the Digital Image Processing System for SSTV is shown in Figure 1. It is built around the Von Neumann architecture of the IEEE standard 696 (S-100) bus. Both general purpose computer boards and special purpose image interface boards are plugged into the bus. Image input and output devices are interfaced to the bus via the two image interface boards.

It is important to note that with the exception of the two image interface boards, the system is simply a general purpose S-100 computer. The two image interface boards would convert any S-100 computer into a Digital Image Processing System for SSTV. The system can even be used without the SSTV Interface Board as a powerful TV input/output digital image processing system.

The TV Camera could be any standard CCTV camera. The one used in this system is made by "Roberts," is all solid state (except the vidicon) and cost the author $60, used.

Either a Video Monitor or standard TV set modified for direct video input can be used for the image display. The author's monitor was made by Electrohome, and cost $35, used. The Video Select Switch allows manual selection of image or alphanumeric display.

The ASCII Keyboard allows operator entries for execution and debugging of programs.

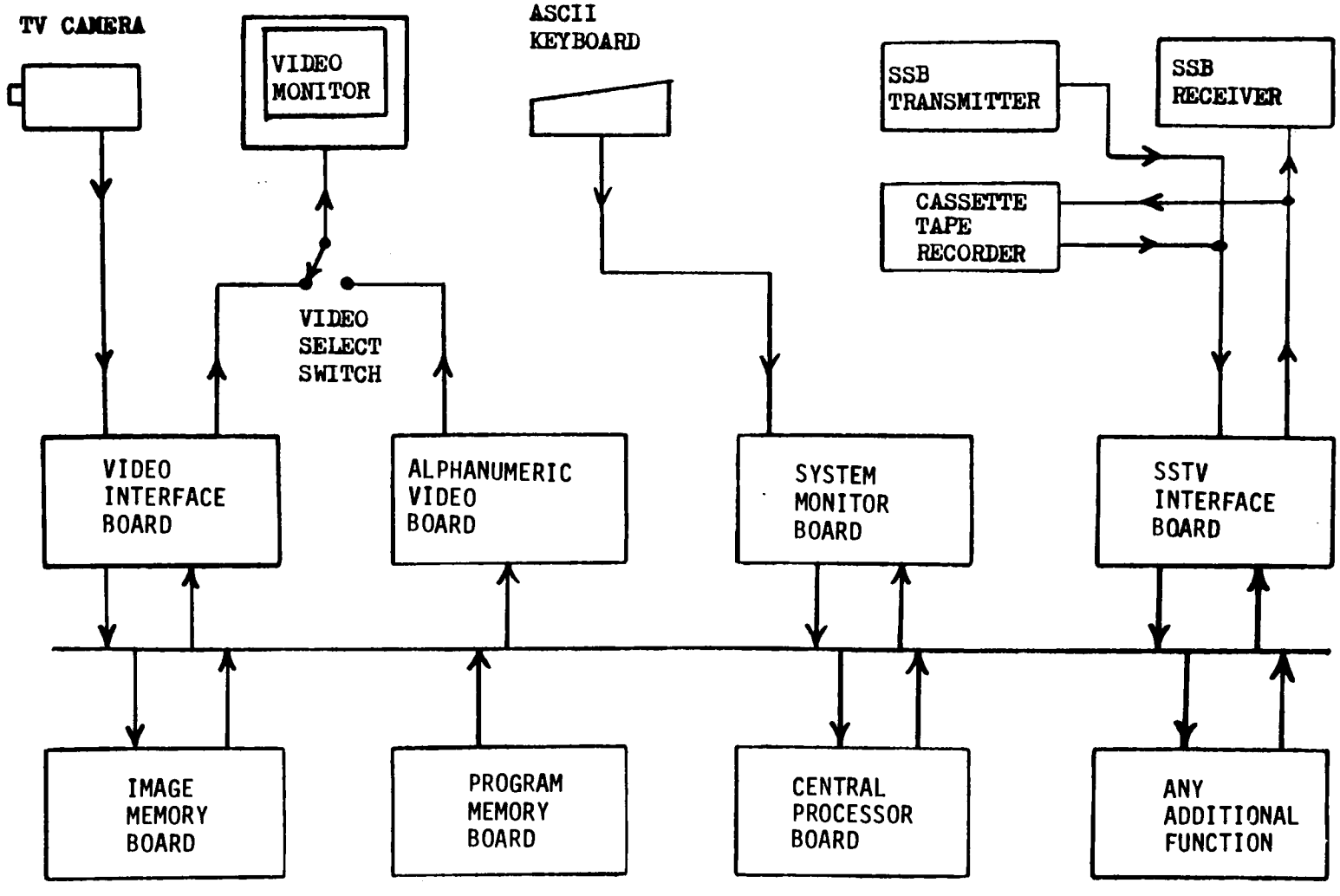

FIGURE 1 - SYSTEM BLOCK DIAGRAM

FIGURE 1B - Photograph of Digital Image Processing system for SSTV, showing the video monitor, keyboard, cassette tape recorder, and S-100 computer including the two image interface boards.

Any amateur radio SSB receiver and SSB transmitter can be used with the SSTV Digital Image Processing System to receive and transmit images in SSTV format. The SSTV Interface Board plugs into the speaker jack of the receiver and the microphone jack of the transmitter.

The Audio Cassette Tape Recorder is used to store images using exactly the same tones as SSTV. Although an inexpensive recorder can be used, wow, flutter, and other audio distortions found in low quality recorders can degrade tape recorded images.

The Video Interface Board was the most challenging and critical portion of the system. It interfaces the video input and output signals to the image memory using direct memory access (DMA), which will be explained in detail in the next chapter.

The Alphanumeric Video Generator Board is a Solid State Music model VB-1B. It provides computer to operator communications for programming and debugging as a CRT terminal or teletype would do.

The System Monitor Board was designed by the author, but has no special imaging functions. It provides for keyboard input, 1024 bytes of RAM, single step hardware, and an unused serial port.

The SSTV Interface Board was also designed by the author. It uses programmable counters and a program running in the CPU to either convert audio tones to the appropriate gray level for SSTV receive, or, alternately, convert the

gray level of each pixel to the appropriate audio tone for SSTV transmit. Chapter 6 gives a detailed description of the board.

The Image Memory Board is a general purpose 8192 x 8 bit static RAM memory board, made by Ithaca Intersystems. It is no different from any other memory on the bus, except that it is used to store image data.

The Program Memory Board, made by Ithaca Intersystems, has sockets for 24 PROM's (Programmable Read Only Memory). Image processing programs, as well as SSTV and character generator look up tables are stored semi-permanently here. To provide a high quality, perfectly repeatable image for testing and demonstrations, four PROM chips contain the Boy-Dog picture shown later in this paper. In less than 1/2 second the Boy-Dog image data can be moved into image memory by a special program.

The Central Processor Board, also made by Ithaca Intersystems, uses a Zilog Z-80 clocked at 2MHz. The board also has a 2K byte assembly language monitor program on PROM.

Of the 22 slots available on the system bus, only 7 are used. Any additional S-100 compatible board could be plugged in, such as: another CPU board, more memory, a floppy, hard, or Winchester disk interface, a floating point processor, or input/output for process control (to name a few).

The separate electronic interfaces or computer boards were described as subsystems of the complete integrated Digital Image Processing System for SSTV. The two image interface boards are described in the next chapters.

CHAPTER 5. VIDEO INTERFACE BOARD

SECTION 5A. BOARD BLOCK DIAGRAM

The Video Interface Board provides direct memory access (DMA) interface between memory on the S-100 bus and RS-170 composite video input and output. It can convert a single field of the video input signal into 16,384 pixel samples of 4 bits each, storing the pixel values in memory. It can also repeatedly read the pixel values from memory to create a continuous video output signal to be fed to a video monitor for viewing.

DMA was used because there was not enough room for all the memory chips on the board, higher resolution could be attained by adding more memory boards to the system (and timing changes), and other DMA devices such as disk drivers could access the same memory. In order to generate a continuous video output signal, digital pixel data must be converted to analog voltage levels. This requires the transfer of digital image data from the Image Memory, through the S-100 bus, to the Video Interface Board. Image data is generated during 70% of each scan line, and another 10% is needed to synchronize DMA control transfer. This means the CPU will have access to the bus for only 20% of the time, which would seriously degrade program execution speed.

However, displaying the low resolution SSTV image does not require 80% of the bus time. Converting the 128 line digital image to the RS-170 output signal with 262.5 lines per field requires that each line of the 128 line digital image be sent out as a pair of two identical lines. Since there are two interlaced (and identical, in this case) fields per frame, each line of the digital image becomes a wide line on the Video Monitor, actually consisting of four identical scan lines.

Because the second line of each line pair is identical to the line just before it, the Video Interface Board does not need to read the same 128 pixels out of image memory again. Instead, during the first line of each line pair, while image data is being read from the image memory for output, it is simultaneously written into the Line Buffer Memory. During the next RS-170 horizontal scan line (the second line of a line pair) the 128 digital pixel values are read out of the Line Buffer Memory to produce a video output line identical to the one before it. The Video Interface Board only uses the S-100 bus during every other line, allowing the CPU to use the bus 60% rather than 20% of the time, for a 3X improvement in program execution speed.

Using the interleaved method described above, the Video Interface Board and CPU actually "time share" the Image Memory, but at such a fast switching rate (7.875 KHz) that both functions appear to operate simultaneously. When a computer program changes a pixel value, that change will be

visible on the Video Monitor within 1/60 second. The image is also visible during SSTV transmit and SSTV receive.

A block diagram of the video interface board is shown in Figure 2.

The timing and control logic provides signals to all other portions of the board, and exchange of signals with the CPU board, as well as the video output synchronization circuits.

The pixel and line counters provide the digital address information to synchronize a particular memory address with the corresponding spot on the video monitor or camera target.

The address buffers put the address information on the system bus whenever the board is operating as the bus master.

Within the image memory, the line buffer memory, and the internal data bus, two adjacent 4-bit pixels are always stored and handled together as an 8-bit byte. This conserves memory address space and allows slower bus clocking for a given bit bandwidth, and the use of slower, less expensive, memory.

The line buffer memory is written into during even numbered lines (0,2,4,---) and read from during odd numbered lines. In the display mode, image data comes (two pixels at a time) from the external image memory, through the S-100 data bus, through the input data bus buffers to the internal data bus. The two pixels get written into the line buffer

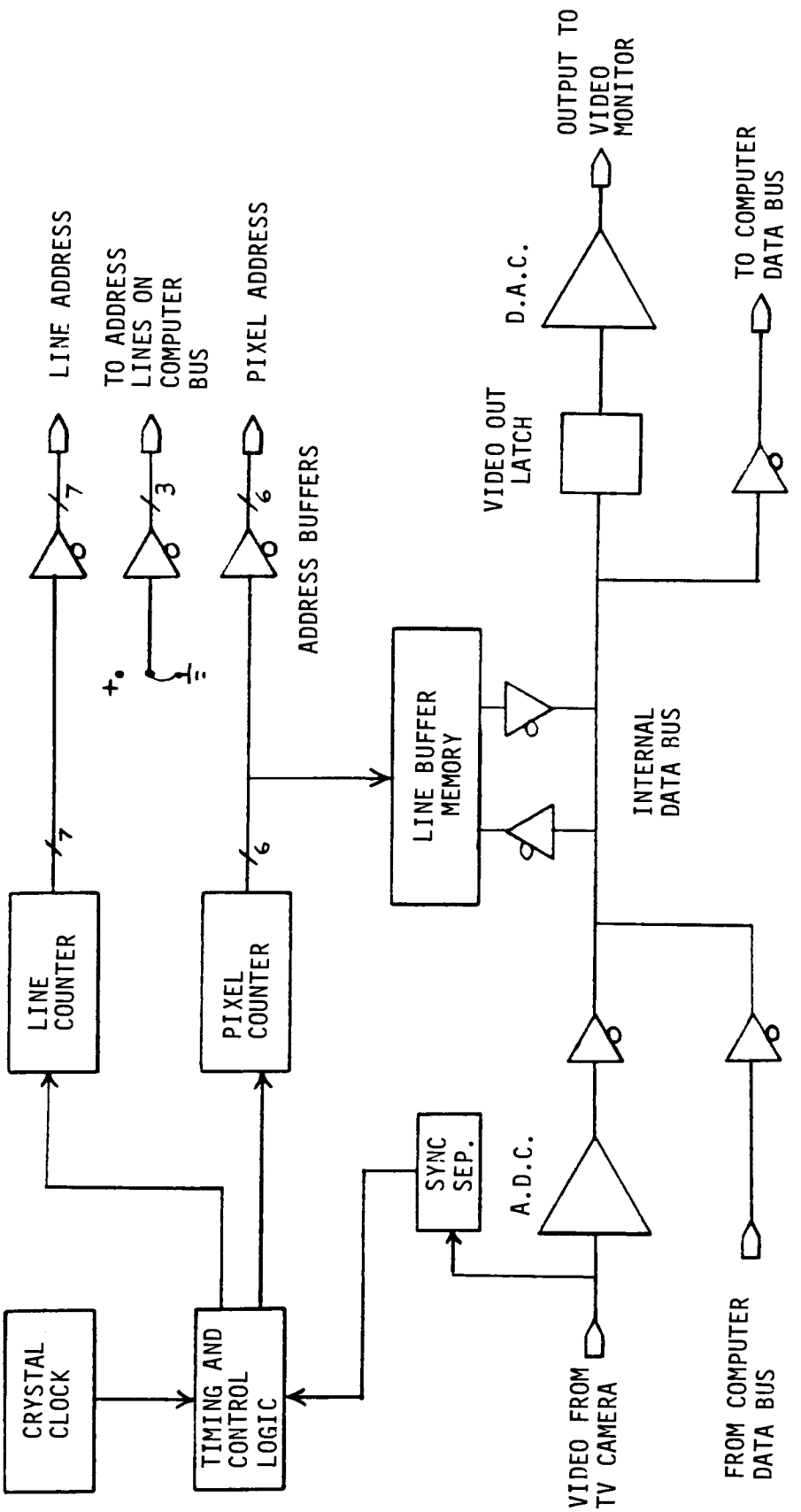

FIGURE 2 - BLOCK DIAGRAM OF VIDEO INTERFACE BOARD

memory and simultaneously are latched into the output buffers for the output digital to analog converter to create the video output signal.

When this board operates in the video input mode, the signal from the TV camera is digitized in the ADC and then simultaneously fed to the external image memory, the video DAC, and the internal line buffer memory. Every other line is stored in a manner similar to the display mode. When the image grab mode is stopped, the last image field thus acquired will be captured in the system image memory.

SECTION 5B. TIMING AND CONTROL LOGIC

Figure 3 is the schematic diagram of the timing and control logic. Throughout this paper, signals that are active in the low (0) state may be defined either with an overscore or with an asterisk post script. Thus $\overline{HS}$ is the same signal as HS*.

All timing and control signals are derived from the master clock frequency crystal at 14.31818 MHz. The fine frequency adjust potentiometer allows fine "pulling" of the crystal frequency. U3 divides the master clock frequency by seven to provide 2.04545 MHz. U2 is a complex MOS LSI chip designed to generate all of the timing signals for a TV camera. Multiplexer U5 selects whether the board will use sync signals from U2 in display mode, or those derived from the input composite video signal while in the "grab" mode. These four signals are composite sync (CS*), vertical sync (VS*), horizontal sync (HS*), and composite blanking (CB*). The two D flip flops of U23 provide that changes of the read/store* signal and DMA/DMA* signal can only occur in sync with HS*.

The master clock frequency is divided by 5 at U4 to produce the pixel clock. The resultant pixel clock frequency of 2.863636 MHz (349.2ns/pixel) was chosen to produce the proper active line time to create a 1:1 aspect ratio.

The pixel clock is divided by two by U24 to provide the pixel select (PXS) signal. On this board, pixel data is

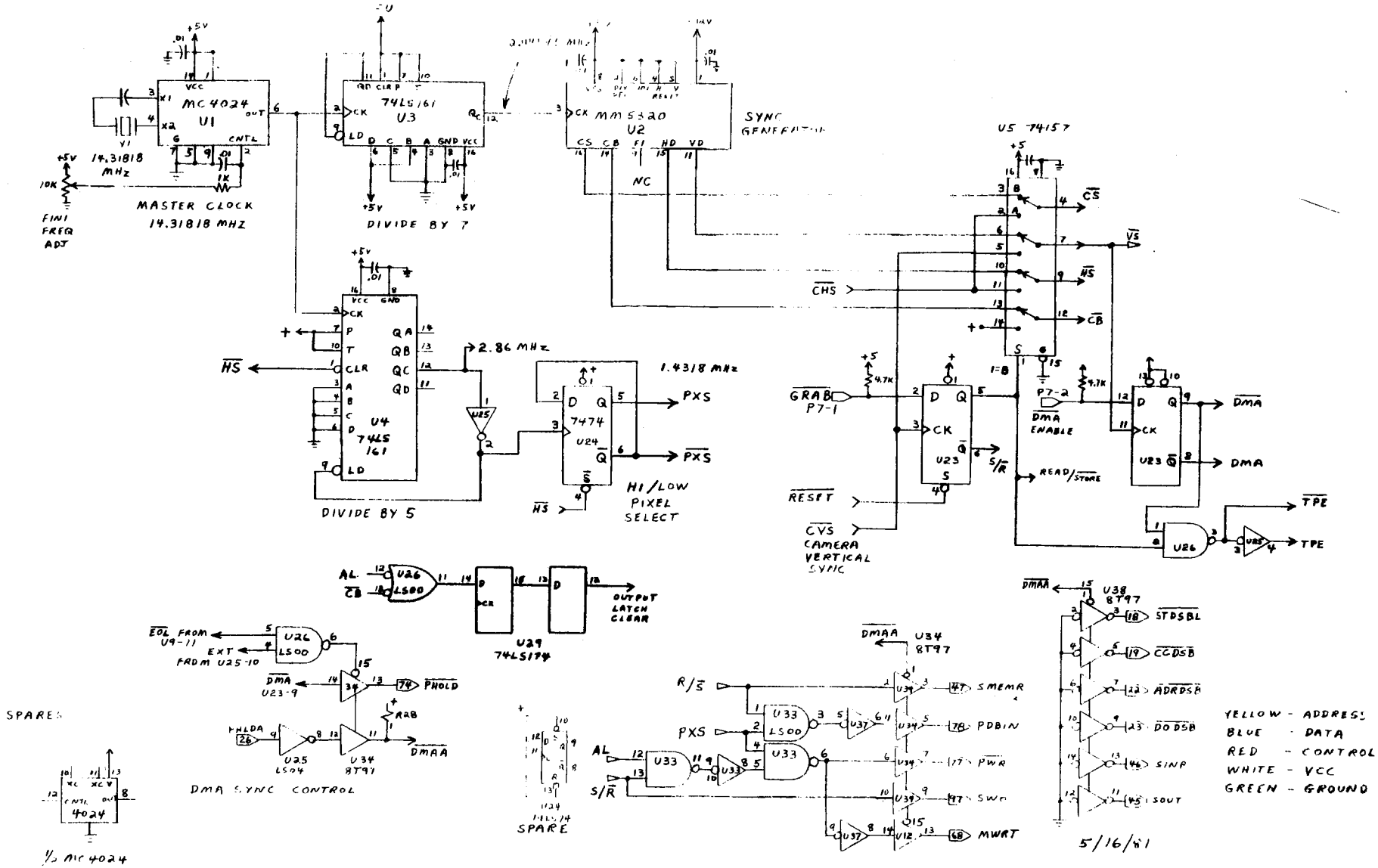

FIGURE 3- TIMING AND CONTROL LOGIC

handled as two groups of 4 bit pixel data combined into an 8-bit byte. PXS is low at the time for the lower (first) pixel to be input or output, and high for the upper (second) pixel.

The lower portion of Figure 3 shows the DMA handshake and bus control logic. At the beginning of a horizontal line in which this board must access the bus, it sends a PHOLD* signal to the CPU board, to request the bus. When the CPU board has released the bus, it returns the PHLDA signal (hold acknowledge), which is used to enable the bus drivers. To control the S-100 bus as a bus master, several internal signals (and some fixed states) are also put on the bus by U34 and U38.

Figure 4 shows the pixel counters and line counters. The pixel counters provide a count of 0 to 63 to the line buffer memory (two pixels per address) on lines BA0 through BA5 (board address). When the buffers of U6 are enabled, these six address lines also go out to the S-100 address lines. When HS* is active, the pixel counters get loaded with the binary number 183, so that 9 counts later, when the count reaches 192, the "active line" signal goes high. This centers the active image area horizontally on the display CRT. After 64 more counts (128 pixels) the END OF LINE* signal goes low as shown in Figure 5.

The line counters (Figure 4) are clocked once per line by HS*. Since every line is used twice, the lowest output bit toggles the board between the internal and external

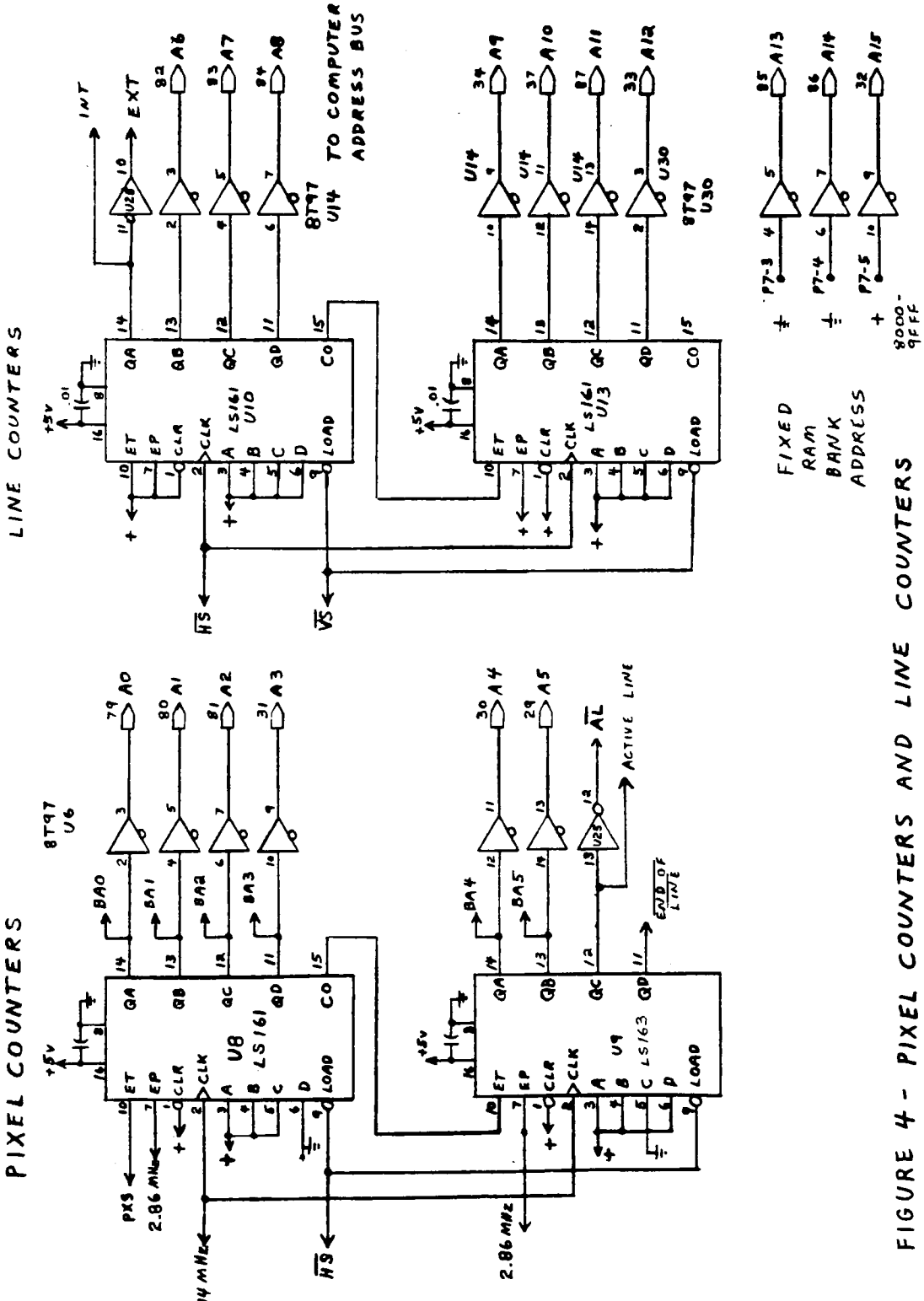

FIGURE 4 - PIXEL COUNTERS AND LINE COUNTERS

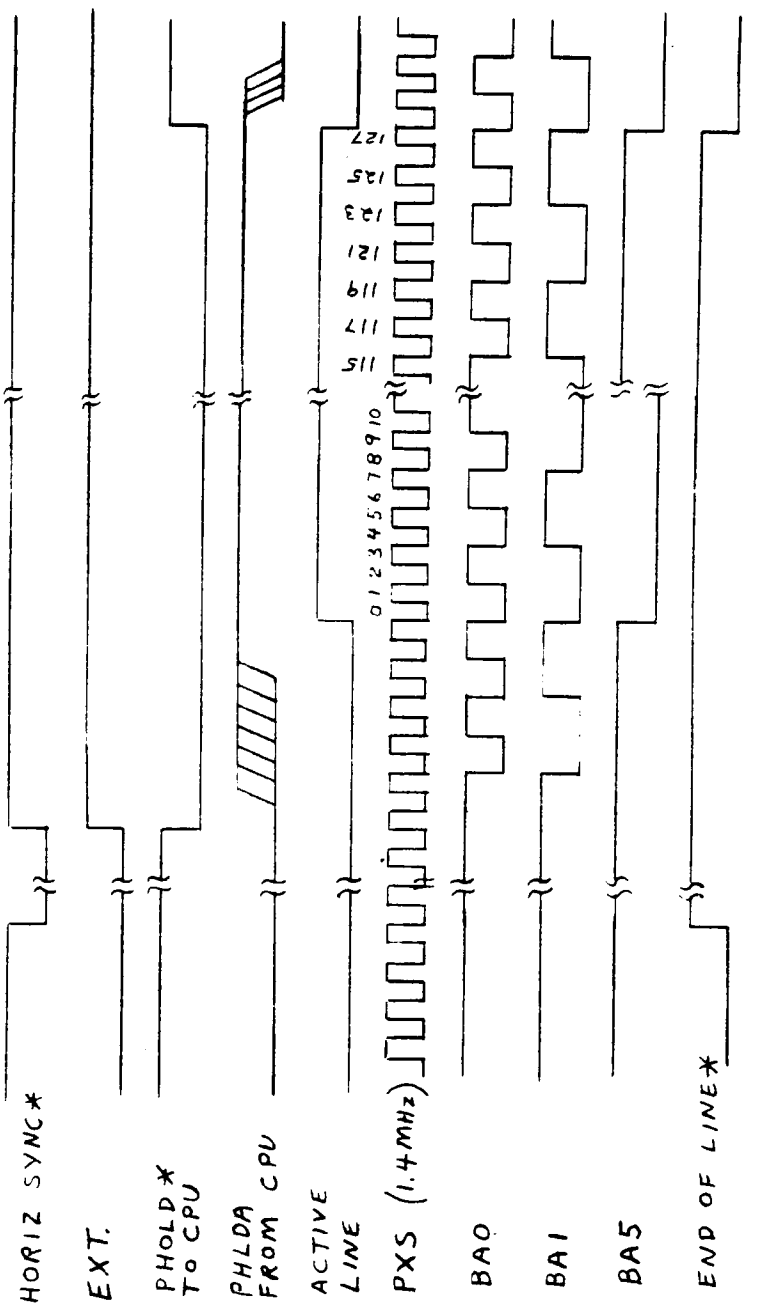

FIGURE 5-HORIZONTAL LINE TIMING

modes. The other outputs of the line counters are not used on the board, but in the external mode, get buffered onto the S-100 address bus to select the proper line of image memory data.

The fixed RAM bank address buffers are jumper selectable to select which 8K byte block of system memory will be used as image memory.

SECTION 5C. LINE BUFFER MEMORY AND DATA BUS

The Line Buffer Memory is the key to this board's ability to use less than 40% of the system bus time by alternately storing then outputting each line. The internal data bus (Figure 6) allows three different modes: 1. in display mode on even lines, data from image memory on the computer bus goes to the line buffer memory and to the output latches (U16 and U20) for output to the video DAC; 2. in grab mode on even lines, data from the video ADC latches goes into the line buffer memory, to the output latches, and to the computer data bus to be written into external image memory; 3. on odd numbered lines, whether in "grab" or "display" mode, pixel data that was just written into the line buffer memory on the previous line, is sent to the output latches in order to repeat that line.

U29 and U21 are clocked on opposite edges of PXS, and thus acquire every other pixel from the video ADC. U17 keeps the low pixel data stable on the data bus while U29 changes.

When the board is put in the no DMA mode, the TPE* (test pattern enable *) signal activates the test pattern buffers, which generate a gray scale video output, useful for testing and calibrating.

Output latch buffers U16 and U20 have 3 state outputs. Their outputs are alternately enabled (by PXS and PXS*) onto the 4 bit bus to the video DAC. The output latch clear signal is a delayed logical AND of ACTIVE LINE* and

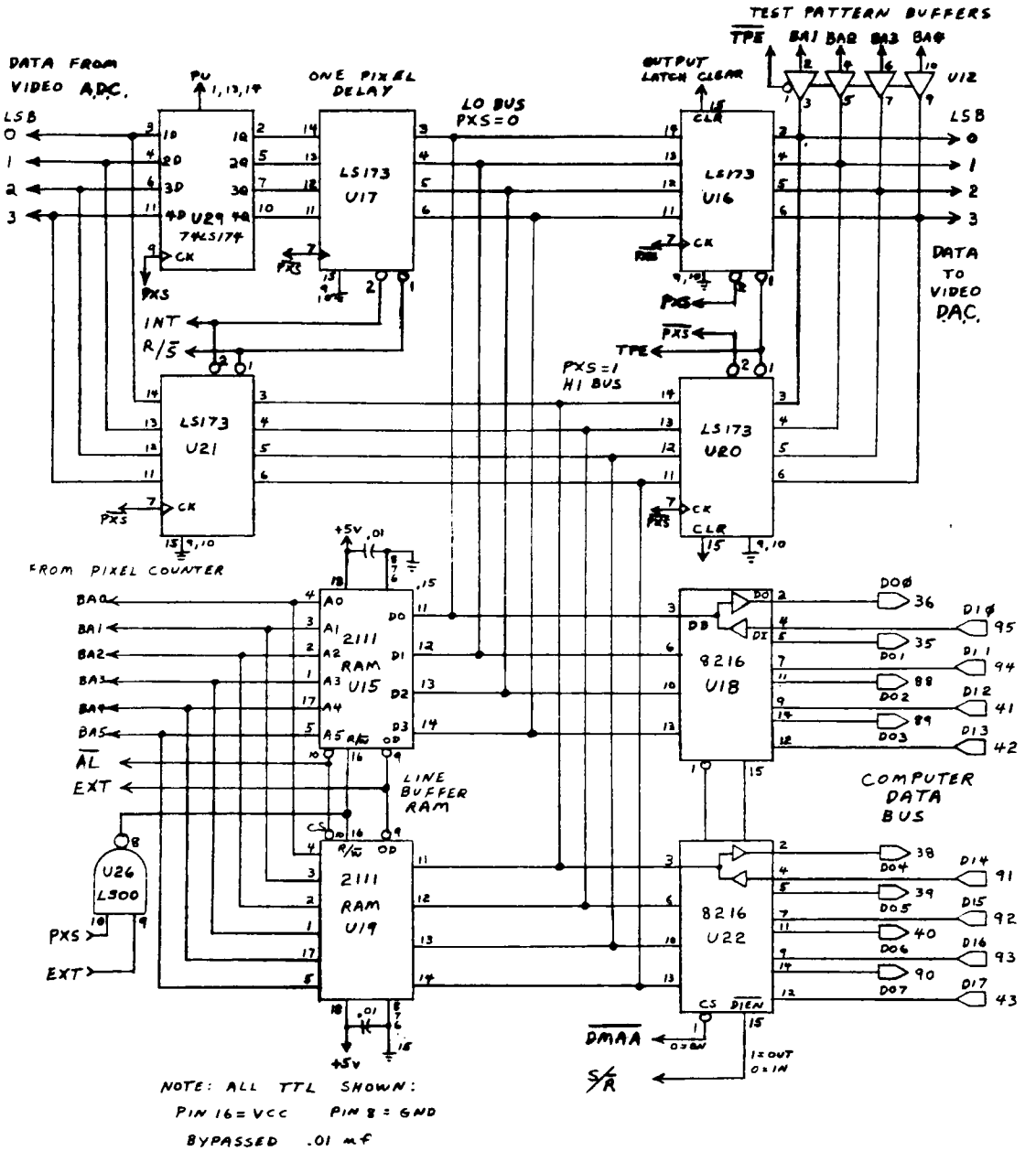

FIGURE 6 LINE BUFFER MEMORY

COMPOSITE BLANKING*, which causes the digital pixel value of zero (black) to be sent out during the left and right borders and vertical blanking time.

SECTION 5D. VIDEO INPUT CIRCUIT

The video input circuit (Figure 7) consists of sync separator and analog to digital (ADC) circuits.

Transistors Q3 and Q4 and the passive components around them strip the active low sync signal from the video input. The circuit has the proper time constants to separate the fast horizontal (15KHz) signal from the slower horizontal (60Hz) sync signal. The sync signals go to the sync multiplexer in the timing and control logic, since they are only used during "grab" mode.

U40 is used as an inverting op amp to provide the proper signal levels needed by the ADC, and also provide separate contrast and brightness controls.

The ADC, U42 is a 4-bit flash converter made by TRW (TDC 1021 J). It can operate at up to 30 million samples per second but is clocked here at only 2.8 MHz. The clock signal comes from the timing circuitry. The digital value of a pixel is available on the outputs 33ns after the rising edge of the clock. The diodes and -6v regulator are for proper biasing and protection of the ADC device.

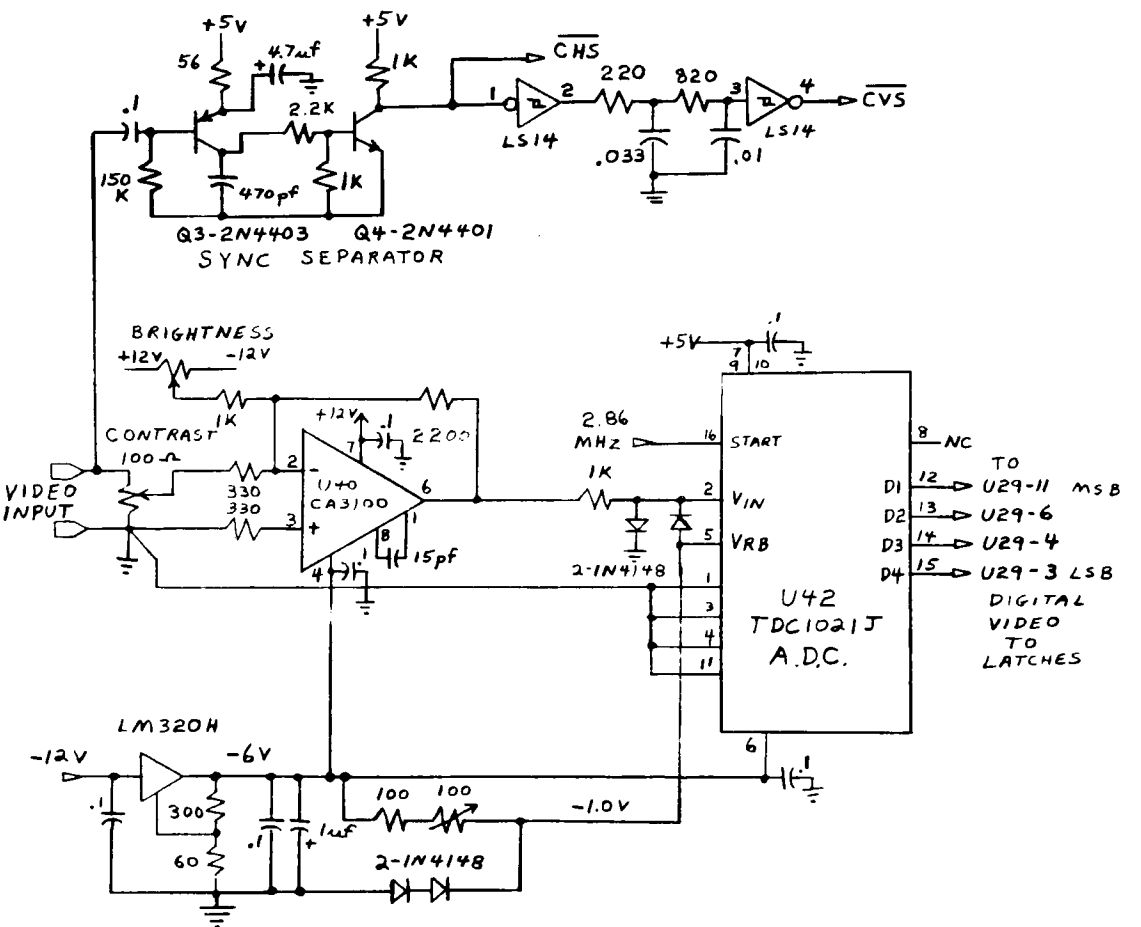

FIGURE 7- VIDEO INPUT CIRCUIT

SECTION 5E. VIDEO OUTPUT CIRCUIT

Figure 8 shows the video output circuit, a digital to analog converter (DAC). The input is 4 bit binary, and the output is an RS-170 compatible composite video signal.

Transistor Q1 and the passive components around it form an adjustable constant current source. The "white amplitude" pot allows adjustment of the current out of the collector of Q1. Transistor Q2 is an emitter follower output buffer. Since the emitter of Q2 is tied to the current source, the output voltage then is a function of the resistance from that node to ground.

U31 and U32 together form a one-of-sixteen decoder (open collector). The input binary code (the pixel value) causes the one selected output of U31 or U32 to be low, grounding the resistor string at the proper tap to produce the proper output voltage.

Constructing the resistor string from 16 separate resistors allows the choice of any desirable monotonic transfer function shape. This could provide a gamma correction for the CRT gamma, or any other system non-linearities. The initial choice was a linear function with all resistor values equal. This has been working fine with the author's Video Monitor. The resistors are mounted on dual inline package headers for simple replacement.

Diode CR1 clamps the current source whenever CS* (composite sync) is active to insert the sync signal on the video output.

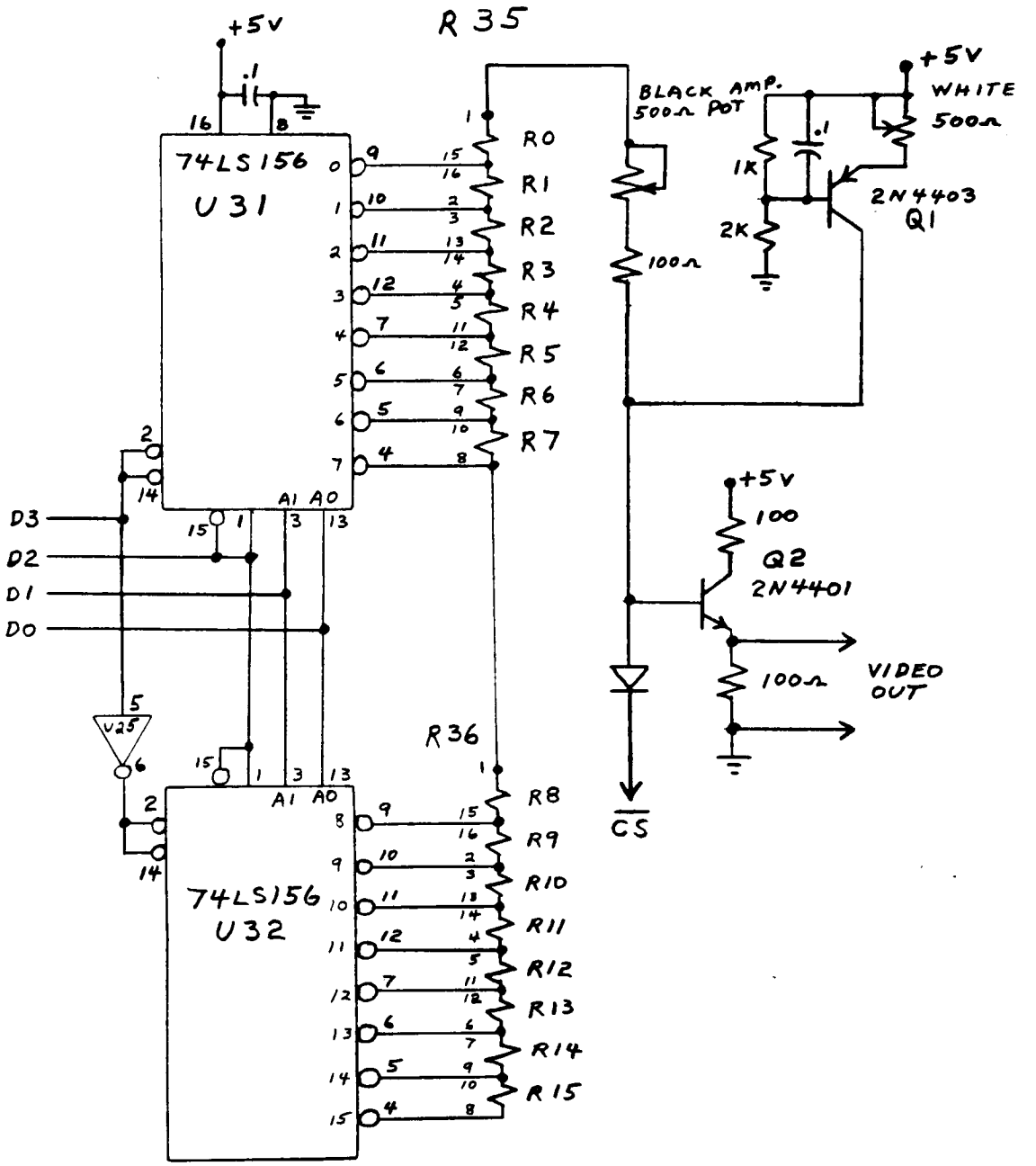

FIGURE 8- VIDEO OUTPUT CIRCUIT

CHAPTER 6. SLOW SCAN BOARD

SECTION 6A. THE SLOW SCAN SIGNAL

The SSTV "video" signal is more like an audio signal than video. It is a constant amplitude but frequency modulated tone. Sync is 1200Hz, black is 1500Hz, and white is 2300Hz. Analog SSTV systems allow a continous gray scale from 1500Hz to 2300Hz. Digital systems use 16 steps of 50Hz from 1525Hz to 2275Hz.

In 60 Hz power countries, the horizontal sweep period of SSTV is 1/15 second (4/60); in 50Hz power countries, it is 1/16.7 second (3/50). Receiving with the wrong frequency results in losing a few pixels on the right edge in one case, or a slightly narrower image in the other case, both causing a slightly distorted aspect ratio. The SSTV horizontal sync pulse width is defined as 5ms, vertical sync as 30ms.

In the U.S. digital standard, the 1/15 second (66.67ms) line time is divided into 139 pixel periods by a 2085 Hz clock, (480us per pixel). 128 pixel periods are image data, 11 pixel periods are horizontal sync.

This FM "video" signal has voice grade qualities. When connected to the microphone input of an AM SSB transmitter, the resultant signal is a constant amplitude RF carrier that swings from +1200Hz to +2300Hz (or -1200Hz to -2300Hz if using lower sideband), thus FM. In the receiver, this moving RF carrier is mixed with a beat frequency oscillator, just as AM SSB voice, to produce the audio output.

Most analog SSTV receiving systems convert the FM video to a voltage by limiting the signal to a constant amplitude and then feeding it to an audio discriminator. The output of the discriminator is a voltage level that corresponds to brightness. In the analog SSTV systems, this voltage would directly control the CRT spot intensity. The two digital systems described in Chapter 2 feed the analog voltage to an analog to digital converter. The voltage gets sampled in time and converted to a binary number that is stored in memory to represent pixel brightness.

The analog method has several faults. The components used in the filters and discriminator are subject to temperature and aging drift. Fixed calibration is almost impossible. The necessary filtering causes a loss of high spatial frequency information. Experimenting with different tone frequencies or modulation modes would require changing analog components. This could only be done in a quantitative, controlled manner if additional calibration equipment were used.

SECTION 6B. SSTV TRANSMIT PROCESS

A much better, and all digital, method of decoding the FM video signal would be to measure the period of each audio cycle, and then use a look-up table to find the corresponding digital brightness level. The similar digital transmit method would be to digitally convert the 4 bit pixel value to the appropriate period to synthesize a sine wave of the desired frequency.

A programmable timer/counter chip with three separate counters is used for both SSTV receive and transmit, with multiplexers to change the signal routing in the two different modes. For purposes of explanation, a separate block diagram is shown for each mode, although some components are used in both modes.

The SSTV transmit block diagram is shown in Figure 9; and SSTV transmit flow chart in Figure 10. The SSTV transmit program is in Appendix A. The 2MHz crystal controlled clock signal from the CPU board is cleaned up, then goes to the clock input of both counters. The pixel clock divider is initialized to divide by 959 to provide the pixel clock frequency of 2085Hz. Each rising edge of the pixel clock sets the sample time latch, which can be tested by the CPU with an input statement. If set, the program clears the latch and sends the proper divisor value to the pixel frequency divider. This divisor value is taken from a lookup table by the SSTV transmit program to correspond to the proper pixel value or sync.

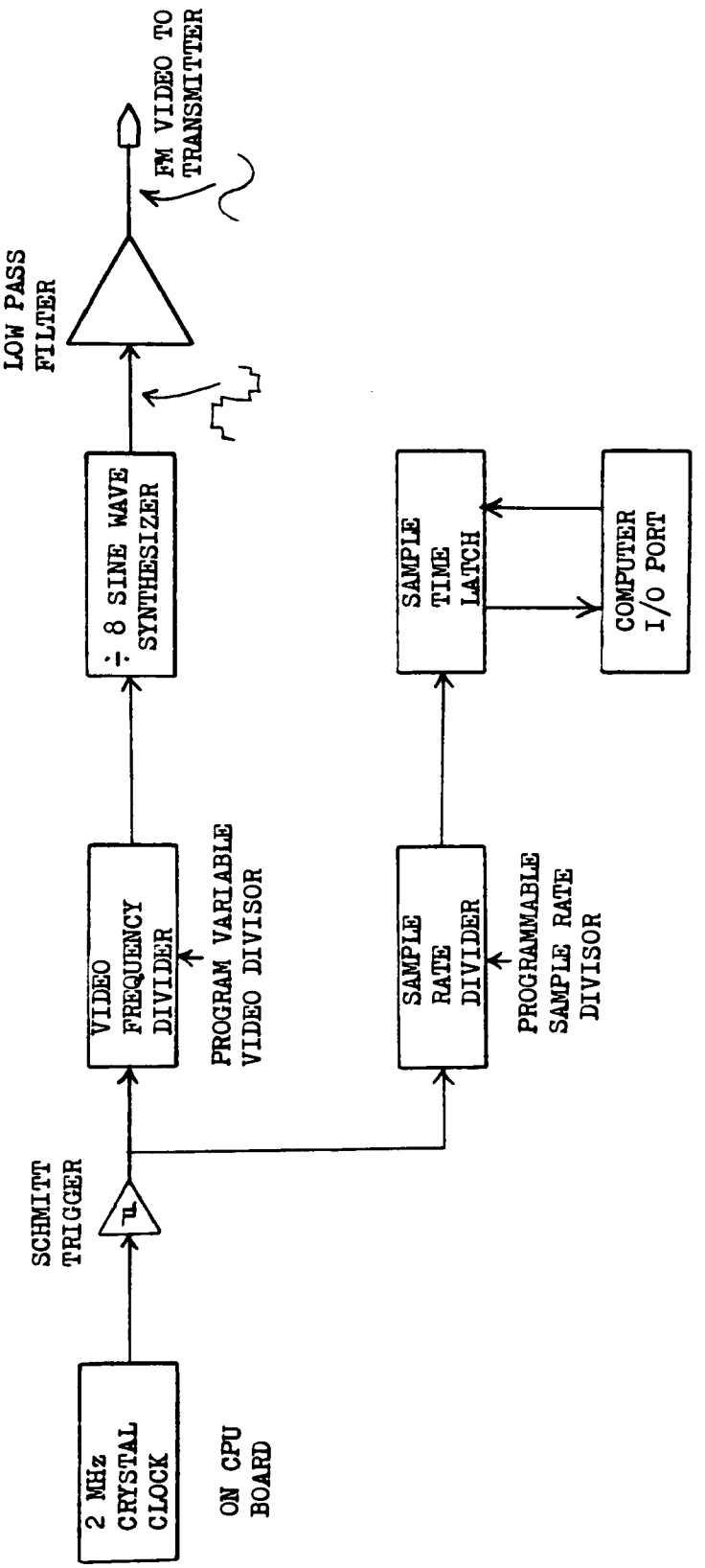

FIGURE 9 - SSTV TRANSMIT BLOCK DIAGRAM

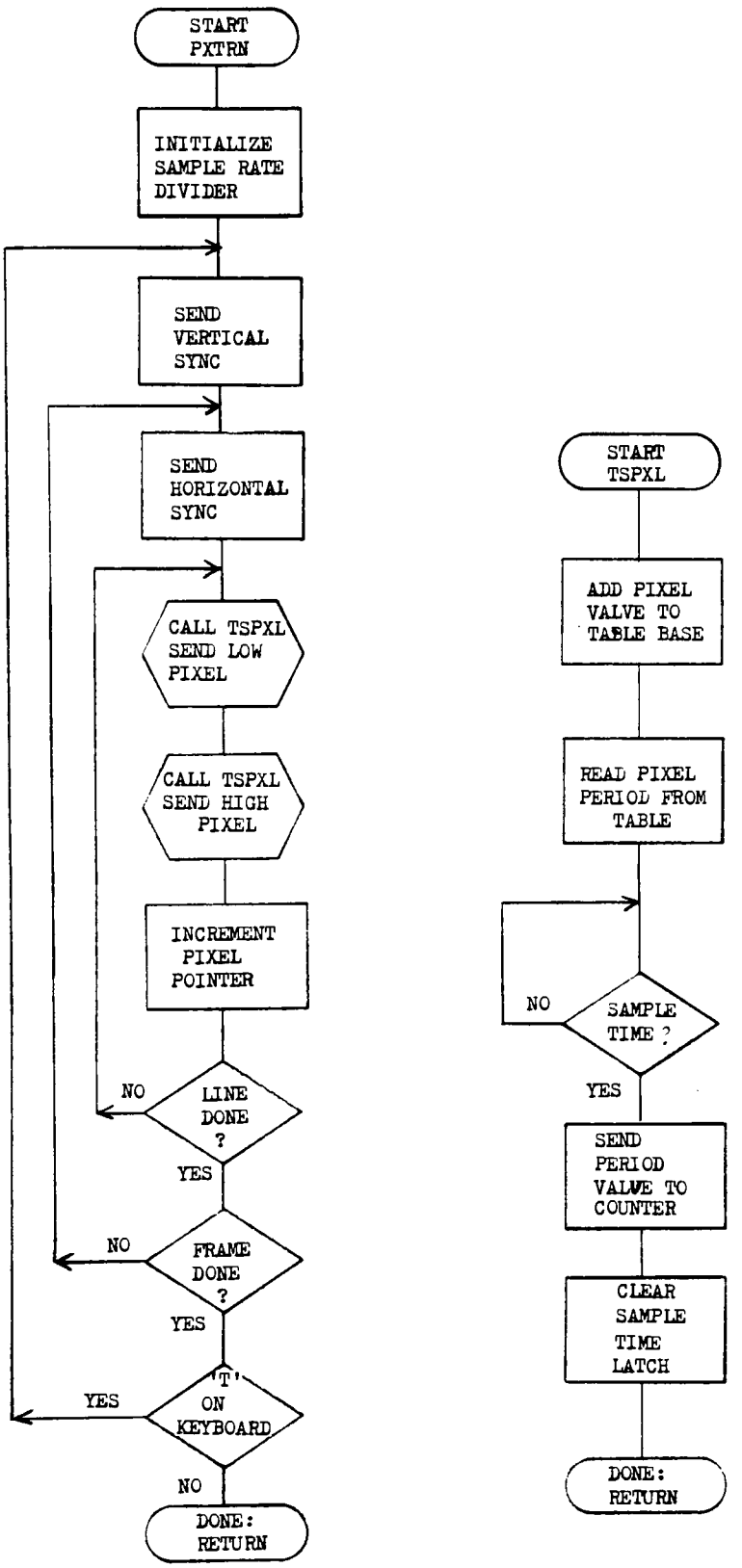

FIGURE 10 - SSTV TRANSMIT FLOW CHART

The output of the pixel frequency divider is actually 8 times the desired output frequency. The sine wave synthesizer creates a stepped sinusoid approximation at 1/8th its input frequency. The low pass filter cleans up the sine wave before it goes to the transmitter or tape recorder.

SECTION 6C. SSTV RECEIVE PROCESS

Figure 11 shows the SSTV board when switched to the receive mode, and the program flow chart is Figure 12. The SSTV receive program is listed in Appendix B. One counter and the sample time latch are used as described before, to generate the 2085Hz pixel clock and indicate to the CPU that it is time to take a sample.

The incoming SSTV FM video is first amplitude limited, and then "squared up" by the Schmitt trigger comparator. This binary signal goes to the enable of one counter, and its complement to the enable of the other counter, so that one counts during positive half cycles, the other counts during negative half cycles. Both are clocked at 250 KHz.

The squared SSTV signal also goes to a circuit that causes an interrupt to the computer on both edges of the squared SSTV signal.

The SSTV receive program processes the interrupt by reading the count out of whichever half cycle counter just stopped, and storing the count in memory. When the program detects that the sample time latch is set, it adds the last positive half cycle count to the last negative half cycle count. It uses the sum to address a look-up table to find the appropriate pixel value. That pixel value is then put into the appropriate location in image memory.

As each pixel value is put in memory, a single bit is sent to the sync integrator latch, set to one if the incoming SSTV signal is within frequency range defined as sync.

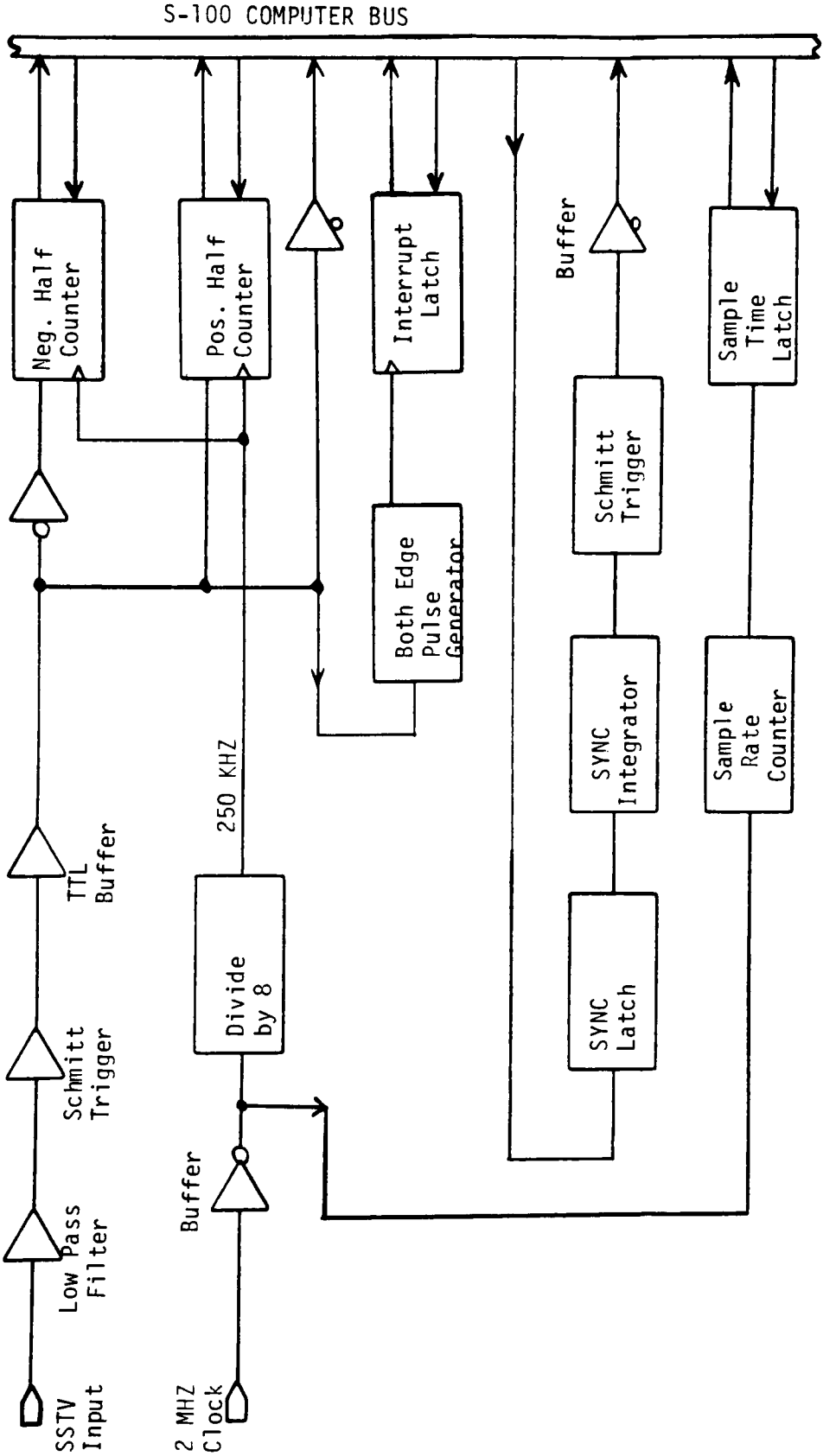

FIGURE 11 - SSTV RECEIVE BLOCK DIAGRAM

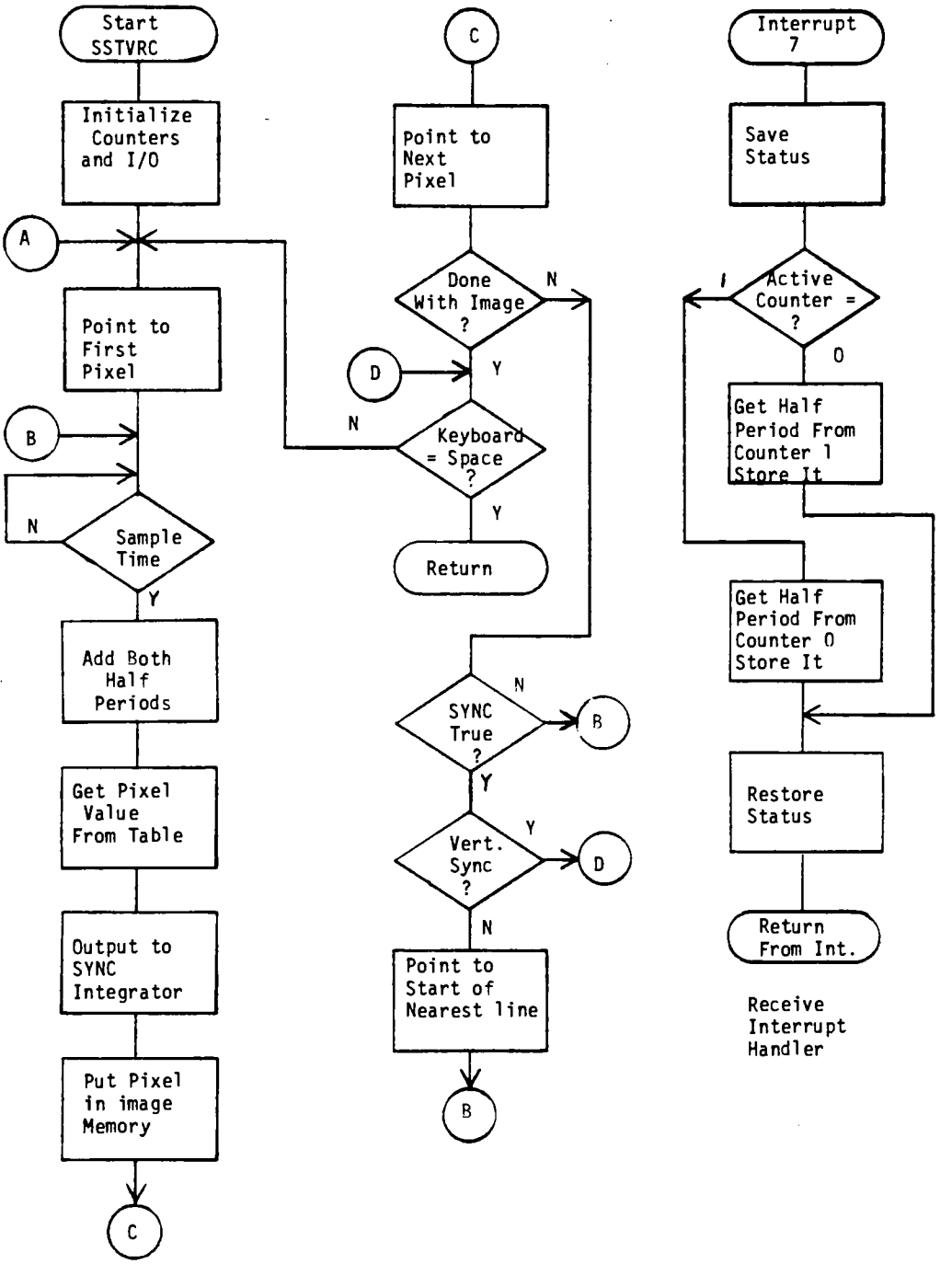

FIGURE 12 - SSTV RECEIVE FLOW CHART

This signal goes to an RC integrator (about 2.5ms), then to a Schmitt trigger, and then to an input port bit. When the program detects that the integrated signal indicates sync, it modifies the receive pixel pointer (a program variable) to cause the next pixel to be put on the beginning of the nearest line.

When the program detects that the integrated sync signal has been active for more than 30 ms it performs a vertical sync test. If the space bar on the ASCII Keyboard has been pressed, the program will return to Pixmon. Otherwise, the pixel pointer will be reset to the upper left corner to receive a new image frame.

SECTION 6D. SSTV BOARD CIRCUIT DESCRIPTION

The schematic diagram is shown in Figure 13 and Figure 14. The operation of the board has just been described, so this explanation will be brief.

Devices U10, U14, and U18 are standard address decoding logic, U2 and U3 are bidirectional bus drivers. U1 is an Intel 8255 programmable input/output chip. U4 is an Intel 8253 programmable timer. It contains three separate 16-bit counters which can be read from, or written to, by the CPU.

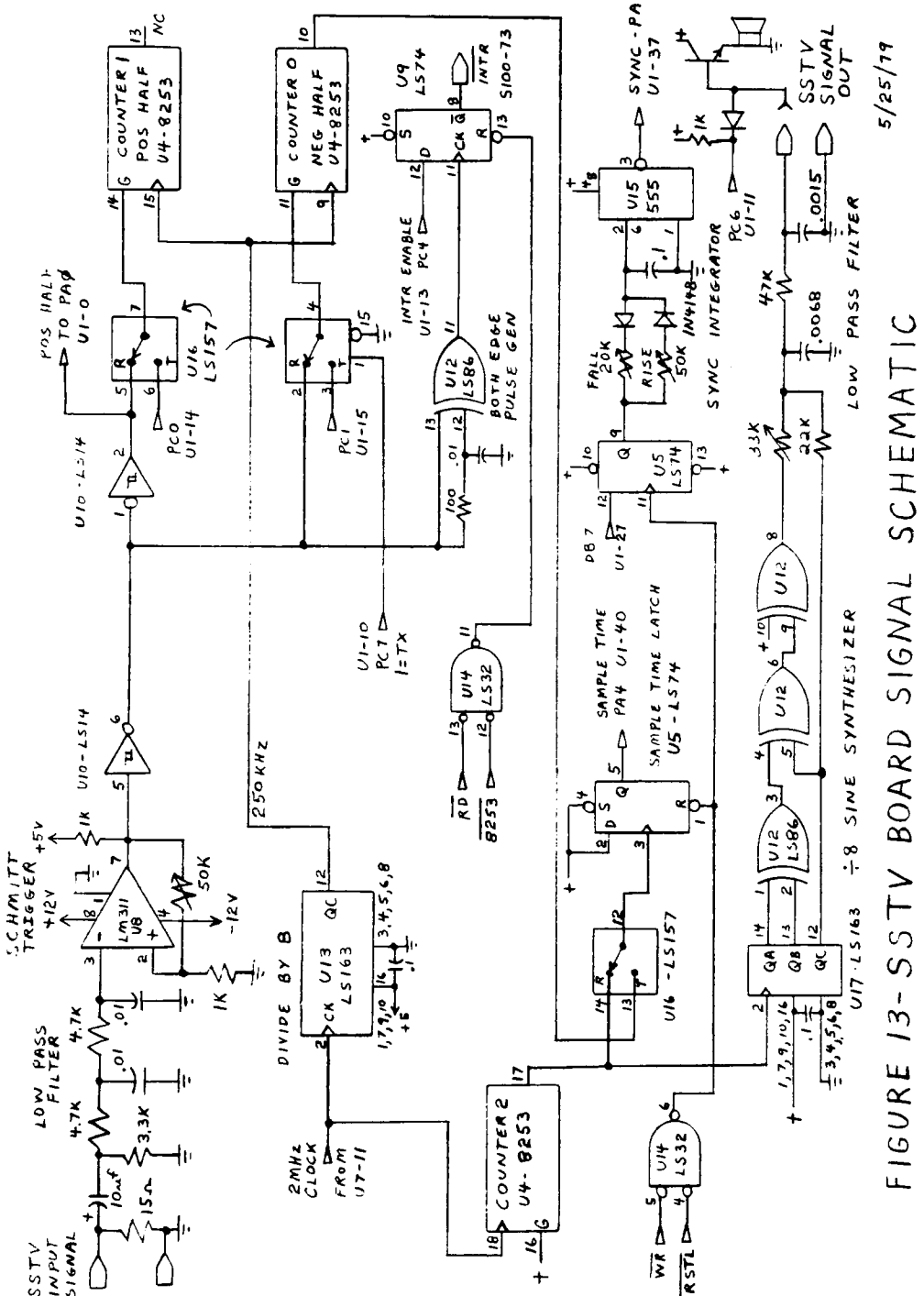

FIGURE 13- SSTV BOARD SIGNAL SCHEMATIC

5/25/79

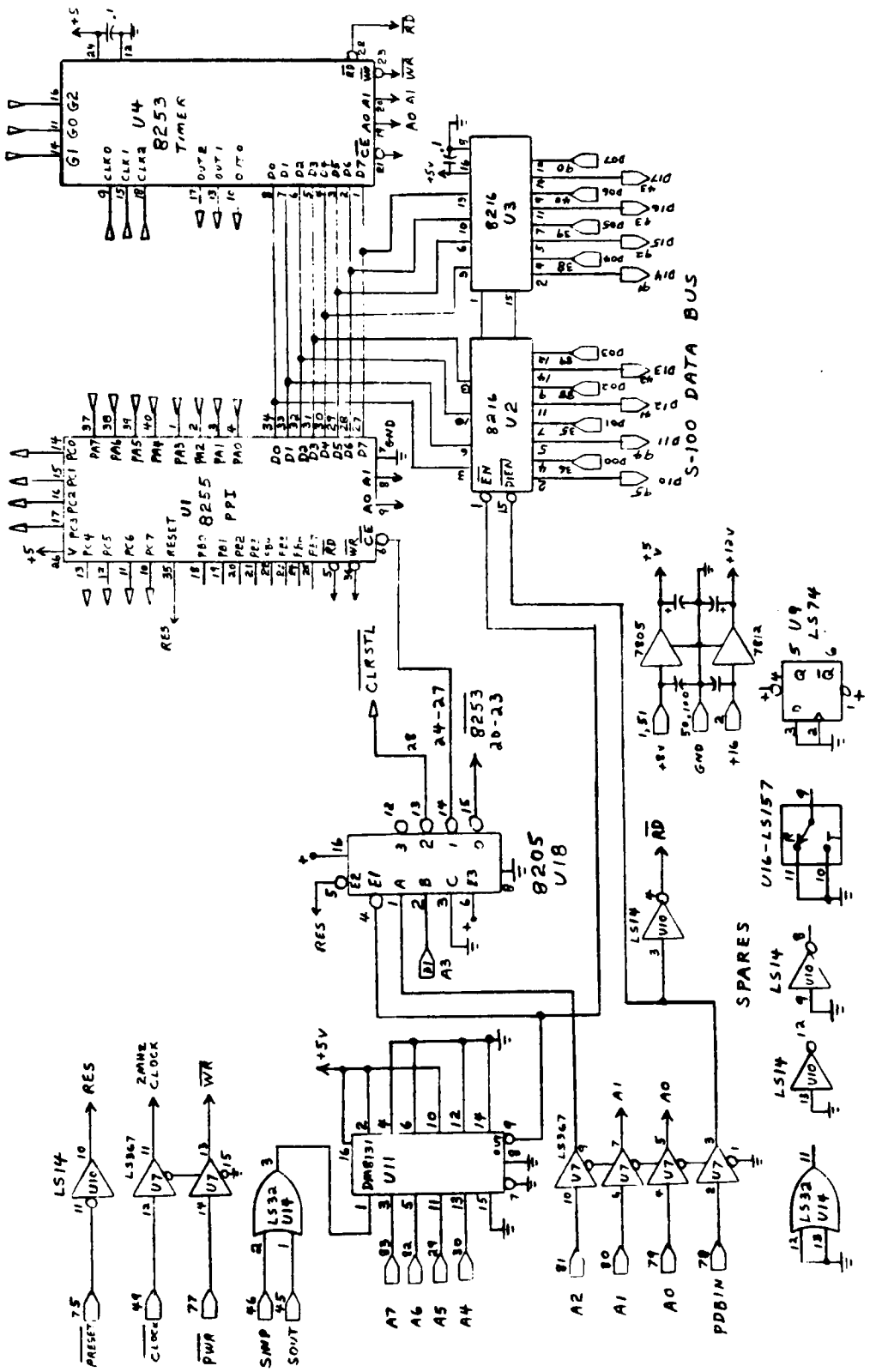

FIGURE 14 - SSTV BOARD BUS INTERFACE

CHAPTER 7. SOFTWARE

SECTION 7A. PROGRAM DEVELOPMENT

The system described in this paper does not truly stand alone. In the present configuration, there is very little program development capability. To enter, edit, assemble, and compile programs, a floppy disk drive, more memory, and appropriate programs would be needed at an additional cost of several hundred dollars.

All programs used in this study were entered and assembled on a separate computer. The programs were electrically programmed into ultraviolet erasable programmable read only memories (EPROM), which were then plugged into the image processing system for final debugging. This method allows the program to be permanently resident, and therefore, immediately executable after power up.

Structured programming techniques were used, with programs and modules written as independent blocks. Most of the image processing routines written for this study can be executed in either of two modes, as a subroutine called by a larger, more complex program, or as a single function executed immediately after the appropriate command is typed on the keyboard.

All of the programs were written in 8080 assembly language, with macros written to perform the Z80 operation codes which are not part of the 8080 instruction set. If

the system had a disk drive, high level languages such as Fortran, Pascal, or BASIC could be used.

The main line program is called Pixmon for "picture monitor." It initializes variables and hardware devices and waits for one of 26 single letter commands. While it is waiting for the command, a 2 pixel cursor blinks on the left edge of the video monitor image. Some functions are executed immediately, others require one or more additional characters. When finished, each function does a return, giving control back to Pixmon to wait for the next command. Figure 15 is a list of commands in Pixmon. The next section gives details of each command function.

```
CTABLE:   ;PIXMON COMMAND TABLE, LISTED ALPHABETICALLY

 ALPHA    ; A ALL ALPHANUMERIC ROUTINES, VECTOR ON SECOND KEY
 BITMSK   ; B MASK OFF BITS (ENTER HEX BIT PATTERN)
 CKBD     ; C CHECKERBOARD TEST PATTERN
 DSTBN    ; D DISTRIBUTION DATA COLLECTION AND GRAPHING
 SHRINK   ; E SHRINK PIX TO 1/4 SIZE, MAKE 4 COPIES
 FILLCM   ; F FILL PIX MEM WITH SAME VALUE (ENTER HEX VALUE)
 GRAYCM   ; G GRAY SCALE TEST PATTERN (VERTICAL BARS)
 HDERIV   ; H HORIZONTAL FIRST DERIVATIVE
 INVERT   ; I INVERT PIX (UPSIDE DOWN)
 JUMBLE   ; J RANDOM PIXEL VALUES('0' FIRST FOR STANDARD INIT VALUE)
 CLIP     ; K CLIP (THRESHOLD) TO BINARY PIX (ENTER HEX THRESHOLD)
 LOAD     ; L LOAD BOY-DOG DEMONSTRATION IMAGE FROM ROM
 GETCM    ; M ASSEMBLY LANGUAGE MONITOR PROGRAM
 NEGATE   ; N NEGATE PICTURE (ONE'S COMPLEMENT)
 ORDRLY   ; O INITIALIZE RANDOM NUMBER
 NAYBOR   ; P NEIGHBORHOOD AVERAGING NOISE REMOVAL
 NOISE    ; Q PUT RANDOM BLACK AND WHITE SPOTS ON SCREEN ('0' FIRST TO INIT)
 PXREC    ; R RECEIVE SSTV RR=RECEIVE, RS=SPECTRUM, RH=HALF SPEED
 SAMPLE   ; S CONTROL GRAB / STORE MODE
 PXTRN    ; T TRANSMIT SSTV: TT=STANDARD, TH=HALF SPEED
 XLAT     ; U TABLE TRANSLATE: UU= PREVIOUS TABLE, UB = ADD 1
 CONVLV   ; V 3X3 CONVOLUTION: SEE LIST BELOW
 ETCH     ; W WRITE WITH BLINKING CURSOR
 TRNSPZ   ; X TRANSPOSE PICTURE MATRIX ACROSS DIAGONAL
 DEMO     ; Y DEMONSTRATION MODE (SELF CAPTIONING)
 ZOOM     ; Z ZOOM IN ON A QUADRANT OF PICTURE(0 TO 4, 4 IS CENTER)


;COVOLUTIONS: V=PREVIOUS MATRIX, E= ROBERTS EDGE DETECT
;PRESET KERNELS: 1,2=LO PASS        3 TO 6 = HI PASS

;  1:       2:        3:          4:          5:          6:          7:
;  1,1,1    0,1,0     0,-4, 0     0,-2, 0    -2, 0,-2     0,-7, 0     0, 0, 0
;  1,1,1    1,4,1    -4,24,-4    -2,16,-2     0,16, 0    -7,36,-7     0,16,-2
;  1,1,1    0,1,0     0,-4, 0     0,-2, 0    -2, 0,-2     0,-7, 0     0,-2,-4

;              FIGURE 15 - PIXMON COMMAND LIST
```

SECTION 7B. BRIEF PROGRAM DESCRIPTIONS

A: ALPHABETICS - provide for putting input or stored alphanumeric characters on the image stored in image memory to allow captioning. Image data is lost wherever character data is written.

B: BIT MASK - clear specified bits of image data by "anding" with the second input parameter.

C: CHECKERBOARD - generate a checkerboard test pattern in image memory.

D: DISTRIBUTION - collect data on the distribution of pixel values, and can graph the data if requested.

E: SHRINK - reduce an image to 64 x 64, then duplicate that image in all four quadrants.

F: FILL - fill image memory with a single specified value.

G: GRAY SCALE - generate a 16 step gray scale on the top edge of an image, or the whole image.

H: HORIZONTAL FIRST DERIVATIVE - create a new image that is the horizontal first derivative of the original, useful for analyzing frequency content of an image.

I: INVERT - move image data to turn the picture upside down.

J: JUMBLE - fill the image memory with pseudo-random pixel values.

K: CLIP - threshold the image at the specified threshold to create a binary (2 level) image.

L: LOAD - load the Boy-Dog demonstration image into image memory.

M: MONITOR - go to assembly language monitor program.

N: NEGATE - create a negative image.

O: ORDERLY - initialize pseudo-random numbers used for J and Q commands.

P: NEIGHBOR - neighborhood averaging noise filter.

Q: NOISE - puts 100 black pixels and 100 white pixels in pseudo-random locations in image memory each time key is pressed.

R: RECEIVE SSTV - from ham receiver or cassette tape.

S: SAMPLE - control grab/store mode.

T: TRANSMIT SSTV - to ham transmitter or cassette tape.

U: TRANSLATE - point processing by translating pixel values through a 16-element look-up table.

V: CONVOLVE - perform filtering with a 3x3 convolution kernel or do Roberts edge detection.

W: WRITE - cursor movement and tracing.

X: TRANSPOSE - transpose the image matrix across the diagonal; vertical data is now horizontal and vice versa.

Y: DEMONSTRATION - a free running, self captioning mode to demonstrate the system's capabilities.

Z: ZOOM - expand data from any quadrant, or center, of image to fill the screen. Rearranges image data, information is lost.

SECTION 7C. POINT PROCESSING

Point processing is a very powerful, yet simple, digital image processing technique. As defined by Pratt[8], it is called point processing because the new value of each point (pixel) is a function only of that point's previous value. The function could be linear, quadratic, exponential, logarithmic; any definable function.

In practice, rather than performing the calculations separately for each pixel, a look-up table is created. In the case of this sytem, the input pixel value becomes the index in a 16 element look-up table. The new pixel value is the selected element from the table.

This technique is commonly used to increase the contrast of an image (or portion of an image). It was this technique that first allowed the detection of an active volcano on Io, one of Jupiter's moons. Figure 16 shows before and after images of point processing to increase contrast on an image received from the JPL (Jet Propulsion Laboratory of California Institute of Technology) Amateur Radio Club.

Another application of point processing is to find all pixels of a certain brightness level. Figure 17 shows the result of having level 7 changed to 15, using the 16 element lookup table of 0, 1, 2, 3, 4, 5, 6, 15, 7, 8, 9, 10, 11, 12, 13, 14. Notice that this table has not destroyed any information but has just encoded the levels differently.

FIGURE 16A - Photograph of Saturn taken by Voyager 1, as received via SSTV with low contrast.

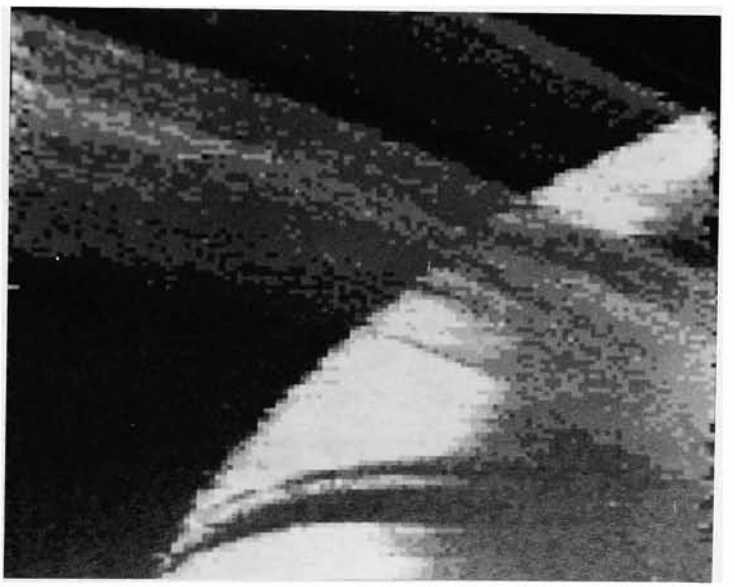

FIGURE 16B - The same image as above, after contrast expansion by point processing.

FIGURE 17A - Original Boy-Dog Image

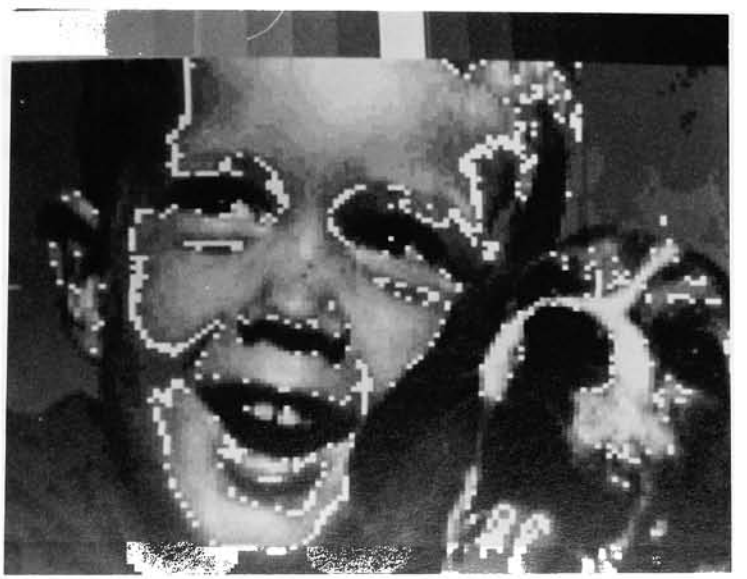

FIGURE 17B - Photography of Boy-Dog image after point prccessing to accentuate level 7 (mid-gray) by making it white, using the translation table: 0, 1, 2, 3, 4, 5, 6, 15, 7, 8, 9, 10, 11, 12, 13, 14.

SECTION 7D. CONVOLUTION

Perhaps the most powerful technique for digital image processing is convolution. Convolution allows filtering an image in the two dimensional spatial frequency domain with a simple "multiply and add" algorithm without first converting the image data to the frequency domain. The characteristics of the filter specify a set of convolution coefficients called the "kernel." In general, positive coefficients perform integration (low pass), while having some negative coefficients perform differentiation (high pass). Only a 3x3 convolution program was implemented. The program could easily be modified for larger kernels if desired, or the 3x3 algorithm could be used several times to effect larger kernels as described by Abramatic and Faugeras.[9] Convolution is covered thoroughly by Oppenheim[10] and Pratt.[8]

The convolution program takes about 40 seconds to convolve the 3 X 3 kernel with the 128 X 128 pixel image. It must perform 147,456 software multiplications (3 x 3 x 16,384 pixels). In order to save memory, yet not use modified data, three lines of image data are removed from image memory to a scratch pad memory. The convolution reads from this scratch pad data, but immediately puts the output pixel back into image memory.

The convolution program listing appears in the appendix. Several example photographs appear in Figure 18.

FIGURE 18A - Boy-Dog image after convolution with low pas kernel: 0, 1/8, 0,
1/8, 1/2, 1/8,
0, 1/8, 0

FIGURE 18B - Original Boy-Dog image after convolution with high pass kernel:
0, -1/2, 0
-1/2, 3, -1/2
0, -1/2, 0

SECTION 7E. NEIGHBORHOOD AVERAGE NOISE FILTER

This program takes advantage of the fact that noise does not need to be removed unless it is objectionable. In SSTV, impulse noise usually causes completely black or completely white pixels. A black pixel in a dark area, or a while pixel in a bright area, may not be objectionable.

This program is based on a method explained by Pratt.[11] For each pixel of the image, the eight pixel "neighbors," (those in contact with the sides or corners of the center pixel) are averaged. The center pixel value is subtracted from this average. If the absolute value of the difference is greater than a specified threshold, the center pixel is replaced by the average. This method is superior to low pass filtering because it only modifies the image where modification appears to be necessary.

A program to generate 100 black and 100 white pixels at pseudo-random locations was written to test the noise filter program.

Figure 19 shows photographs of before and after noise filtering. Note that some original high frequency information was destroyed, such as the reflection in the boy's eyes. Note also that white spots were left in bright areas, and black spots were left in dark areas.

A listing of this program appears in the Appendix.

Figure 20 includes photographs of images after execution of some of the other programs described in section 7B.

FIGURE 19A - Boy-Dog image after 100 black spots and 100 white spots were written into pseudo-random pixel locations by a program to simulate impulse noise.

FIGURE 19B - Above image after execution of Neighborhood Average Noise Filter program.

Bit Mask, Remove LSB
image is now 128 x 128 x 3

Shrink then Zoom
image is now 64 x 64 x 4

2:1 Zoom

Negate

Original helicopter image

Helicopter image after
Roberts Edge Detection and Clip

FIGURE 20

CHAPTER 8. CONCLUSION

A powerful, but low cost, digital image processing system has been conceived, designed, constructed, programmed and made operational. A unique "line buffer" architecture and the use of the most advanced 8-bit MOS microprocessor give the system powerful software capabilities. Costs have been kept down by limiting the image quality to SSTV standards and by employing the computer bus with the widest selection of options and vendors.

A unique, all digital, method of encoding and decoding SSTV signals allows experimentation and measurement by changing program variables.

An easily expanded base of image processing programs has been developed. These include powerful functions such as two dimensional convolution, point processing, and neighborhood average noise removal, as well as a host of "special effects" programs such as invert, shrink, zoom, and cursor tracing.

The system has been demonstrated to hundreds of hams, computer hobbyists, and professionals in several cities, and as such has helped spread a basic understanding of digital image processing. Many persons are duplicating parts of the hardware, and are using the programs. The system is a powerful tool to use digital image processing for SSTV.

REFERENCES

1. Federal Communications Commission Regulations part 97. "Amateur Radio Service"
2. Kirsch, R. A., L. Cahn, L. C. Ray, and G. H. Urban, "Experiments in Processing Pictorial Information with a Digital Computer," Proceedings of the Eastern Joint Computer Conference, 1957.
3. Nagao, M., et al. "An Interactive Picture Processing System on a Minicomputer," Proceedings, 1974 International Conference on Pattern Recognition and Image Processing.
4. McDonald, Copthorne - "A New Narrow Band Image Transmission System," QST August, September, 1958.
5. Steber, George, Dr. "SSTV to Fast-Scan Converter," QST, March, May, 1975.
6. Abrams, Clayton. "Display SSTV Pictures on a Fast Scan TV," Ham Radio Magazine, July, 1979.
7. Electronics Industries Association, "Electrical Performance Standards - Monochrome Television Studio Facilities, EIA Standard RS-170."
8. Pratt, William K., "Digital Image Processing" John Wiley and Sons, 1978.
9. Abramatic, J. F., and Faugeras, O., "Design of Two Dimensional FIR Filters from Small Generating Kernels," Proceedings, 1978 PRIP Conference, P123.

10. Oppenheim, A. V., and Schafer, R. W., "Digital Signal Processing," Prentice Hall, Inc., 1975.

11. Pratt, William K., "Course Notes, Digital Image Processing," Integrated Computer Systems, 1979.

APPENDIX A - SSTV TRANSMIT PROGRAM LISTING

```
;TRANSMIT IMAGE DATA VIA SSTV MODE
PXTRN:
  LXI     H,TINIT ;POINT TO TRANSMIT INIT DATA
  CALL    PXINIT  ;INITIALIZE COUNTERS AND HARDWARE
  CALL    BLINKH  ;GET SECOND COMMAND CHARACTER
  CPI     'H'     ;HALF SPEED MODE ?
  JNZ     TSPXB
  MVI     A,240   ;120 * 2, HALF STANDARD RATE
  OUT     CNT0
TSPXB:
  LXI     H,PXMEM ;POINT TO IMAGE MEMORY
  LXI     D,PXMEM
  LDAX    D       ;GET FIRST 2 PIXELS
  MOV     B,A
;VERT SYNC COUNT, ONE WHOLE LINE, 139 PIXEL PERIODS
  MVI     C,139
TSPXC:
  MVI     A,208   ;COUNTER VALUE FOR SYNC FREQ
  OUT     CNT2    ;TO SINE SYNTHESIZER
TSPXD:
  IN      PORTA
  ANI     10H     ;SAMPLE TIME LATCH BIT
  JZ      TSPXD   ;WAIT FOR SAMPLE TIME LATCH
  OUT     CLRSTL  ;CLEAR THE LATCH
  DCR     C       ;PIXEL COUNTER
  JNZ     TSPXD   ;KEEP SENDING SYNC UNTIL ZERO
  MOV     A,B     ;FIRST 2 PIXELS ON LINE
  STAX    D       ;PUT BACK IN PIX MEM
  MOV     B,M     ;FIRST PIXEL THIS LINE
  MOV     C,B     ;SAVE BOTH PIXELS
;PUT TRANSMIT CURSOR IN IMAGE MEMORY (TEMPORARILY)
  MVI     M,0FH   ;SPOT ON BEGINING OF ACTIVE LINE
  MOV     D,H     ;SAVE CURSOR POINTER
  MOV     E,L
TSPXE:
  MOV     A,C     ;2 PIXELS
  ANI     0FH     ;MASK, REMOVE UPPER PIXEL
;TRANS LO PIXEL
  CALL    TSPXL
  MOV     A,C     ;2 PIXELS
  RRC
  RRC
  RRC
  RRC
  ANI     0FH
```

APPENDIX A- CONTINUED

```
;TRANS HI PIXEL
  CALL    TSPXL
  INX     H         ;POINT TO NEXT 2 PIXELS
;SAVE NEXT PIXEL
  MOV     C,M
;TEST FOR END OF LINE
  MOV     A,L
  ANI     63        ;START OF NEW LINE ?
  JNZ     TSPXE     ;NO, JUST SEND NEXT 2 PIXELS
;DONE WITH LINE, SEND SYNC FREQ
  MVI     A,16      ;PIXEL VALUE CODE FOR SYNC
  CALL    TSPXL     ;WILL SEND SYNC

;TEST IF PICTURE DONE
  MOV     A,H
  CPI     HIGH PXDUN
  JZ      TSPXF     ;TEST FOR EXIT IF DONE
;SET UP FOR NEXT LINE
  MVI     C,10      ;11 SAMPLE TIMES TOTAL FOR HORZ SYNC
  JMP     TSPXD
TSPXF:
  LDA     KBIN      ;GET CHAR FROM UART
  CPI     NOT 'T'
  JZ      TSPXB     ;IF T, SEND ANOTHER FRAME
;TURN OFF TRANSMITTER,BUT KEEP SYNC TONE RUNNING
  MVI     A,(TXMODE AND(NOT(40H))) ;TRANSIT BIT OFF
  OUT     PORTC
  RET               ;BACK TO PIXMON

TSPXL:
;WAIT FOR SAMPLE TIME LATCH, THEN SET COUNTER FOR PIXEL FREQ
;GET PIXEL PERIOD FROM TABLE FIRST, TO HAVE IT READY
  XBDH
  MVI     H,HIGH(TXTBL) ;POINT TO PIXEL PERIOD LOOKUP TABLE
  ADI     LOW(TXTBL) ;ADD PIXEL VALUE
  MOV     L,A       ;POINT TO TIME VALUE
  MOV     L,M       ;TIME VALUE IN L
TSPXM:
;WAIT FOR SAMPLE TIME LATCH TRUE
  IN      PORTA
  ANI     10H
  JZ      TSPXM     ;LOOP TIL TRUE
  OUT     CLRSTL    ;CLEAR LATCH
  MOV     A,L       ;PIXEL PERIOD TIMER VALUE (*1/8)
  OUT     CNT2      ;TO COUNTER
  XBDH
  RET
```

APPENDIX A- CONTINUED

```
TXTBL:
   DB       168,160,155,150,146      ;PIXEL VALUES 0,1,2,3,4
   DB       142,139,135,131,128      ;5,6,7,8,9
   DB       125,122,119,116,113      ;10,11,12,13,14
   DB       108                      ;15=WHITE
   DB       208                      ;SYNC, 1200 HZ
```

APPENDIX B - SSTV RECEIVE PROGRAM LISTING

```
PXINT:  ;PICTURE REC INTERRUPT
   XAF              ;USE A PRIME
   IN       PORTA   ;WHICH COUNTER IS ACTIVE ?
   ANI      8       ;IF 0 IS ACTIVE, GET COUNT FROM 1
   JZ       PXNT2
   IN       CNT0    ;NEGATIVE HALF CYCLE
   STA      PRVCN   ;STORE COUNT IN MEM
   XAF              ;BACK TO UNPRIME REGISTERS
   EI               ;RE-ENABLE INTERRUPTS
   RET
PXNT2:
   IN       CNT1    ;POSITIVE HALF CYCLE
   STA      PRVCN+1 ;STORE NEW VALUE
   XAF              ;BACK TO UNPRIME REGISTERS
   EI               ;RE-ENABLE INTERRUPTS
   RET

;RECEIVE SLOW PICTURE
SSTVRC: MVI      A,87H   ;CONTROL WORD FOR INTERFACE CHIP
   STA      0FFE3H  ;8255 CONTROL PORT
   LXI      H,RINIT
   CALL     PXINIT
SSTVR2: CALL     RSPXA3
   JMP      SSTVR2

PXREC:
   CALL     BLINKH  ;GET ANOTHER LETTER
   CPI      'S'     ;DISPLAY SPECTRUM ?
   JZ       PIXTRM
;OTHERWISE RECEIVE
   PUSH     PSW     ;SAVE INPUT CHARACTER
   LXI      H,RINIT ;INITIALIZE COUNTERS AND I/O FOR REC
   CALL     PXINIT
   POP      PSW     ;SECOND CHAR IN REG A
   CPI      'E'     ;EUROPE SWEEP SPEED ?
   JNZ      RSPXA1
   LXI      H,870   ; 1/16 SECOND PER LINE
   JMP      RSPXA2
RSPXA1:
   CPI      'H'     ;HALF SPEED MODE ?
   JNZ      RSPXA3  ;LEAVE AS IS IF NOT
   LXI      H,1918  ;959*2, HALF THE NORMAL FREQUENCY
```

APPENDIX B - CONTINUED

```
RSPXA2:
  MOV     A,L
  OUT     CNT2
  MOV     A,H
  OUT     CNT2
RSPXA3:
  LXI     H,PRVCN ;POINT TO LAST COUNTER VALUE
  MVI     D,HIGH(RECTBL)  ;HI POINTER TO LOOKUP TABLE
  XBDH            ;SWAP REGISTER SETS
RSPXA:
  LXI     H,PXMEM ;POINT TO IMAGE MEMORY
  MVI     B,0     ;CLEAR HI/LO FLAG
RSPXD:
  MVI     C,0     ;CLEAR SYNC COUNT
RSPXB:
  XBDH            ;SWAP REGISTER SETS
;HL NOW POINTS TO PRVCN, D HAS HI BYTE OF RECTBL

;WAIT FOR SAMPLE TIME LATCH OUTPUT TRUE
RSPXC:
  IN      PORTA
  ANI     10H     ;SAMPLE TIME LATCH BIT
  JZ      RSPXC
;FORM NEW VALUE
  MOV     A,M     ;COUNTER 0
  INR     L       ;POINT TO COUNTER 1 VALUE
  ADD     M
  DCR     L
  MOV     E,A     ;POINTER IN TABLE
  LDAX    D       ;GET BRIGHTNESS VALUE FROM TABLE
  XBDH            ;CHANGE REGS BACK
;OUTPUT TO SYNC INTEGRATOR AND CLEAR SAMPLE TIME LATCH
  OUT     CLRSTL
;PUT PIXEL VALUE INTO PICTURE MEMORY
  RRD             ;ROTATE NIBBLE INTO MEM
  BIT     0,BR
  JNZ     RSPXF
  INR     L       ;MOVE POINTER TO NEXT 2 PIXELS
  JNZ     RSPXF
  INR     H
  MOV     A,H
  CPI     20H+(HIGH PXMEM) ;IS H OUTSIDE OF VALID PIX MEMORY ?
  JC      RSPXF   ;CONTINUE IF STILL VALID
 LXI     H,PXEND ;HOLD ON LAST PIXEL UNTIL VERT SYNC
```

APPENDIX B - CONTINUED

```
;TEST KEYBOARD TO SEE IF OPERATOR WANTS TO STOP
  LDA     KBIN     ;GET INVERTED CHAR
  CPI     NOT (' ') ;SPACE CHAR IS SIGNAL TO STOP
  JZ      RXEXIT   ;EXIT RECEIVE FUNCTION IF TOLD TO STOP
  CPI     NOT('V');IS IT V FOR VERT SYNC ?
  JZ      RSPXA    ;START OVER AT TOP IF SO
  JMP     RSPXB    ;OTHERWISE, JUST GET NEXT PIXEL
RSPXF:
  IN      PORTA
  RLC              ;PUT SYNC INTEGRATOR BIT INTO CARRY
  JC      RSPXD    ;1=NOT SYNC, GET NEXT PIXEL
  INR     C        ;SYNC COUNT
  MOV     A,C
  CPI     30       ;VERT SYNC ?
  JNC     RSPXV    ;PROCESS VERT SYNC
;HORIZONTAL SYNC
  MOV     A,L      ;LOW PART OF PIXEL POINTER
  ADI     32       ;ADD HALF LINE BEFORE TRUNCATION
  JNC     RSPXK
  INR     H        ;IF ADD CAUSED CARRY
RSPXK:
  ANI     NOT 63   ;TRUNCATE TO START OF LINE
  MOV     L,A
  MOV     B,A      ;CLEAR LSB OF HI/LO FLAG
  JMP     RSPXB

;VERT SYNC
RSPXV:
  LDA     KBIN
  CPI     NOT (' ')        ;SPACE IS COMMAND TO STOP
  JNZ     RSPXA    ;START ANOTHER FRAME IF NOT TOLD TO STOP
RXEXIT:
  MVI     A,8FH    ;TRANSMIT MODE
  OUT     PORTC    ;I/O MODE SELECT
;SET OUTPUT TONE FOR SYNC
  MVI     A,208    ;1200 HZ
  OUT     CNT2
  RET

  END
```

APPENDIX C - POINT PROCESSING PROGRAM LISTING

```
XLAT:
  CALL     BLINKH  ;HIGH BLINK, RETURN KEYBOARD CHARACTER
  CPI      'U'     ;IF U , USE PREVIOUS TABLE
  JZ       XLAT3
  CPI      'B'     ; B IS FOR BRIGHTER
  JNZ      XLAT1   ;IF NOT B, GET NEW TABLE FROM OPERATOR
  CALL     XLB     ;FILL TABLE WITH VALUES TO ADD 1
  JMP      XLAT3
XLAT1:          ;GET 16 VALUES FROM KEYBOARD
  MVI      C,16
  LXI      H,XTABLE
XLAT2:
  CALL     HEXIN   ;GET HEX CHARACTER
  ANI      0FH     ;REMOVE UPPER BITS
  MOV      M,A     ;PUT INTO TABLE
  INX      H      ;NEXT TABLE LOCATION
  DCR      C      ;DONE WITH 16 ?
  JNZ      XLAT2   ;LOOP IF NOT
XLAT3:
  LXI      H,PXMEM        ;IMAGE MEMORY
  LXI      B,XTABLE       ;POINT TO LOOKUP TABLE
  MOV      D,C
XLAT4:
  XRA      A      ;CLEAR A
  RLD             ;ROTATE LOW PIXEL INTO A
  ADD      D      ;ADD LO TABLE BASE TO PIXEL
  MOV      C,A    ;BC NOW POINTS TO TABLE(PIXEL)
  LDAX     B      ;GET NEW PIXEL VALUE FROM TABLE
  RLD             ;NEW PIXEL TO IMAGE MEMORY, NEXT PIXEL OUT
  ADD      D      ;ADD AS ABOVE
  MOV      C,A
  LDAX     B      ;NEW PIXEL IN A
  RLD          ;NEW PIXEL TO IMAGE MEMORY
  INR      L      ;POINT TO NEXT 2 PIXELS
  JNZ      XLAT4   ;LOOP IF L IS NOT 0
  INR      H      ;INCREMENT HIGH BYTE OF PIXEL ADDRESS
  MOV      A,H
  CPI      HIGH(PXDUN)     ;TEST IF DONE
  JNZ      XLAT4          ;LOOP IF NOT DONE
  RET
```

APPENDIX C - CONTINUED

```
XLB:     ;CREATE TABLE TO ADD 1 TO EACH PIXEL VALUE
  LXI      H,BTABLE       ;SOURCE
  LXI      D,XTABLE       ;DESTINATION
  LXI      B,16          ;BYTE COUNT TO MOVE
  LDIR          ;MOVE TABLE DATA
  RET

;BTABLE- BRIGHTER TABLE- ADDS 1 TO EACH PIXEL VALUE
;ZERO BECOMES 1, 15 STAYS 15, NO WRAP AROUND
BTABLE:
  DB     1,  2,  3,  4,  5,  6,  7,  8
  DB     9, 10, 11, 12, 13, 14, 15, 15

END
```

APPENDIX D - CONVOLUTION PROGRAM LISTING

```
CONVLV:
   CALL     BLINKH  ;GET SECOND CHARACTER
CONVL2:
   CPI      'V'     ;V TWICE MEANS USE PREVIOUS COEFFS
   JZ       CNV2
   CPI      'E'     ;E - USE ROBERTS EDGE DETECT FUNCTION
   JZ       EDGDET
;IF NEITHER V NOR E, USE NUMBER TO SELECT A PRESET ARRAY
   SUI      30H
   RC
   RZ
   LXI      H,MTABLS-11
   LXI B,11         ;11 ENTRIES PER TABLE
;LOOP UNTIL PROPER TABLE ADDRESS IS FOUND
CNV1:
   DAD      B
   DCR      A
   JNZ      CNV1
;MOVE TABLE FROM ROM INTO RAM
   LXI      D,OFFSET        ;DEST
   LDIR
CNV2:
;SET UP POINTERS FOR WRITING BACK TO IMAGE
   LXI      H,PXMEM
   MVI      B,0
   XBDH
;GET FIRST LINE INTO BUFR2 AND BUFR3
   CALL     LINONE
CNV3:
   CALL     NXTLIN  ;SHIFT BUFFERS, GET NEXT LINE

CNV4:
;SET UP POINTERS FOR BEGINNING OF LINE
   LXIX     BUFR1   ;PIXEL POINTER
   MVI      C,128   ;PIXELS PER LINE

CNV5:
;CONVOLVE TO FIND ONE NEW PIXEL
   LHLD     OFFSET  ;INITIAL SUM IN HL
   LXIY     COEFS
   CALL     DO3MPY  ;TOP LINE
   CALL     DO3MPY  ;CENTER LINE
   CALL     DO3MPY  ;BOTTOM LINE
   DAD      H
   DAD      H
   DAD      H
   DAD      H
   DAD      H
```

APPENDIX D- CONTINUED

```
;LIMIT PIXEL VALUE FROM 0 TO 15
    MOV     A,H     ;HI BYTE OF SUM
    ORA     A
    JP      CNV6    ;GO AROUND IF POSITIVE
    XRA     A       ;MAKE ZERO IF VALUE WAS NEG
    JMP     CNV7
CNV6:
    CPI     16      ;TOO BIG ?
    JC      CNV7
    MVI     A,15    ;MAKE IT 15 IF LARGER
CNV7:
    XBDH    ;GET PIXEL OUTPUT POINTERS
    RRD     ;ROTATE PIXEL INTO MEMORY
    INR     B       ;UPPER / LOWER FLAG
    BIT     0,BR    ;IS LSB SET ?
    JNZ     CNV8    ;IF SET, DONT INX H
    INX     H
CNV8:
    XBDH            ;INPUT POINTERS
    LXI     D,-389  ;CORRECT BACK TO NEXT UPPER LEFT PIXEL
    DADX    DE
    DCR     C       ;DONE WITH LINE ?
    JNZ     CNV5    ;LOOP IF NOT

;DONE WITH LINE: TEST IF END OF PIX
    LHLD    LINADR
    LXI     D,64    ;MOVE DOWN ONE LINE
    DAD     D
    SHLD    LINADR
    MOV     A,H
    CPI     HIGH (PXDUN)
    JNZ     CNV3    ;LOOP FOR NEXT LINE
    MOV     A,L
    ORA     A       ;DO LAST LINE ?
    JZ      CNV4    ;YES, DO IT
    RET             ;BACK TO PXMON
```

APPENDIX D- CONTINUED

```
   DO3MPY: ;MAKES SUM OF PRODUCTS OF THREE COSECUTIVE PIXELS
   CALL    DO1MPY
   CALL    DO1MPY
   CALL    DO1MPY
   LXI     D,127    ;POINT TO NEXT LINE
   DADX    DE
   RET

DO1MPY: ;ADDS ONE PRODUCT TO SUM
   LDRX    AR,0     ;GET PIXEL FROM BUFFER MEMORY
   INXX
   LDRY    ER,0     ;GET COEFFICIENT FROM TABLE
   INXY
   ORA     A        ;TEST IF ZERO
   RZ
   MOV     B,A
   MOV     A,E
   ADD     A        ;TEST FOR ZERO,SET C IF NEG
   RZ
   SBB     A        ;ALL ONES IF CARRY SET, ZERO OTHERWISE
   MOV     D,A      ;SIGN EXTENSION OF E NOW IN D
MPY2:
   DAD     D
   DJNZ    MPY2
   RET

   END
```

APPENDIX E - NEIGHBORHOOD AVERAGE NOISE FILTER

```
NAYBOR: ;NEIGHBORHOOD AVERAGE NOISE FILTER
;SET UP POINTERS FOR WRITING BACK TO IMAGE
  LXI     H,PXMEM
  MVI     B,0
  XBDH
;GET FIRST LINE INTO BUFR2 AND BUFR3
  CALL    LINONE
NAY3:
  CALL    NXTLIN  ;SHIFT BUFFERS, GET NEXT LINE
NAY4:
  LXI     H,BUFR1 ;PIXEL POINTER
  MVI     C,128   ;PIXELS PER LINE
;LOOP FOR TESTING EACH PIXEL
NAY5:
  LXI     D,128   ;OFFSET VALUE TO MOVE DOWN ONE LINE
  MOV     A,M     ;UPPER LEFT PIXEL
  INX     H
  ADD     M       ;UPPER CENTER PIXEL
  INX     H
  ADD     M       ;UPPER RIGHT PIXEL
  DAD     D
  ADD     M       ;LEFT CENTER PIXEL
  INX     H
  MOV     B,M     ;CENTER PIXEL, SAVE IN B, DONT ADD TO SUM
  INX     H
  ADD     M       ;RIGHT CENTER PIXEL
  DAD     D
  ADD     M       ;LOWER LEFT PIXEL
  INX     H
  ADD     M       ;LOWER CENTER PIXEL
  INX     H
  ADD     M       ;LOWER RIGHT PIXEL

  LXI     D,-261  ;FOR NEXT PASS, CORRECT INPUT POINTER
  DAD     D
;AVERAGE = SUM/8
  SRL     AR      ;DIV BY 2
  SRL     AR      ;DIV BY 2
  SRL     AR      ;DIV BY 2 AGAIN
  MOV     D,A     ;SAVE AVERAGE
;A = AVERAGE-CENTERPIXEL
  SUB     B       ;SUBTRACT CENTER PIXEL VALUE
;IF A < 0 THEN A = - A
  JNC     NAY6    ;IF POS, DONT COMPLEMENT
  CMA             ;MAKE TWOS COMPLEMENT IF NEGATIVE
  INR     A
```

APPENDIX E -CONTINUED

```
NAY6:
;IF A <= DELTA THEN NEWPIXEL = CENTERPIXEL
; ELSE NEWPIXEL = AVERAGE
  CPI     5         ;SET CARRY IF LESS THAN 5
  MOV     A,B       ;ORIGINAL CENTER PIXEL VALUE
  JC      NAY7      ;WITHIN DELTA, RESTORE CENTER PIXEL VALUE
  MOV     A,D       ;AVERAGE OF NEIGHBORS IF OUTSIDE DELTA
NAY7:
  XBDH    ;POINTERS FOR OUTPUT PIXEL
  RRD     ;ROTATE PIXEL INTO MEMORY
  INR     B
  BIT     0,BR      ;IF UPPER HALF JUST DONE, INX H
  JNZ     NAY8
  INX     H
NAY8:
  XBDH
  DCR     C         ;DONE WITH LINE ?
  JNZ     NAY5      ;LOOP IF NOT

;TEST IF END OF PIX
  LHLD    LINADR
  LXI     D,64      ;MOVE DOWN ONE LINE
  DAD     D
  SHLD    LINADR
  MOV     A,H
  CPI     HIGH (PXDUN)
  JNZ     NAY3      ;LOOP FOR NEXT LINE
  MOV     A,L
  ORA     A         ;DO LAST LINE ?
  JZ      NAY4      ;YES, DO IT
  RET               ;BACK TO PXMON

  END
```