NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

THE UNIVERSITY OF ALBERTA

A VIDEO DIGITIZER FOR SCOLIOSIS STUDIES

BY

© STEVEN HAROLD SLUPSKY

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF: ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL, 1988

ISBN 0-315-45641-8

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: STEVEN HAROLD SLUPSKY

TITLE OF THESIS: A VIDEO DIGITIZER FOR SCOLIOSIS STUDIES

DEGREE: MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED: 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Steve Slupsky

#1201, 1 - Michener Park

Edmonton, Alberta

T6H 4N1

Date: June 6, 1988

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled A VIDEO DIGITIZER FOR SCOLIOSIS STUDIES submitted by STEVEN HAROLD SLUPSKY in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

M. B. Wunda

Supervisors

Date: May 30/88

To my wife, Cindy Lynn Slupsky ...

Abstract

Scoliosis is a deformity of the spine characterized by a lateral curvature. Research has indicated that topographic techniques can be used to describe the disorder and monitor its progression. Presently, this technique is not used by physicians for making theraputic decisions because of the length of time involved in generating quantitative results.

This thesis describes a video image acquisition system that can be used to reduce the time required to quantify topographic details of the trunk and aid in the diagnosis, monitoring and research of scoliosis. The system integrates the capability of large, expensive grey-scale image acquisition equipment into a small, low-cost diagnostic imaging tool using current state-of-the-art technologies and design techniques.

Acknowledgement

I would like to thank Doug Hill for his helpful suggestions and in the preparation of this thesis. I would like to thank my supervisors Nelson Durdle and Jim Raso, for their advice during the course of my graduate degree program. Finally, I would like to thank my wife for her support and help with proof reading this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A/D	Analog to Digital
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
DMA	Direct Memory Access
GPIA	General Purpose Interface Adaptor (IEEE-488)
GPIB	General Purpose Interface Bus (IEEE-488)
IEEE	Institute of Electrical and Electronic Engineers
K	Kilo (2^{10} = 1 024)
M	Mega (2^{20} = 1 048 576)
LSB	Least Significant Bit
LSI	Large Scale Integration
MSB	Most Significant Bit
MSI	Medium Scale Integration
ns	Nanosecond
PADIM	Peripheral Analog to Digital Interface Module
PTM	Programmable Timer Module
SYNC	Synchronization
uP	Microprocessor
us	Microsecond
VLSI	Very Large Scale Integration

1. Introduction

Scoliosis is a lateral deviation of the spine in the frontal plane accompanied by axial rotation of the vertebrae. Monitoring and measuring the degree of curvature of the spine is necessary to determine the treatment required. The internal spinal configuration can be accurately determined using radiographs. A radiograph is essential for characterizing the severity of the disorder; however, for the purposes of ongoing therapy where a physician may require weekly or monthly data on progression of the disorder, this method is unacceptable because of radiation exposure. A technique that employs ordinary light to generate a moire fringe pattern to derive surface topographic information is used in some centres. This technique has a promising future as a diagnostic/monitoring tool for scoliosis but presently is limited because of the time required to produce quantitative results.

One of the mysteries of spinal deformity is how someone with a mild scoliosis may exhibit a very severe trunk deformity while someone else with a severe scoliosis may possess only a mild to moderate degree of deformity. Generally, this cosmetic deformity is the major concern of

those affected and causes them to seek corrective measures.

The use of moire fringes for topographic analysis of the human spine is a growing field of biomedical research. Moire[1] fringe patterns are a form of structured light that are generated by passing light through a grating and onto a surface. Elevations and depressions of the surface of the back cause the lines to distort and take on a wavy appearance. The moire fringe pattern can represent three dimensional topographic information in two dimensions (Real [2]). For scoliosis diagnosis, one basic method used to generate a moire pattern involves projecting a line pattern onto the back of a subject and this combines with a grating of equal spatial frequency to create a moire pattern. Then, using a system to digitize the image, image processing techniques can aid in the detection and diagnosis of scoliosis. Another topographic technique proposed by researchers (Stokes, et al, [26]) is to project a line pattern onto the back, digitize the image and process this directly bypassing the generation of a moire fringe pattern. All the necessary topographic details are present with the line pattern, and applying a geometric transformation to the image will provide a method of constructing a contour map of the back.

It is the purpose of this thesis to describe a system

---

1 The word Moire originated in the French language and is an adjective that means to have a wavy or watery appearance.

of hardware and software to aid with digital acquisition of video images; in particular, the image of the deformed trunk of a child with scoliosis. The design and development of the video digitizer was motivated by the need for high speed acquisition of video images to reduce the time required to quantify the topography of the back. The current system used for scoliosis research at the Glenrose Hospital in Edmonton, Alberta, employs a photographic method to obtain a line pattern image and then, with the aid of a digitizing tablet, digitizes the pattern. Additional image processing techniques are used to determine the topography of the trunk and the relationship of the spine to the topography.

Inaccuracies of such a time consuming system can affect the results substantially. To minimize the time required to obtain an image, a system has been developed whereby a 512 x 484 pixel image is collected in real time using a video camera and a microprocessor based digitizer. The digitized image is then transferred to a host computer through an instrumentation bus where it can be further processed (Figure 1.1).

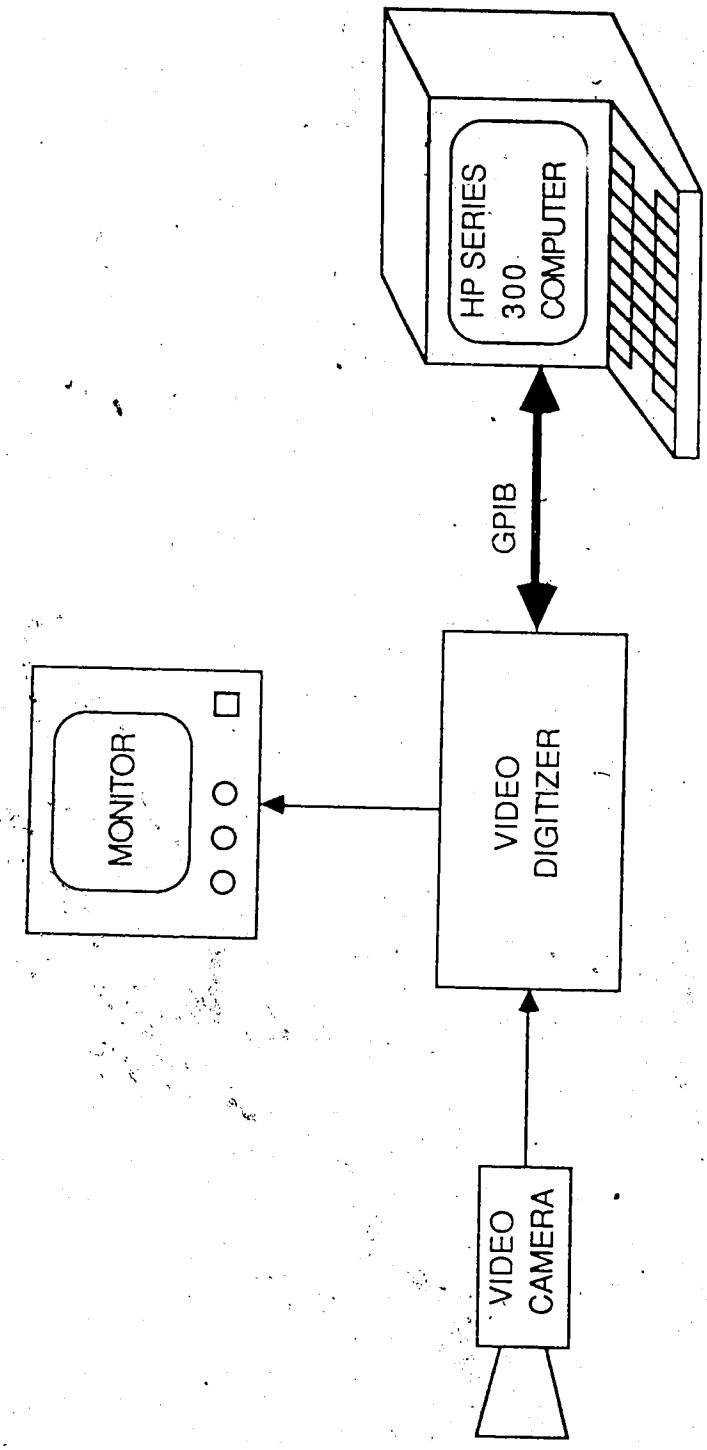


Figure 1.1

Block Diagram of System.

2. System Hardware

2.1 Overview

The hardware system described herein was devised to capture a video frame and store it in digital memory where a computer could extract the digital information and reconstruct the original image. The hardware must synchronize with a standard video frame format by conditioning the video signal and extracting the synchronization information. The system must then control an analog sampling interface that samples the video information in real time and stores the information in digital memory. Finally, the hardware must provide an interface to a host computer. Figure 2.1 illustrates a block diagram of the digitizer.

2.1.1 Composite Video Signals

The objective is to be able to digitize a standard NTSC monochrome video frame made up of two 262.5 line fields. One field of a video frame represents 262.5 lines scanned from top to bottom. The second field represents another 262.5 lines scanned from top to bottom; however, the lines of the second field fall between the lines of the first. This is

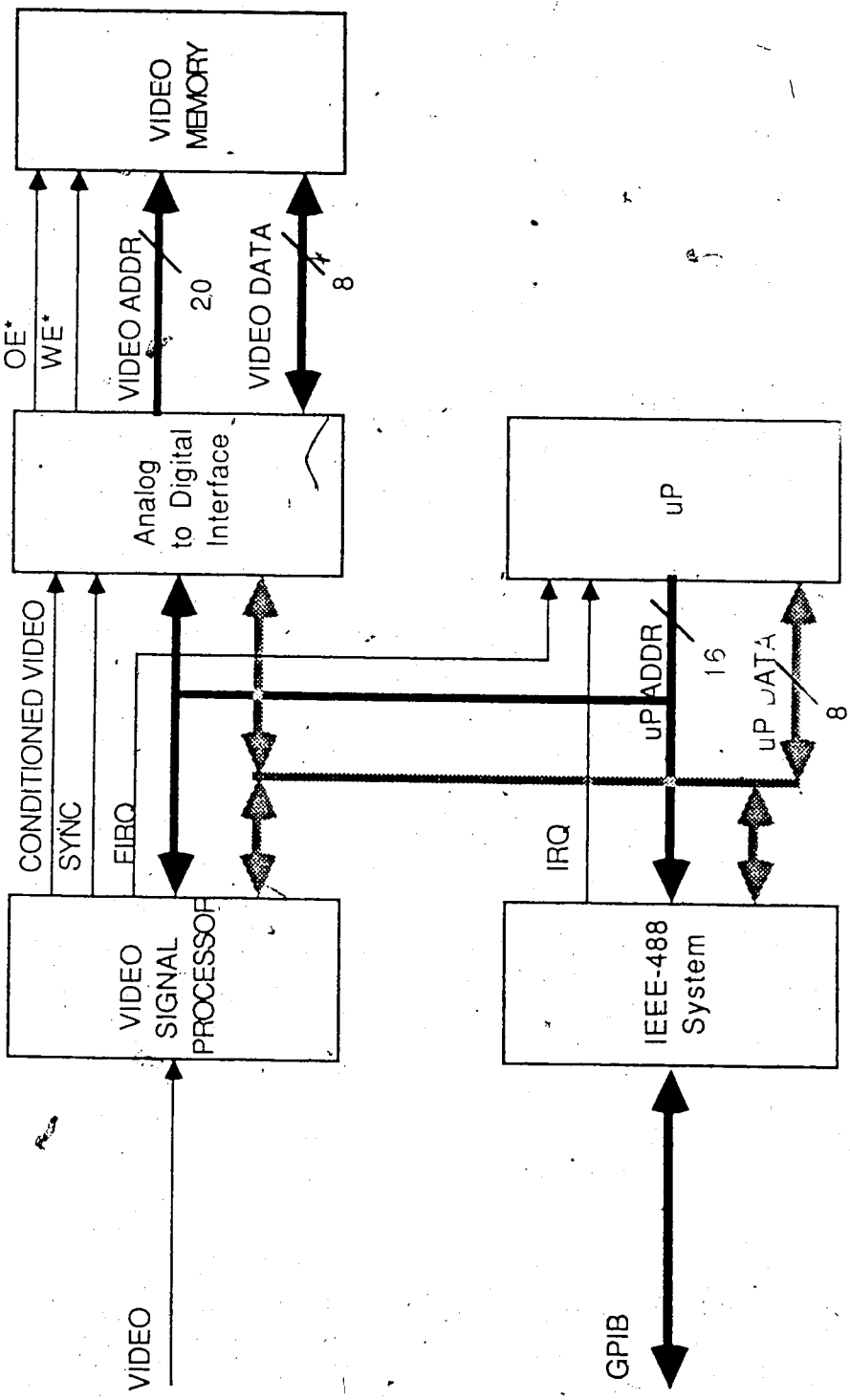


Figure 2.1

Block Diagram of Video Digitizer.

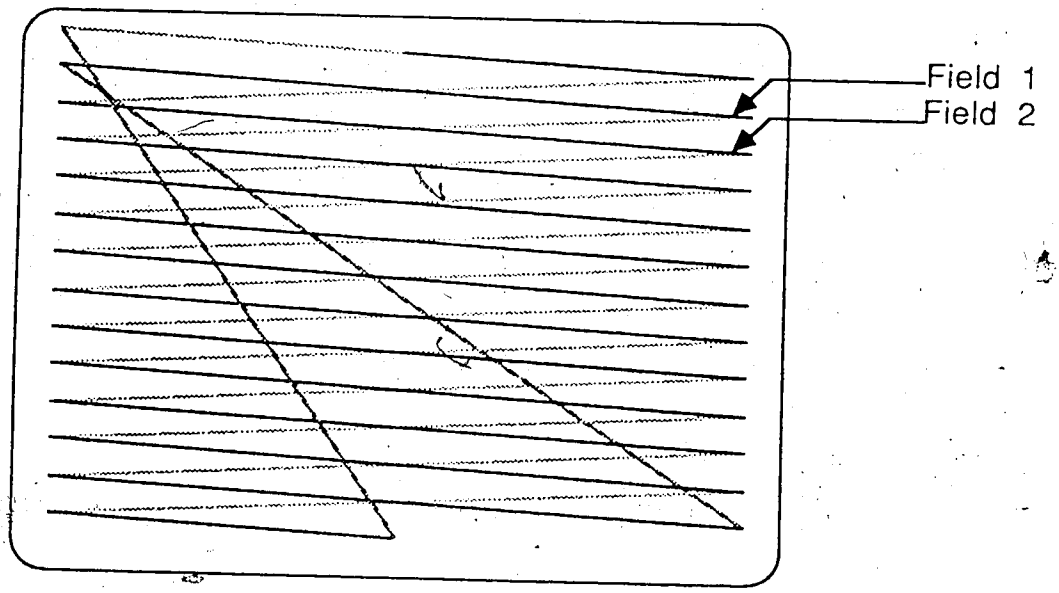


Figure 2.2

Video Frame (Interlacing)

called an interlaced video frame (Figure 2.2).

Referring to Figures 2.3 and 2.4, the video signal has three major components: the horizontal synchronization pulse, the vertical synchronization pulse, and the video information itself.

To scan a line, the scanning beam moves from the left to the right of the image. When the beam reaches the right most edge of the image, it is turned off and the beam retraces back to the left most edge of the image as a result of the horizontal synchronization pulse.

At the end of each field, there is a vertical synchronization sequence that restores the scanning beam to the top of the picture (vertical retrace). This sequence is also responsible for synchronizing the two fields so that the lines of the second lie between the lines of the first.

To digitally represent the video frame, only the video information is digitized. This process involves synchronizing the sampling process with the inherent synchronization of the video signal. There are three levels of synchronization: horizontal, vertical and field synchronization. The first two synchronization levels have corresponding synchronization pulses built into the video signal (Figure 2.4). The field synchronization requires a detailed analysis of the difference between the two fields of an interlaced video frame.

There are two noticeable differences between the two

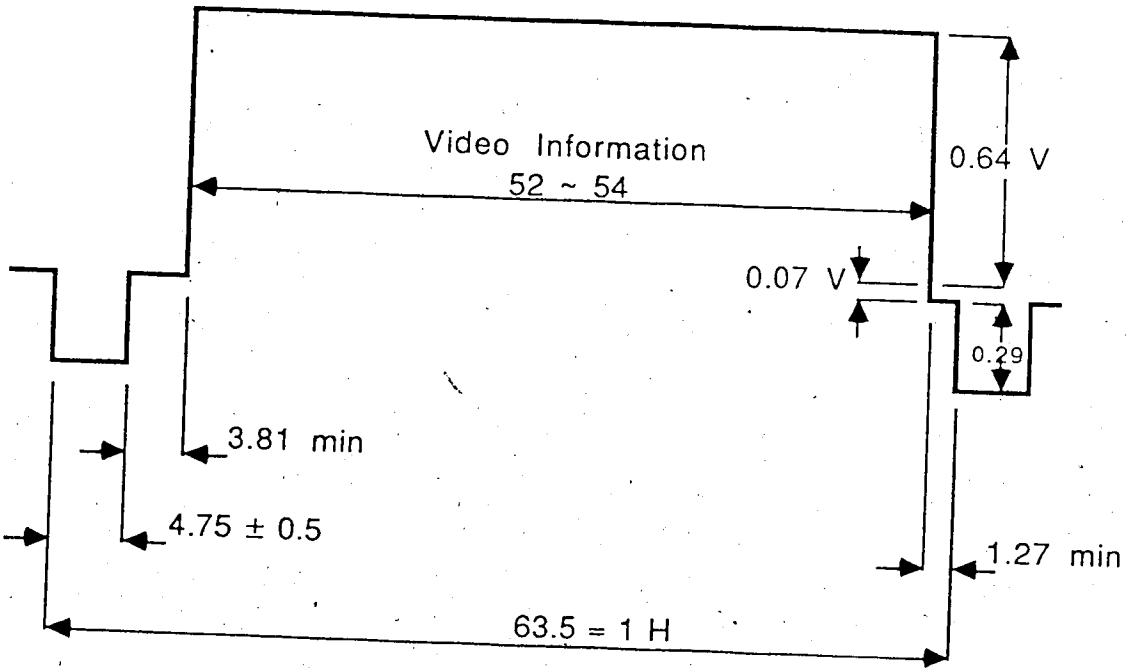
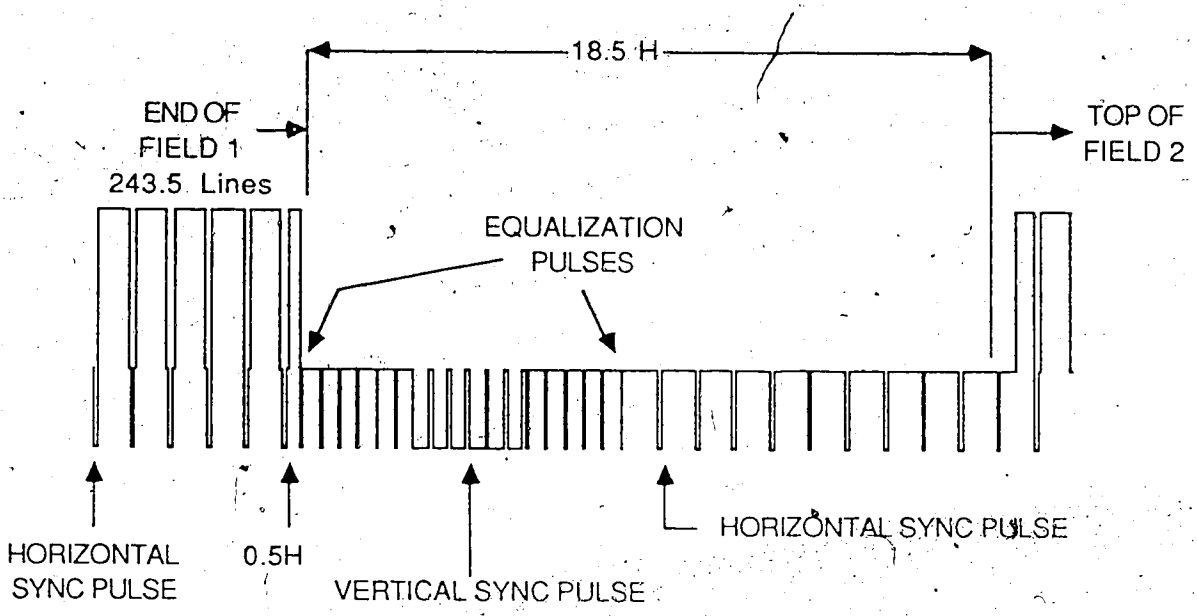
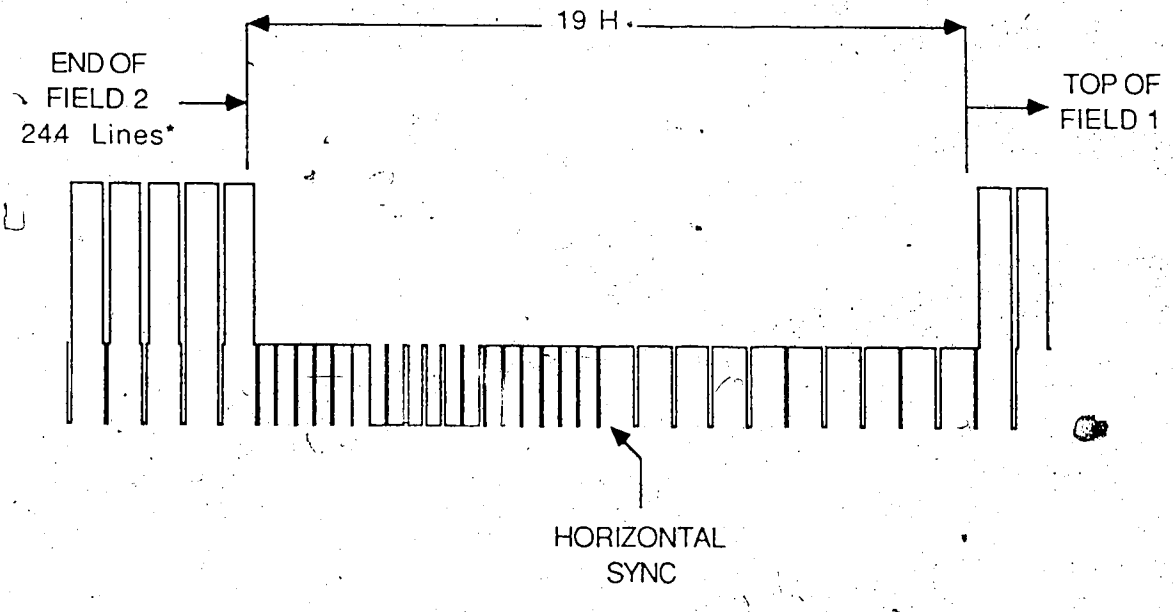


Figure 2.3

Compo_ite Video Line





* Note: Only 243.5 of the 244 lines are visible.

Figure 2.4

Vertical Retrace of Both Fields

fields concerning synchronization signals near or during the vertical synchronization pulse. First, the duration of the vertical synchronization sequence at the end of the first field is 18.5 H (1 H = 63.5 us which represents the period of one horizontal line) and 19 H at the end of the second. The first horizontal synchronization pulse to occur after the vertical synchronization pulse in the first field is 3.5 H after the vertical sync whereas it is 3 H after the vertical sync in the second field. The second difference is between the last visible lines of video information with respect to horizontal and equalization synchronization pulses of the two fields (Figure 2.4). Distinguishing fields is accomplished by recognizing that at the end of the first field, the last line is only 0.5 H in duration. The latter method of detection is used because it requires less hardware.

The next step is to consider a mechanism for digitizing the video information.

2.1.2 Communications Requirements

At the onset of this project, it was decided that the video digitizer would interface with an existing Hewlett Packard Model 300 computer system. The HP computer system provided the capability of communication through an RS-232 serial communications port or an IEEE-488 instrumentation bus.

Consideration was given to both standards. RS-232, although universal among personal computers, was rejected because of the large volume of data required to be transferred between the video digitizer and the computer system. An RS-232 port could not meet the required speed of data transfer. The IEEE-488 instrumentation bus standard has high speed data transfer capability and, while not universally incorporated as a standard feature, is available on most personal computer systems as an add-on peripheral. These features contributed to overcoming concern regarding increased hardware and software complexity in realizing the IEEE-488 standard.

2.2 Microprocessor

The video digitizer uses the MC68B09 microprocessor to control the hardware (Figure 2.5). The MC6809 series of microprocessor was chosen because of familiarity. The main tasks of the microprocessor are the coordination of data transfer and system control, so there is no requirement for a more powerful processor. The microprocessor does not perform the actual data transfer but programs the hardware (Analog to Digital Interface) to accomplish this. This is because of the high data transfer rates required to sample the video signal in real time and maintain the desired resolution. It is too time consuming to have a microprocessor controlling the A/D converter, reading its

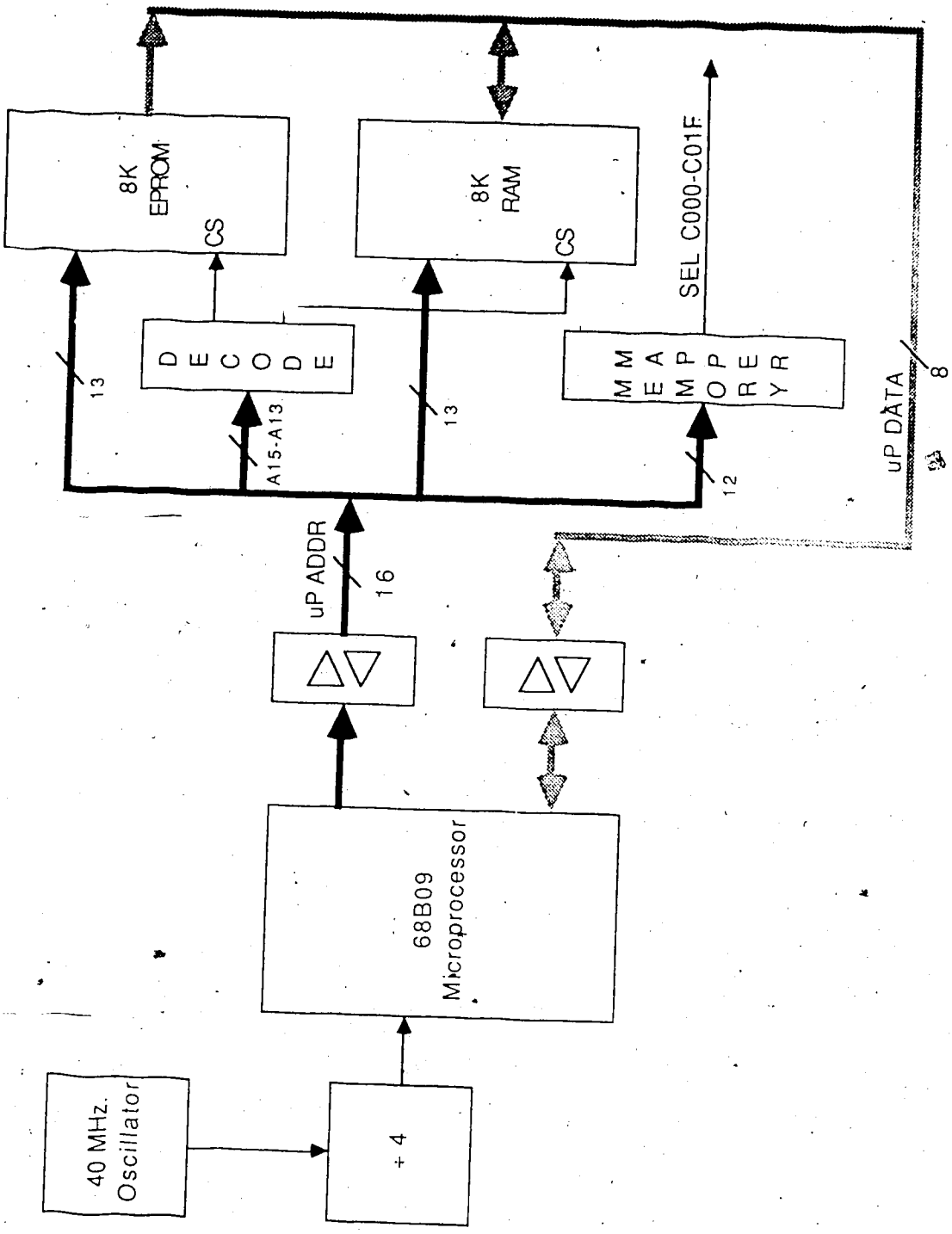


Figure 2.5

Block Diagram of Microprocessor.

data and storing it in memory.

The 'B' version of the 6809 (MC68B09) was used because it is faster than the other versions of the 6809 (2 MHz v.s. 1 MHz). A 2764 (8K x 8) EPROM is used to store the MC68B09's program and a 6264 (8K x 8) RAM is reserved for the microprocessor. An MC68B40 Programmable Timer Module is used to detect and measure the video synchronization pulse and timing signals. The 74ALS245 buffers are used to buffer the outputs of the 68B09.

The 68B09 oscillator (EXTAL) input is driven by a 10 MHz clock derived from a 40 MHz oscillator and a divide-by-four circuit. The 68B09's internal circuitry divides the 10 MHz clock by 4 to 2.5 MHz, thus giving a microprocessor cycle time of 400 ns.

2.3 A/D Acquisition Interface

The A/D acquisition interface is the heart of the digitizer. Consisting of an A/D converter, reference voltage circuitry, and an Application Specific Integrated Circuit (ASIC), this interface controls the flow of data from the A/D converter to the video memory (Figure 2.6).

2.3.1 A/D Converter

The analog to digital conversion circuitry must be able to digitize a 512 x 484 pixel video image in real time. The video information portion of a video line is about 52 us.

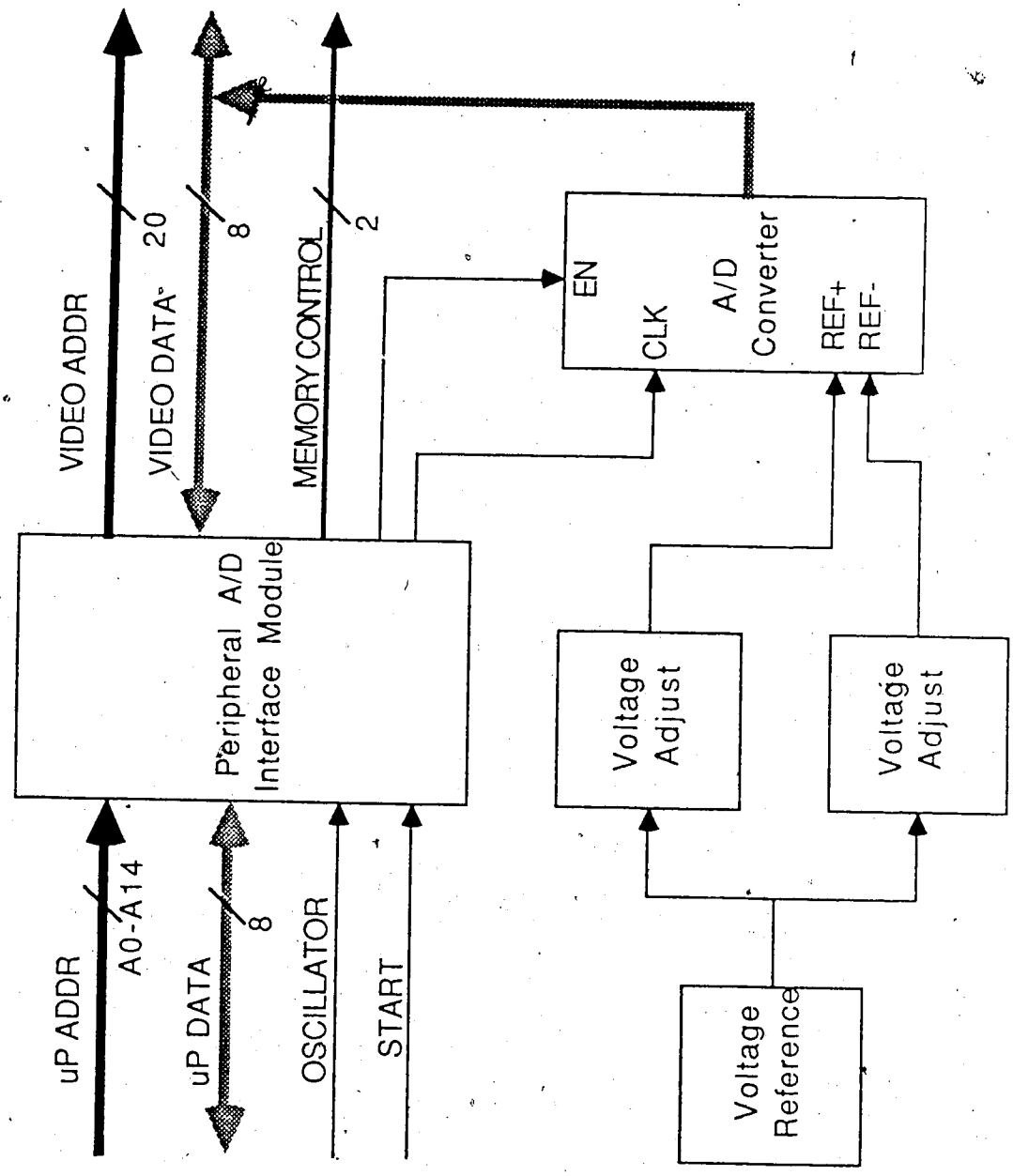


Figure 2.6

Block Diagram of Analog to Digital Interface

To acquire 512 pixels for one video line in 52 us requires a conversion rate of 10 MHz. A flash A/D converter must be used to overcome this speed constraint. An 'n' bit flash converter uses an internal chain of 2^n voltage comparators, each with its own reference voltage, to quantize the signal. The resulting quantized signal is then encoded into an 'n' bit binary number. This requires only one clock cycle to perform a conversion while other types, such as successive approximation, require several cycles.

To account for such aspects as varying lighting conditions, low contrast images and spatially variant contrast images due to skin reflectivity, 256 quantization levels (8 bits) are required to sufficiently represent the video signal.

2.3.2 Peripheral Analog to Digital Interface Module

The Peripheral A/D interface module (PADIM) controls the flow of data from the high speed A/D converter to video memory (Figures 2.7 and 2.8). In essence, it is a high speed, 10 megabyte per second, direct memory access (DMA) controller. The module is programmed with the base address of the memory transfer and the number of bytes to be transferred. In this application, the device is programmed with the above information for every video line (Figure 2.9). Therefore, the sampling process can be synchronized to the start of video information and proceed for a fixed

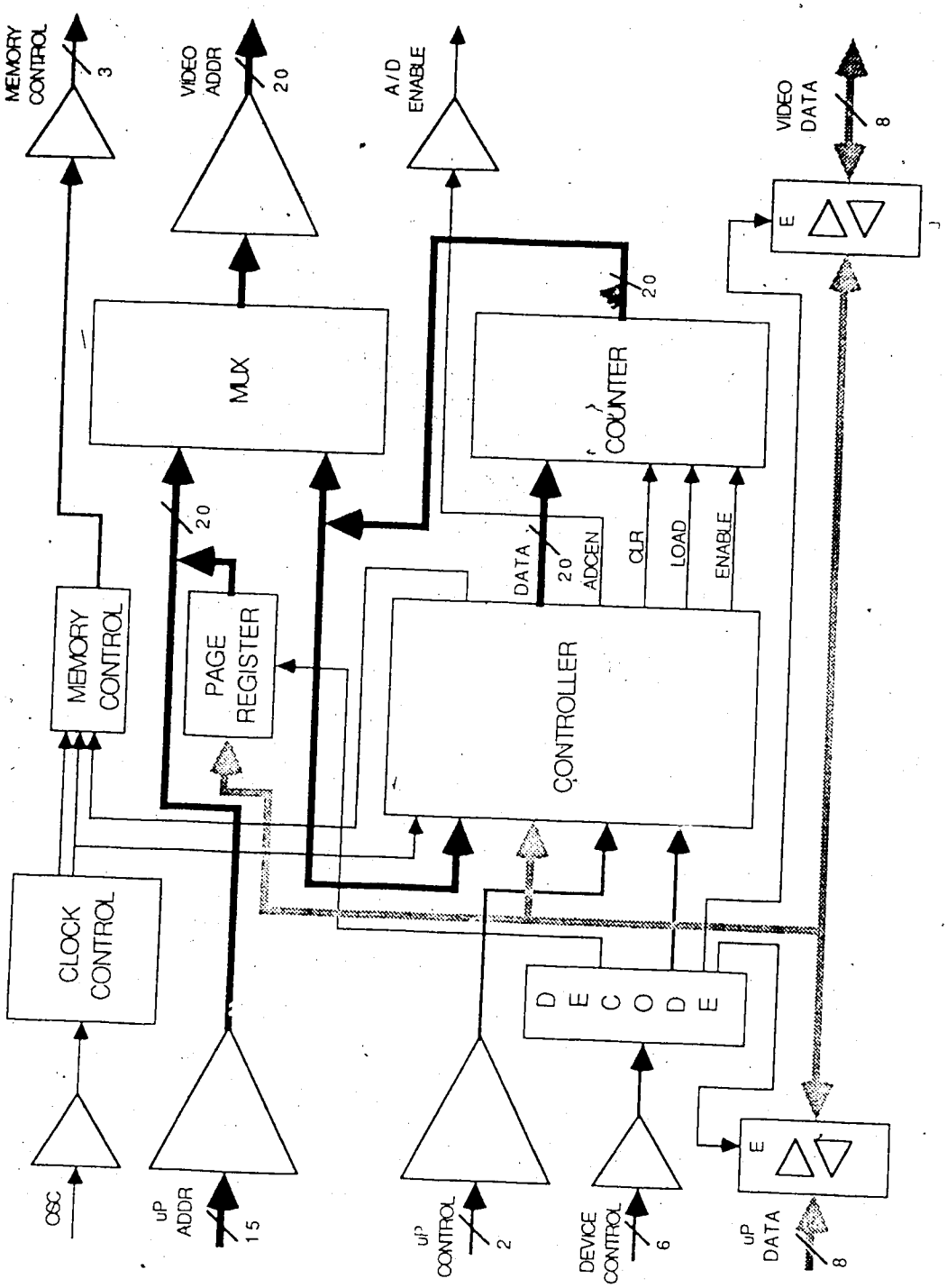


Figure 2.7

Peripheral Analog to Digital Interface Module.

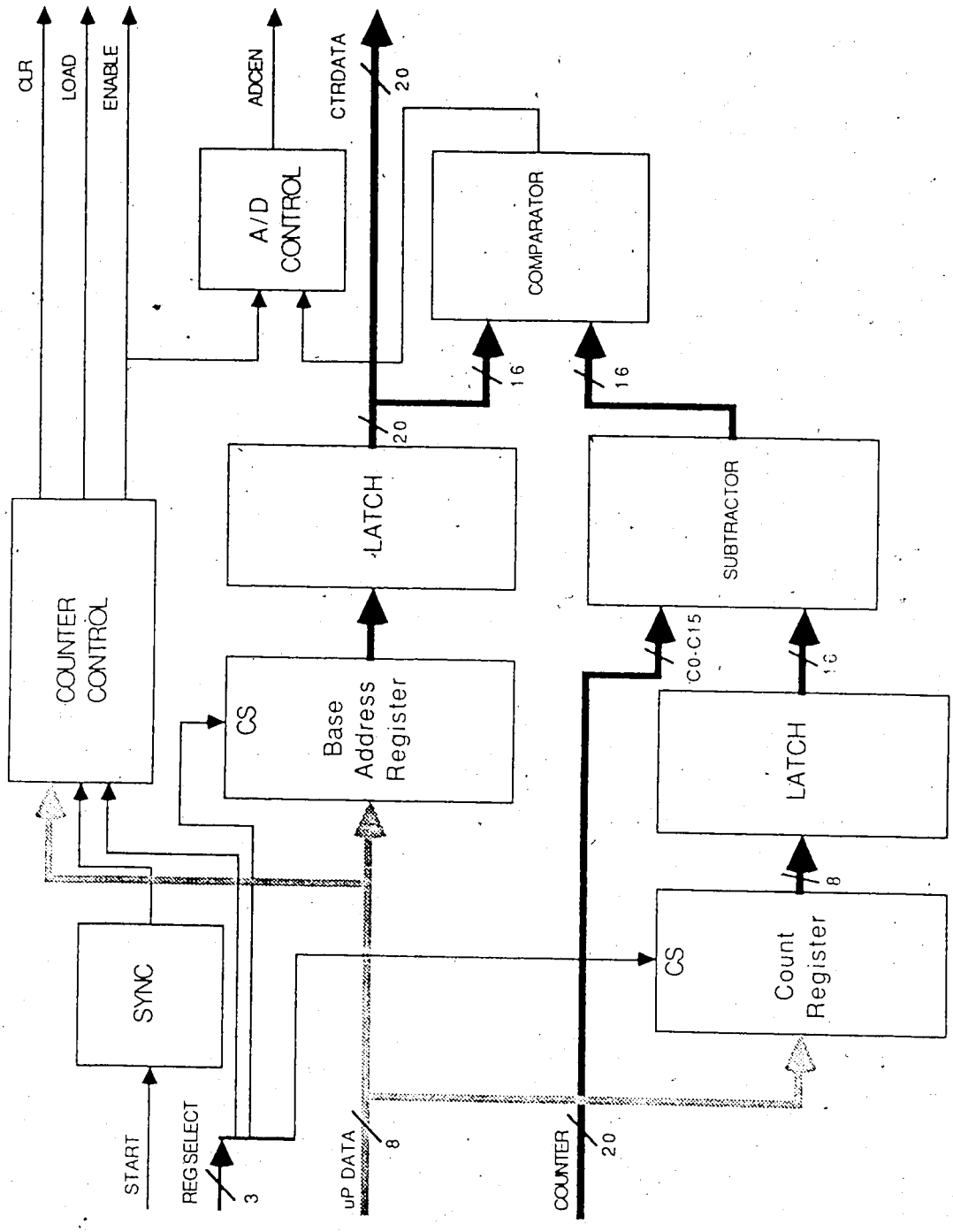


Figure 2.8

Block Diagram of Controller.

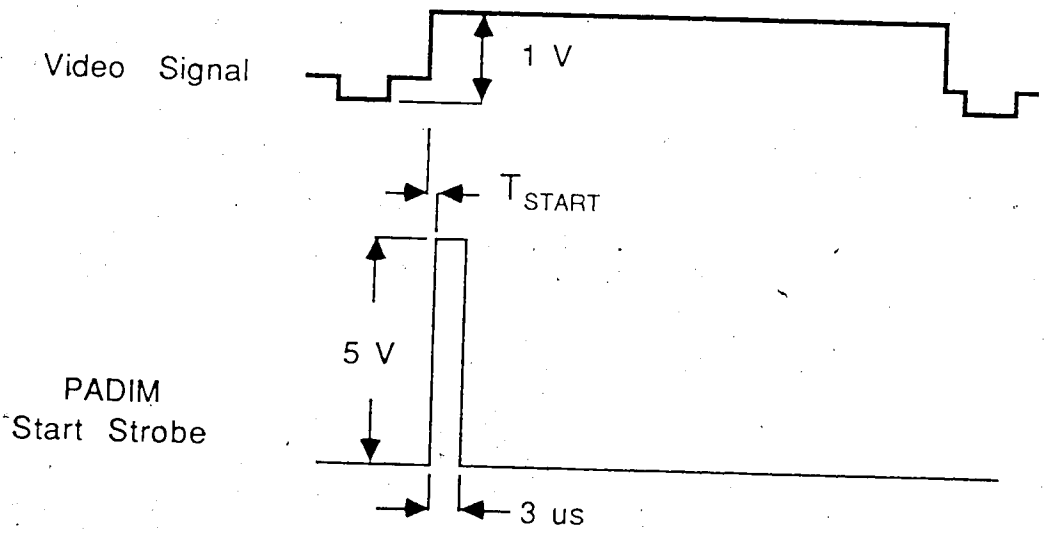


Figure 2.9

Video Line Sample Timing

time. This eliminates the sampling of the synchronization pulses.

The interface module also provides an interface between the microprocessor and the video memory. This interface is provided through a 32K memory block in the microprocessor address space. The microprocessor programs the interface module with the appropriate page to access all video memory. The video memory address space of the PADIM was designed to accommodate up to 1 megabyte of RAM for expandability purposes. In this application, only 256K of this address space is occupied by RAM.

Another feature of the interface module is the ability to program a delay register that accommodates the pipelining effect of the A/D converter. When the first sample is taken, the encoded binary number may not appear immediately, but at some fixed number of samples later, at the output of the A/D converter. The interface module can accommodate a delay of 1 to 8 sample periods.

2.3.2.1 Discrete MSI vs ASIC Gate Array

Three methods of implementing the PADIM were considered: (1) implementation using discrete off-the-shelf medium scale integration (MSI) components, (2) a full custom CMOS integrated circuit, or (3) an application specific integrated circuit (ASIC) gate array.

2.3.2.1.1 Design Considerations

Implementation of the three methods were evaluated.

A). MSI:

A preliminary design was prepared using MSI components to evaluate its feasibility. The PADIM consisted of an estimated 40 MSI components, 652 pins and a board area of about 130 cm^2. This yielded a total chip count of about 80 chips to implement the entire digitizer.

B). ASIC:

An ASIC gate array, while increasing design time, does provide several advantages. The total chip count can be halved to a more reasonable 39. Power consumption is reduced by an appreciable amount and, due to the reduced pin count, reliability increased and board area was significantly reduced to 10.9 cm^2. The gate array is, however, not without its disadvantages; some of these are increased design time, increased cost, and no design changes are possible once the circuit is fabricated in silicon.

C). Custom CMOS:

The full custom implementation has advantages with respect to increased security, but has a very long and complex design time, a very large cost for low volume production, and no further design changes can be made once the circuit has been fabricated in silicon.

After careful consideration of the available resources, it was decided that the gate array approach best suited the

needs for the project.

2.3.2.1.2 Design Procedures

LSI Logic's design procedures and macrocell manuals were studied. These manuals outline the design procedure from paper to reticle and provide a description of their software programs (ie: LSED, LCMP, LVER, LSIM, LTOG, LCHK, and LTEST; Figure 2.10).

The first stage in development of a gate array is the initial design layout. Using a schematic editor (such as LSI Logics 'LSED') a design and a list of the network interconnections, called a netlist, is generated. This netlist information is used later to electronically breadboard or simulate the circuit behaviour.

After design layout, the netlist must be compiled into the necessary format for the network simulator. This compilation process analyzes the network parameters such as fanout and fanin. Information on these network parameters is used to generate delay information necessary for simulation. If there is an error in the design, such as very long rise times, it is reported at this stage in design. If such an error occurs, the affected area of the circuit must be redesigned.

After network compilation, the design is subject to an electrical verification process ('LVER' program). This program checks the electrical feasibility of

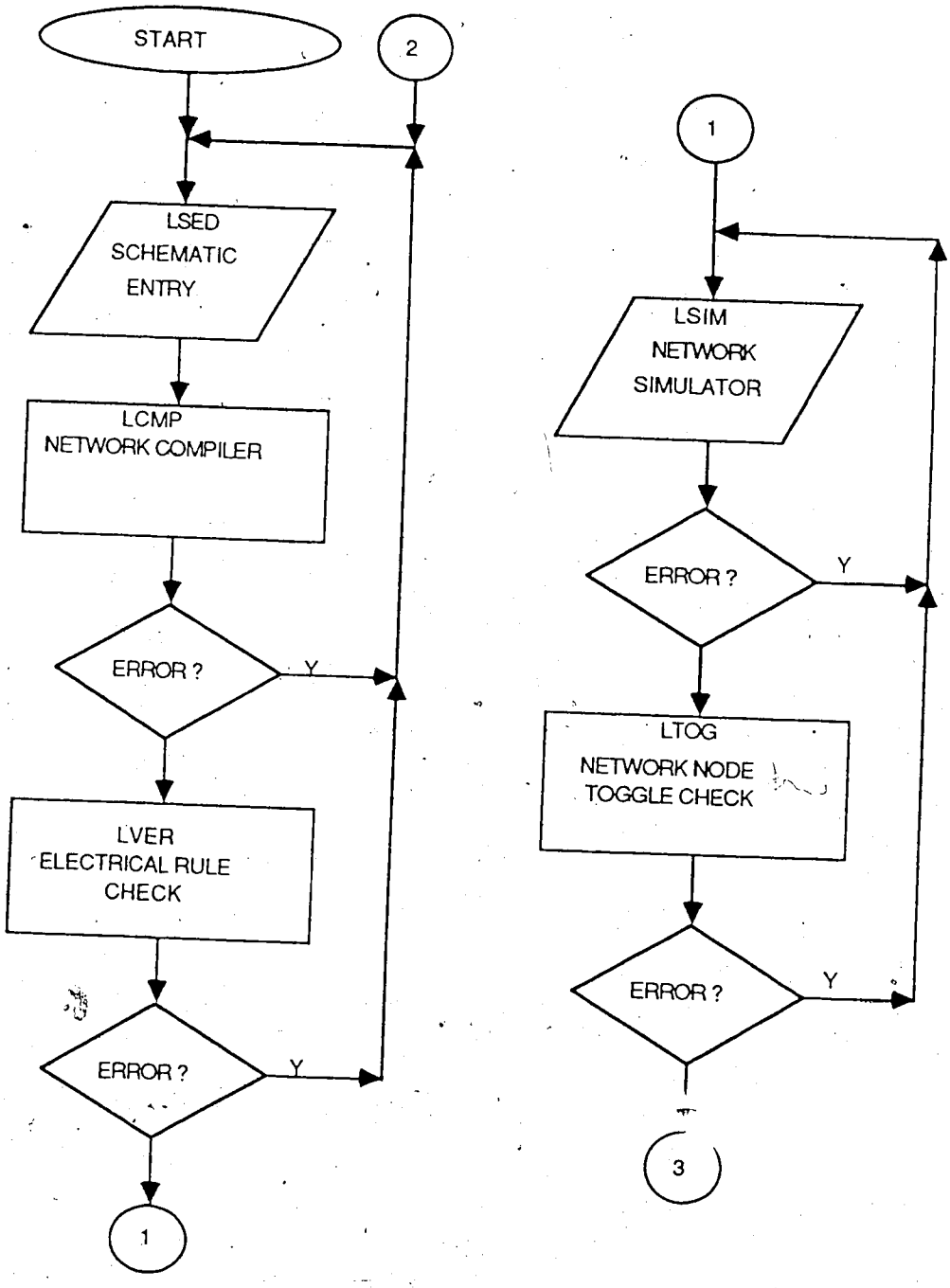


Figure 2.10

Design Procedure Flowchart.

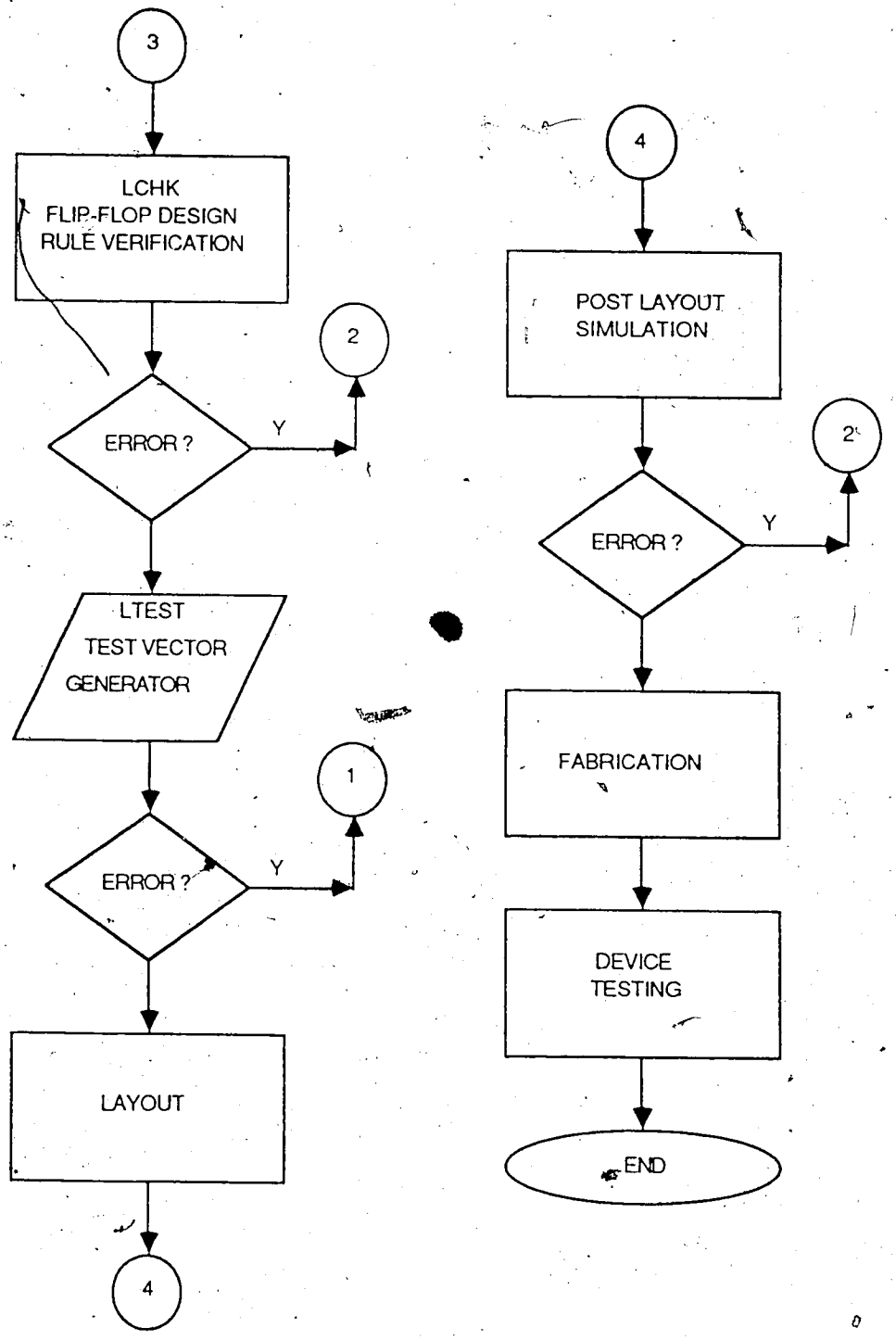


Figure 2.10 (cont).

Design Procedure Flowchart

interconnections between cells and also verifies that the gates and pads used in the design are available for the array type being used. Again, if an error occurs, it must be corrected and the netlist regenerated and compiled.

The design is now ready to undergo simulation. The simulator uses the compiled network and a simulation control language (SCL) program. The SCL program verifies network performance. If, during simulation, an error occurs due to SCL program syntax, the SCL program must be corrected and the network re-simulated. If the error indicates a design problem, then the design procedure continues with necessary modifications to the design layout.

When the simulation is complete, the results are analyzed for toggle information (LTOG). This program checks that every node in the network has been toggled to ensure that 100% testability[2] has been achieved. However, testability must have been designed into the network initially. If toggle analysis is less than 100%, the designer must supply additional test vectors to the SCL program and re-simulate.

The next step is to check for design rule violations (LCHK program). This software analyzes the network for errors such as set-up and hold time violations on flip-flops. If a violation occurs, the net and element involved

---

[2] Testability refers to the ability of the tester, a device that tests post fabrication circuit performance, to verify every electrical interconnection of the gate array.

are reported, and the design must be modified.

When the circuit has been successfully simulated, checked for design rule violations, and a set of test vectors produced that achieve 100% node toggle coverage, the bonding diagram and bonding command file are generated. This includes the necessary information to make the interconnections between the die and package.

The LTEST program is the next phase in development. This program extracts test vectors from the simulation control programs and generates a test program for the automatic tester. If an error occurs, it may be necessary to modify the simulation control programs or the design itself.

The design is laid out after the test program is complete. This generates the actual cell interconnection information for the reticles. After layout, the actual wire lengths are computed and used in a post-layout simulation. This involves executing the simulation, design rule check and test program generation programs and verifying the proper operation of the design.

When the device design procedure is complete, the array is fabricated and the chip is subjected to design testing. This involves the automatic tester and the test program generated by the LTEST program.

2.3.2.2 Gate Array Design

A block diagram of the gate array is shown in Figures 2.7 and 2.8. The gate array consists of four major blocks: the CLOCK CONTROL, MUX, COUNTER, and CONTROLLER.

The purpose of the gate array is to transfer data from the A/D converter to the video memory system. Operation of the device is as follows. Before a transfer occurs, the device must be programmed with the appropriate memory address of where the memory transfer is to occur (Base Address) and the amount of data to transfer (Count). The device can be triggered asynchronously by an external source or synchronously by the microprocessor. Once the PADIM has been triggered, the A/D converter is enabled and memory addresses are generated by the device in synchronization with conversions of the A/D converter. During this process, the PADIM generates the necessary RAM control signals to write the result of a conversion into memory.

The COUNTER module was designed to generate the memory addresses. This module is controlled by the CONTROLLER and was designed to operate at the required speed of 10 MHz.

The heart of the PADIM is the CONTROLLER module (Figure 2.8). This module coordinates the activity of the MUX and COUNTER modules. It contains the Base Address Register, the Count Register and the logic required to monitor and control the COUNT module. It also controls logic to synchronize an external asynchronous event to the

CONTROLLER. Upon receiving a signal to start the memory transfer, the CONTROLLER initializes the COUNTER module, selects the COUNTER module for output via the MUX module, and enables the A/D and RAM control signals. As the transfer progresses, the CONTROLLER module monitors the output of the COUNTER module. When the CONTROLLER detects that the required number of bytes have been transferred, the COUNTER is stopped and the A/D and RAM control signals are disabled.

The MUX module controls the address appearing on the video memory address lines. The MUX selects between the address produced by the microprocessor address line and the page register or the output of the COUNTER module.

The CLOCK module was designed to generate the necessary timing for the controller and RAM control signals. A 10 MHz clock source is required because the data transf rate is 10 MHz. The RAM access time is 10(s and the write strobe time is 70ns, therefore a write strobe with a duty cycle of 25% would satisfy the RAM access requirements. To accomplish this, a 40 MHz external source was used and the internal circuitry of the clock module generates a 50% duty cycle 10 MHz clock and a 25% duty cycle 10 MHz signal using shift registers.

Schematic diagrams of the gate array are shown in appendix E.

2.3.2.3 Gate Array Implementation

When the gate array had been successfully designed and simulated, the device was fabricated. This process involved sending the layout information to LSI Logic to have the reticles made following which the actual fabrication and packaging process took place.

Subsequently, the peripheral A/D interface module was implemented into the video digitizer system board. The necessary connections were made to the address, data, and control buses of the system.

2.4 Video memory

The video memory subsystem is comprised of 256K of 100ns CMOS static ram and an address decoder (74F138). To accommodate the high speed data transfer (10 MHz), it was necessary to use high speed ram. Static RAM was chosen for its ease of use and cost which is comparable to high speed dynamic RAM.

The memory decoder circuitry consists of a 74F138. This is used to decode the 256K memory address space into 32K blocks. Each static RAM module has a 32K by 8 configuration.

2.5 Video signal processor

The video signal processor conditions the incoming

video signal and generates synchronization and timing signals (Figure 2.11). The signal conditioning provides a low pass filtered video signal to the A/D converter to reduce aliasing effects. The signal also drives circuitry that extracts synchronization pulses. These pulses are routed to a timer module that performs pulse width measurements to determine the type of synchronization pulse.

2.5.1 Video signal conditioning

The video signal is initially level shifted using a diode clamping circuit to reduce and provide a constant dc offset. The signal is low pass filtered (3dB frequency = 1.59 MHz < Nyquist Frequency of 5 MHz) to reduce aliasing and then buffered through a LF357 op-amp. After buffering, the synchronization pulses are separated using a LM339 voltage comparator to generate a composite synchronization signal. From this signal, the horizontal synchronization pulses are separated by a differentiator and voltage comparator circuit.

2.5.2 Timing

Special circuitry was required to generate a pulse to indicate the start of a video line (Figure 2.12). The circuitry accommodates for the blanking delay of the video signal. The blanking delay circuit, consisting of a 74LS221 multivibrator plus discrete capacitors and resistors and a

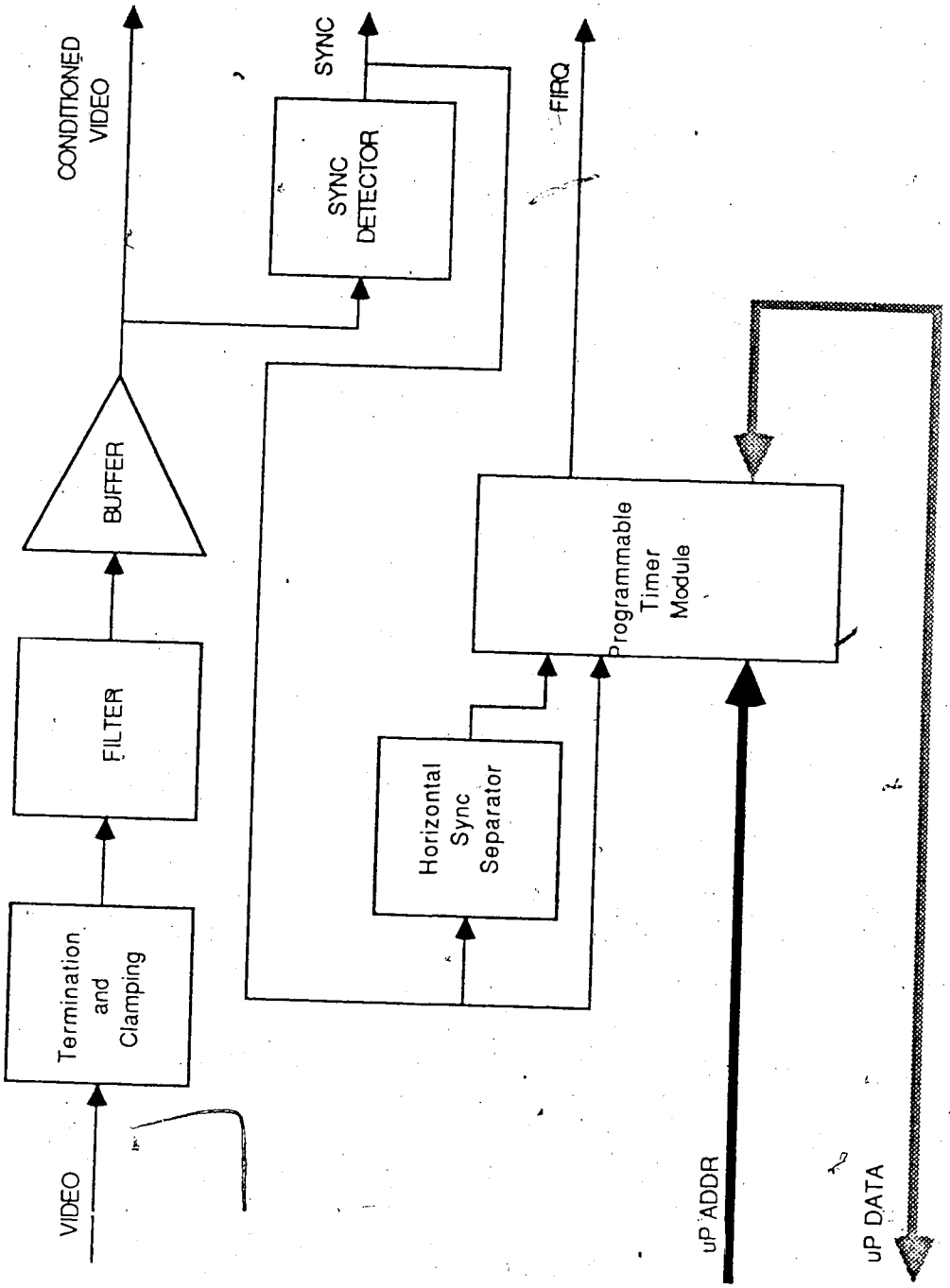


Figure 2.11

Block Diagram of Video Signal Processor

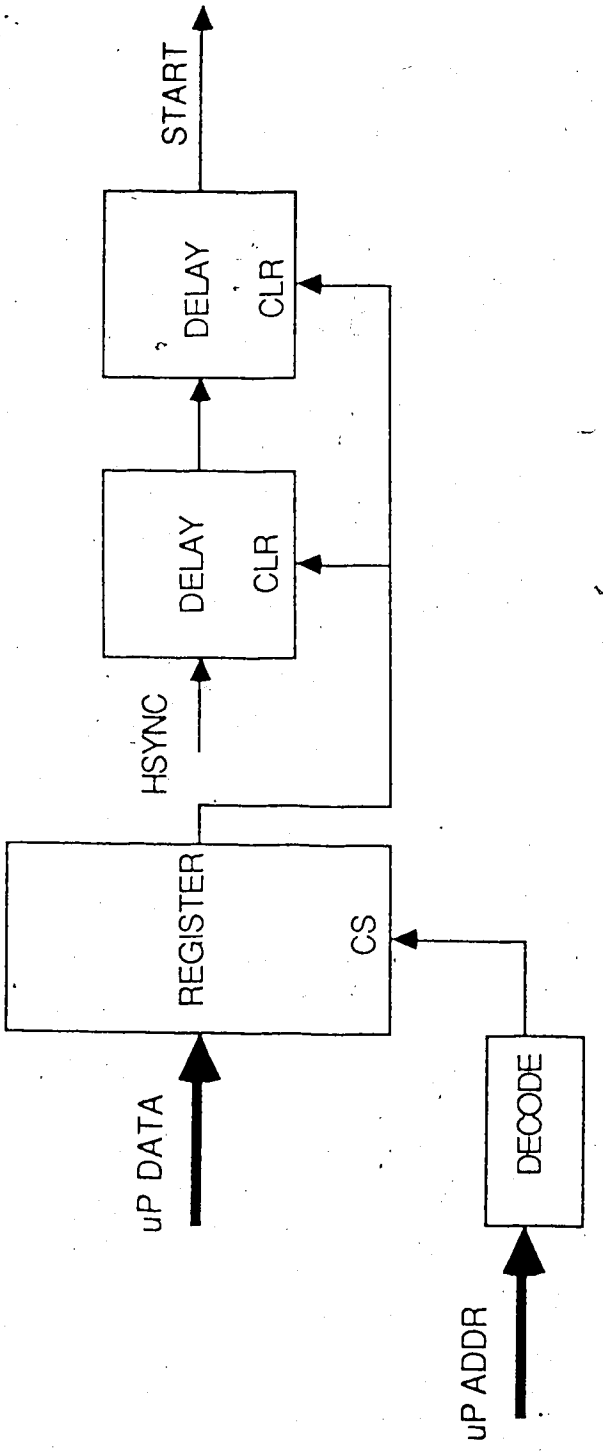


Figure 2.12

Block Diagram of Blanking Delay Circuit.

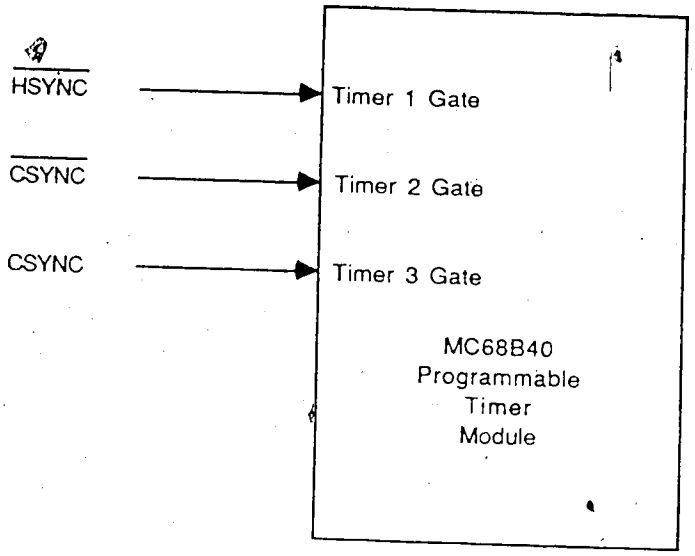
74ALS574 register, synchronizes with the horizontal synchronization pulse to generate the start signal.

The composite and horizontal synchronization signals are connected to the gate inputs of a MC68B40 programmable timing module (PTM). The PTM measures pulse widths and thus discerns between vertical, horizontal and equalization synchronization pulses. Figure 2.13 (PTM connection and programming details) details the programming model for the PTM.

2.6 Communications Subsystem (IEEE-488)

The IEEE-488 interface is built around the Motorola MC68B488 General Purpose Interface Adapter (GPIA) controller chip (Figure 2.14). This chip performs handshaking and protocol administration required by the standard. The bus is connected to the GPIA through two MC3447 bus transceivers. These transceivers have built in GPIB bus terminations and interface logic to interface with the GPIA.

A DIP switch connected to the GPIA through a 74ALS574 latch, is used to set the GPIB address of the video digitizer.



Timer 1: G1 <-- $\overline{\text{HSYNC}}$

- This timer is used as a horizontal sync pulse detector. The timer must be initialized for pulse width (PW) measurement: PW > Time Out = 3.5 us, interrupt enabled.

Timer 2: G2 <-- $\overline{\text{CSYNC}}$

- This timer is used to detect vertical sync and equalization pulses. For vertical sync pulse detection, the timer must be set up for pulse width measurement: PW > Time Out = 20 us, interrupt enabled. For equalization pulse detection, the timer must be set up for pulse width measurement: PW < Time Out = 3.5 us, interrupt enabled..

Timer 3: G3 <-- CSYNC

- Field detection is the purpose of this timer. The timer must be set up for pulse width measurement: PW < Time Out = 45 us, interrupt disabled. A field is detected by reading the interrupt flag associated with this timer when an equalization pulse is recorded. If the interrupt flag is set, the current video field is #1.

Notes: CSYNC = Composite SYNChronization signal
HSYNC = Horizontal SYNChronization signal

Figure 2.13

Programmable Timer Module Programming Details

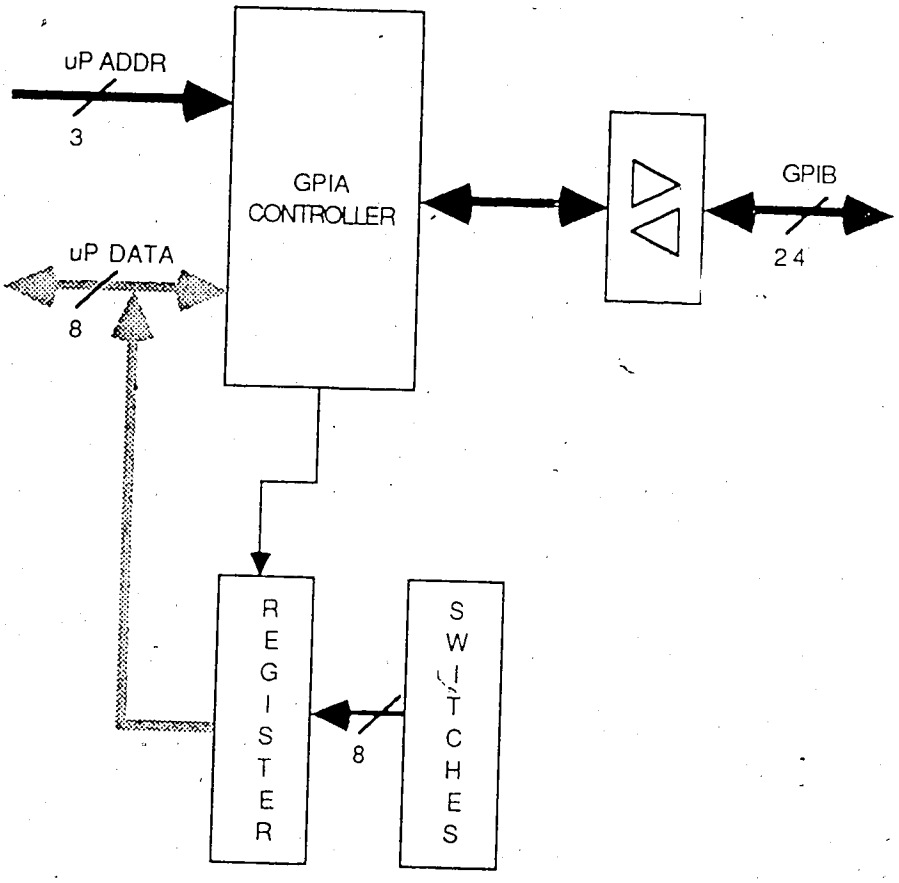


Figure 2.14

Block Diagram of IEEE-488 System.

3. System Software

3.1 Overview

The system software executes on the local MC68B09 microprocessor. It controls and synchronizes the functional blocks of the video digitizer system and is written in the microprocessor's native assembler code.

After a hardware reset, microprocessor program execution is vectored to the start of the video digitizer code (Figure 3.1). The program initializes the microprocessor system, the GPIA, and the video system. The video system initialization (Figure 3.2) resets the video signal processor (PTM), the analog to digital interface (PADIM), and initializes the default sampling parameters.

After the video digitizer has been initialized, the software enters the main loop. The Main Loop module (Figure 3.3) monitors the IEEE-488 instrumentation bus for a message. The received message is analyzed when the bus completes a transaction with the video digitizer configured as a listener. The microprocessor will determine the validity of the command and process it if valid, otherwise it will send a message to the controller, via the send response module, indicating an error (Figure 3.4).

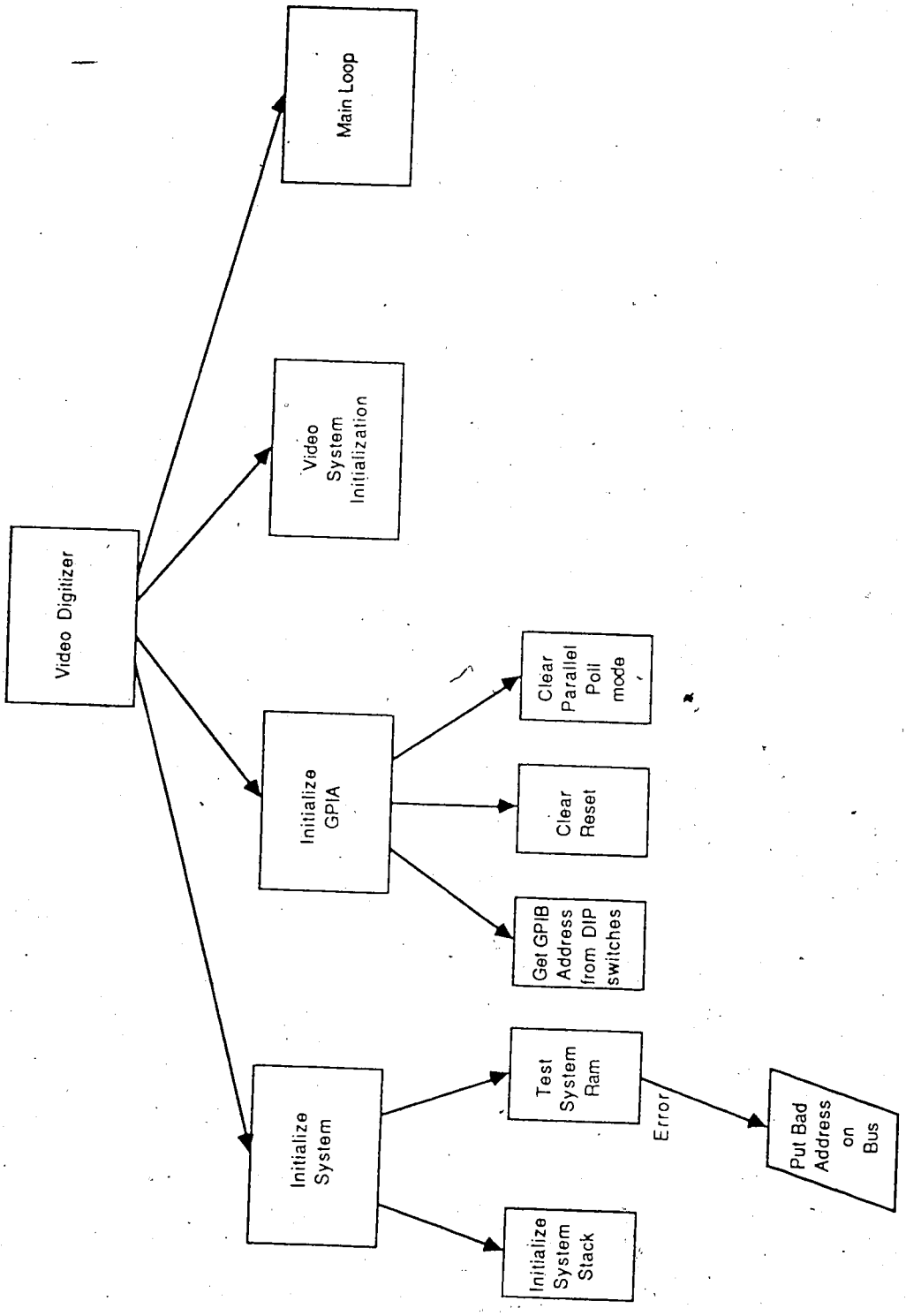


Figure 3.1

Structure Diagram of the Video Digitizer Software

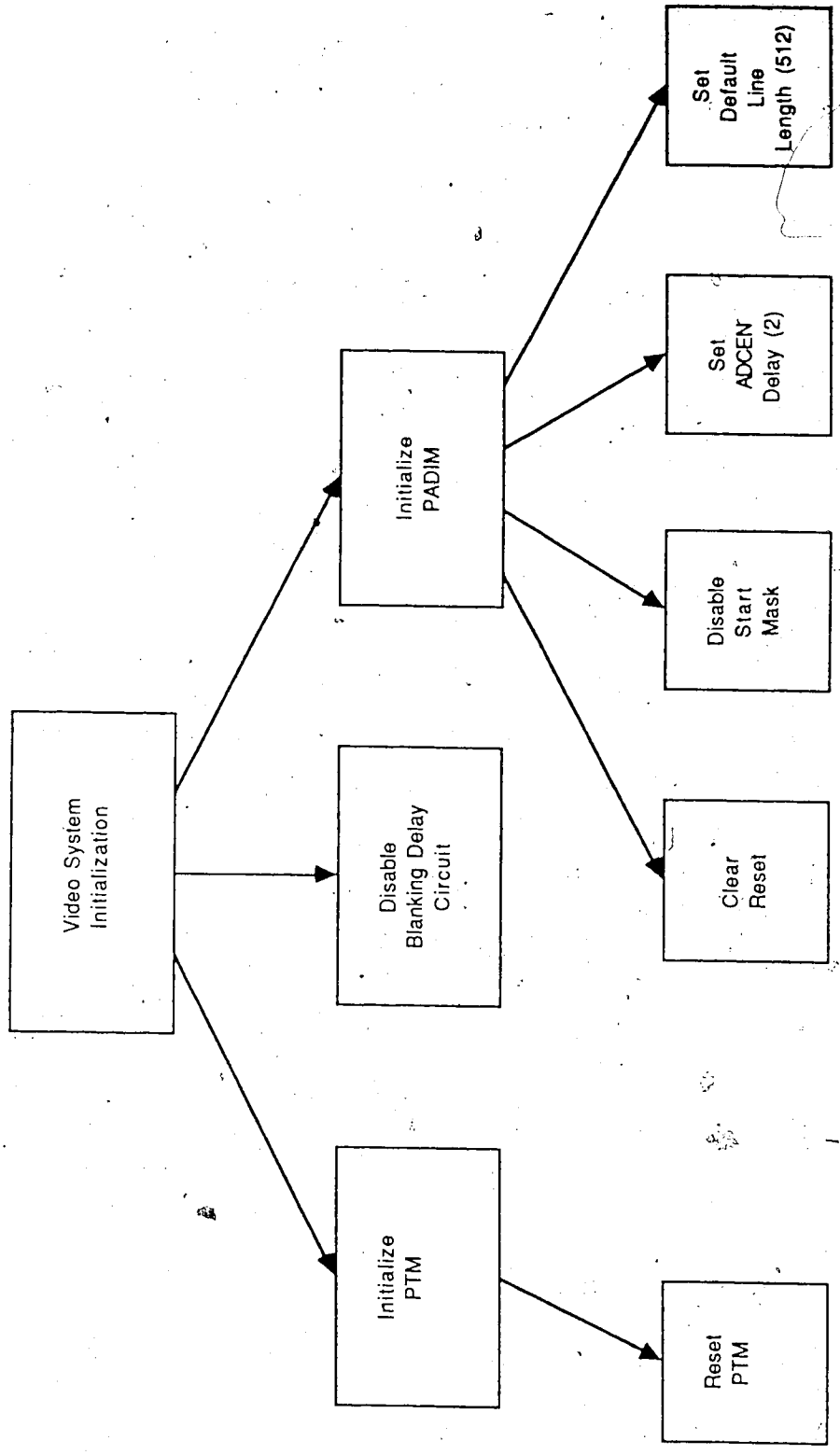


Figure 3.2

Video System Initialization Module

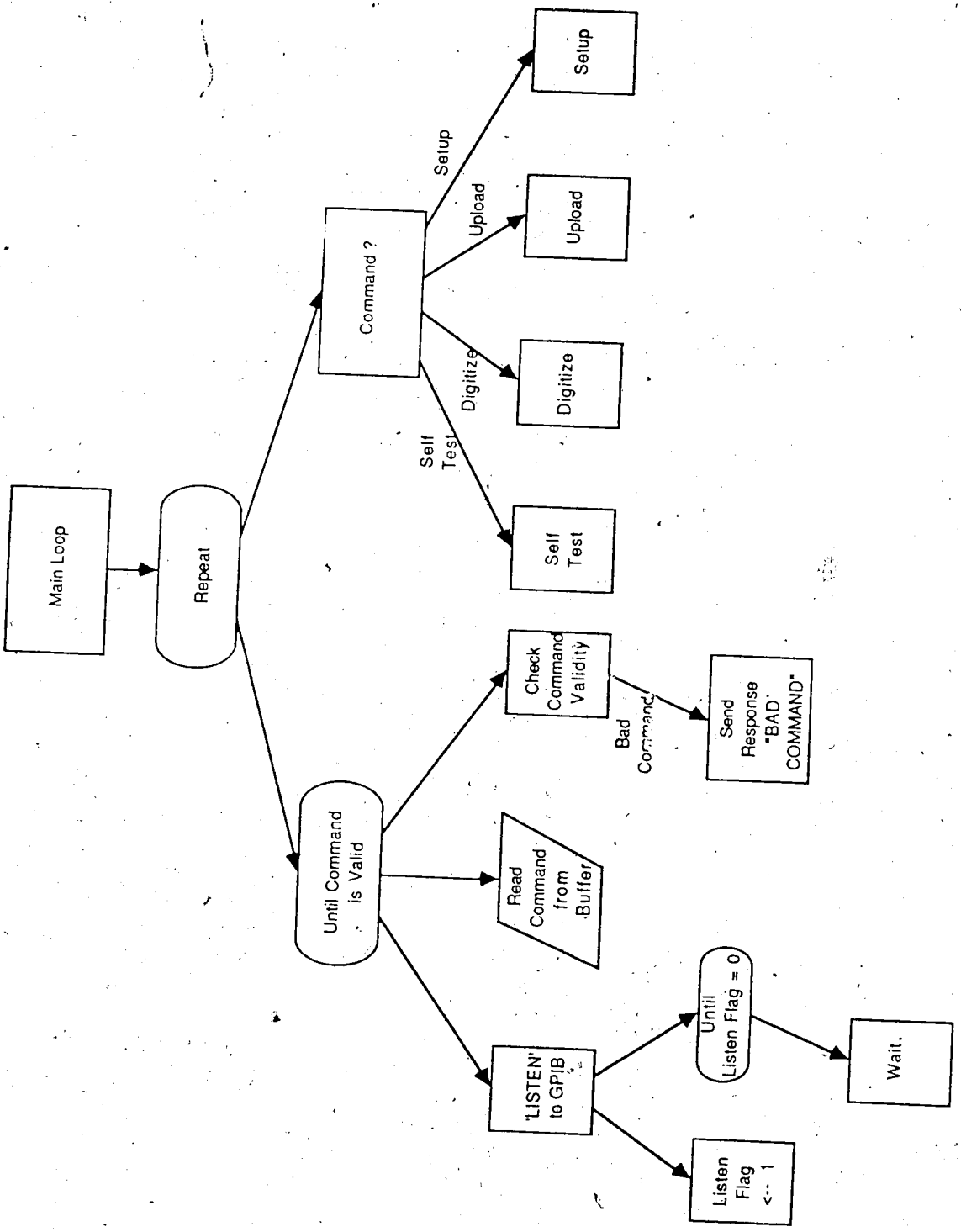


Figure 3.3

Main Loop Module

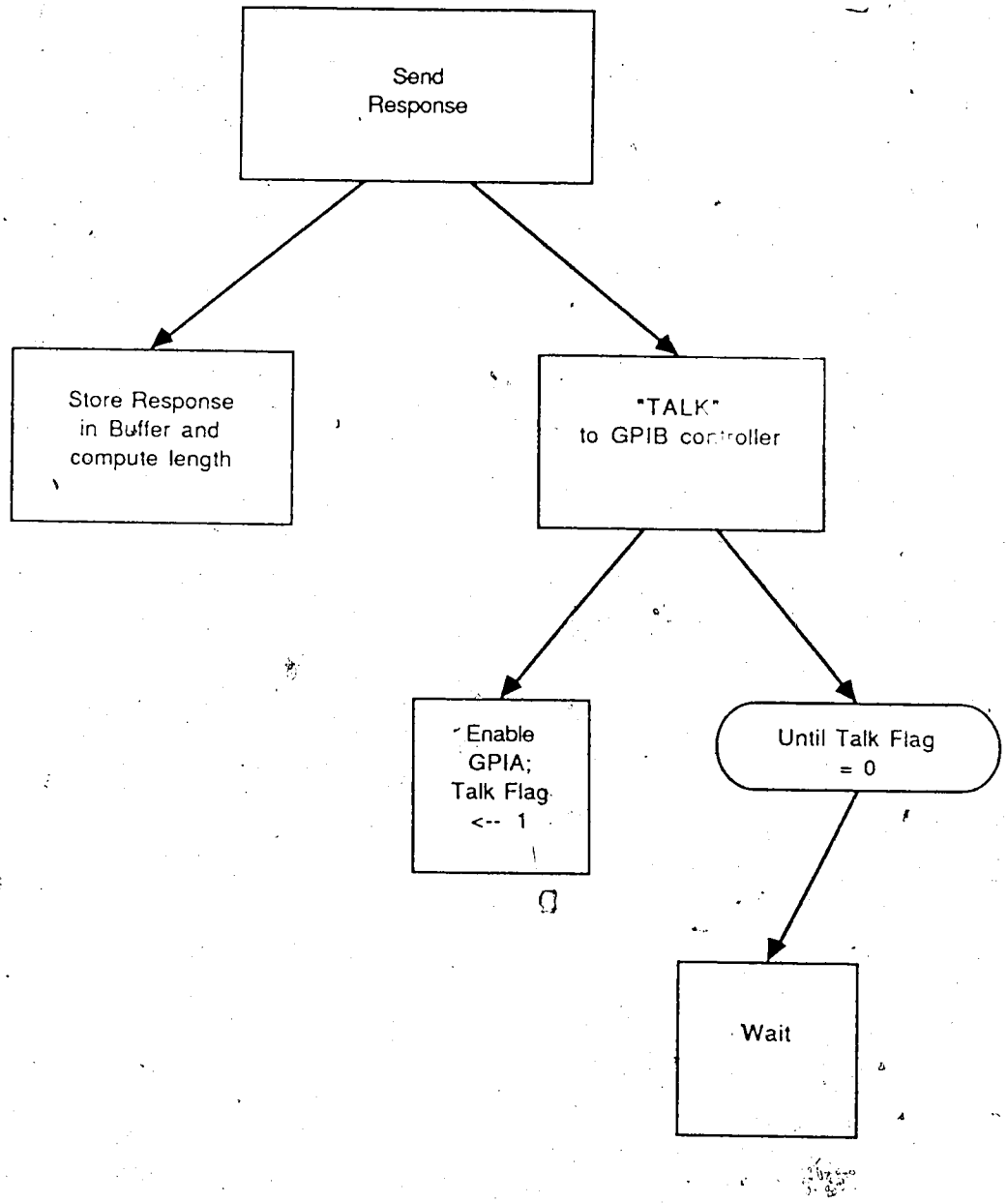


Figure 3.4

Send Response Module

3.2 Digitize Module

This routine is executed when the main loop receives a command to digitize a video frame. The digitization software (Figure 3.5) sets up the start address for the video frame, digitizes the video frame and sends a response to the host computer indicating the completion of the frame acquisition.

The Digitize Frame module (Figure 3.6) is the root module for video frame acquisition. Control is first passed to an initialization module, Video Signal Processor Initialization (Figure 3.7), which configures the PTM to detect equalization, horizontal, and vertical synchronization pulses. When the PTM detects these pulses, the microprocessor is alerted with the assertion of an interrupt line. After the initialization, the Digitize Frame module calls the Frame Synchronization module. The detection of synchronization pulses and subsequent interrupts, provide video frame synchronization information to the Frame Synchronization module. Once frame synchronization has been established, the module digitizes both video fields using the Digitize Field module.

3.2.1 FIRQ Interrupt Service Module

The MC68B09 microprocessor has three interrupt levels: (1) a non-maskable interrupt, (2) a standard interrupt and (3) a fast interrupt. The fast interrupt sequence is

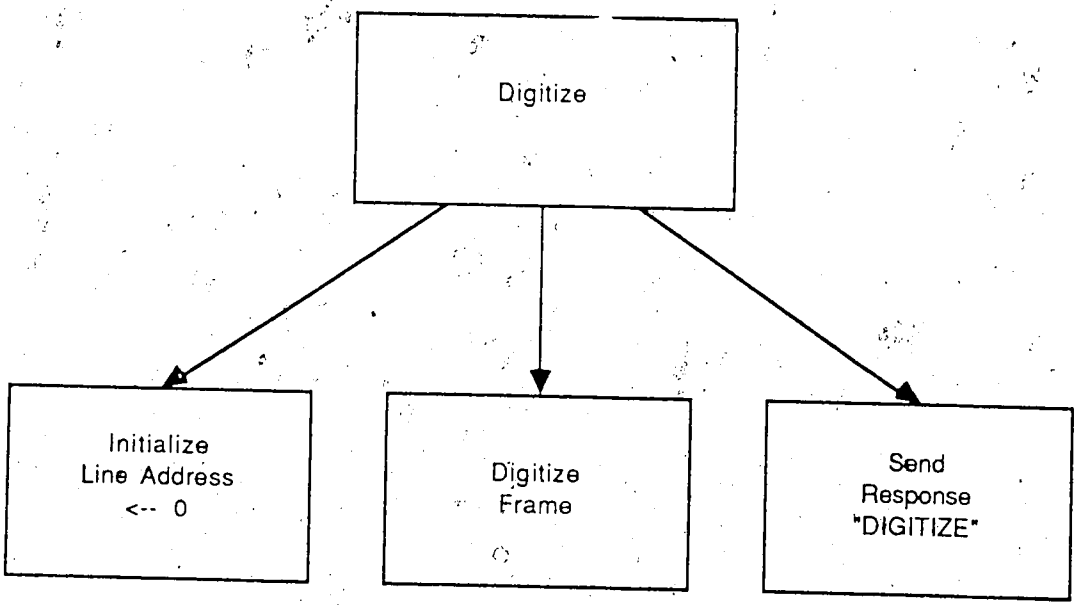


Figure 3.5

Digitize Module

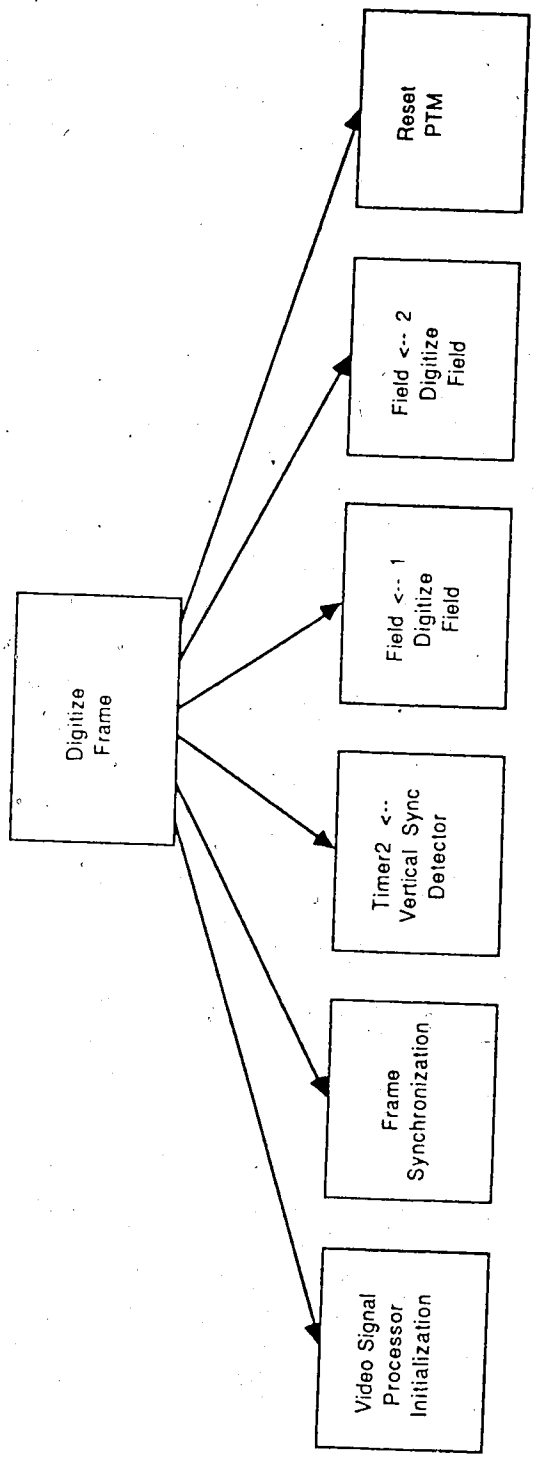


Figure 3.6

Digitize Frame Module

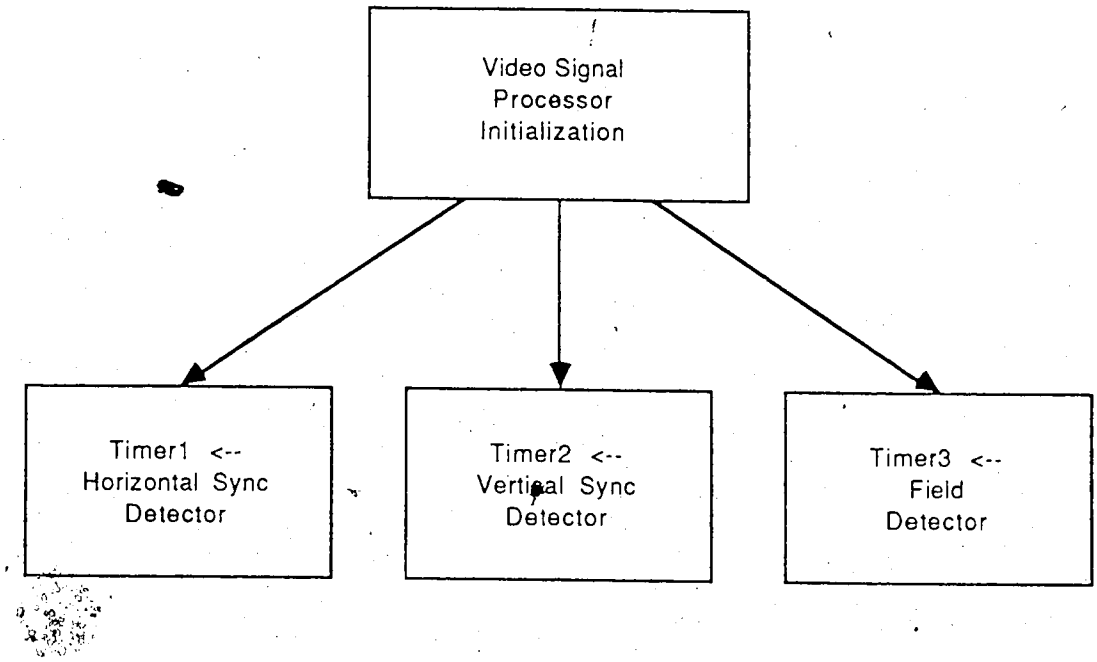


Figure 3.7

Video Signal Processor Initialization Module

initiated by a device attached to the FIRQ line of the microprocessor and has priority over the standard interrupt (IRQ). An FIRQ is fast in the sense that when the microprocessor recognizes the interrupt, the microprocessor stacks only the program counter and condition code register. This reduction in stacking operations compared to the NMI and IRQ increases interrupt response time by a factor of 2.

The FIRQ is used by the video digitizer to link the video signal processor to the microprocessor. This link allows the PTM to quickly alert the microprocessor when it detects a synchronization pulse. Since synchronization pulses may occur as little as 32 us (0.5 H) apart, the microprocessor must respond and process the interrupt in less than 32 us. A structure diagram of the FIRQ service routine is shown in Figure 3.8. In addition to fast execution, the interrupt service routine must provide a way of informing other software modules that a synchronization pulse has occurred. This is accomplished by using a flag that the other modules can poll; however, the longest period of time between successive polls of the flag by a module must not exceed the peri- -etween synchronization pulses specific to that module. That is, during the process of digitizing video lines, the software is synchronized to the horizontal synchronization pulses; thus, the longest time allowed between successive polls is 63.5 us (1 H). To achieve minimal interrupt service time, the service

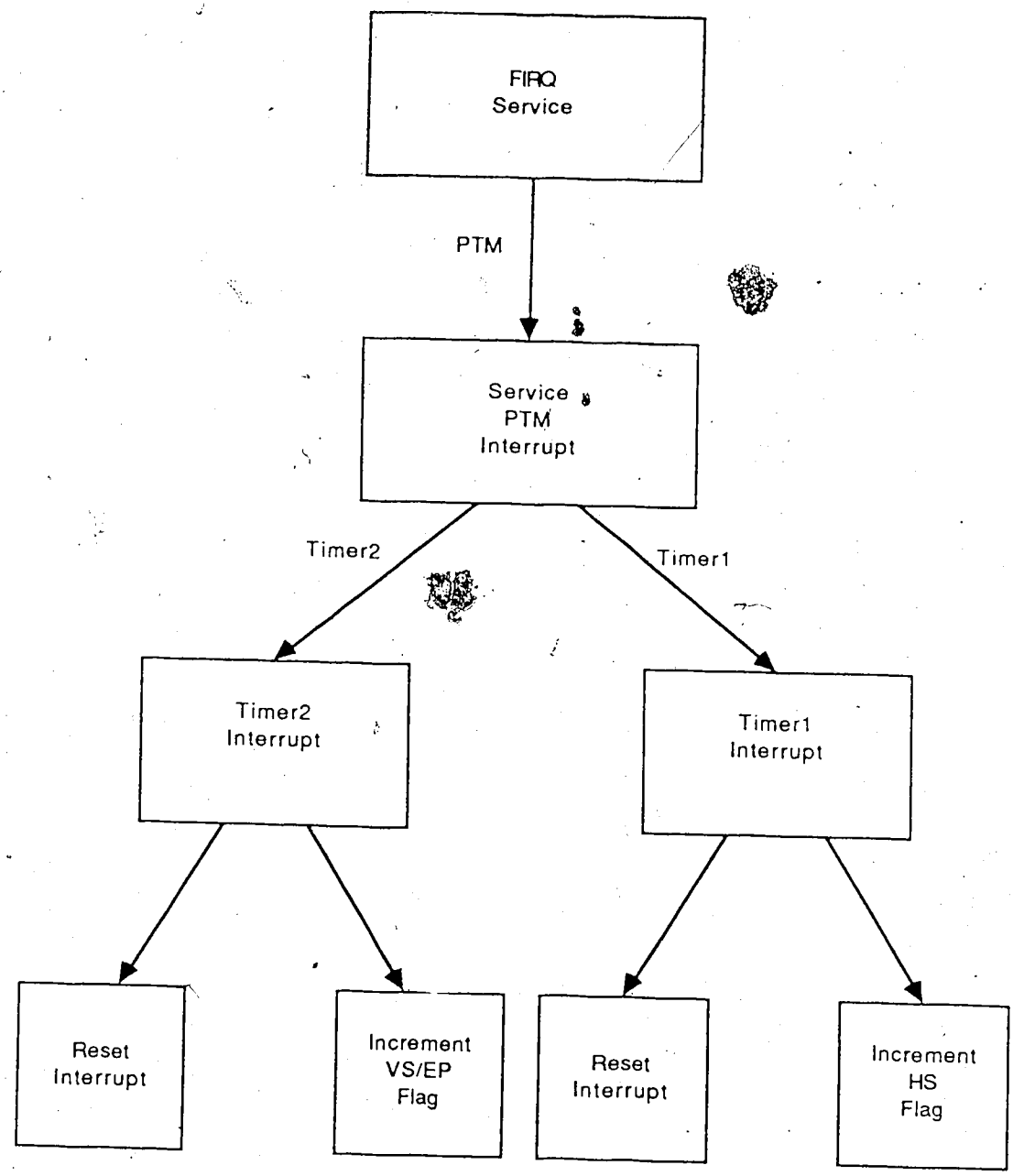


Figure 3.8

FIRQ Service Module

routine acknowledges the interrupt and increments a flag associated with the timer that caused the interrupt.

3.2.2 Frame Synchronization Module

Frame synchronization is accomplished by detecting the end of the second field of a video frame. This procedure is outlined in the flowchart shown in Figure 3.9 (also see Figure 2.4 for vertical synchronization sequence details). First, the program waits for a vertical synchronization pulse followed by a horizontal synchronization pulse. After this, the program waits for an equalization pulse. When an equalization pulse is detected, the module measures the pulse width of the preceding video line (the last visible video line). If the width of this line is 1 H, then the current video field is the second field of the current video frame. If the width of the line is 0.5 H, then the current field is the first field of the video frame.

When the end of the second field of a video frame is detected, the software is synchronized to the frame (Figure 3.10). Notice that the frame synchronization structure diagram shows the software waiting for six horizontal synchronization pulses after the vertical synchronization pulse is detected. This is because five horizontal synchronization pulses occur and are detected during the vertical synchronization. Therefore, to detect the first horizontal synchronization pulse after the vertical

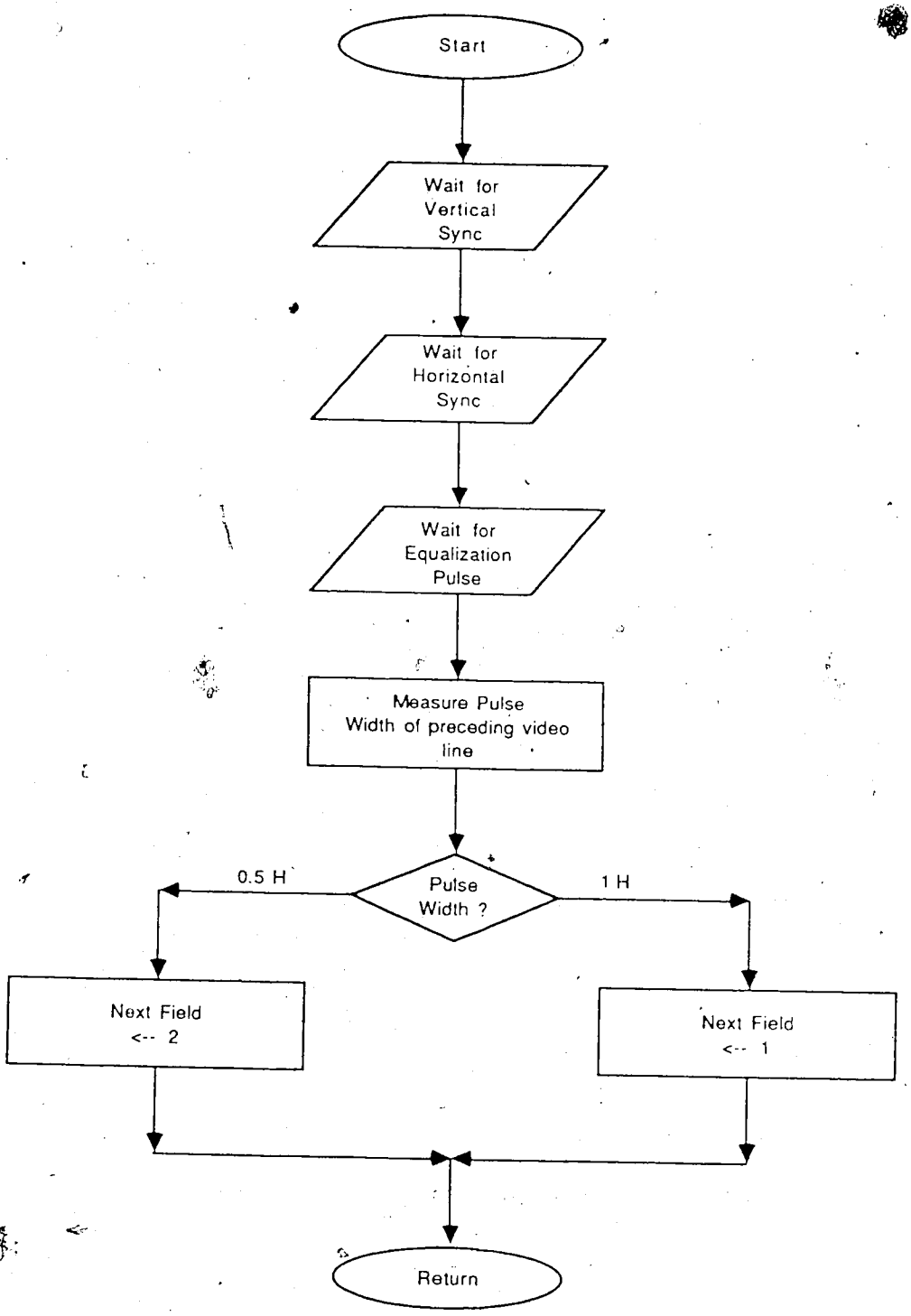


Figure 3.9

Frame Synchronization Flowchart

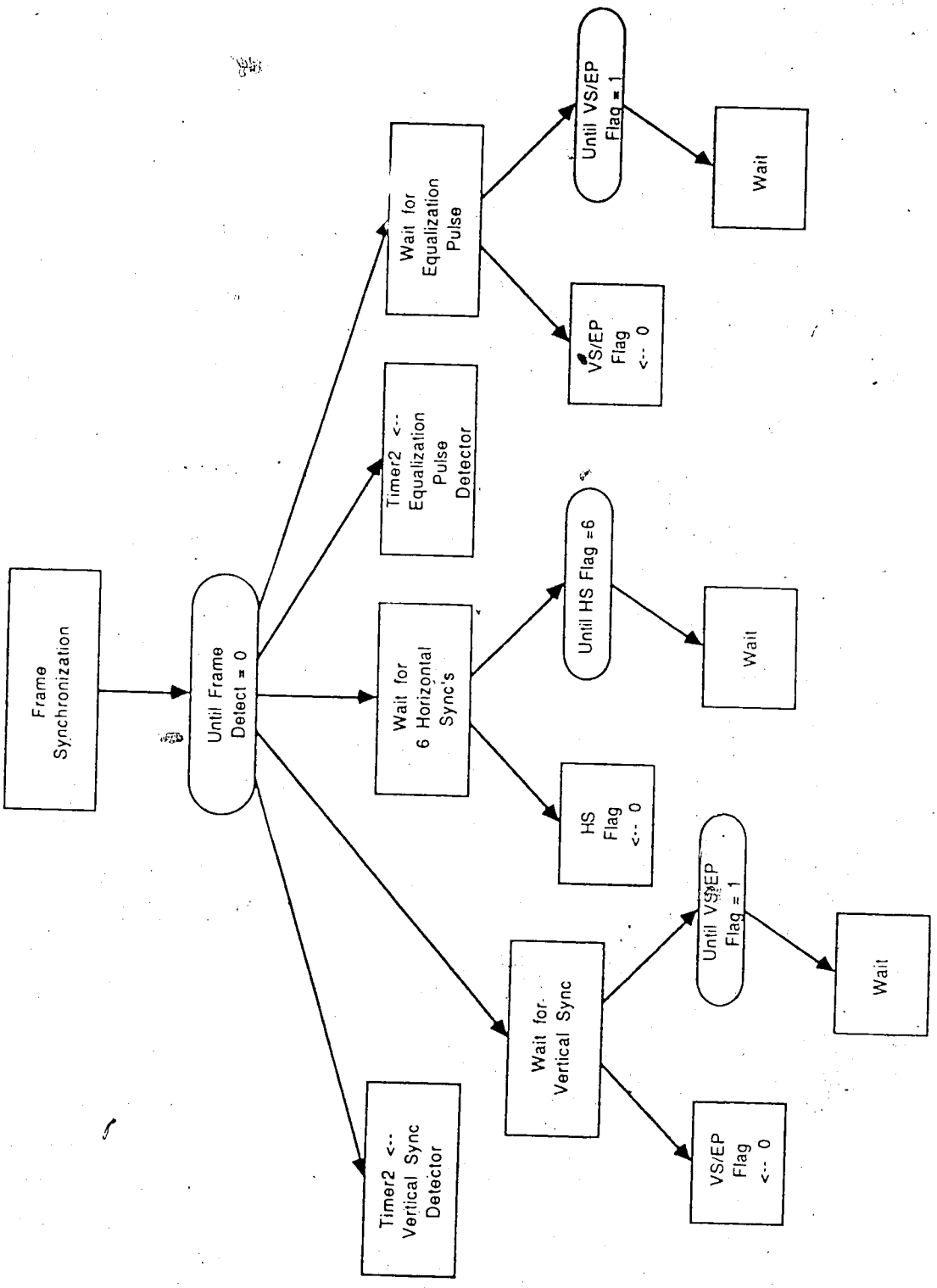


Figure 3.10

Frame Synchronization Module

synchronization pulse, six horizontal synchronization pulses must be detected.

3.2.3 Digitize Field Module

When the software has established synchronization with the video frame, both video fields of the frame are digitized by the Digitize Field module. The Digitize Field module is designed to be field independent; that is, it may digitize the first or second field based on a register parameter (called the field parameter) passed to the module. The parameter (in this case, the A accumulator) must contain the number of horizontal synchronization pulses that occur between the vertical synchronization pulse and the first visible video line which is unique to each video field.

Once control is given to the Digitize Field module (Figure 3.11), the software sets up the base address of the first video line (passed as a variable parameter from the Digitize Frame module) and monitors the number of horizontal synchronization pulses that occur. When the number of horizontal synchronization pulses that have occurred matches the field parameter, the software enables the blanking delay circuit which in turn provides the necessary start signals to the PADIM.

The software now monitors the horizontal synchronization pulses, which correspond to the PADIM start

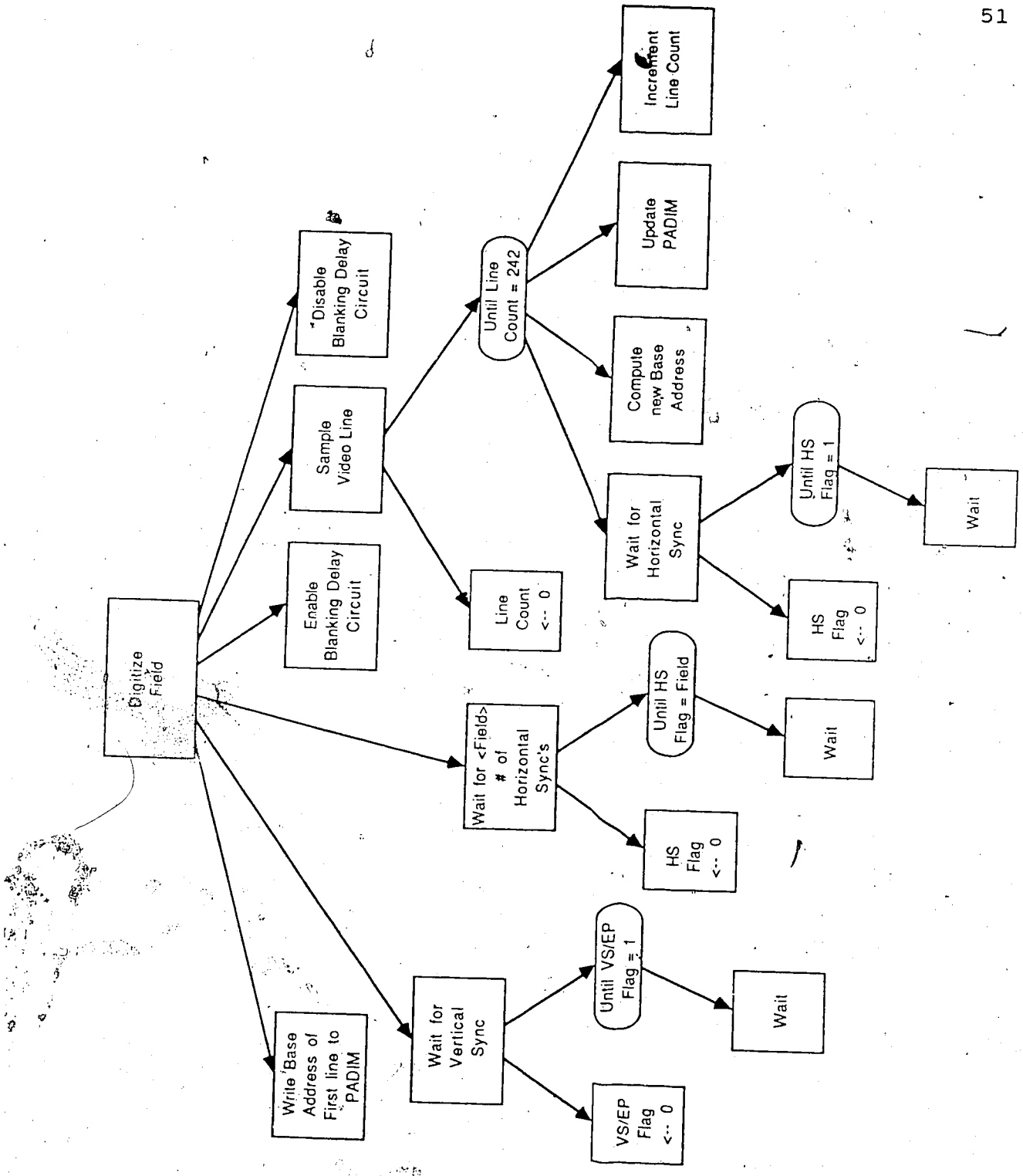


Figure 3.11

Digitize Field Module

signals plus some fixed delay, and reprograms the PADIM with a new base address for each video line. This continues until the program detects that the required number of video lines (242 by default) have been sampled. When all the video lines of the current field have been digitized, the module disables the blanking delay circuit. The time between horizontal synchronization pulse detection (start of a video line) and the next horizontal synchronization pulse is small (63.5 us). Care must be exercised to ensure that the PADIM is updated with the base address of the next video line before the occurrence of the next synchronization pulse.

3.3 IRQ Service Module

The IRQ Service module controls device transactions with the host computer via the IEEE-488 instrumentation bus. This module consists of interrupt driven input, output and GPIB command modules (Figure 3.12). The MC68488 GPIA uses the standard interrupt (IRQ) of the microprocessor system. A transaction between the video digitizer and the host computer must be initiated by the host computer. When the host initiates the bus transaction by sending a command to the video digitizer, a BI (Byte In) interrupt is generated by the GPIA. An interrupt sequence is initiated if the GPIA was enabled as a 'LISTENER' as occurs in the Main Loop module (Figure 3.3). When the BI interrupt

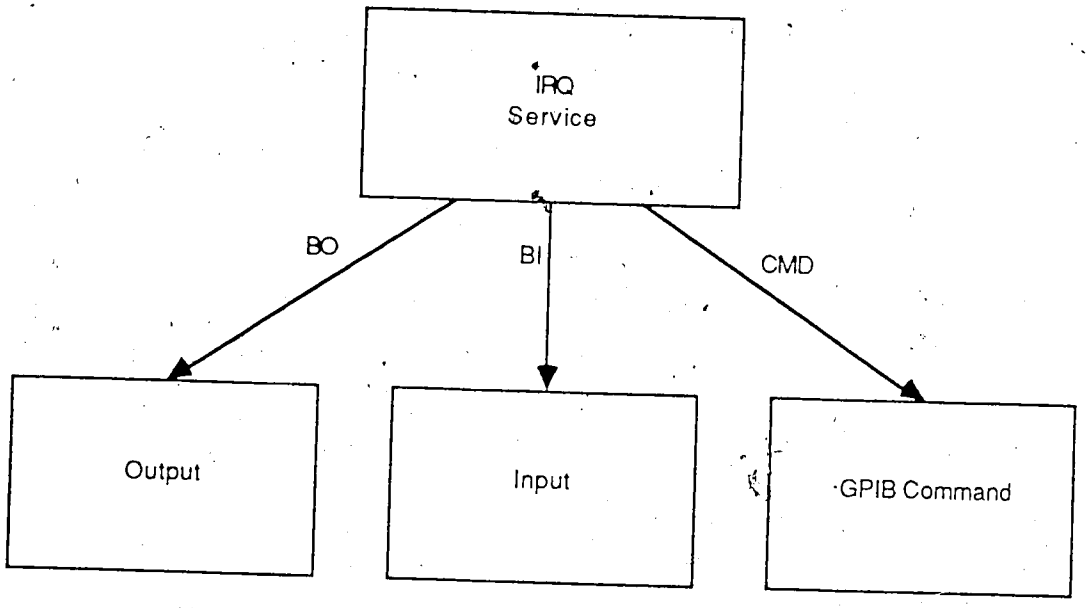


Figure 3.12

IRQ Service Module

is recognized, the IRQ service module directs program execution to the input module (Figure 3.13).

The input module polls the GPIA to see if a byte is ready. When the GPIA indicates so, the software reads the GPIA data-in register and places the data into the input buffer and updates the length attribute. This continues until the module detects that the EOI (End Or Identify) line of the GPIB has been asserted. When the EOI line is asserted, it indicates that the next byte of information will be the last for the current bus transaction. When the assertion of the EOI line is detected, the program reads the last byte from the GPIA, stores the data in the buffer, updates the length, and clears the Listen Flag to indicate that the listen operation has completed.

After the input has been processed, the digitizer responds with a message unique to the command received (Figure 3.4). When the send response module is executed, the response message is stored in a buffer and the GPIA is enabled to make the video digitizer a 'TALKER'. The module polls the status of a flag (called the Talk Flag) to determine if the GPIB transaction has taken place. When the GPIB is ready to accept the video digitizer as a 'TALKER', a BO (Byte Out) interrupt is generated by the GPIA. When this interrupt is generated, the IRQ service module directs program execution to the output module (Figure 3.14).

The output module computes the end of the string and

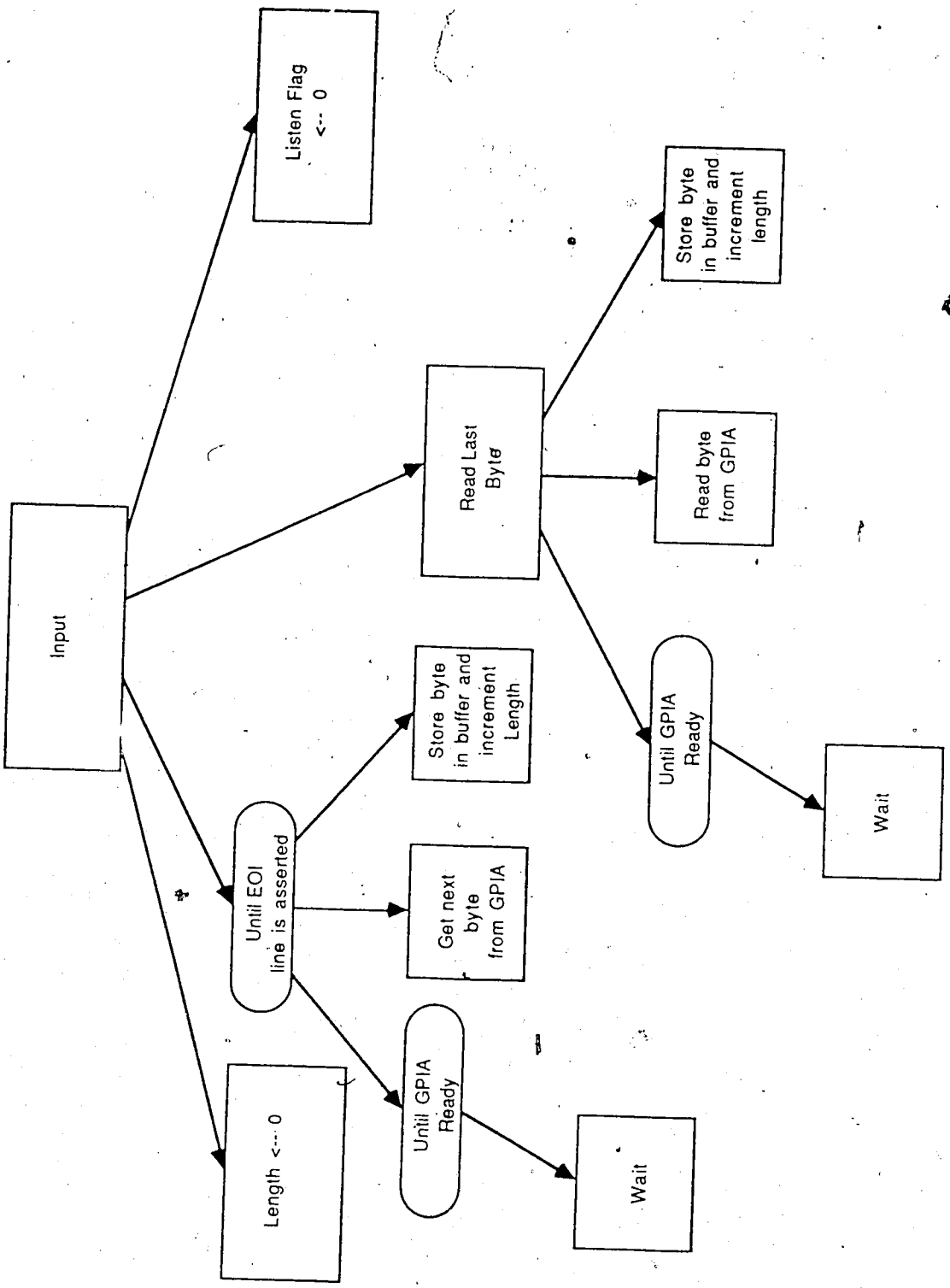


Figure 3.13

Input Module

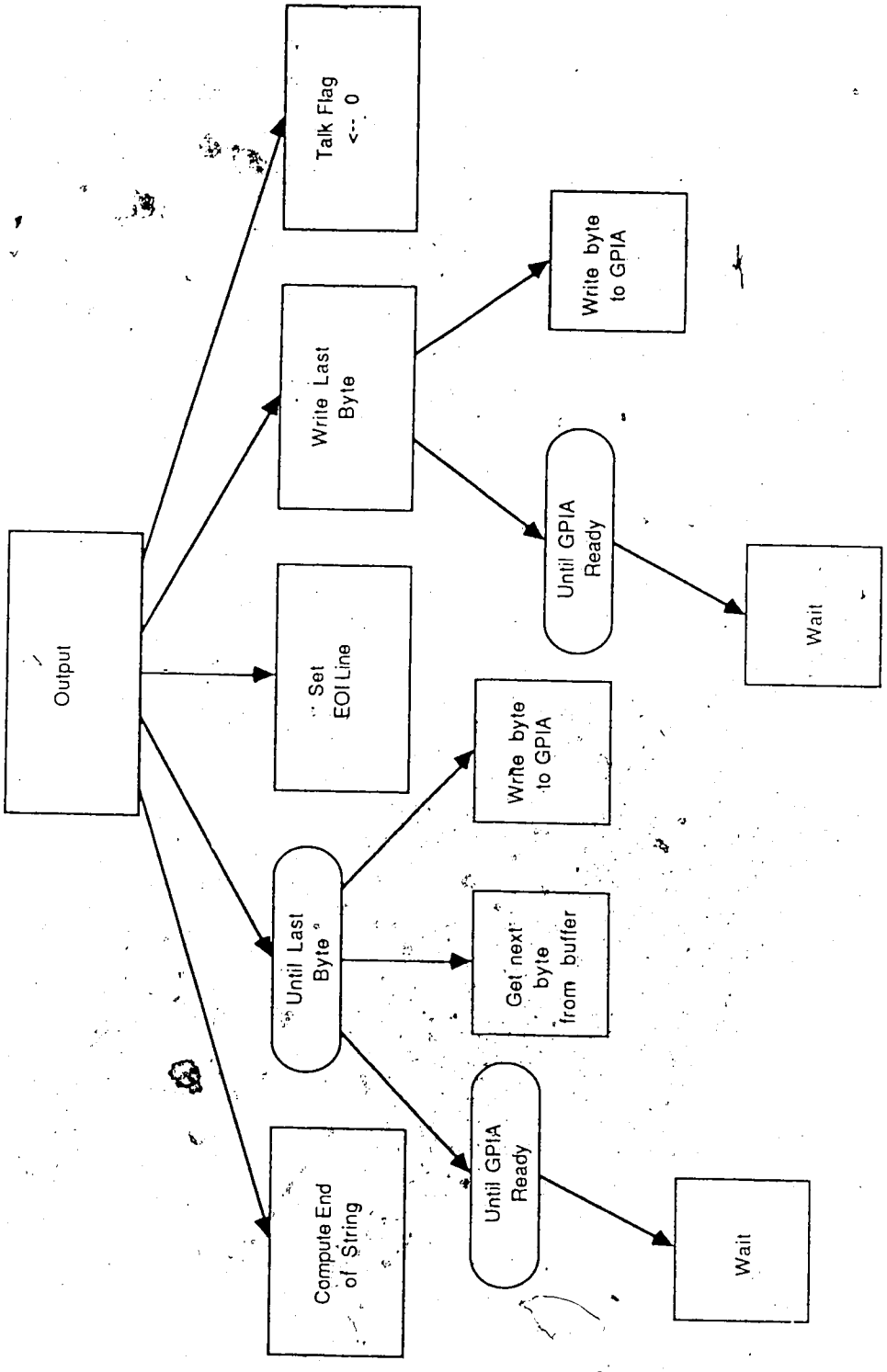


Figure 3.14

Output Module

polls the GPIA for output status. When the GPIA is ready to receive a byte for output, the module extracts the necessary byte from the buffer and writes this to the GPIA data-out register. This procedure continues until the program detects the last byte of the message and then the EOI (End Or Identify) line is asserted and the last byte is written to the GPIA. Finally, the Talk Flag is cleared to indicate the completion of the output.

Another feature built into the IEEE-488 software is a module to handle the Device Clear (DCL) GPIB command. When the GPIA generates an interrupt, the status register is also examined to determine if a GPIB command has occurred. If so, the microprocessor executes the GPIB command module (Figure 3.15). This module checks to see if a DCL command was the cause of the interrupt. If so, the software resets the system stack and initializes the video system.

3.4 Upload Module

The upload module transfers the digitized video information to the host computer system. This software module is part of the video digitizer command set. The host sends an upload command byte followed by a sixteen bit word indicating the video line number. The host computer requests video information on a line by line basis. When control is transferred to this module, the computer retrieves the line number from the input buffer. This

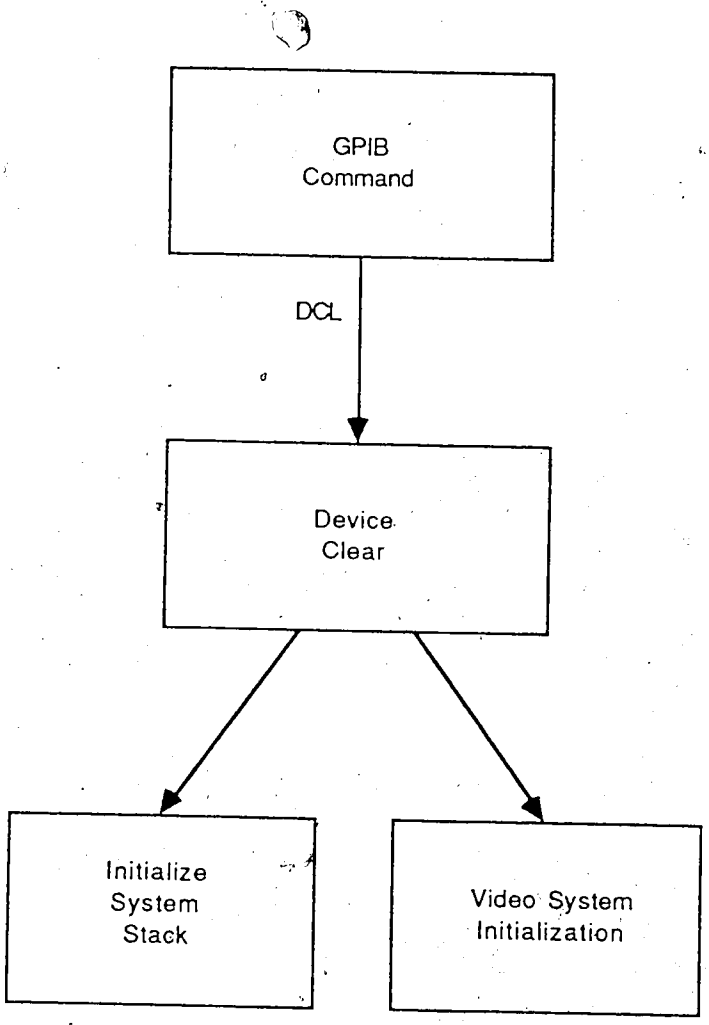


Figure 3.15

GPIB Command Module

information is used to compute a video page and memory address for video RAM. The PADIM is programmed with the video page and the data is read from video RAM and stored in an output buffer. A response message is constructed when the video samples have been loaded into the output buffer. The video information is returned with the response (Figure 3.16).

3.5 Self Test Module

Three self test operations are incorporated in the video digitizer. These tests were implemented to aid in the hardware debugging of the video digitizer and are available to assist in diagnostic testing of the video digitizer in case of hardware problems (Figure 3.17).

The first mode tests the video signal processor and PADIM hardware (Figure 3.18). This test synchronizes the digitizer with the video frame and puts the video digitizer into a repetitive digitization procedure of 256 successive frames. The video digitizer will return a completion byte to the host computer through the IEEE-488 instrumentation bus in approximately 8.5 seconds. During this time, an oscilloscope can be used to monitor the signal paths in the video digitizer hardware.

The second mode is a video RAM test (Figure 3.19). This software module performs a memory test on the video RAM by 'walking' a 1 through each byte. The test result is sent

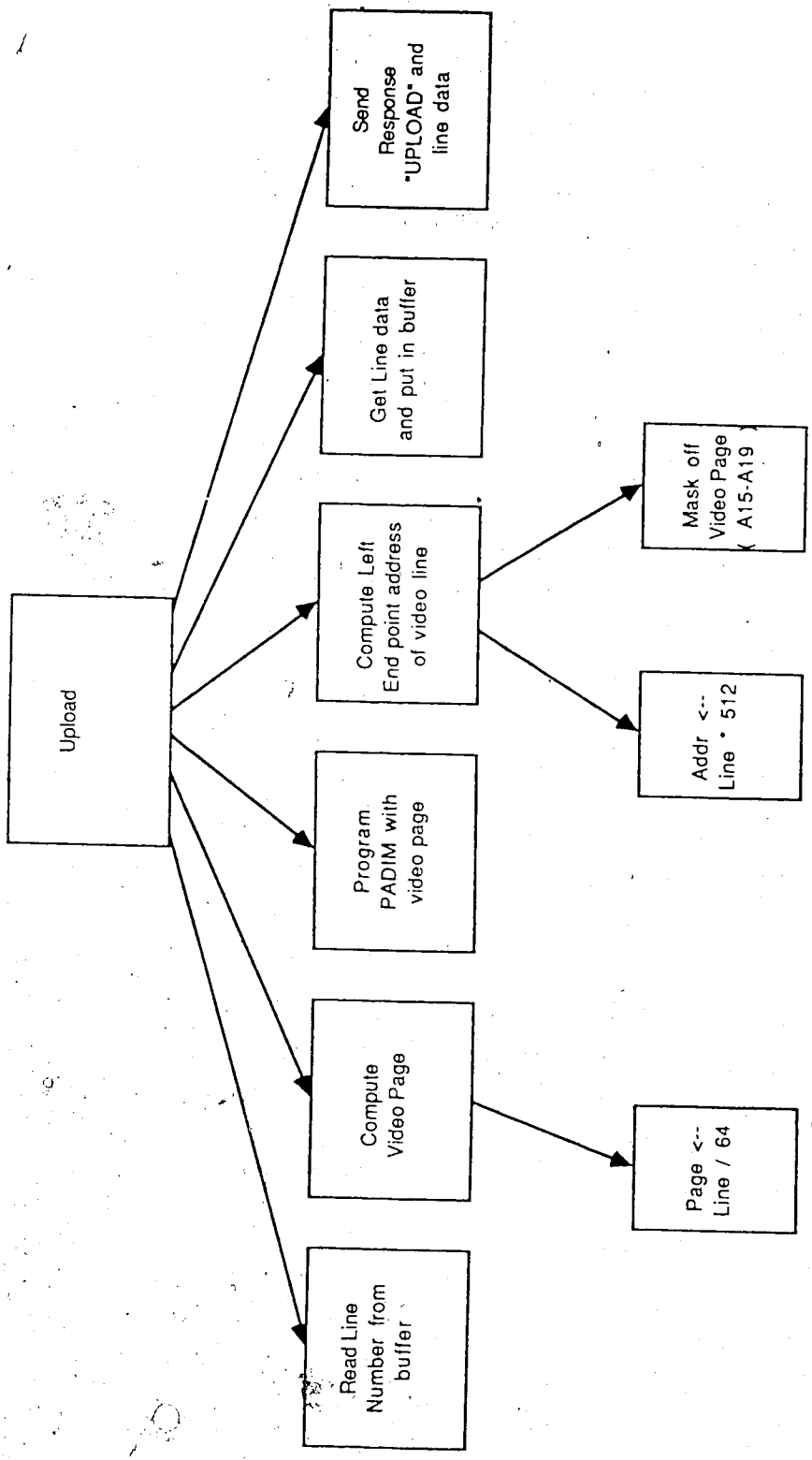


Figure 3.16

Upload Module

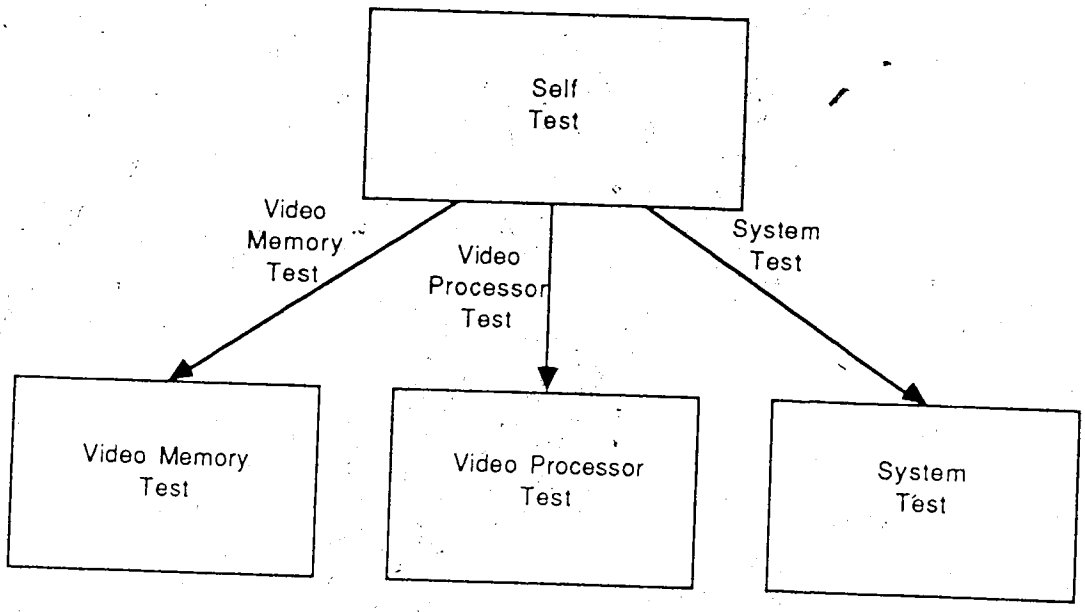


Figure 3.17

Self Test Module

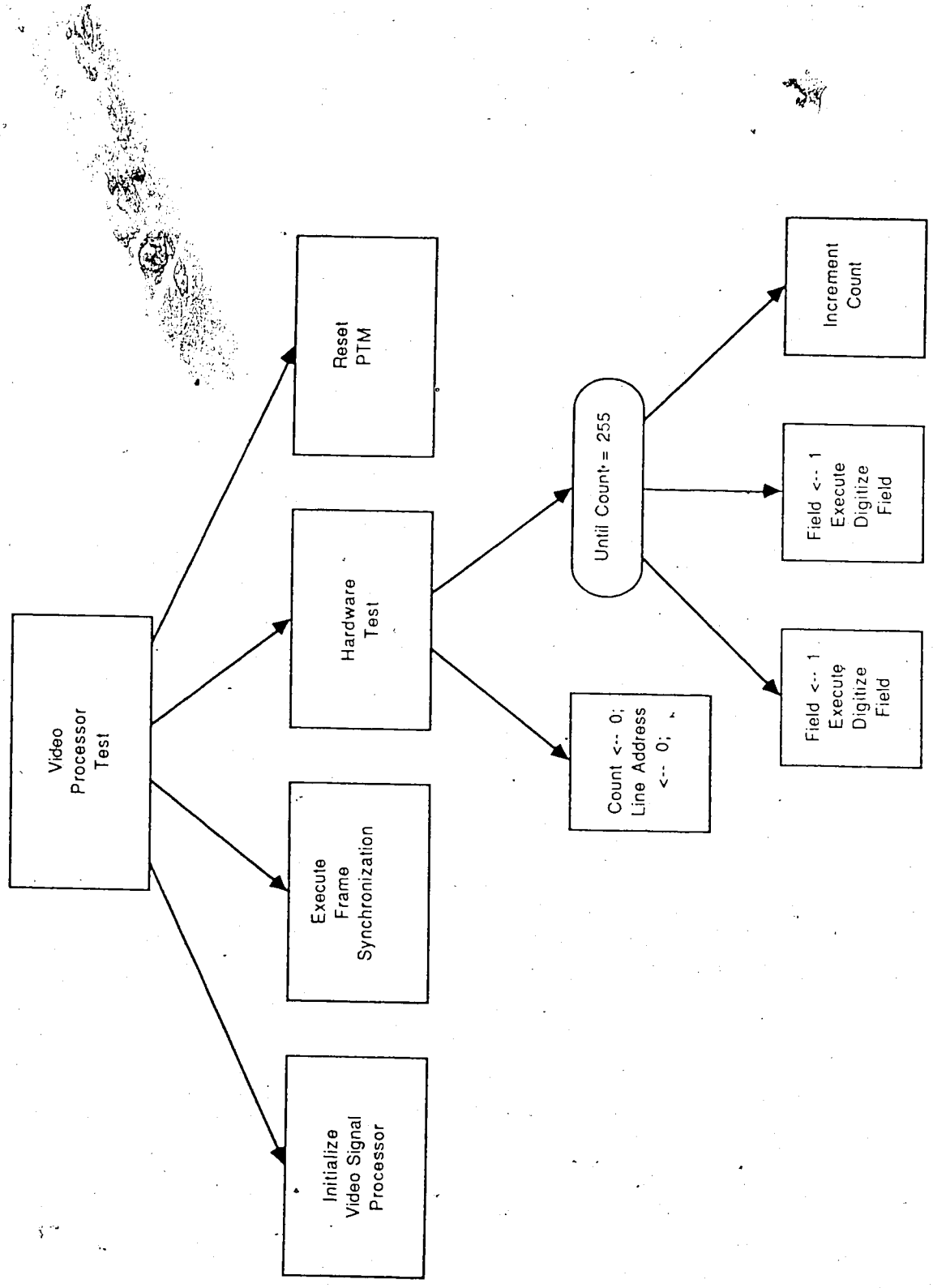


Figure 3.18

Video Processor Test Module

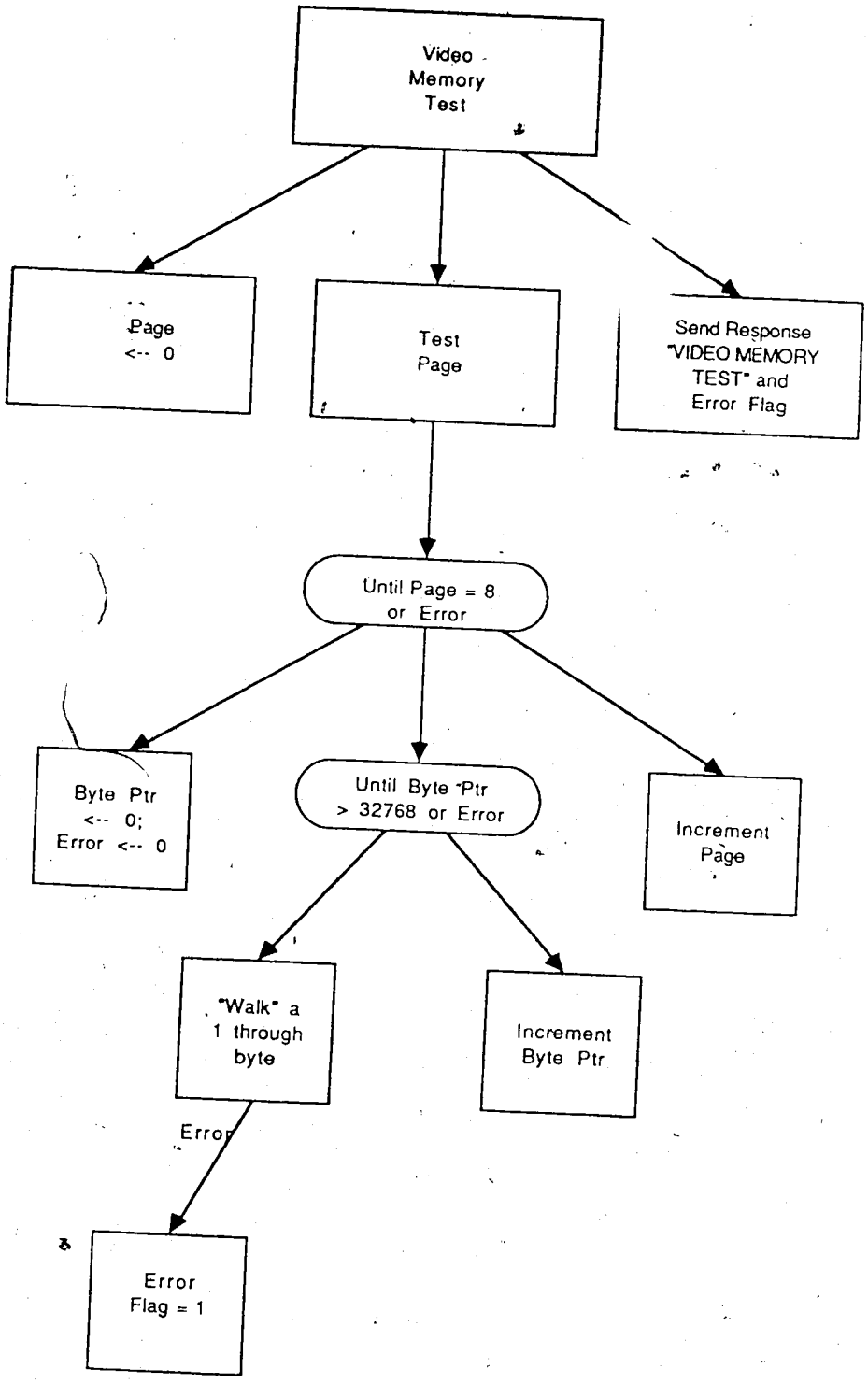


Figure 3.19

Video Ram Test Module

to the host computer. A return code of 1 indicates a RAM test failure and a return code of 0 indicates the video RAM passed the test.

The third test mode is a system test (Figure 3.20). The system test mode fills the video memory with a pattern. A sixteen bit word is stored in successive memory locations in video RAM. The MSB of this word corresponds to the video page and the LSB represents the location of a byte within an 8 bit page. Once completed (as indicated by a response via the IEEE-488 bus) the video memory can be examined by uploading the video memory to detect if an addressing failure has occurred.

3.6 Setup Module

The setup module is part of the command structure of the program (Figure 3.21). When the Main Loop module detects a setup command, the microprocessor is directed to execute this module. The setup module was designed to aid in debugging the hardware and provides a way to modify the default parameters (number of samples per line and the number of lines per field).

Once program control is given to the setup procedure, the software extracts the line count and sample count from the input buffer (these two 16-bit values are supplied with the setup command by the host computer), programs these values into the PADIM and responds to the host.

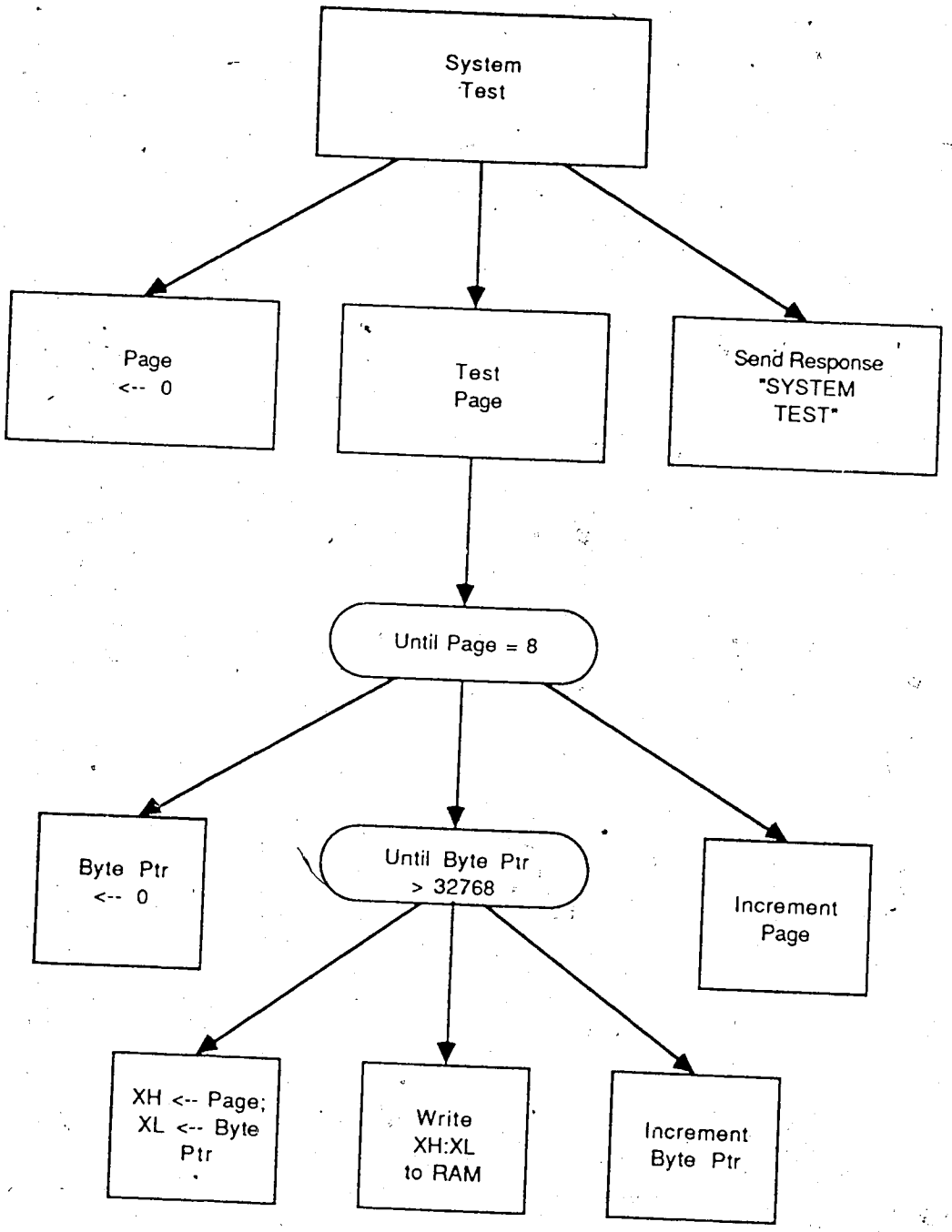


Figure 3.20

System Test Module

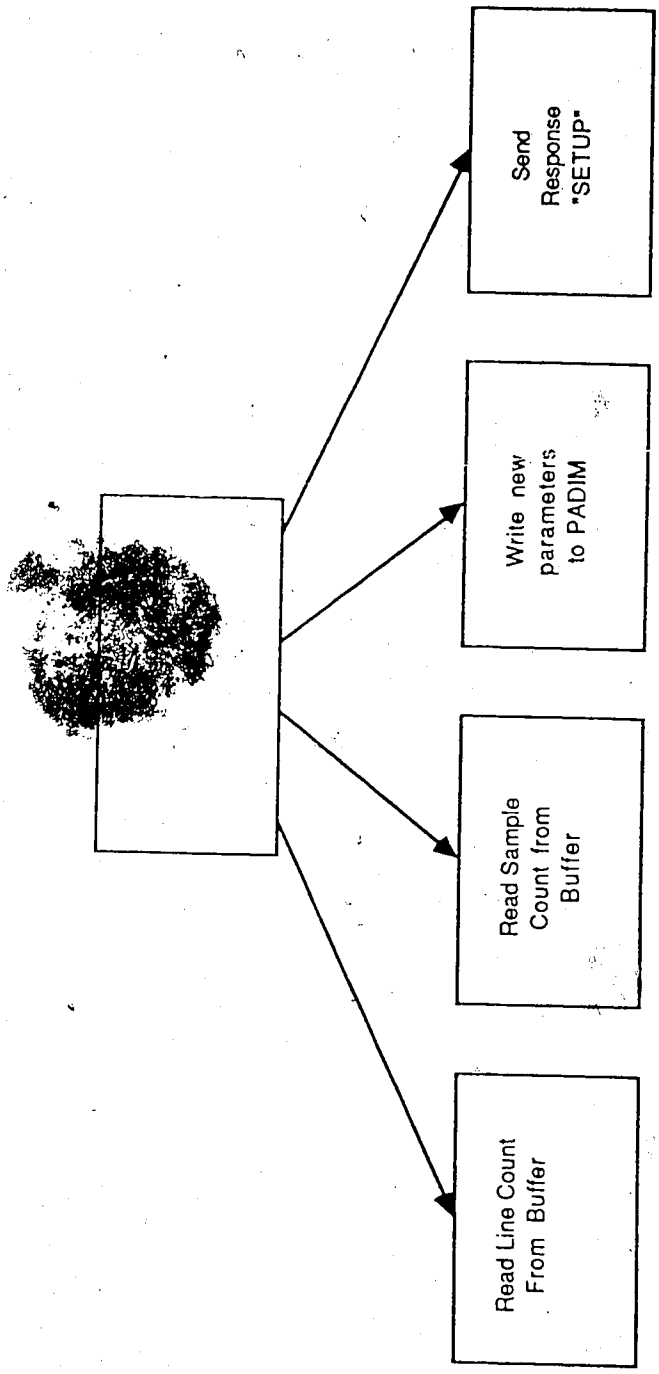


Figure 3.21

Setup Module

4. Prototype Verification and Evaluation

4.1 Initial Prototype Verification

The circuit was prototyped on a vector-board using 3-M prototype components. First, the video signal processing circuit was prototyped and tested by adjusting the corner frequency of the buffer's RC feedback network and the D.C. offset voltage of the clamping circuit (see schematic diagrams in Appendix F). The LM357 op amp is a high performance, high gain FET input op amp. To avoid oscillation a filter is used in the feedback path of the amplifier. The corner frequency was adjusted to maximize the 3dB frequency of the filter so that the output did not oscillate and the addition of this pole did not significantly affect the 3dB frequency of the low pass filter. The clamping circuit provided a constant 0.2V D[redacted] offset voltage and eliminates the D.C. offset fluctuation some poor quality video cameras exhibit. Once the video signal processing interface was successfully tested, the computer and GPIB were assembled. Specialized software was written to verify correct operation of the computer hardware. Critical signal waveforms were analyzed (ie: 40 MHz. oscillator and E) and a logic analyzer was used to

verify the proper execution of the test software.

4.2 IEEE-488 Instrumentation Bus

Software was written to test the HPIB interface subsequent to testing the basic computer hardware. The transmit and receive functions of the communications software were written and tested using an IBM PC computer equipped with an IEEE-488 interface card. This involved sending a message from the PC to the video digitizer. When the video digitizer received the message, it transmitted the same message to the PC where verification was performed to ensure the integrity of the instrumentation bus.

4.3 Self Test

A series of test programs were incorporated into the video digitizer software to verify that the video memory subsystem and the analog to digital interface functioned correctly. Results of the video processing self test are shown in Figure 4.1. This indicates that the A/D converter is being enabled at the proper time and that the software is correctly synchronized with the video frame. The video memory test was conducted and returned a value of 0 indicating a successful memory check. A small program was written to repetitively perform this test and was left to run overnight. The test did not fail once during the 1900 tests performed. The system test verified that the PADIM

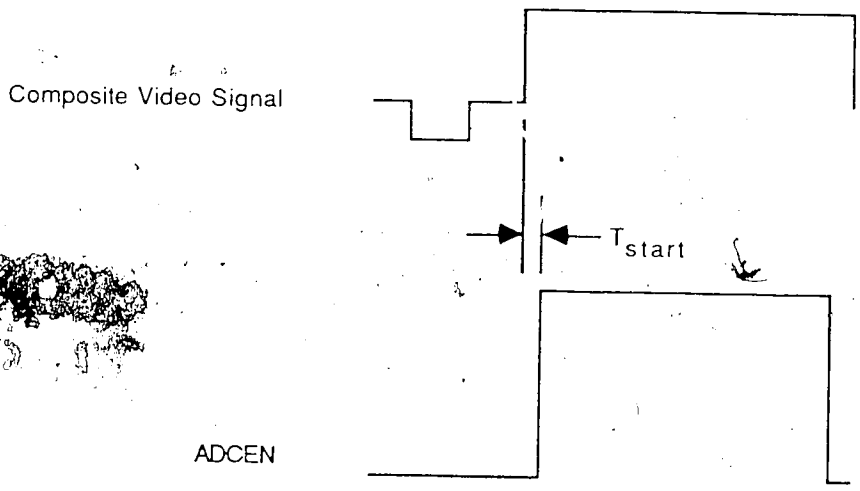
page register was operating properly.

4.4 Frame Digitization

Preliminary tests were done to show that the digitizer performed as expected. These tests involved placing objects in front of the video camera, digitizing them and then analyzing the data. This was done to verify that the video memory was retaining the sample information.

First, a strong light was placed in front of the video camera and the bottom horizontal half of the camera lens was covered (Figure 4.2). The image was digitized and the resulting data was analyzed. Tables 4.1 and 4.2 show the video memory contents for selected video lines. The preceding process was repeated with the obstruction of the left vertical half of the camera lens (Figure 4.3). By analyzing the resulting memory data, shown in Tables 4.3 and 4.4, the obstructing object can be observed by knowing that a low hexadecimal number represents black and a high number is white.

The IBM PC was then programmed to load the data from a digitized image and display it on a graphics screen. Several images of the lab were digitized and displayed in this manner. However, due to the lack of grey scale capability with of IBM PC/XT computer system, these results are not appropriately shown here.



ADCEN timing was measured with respect to the composite video signal and the delay was determined to be equal to the start delay generated by the blanking delay circuit. It was also noted that ADCEN remained low during the vertical sync period.

Figure 4.1

Video Processor Self Test Result

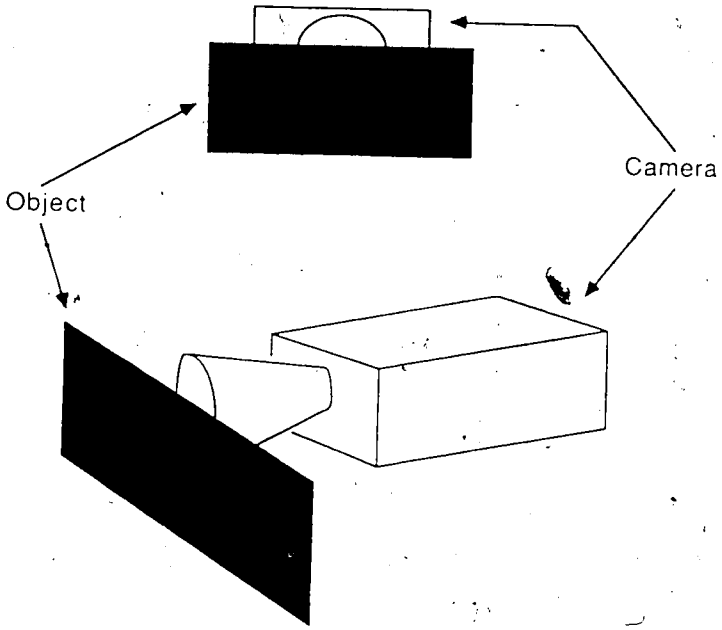


Figure 4.2

Camera - object orientation #1

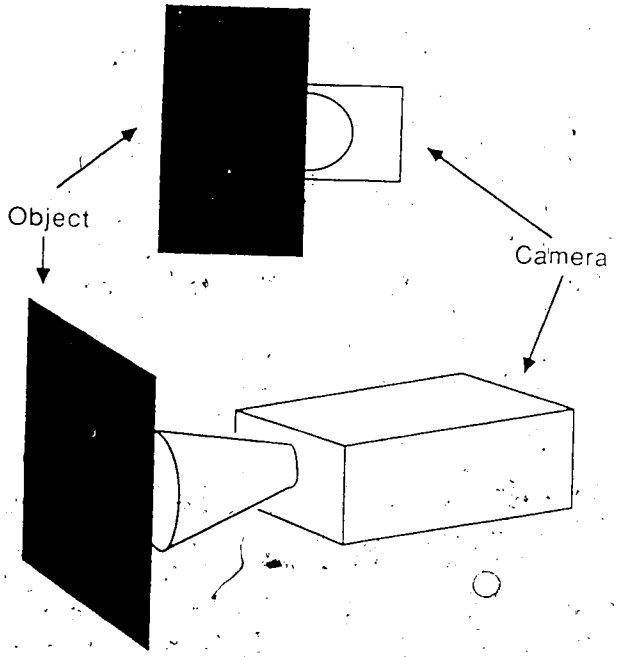


Figure 4.3

Camera - object orientation #2

	Sample											
				2 0	2 1	2 3	2 4	2 6	2 7		5 0	5 1	5 1
	0	1	2 ..	0	5	0	5	0	5	..	9	0	1
0	E8	EA	F4	FF	FF	FE	FF	FF	FF		F2	F4	EF
1	F5	DF	E1	FF	FF	FF	FF	FF	FF		E7	F1	F0
2	F0	F2	EF	FF	FF	FF	FF	FF	FF		F9	F2	F4
.													
.													
100	F0	E8	EA	FE	F8	F0	F6	F2	F9		E8	D0	F2
115	C3	C9	C0	C8	D5	C2	CA	C8	C4		C3	C4	C5
Line 130	80	88	83	90	87	89	82	7F	83		80	83	81
.													
.													
240	10	11	10	13	14	10	13	12	11		11	14	0F
241	0A	0C	09	1B	15	10	0A	10	10		0A	10	09

Table 4.1
Raw video data - orientation #1, field 1

	Sample											
				2 0	2 1	2 3	2 4	2 6	2 7		5 0	5 1	5 1
	0	1	2 ..	0	5	0	5	0	5	..	9	0	1
0	00	00	00	00	00	00	91	FF	FF		F2	F5	F4
1	F0	F0	F4	FF	FF	FF	FF	FF	FF		F0	EF	EF
2	F0	F1	F3	FF	FF	FF	FF	FF	FF		F8	F4	F5
.													
.													
100	EF	EF	EC	FB	FC	FA	F9	F7	F8		F0	ED	EF
115	C3	BF	C2	C4	C9	C6	C0	C2	C4		C2	C0	BE
Line 130	7F	7E	82	85	87	83	84	81	81		82	90	84
.													
.													
240	0F	0E	1D	0F	14	05	13	10	18		11	10	1F
241	11	12	02	10	1C	21	09	13	12		0A	10	0B

Table 4.2
Raw video data - orientation #1, field 2

Sample

Line	0	1	2	..	20	25	30	35	40	45	..	590	591	601
0	EE	DC	CE		F8	D5	B0	94	57	3F		1B	20	12
1	EA	D4	D6		FF	E5	B8	A2	61	44		15	1A	0F
2														
.														
.														
100														
115														
130														
.														
.														
240	FF	FE	FA		FF	FE	F1	B0	42	33		10	10	19
241	FF	FF	F9		FF	FF	F0	A2	47	2D		0C	18	0E

Table 4.3
Raw video data - orientation #2, field 1

Sample

Line	0	1	2	..	20	25	30	35	40	45	..	590	591	601
0	00	00	00		00	00	00	74	4D	2D		10	16	10
1	EC	D6	D4		FF	EE	BD	84	45	38		0C	23	1C
2														
.														
.														
100														
115														
130														
.														
.														
240														
241	FF	FF	FA		FF	FF	CE	6B	45	2F		14	12	14

Table 4.4
Raw video data - orientation #2, field 2

4.5 Clinical Installation

4.5.1 Initial Tests

Upon confirmation that the video digitizer was operating correctly in conjunction with an IBM PC, it was taken to the Glenrose Rehabilitation Hospital and connected to the HP computer system. Software was written in HP BASIC to interface the computer to the video digitizer. Initial tests were performed to ensure proper connections and operation of the video digitizer. This involved execution of the self-test procedures and examination of the results. Initially, the video digitizer did not operate. It was determined that the EOI line of the IEEE-488 bus was not being asserted properly. This problem was rectified with a modification to the BASIC program and then the self tests were executed. First, the video digitizer's video memory was tested; this test passed, then the system test was used to verify the address and page register system, again, this test passed. Finally, the video signal processor test was performed and an oscilloscope was used to verify proper A/D enable operation which was satisfactory.

4.5.2 Line Pattern

After establishing that the digitizer was operating properly, the BASIC program was modified to upload and display the video data of a digitized image. Standard line

patterns projected onto a white background were digitized. This was done to ensure that the BASIC program was reading the information properly from the video digitizer and displaying the line information in the correct order with respect to the two fields of the video frame (Figure 2.3). Photograph's of the original and digitized images are shown in Figures 4.4 and 4.5 respectively. The digitized image has good contrast, however, near the four corners of the frame, the image appears to fade in brightness and contrast. This is caused by the video camera itself and not the video digitizer hardware.

4.5.3 Noise Reduction

A mannequin (Figure 4.6) was used as a test subject. The mannequin image was digitized (Figure 4.7) and the image was processed using two noise reduction algorithms. The first technique was a frame averaging algorithm which averages the image pixels from several frames to produce a filtered image. The principle of the filtering is that noise is random and the averaging function will reduce the magnitude of the noise. In this case, four successive frames were digitized and averaged (Figure 4.8). The photograph illustrates that the filter reduces the magnitude of the noise, particularly around the shoulders, and produces a slightly contoured image. The second technique involved application of a median filter to

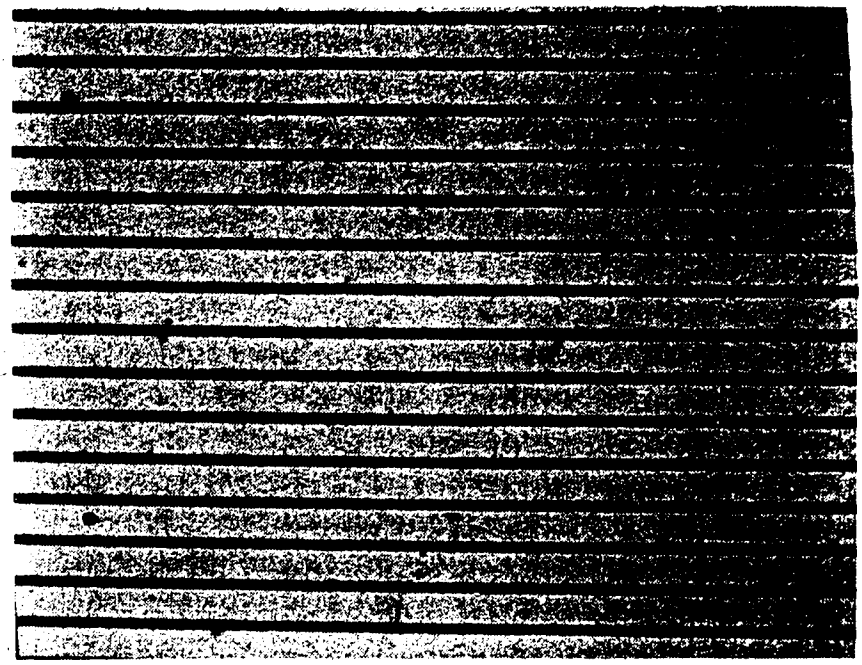


Figure 4.4

Original Horizontal Line Pattern

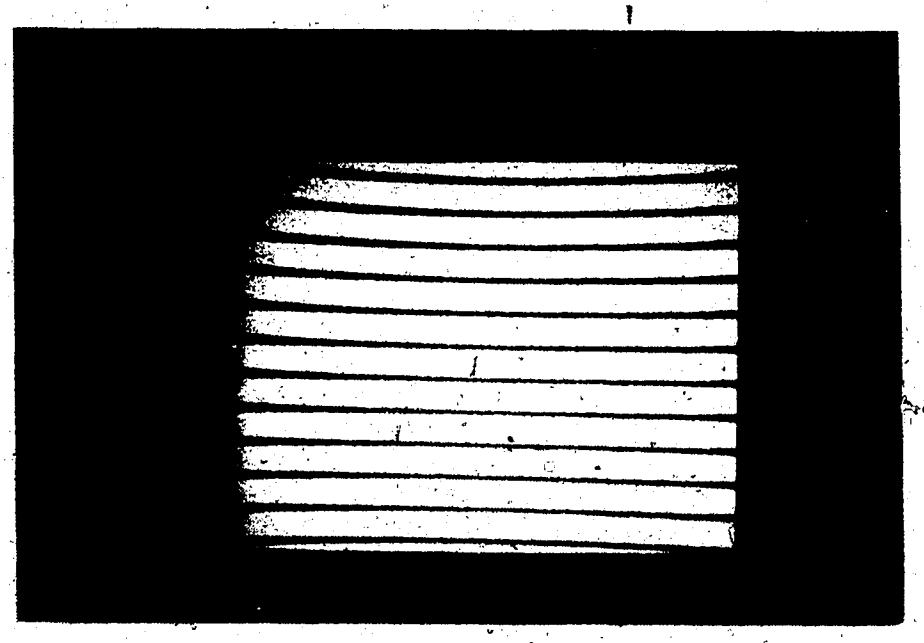


Figure 4.5

Digitized Horizontal Line Pattern

Figure 4.6

Mannequin

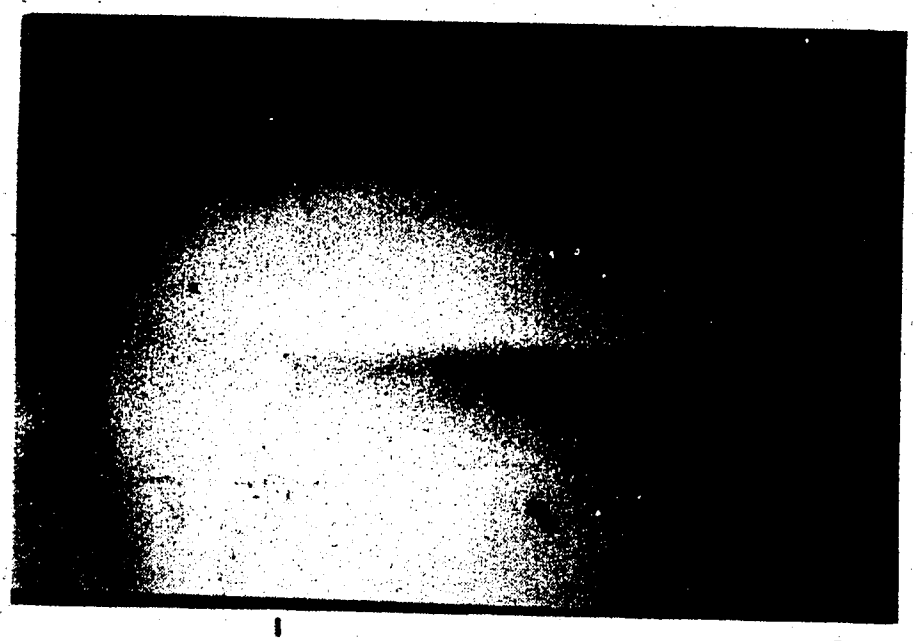


Figure 4.7

Digitized Image of Mannequin

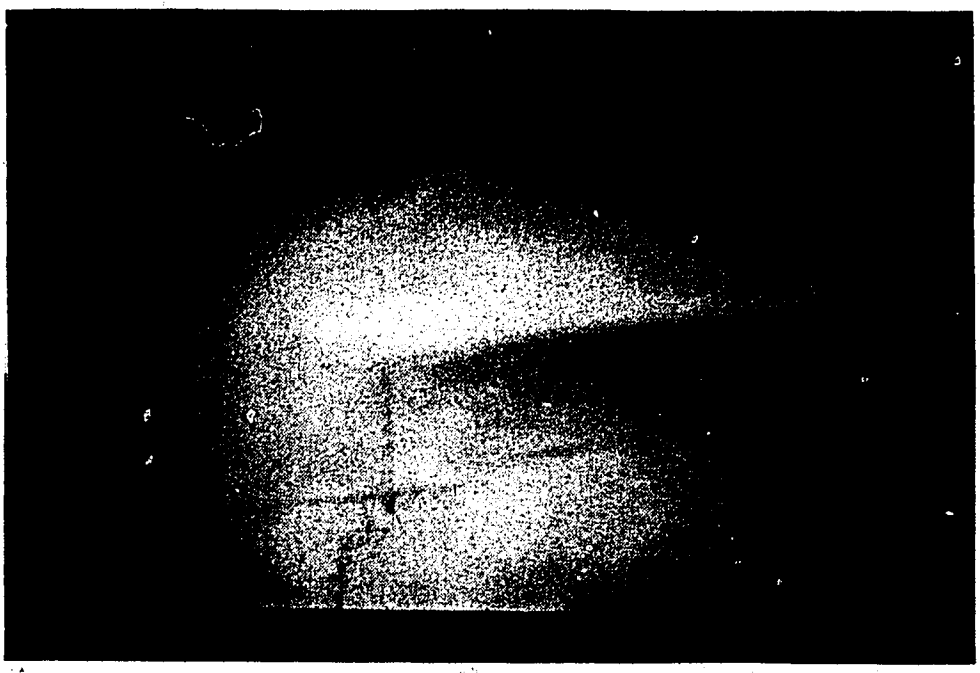


Figure 4.8

Frame Average Filtered Image of Mannequin

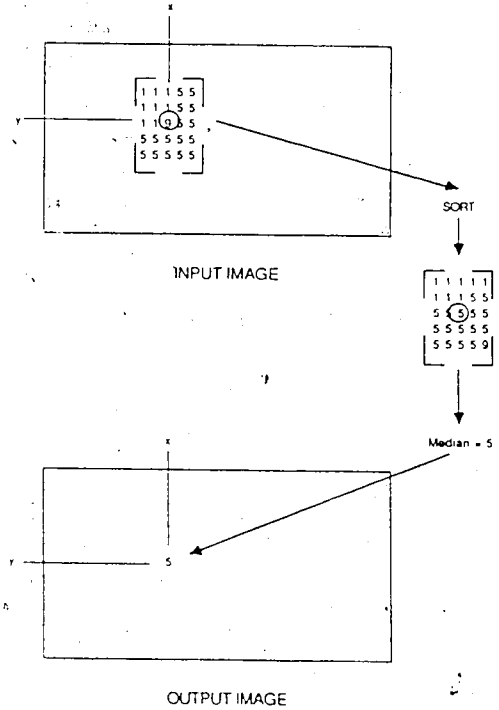


Figure 4.9

Median Filter Algorithm

the image. The median filter is a point operation that places the median value of an area of pixels around a point in the input image into the output image (Figure 4.9). Results of applying a third order median filter (ie: 3 x 3) to the mannequin image above are illustrated in Figure 4.10. Note the large reduction in noise, however, the contouring effect of this filter is significantly more profound than the frame averaging filter.

Noise reduction is important for two main reasons. First, because the prototype video digitizer is prese being used, a certain amount of noise from the digital portion of the circuit is present in the analog secti While the prototype has been designed to minimize noise, better reliability and a significant reduction in noise will occur if a printed circuit board is used. Second, the video camera itself is susceptible to a certain amount of noise.

The frame averaging filter algorithm maintains all of the original information and edge definition of the image. However, several frames of the same image, differing only in noise present, must be available for averaging, or else the edges will become smeared or blurred in the final output. This could present a problem for this clinical application because the images to be digitized are not likely to remain stationary over the time required to digitize and upload four or more images. The median filter has several advantages. It maintains edge definition, reduces noise

significantly, is fast (with respect to using a window function that would require the use of a Fourier Transform) but slower than frame averaging, and requires only one input image.

4.5.4 Mannequin with Line Pattern

A line pattern was projected onto the back of the mannequin to simulate a real clinical application and this image was digitized. The original and median filtered images are shown in Figures 4.11 and 4.12. These photographs illustrate the distortion of the horizontal lines in relation to the curvature of the back. Finally, the procedure was used on a scoliosis patient to illustrate a clinical application and how the horizontal line distortion is related to the cosmetic deformity of scoliosis. The results are shown in Figures 4.13 and 4.14.

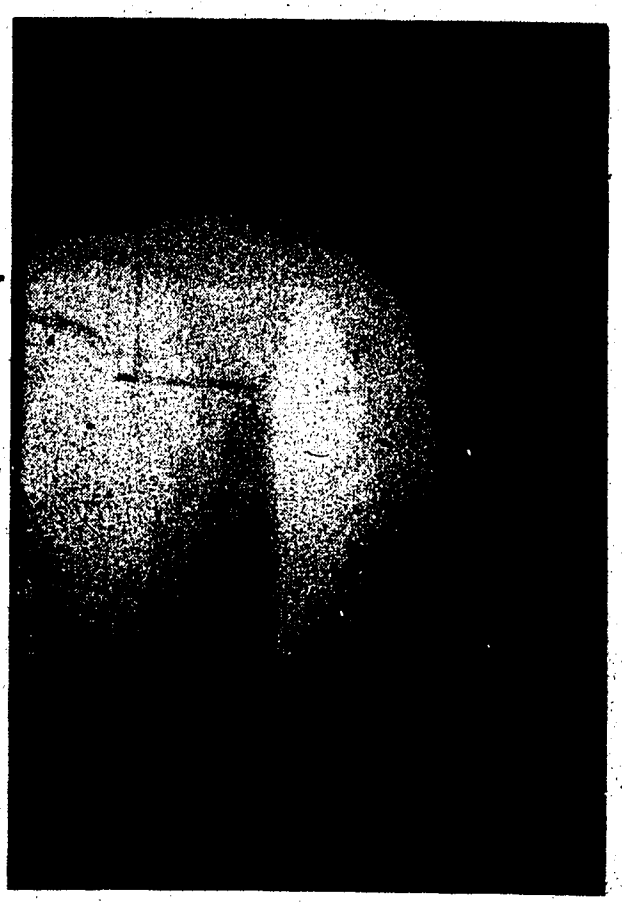


Figure 4.10

Median Filtered Image of Mannequin

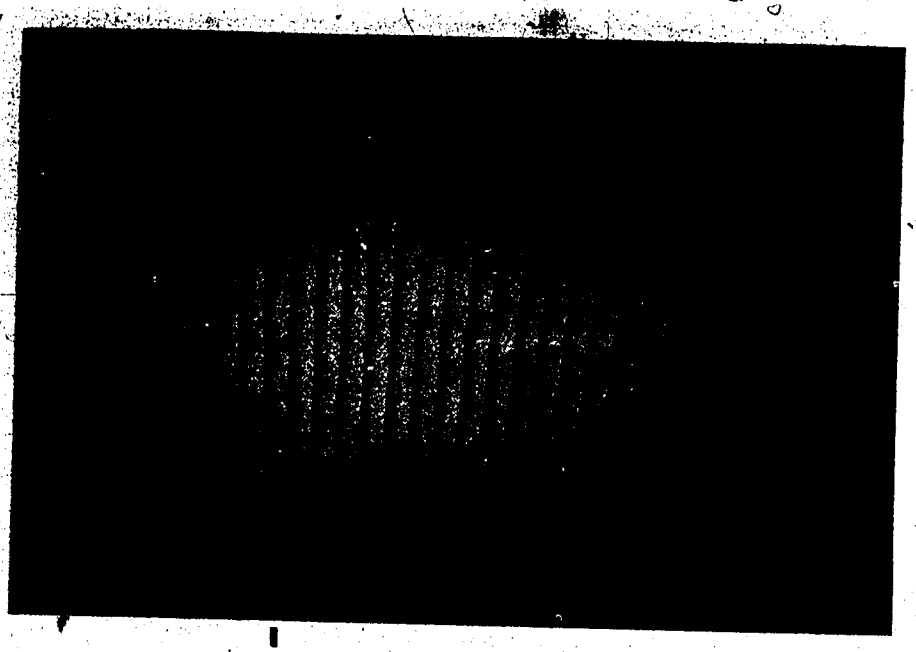


Figure 4.11

Horizontal Line Pattern on Mannequin

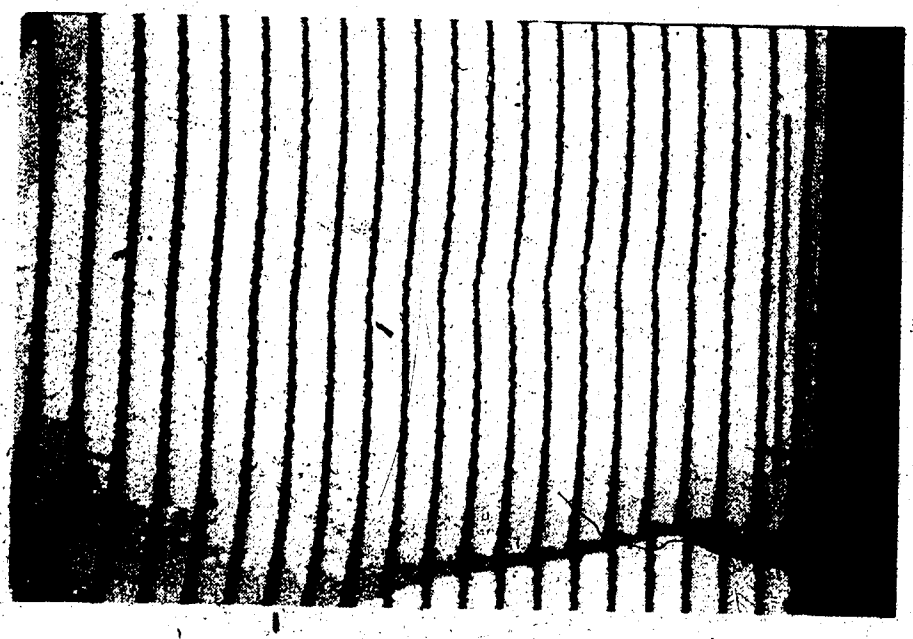


Figure 4.12

Median Filtered Image of Figure 4.11

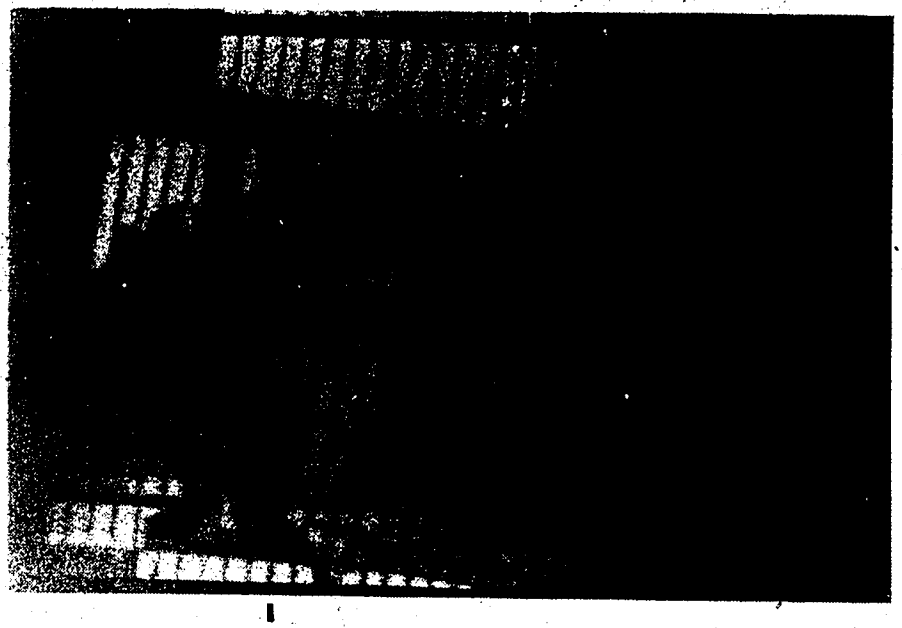


Figure 4.13

Scoliosis Patient

Figure 4.14

Median Filtered Image of Scoliosis Patient

5. Limitations and Recommendations

The described video digitizer has several important limitations that must be expressed. These limitations are: (1) it is a grey scale digitizer with 8-bit resolution; (2) real time image processing is not possible since the digitizer can 'grab' only one frame at a time and then must upload this information to a host computer system; and (3) resolution capability is limited to 512 x 484 pixels.

Enhancement to the capabilities of the present video digitizer can be realized by improvements to hardware, software or a combination of the two.

Some hardware enhancements include the elimination of the microprocessor address and data buffers. Initially, it was expected that these buses would be overloaded but the use of a gate array has eliminated the need for these buffers. The design of a printed circuit board is also recommended. The use of a printed circuit board would significantly increase reliability and decrease noise from the digitizer circuitry.

Software capabilities of the video digitizer can be expanded to provide on-board noise reduction. Additionally, an edge enhancement algorithm would contribute significantly

to the video digitizer for application to scoliosis studies.

Operational performance could be enhanced with the expansion of video RAM and software. The video RAM module presently contains enough memory to hold one video frame. Expansion could provide up to four frames of video images. With software modifications, the video digitizer could be used to digitize four successive video frames. These can be used for frame averaging and could eliminate the need for uploading the video memory between each frame acquisition.

6. Conclusion

The intent of this research was to develop an imaging system to be used in the diagnosis and monitoring of children with scoliosis. The solution required on line data acquisition of video images which can be analyzed to provide details of trunk deformity. This thesis described a system of hardware and software which provide a digitized video image to a host computer for such an analysis.

The system is compact, more reliable and cost effective due primarily to the use of readily available off-the-shelf microcomputer components and a VLSI gate array that is software controllable.

It is also believed that the described system is not limited to use in scoliosis but may be used for a variety of other imaging requirements. However, these uses must not exceed the inherent limitations of the video digitizer itself.

The described grey scale video digitizer is a safe tool for use in the measurement of back surface topography. The video digitizer will make a significant contribution to the growing field of digital imaging techniques in diagnostic and therapeutic scoliosis medicine.

References

[1] REAL, R.R., "Digital Processing of Dynamic Imagery for Photogrammetric Applications", IEEE Trans. on Instr. and Meas., Vol. IM-33, No. 1, March, 1984, pp. 45- 51.

[2] REAL, R.R., "Fast Digital Image Processing for Scoliosis Screening", Proc. - Sixth Annual Conf. IEEE Engineering in Medicine and Biology Society, Los Angeles, 1984, pp. 306 - 311.

[3] REAL, R.R., FUJIMOTO, Y., "Electronic Imaging, Processing and Storage Applied to Scoliosis Screening", Fourth Inter. Symp. on Surface Topography and Spinal Deformity, Lac Ste Marie, Quebec, Canada, 1986, in press.

[4] FLORY, R.E., "Image Acquisition Technology", IEEE Proc., Vol. 73, No. 4, April, 1985, pp. 613 - 637.

[5] Analog Devices, "Data Acquisition Databook 1982", Analog Devices Inc., 1982, Vol. 1, pp. 11-157, Vol. 2, pp. 11-27.

[6] MALMSTADT, ENKE, CROUCH, "Electronics and Instrumentation for Scientists", The Bengamin/Cummings Publishing Co. Inc., 1981, pp. 385.

[7] DUDGEON, Dan E., MERSERGAU, Russel M., "Multidimensional Digital Signal Processing", Prentic Hall Inc., 1984

[8] CIARCIA, Steven A., "Build A Gray-Scale Video Digitizer", Byte Magazine, May 1987, pp. 95 - 106, June 1987, pp. 129 - 138.

[9] Micropower Systems, "Micropower Systems for Data Acquisition and Conversion IC's", Micropower Systems, 1985, pp. 182 - 184.

[10] Motorola Inc., "Linear and Interface Integrated Circuits", Motorola Inc., 1985.

[11] CHIANG, Hai Hung, "Electronic Wave-Forming and Processing Circuits", John Wiley & Sons Inc., 1986, pp. 43 - 44, 392 - 441.

[12] KIVER, Milton S., KAUFMAN, Milton, "Television Electronics", Delmar Publishers Inc., 1983.

[13] NEC Electronics Inc., "Memory Products NEC 1986 Data Book", NEC Electronics Inc., 1986.

[14] Texas Instruments Inc., "ALS/AS Logic Data Book 1986", Texas Instruments Inc., 1986.

[15] Texas Instruments Inc., "The TTL Data Book Volume 2 1985", Texas Instruments Inc., 1985.

[16] Motorola Inc., "Motorola Microprocessors Data Manual", Motorola Inc., 1981

[17] RCA Solid State, "Integrated Circuits For Linear Applications", RCA Corp., 1986.

[18] Motorola AN973, "Avoiding Data Errors with Fast Static Rams", Motorola Inc., 1987.

[19] Motorola AN971, "Avoiding Bus Contention In Fast Access Ram Designs", Motorola Inc., 1987.

[20] LSI Logic Corp., "LSI Logic Databook & Design Manual October 1986", LSI Logic Corp., 1986.

[21] LSI Logic Corp., "LSI Logic Design Procedures", LSI

Logic Corp., 1986.

[22] Motorola Inc., "General Purpose Interface Adapter (GPIA)", Motorola Inc., 1980.

[23] ANDRESEN, Peter L., "How to Interface with the IEEE-488 Bus", Microsystems, September 1984, pp. 88 - 104.

[24] NEWROCK, Richard S., "An IEEE-488 Bus Tutorial", Microsystems, April 1983, pp. 34 - 61.

[25] Intel Corp, "80386 Hardware Reference Manual", Intel Corp., 1986.

[26] STOKES, Ian A. F., ARMSTRONG, J. G., MORELAND, M. S., "Spinal Deformity and Back Surface Asymmetry in Idiopathic Scoliosis", Journal of Orthopaedic Research, Vol. 6, No. 1, pp. 129-137.

Appendix A

Operating Instructions

(1). First connect a camera to the video digitizer's video input (Figure A.1).

(2). Connect the video output to a video monitor.

(3). Remove the 75 ohm termination jumper (only if the video monitor, step 2, has been connected).

(4). Set the GPIB DIP switches to a pre-selected GPIB address.

(5). Next, connect the power supply fo the video digitizer, the camera and monitor.

(6). Turn on the video monitor and camera.

(7). Turn on the video digitizer.

(8). Adjust the camera focus, zoom and aperture settings to

obtain a well lit, focused image.

Before proceeding, ensure that all the proper connections have been made. To digitize the video frame, send the VIDEO DIGITIZE command to the video digitizer via the GPIB (Table A.1). This command causes the video digitizer to synchronize to and digitize the video frame. The video digitizer will reply with a completion message when the frame has been digitized.

Command	Syntax	Response	Notes
Video TEST	<0><0>	<3><0>	
Video RAM TEST	<0><1>	<3><x>	x = 0: ok x = 1: fail
System TEST	<0><2>	<3><0>	
DIGITIZE	<1>	<1>	
UPLOAD	<2><xh><xl>	<2><data>	xh: line # msb xl: line # lsb data: 512 bytes
SETUP	<3><xh><xl> <yh><yl>	<3>	xh: # lines msb xl: # lines lsb yh: # samples msb yl: # samples lsb

Table A.1
Command Summary

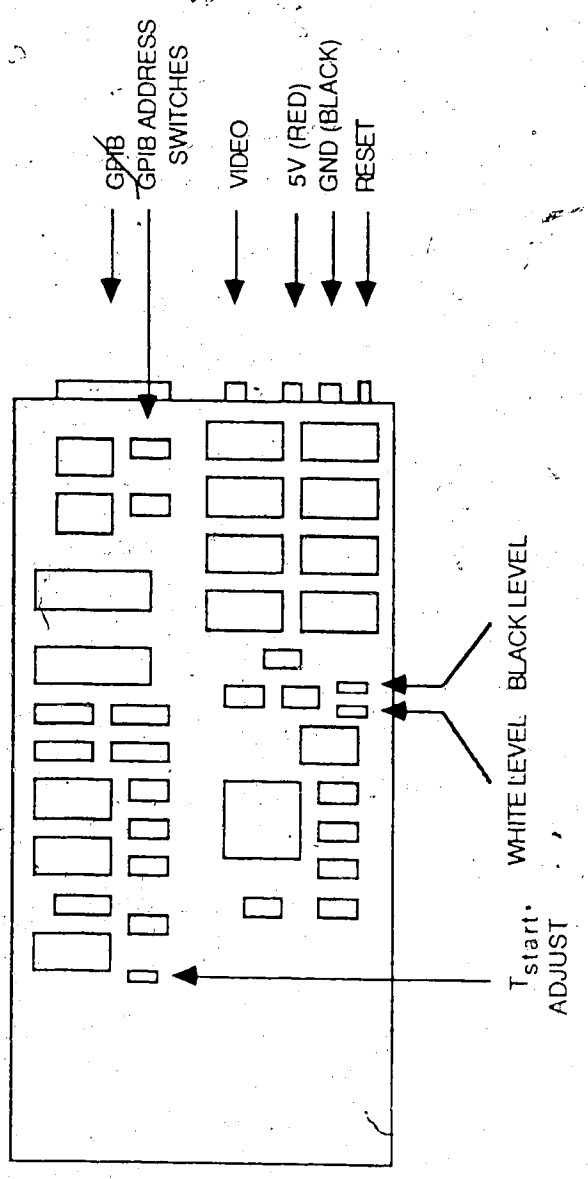


Figure A.1

Video Digitizer Prototype

After the frame has been digitized, the Upload command uploads the video memory to the host computer. This command instructs the video digitizer to send 512 bytes of data starting with the left end point of the line specified in the command. For instance, to upload the first line of video data, the UPLOAD command is sent with the line address 0. The host computer must now receive the data. Note that the total amount of data sent by the video digitizer in response to this command is 513 bytes. The first byte indicates that the information to follow is video data, and the remaining 512 bytes are the video information itself. This command is repeated with new line numbers until all required data has been uploaded.

The white and black reference levels may be adjusted to improve contrast or brightness. Use a small jewelers screwdriver to rotate the potentiometer counter clockwise t increase the reference level. Increasing the white level has the effect of making the white of the image appear darker. Increasing the black level makes the darker parts of the image appear darker.

Appendix B

Service Instructions

The video digitizer has a number of diagnostic test services built into it. These test services are shown in Table A.1. Should any of these test services fail, the following procedures should be taken.

First, repeat the test service to verify the problem. If the test service fails again, ensure the power and camera connections are correct and secure. Verify the installation or removal of the 75 ohm termination jumper as outlined in the operating instructions.

If the problem is intermittent, make note of it with any pertinent environmental details. Examine the history of these problems and consult a technician.

Appendix C

Software Listing

```
;
; NAME:  IMAGER.ASM
; DESCRIPTION:  The following software program is written in MC6809
;    assembler and is designed to run on the grey-scale video digitizer
;    hardware.
; AUTHOR:  Steven H. Slupsky
; REVISION:  April 25, 1988

;*****************************************************************************
;
; System Constants  -  System TO = 400 ns   ( 2.5 MHz. )
;
;*****************************************************************************
HSDM    EQU     %01111010       ; Horizontal Sync Detect Mode
VSDM    EQU     %01111010       ; Vertical Sync Detect Mode
EPDM    EQU     %01011010       ; Equalization Pulse Detect Mode
FDM     EQU     %00011010       ; Frame Detect Mode
HSDTO   EQU     8               ; Horizontal Sync Detect Time Out
VSDTO   EQU     49              ; Vertical Sync Detect Time Out
EPDTO   EQU     8               ; Equalization Pulse Detect Time Out
FDTO    EQU     111             ; Frame Detect Time Out
HSBF1   EQU     14              ; Horizontal Sync pulses Before Field 1
HSBF2   EQU     13              ; Horizontal Sync pulses Before Field 2
ADDLY   EQU     1               ; A/D converter Delay ( PDR )
TSTVID  EQU     0               ; Video Test
TSTRAM  EQU     1               ; Ram Test
TSTSYS  EQU     2               ; System Test
VPAGE   EQU     7               ; Top 32K Page of Video Ram

;*****************************************************************************
;
; System Equates
;
;*****************************************************************************
MPURAM  EQU     $A000           ; MPU Ram Pointer
EOSR    EQU     $BFFF           ; End Of System Ram
VRAM    EQU     $0000           ; Video Ram Pointer
```

```
OUTLEN  EQU     $A100           ; Length of OUTPUT string - 1 ( 2 bytes )
OUTBUF  EQU     $A102           ; OUTPUT buffer pointer ( 2 bytes )
INPLEN  EQU     $A104           ; Length of INPUT string - 1 ( 2 bytes )
INPBUF  EQU     $A106           ; INPUT buffer pointer ( 2 bytes )
SYSSTACK EQU    $C000           ; System Stack
USRSTACK EQU    $BF00           ; User Stack
CMDBUF  EQU     $AF00           ; 256 byte Command Buffer
RESBUF  EQU     $B000           ; 1K byte Response Buffer

DPA     EQU     $A0             ; Direct Page Address
;*******************************************************************************
; the following equates are relative to the DPR
;*******************************************************************************
SYSBYTE EQU     $00             ; System byte
TLKFLG  EQU     $01             ; Talk Flag ( 1 = Talking )
LSNFLG  EQU     $02             ; Listen Flag ( 1 = Listening )
VSDFLG  EQU     $03             ; Vertical Sync Detect Flag
EPDFLG  EQU     $03             ; Equal. Pulse Detect Flag ( shared w/ VSDFLG )
HSDFLG  EQU     $04             ; Horizontal Sync Detect Flag
GP16A   EQU     $10             ; 16 bit General Purpose Register
GP16B   EQU     $12             ;
GP16C   EQU     $14             ;
GP16D   EQU     $16             ;
TMP16   EQU     $18             ; 16 bit temporary register
LAH     EQU     $20             ; Line Address ( high byte )
LAM     EQU     $21             ; Line Address ( middle byte )
LAL     EQU     $22             ; Line Address ( low byte )
LLH     EQU     $23             ; Line Length ( high byte )
LLL     EQU     $24             ; Line Length ( low byte )
FLD     EQU     $25             ; field
LCNT    EQU     $26             ; # of lines to be digitized in a field
GP8A    EQU     $28             ; 8 bit General Purpose Register
DCLFLG  EQU     $29             ; Device Clear Flag

;*******************************************************************************
;
;  GPIA Equates
;
;*******************************************************************************
GPISR   EQU     $C008           ; Interrupt Status Register ( R )
GPIMR   EQU     $C008           ; Interrupt Mask Register ( W )
GPCSR   EQU     $C009           ; Command Status Register ( R )
GPASR   EQU     $C00A           ; Address Status Register ( R )
GPAMR   EQU     $C00A           ; Address Mode Register ( W )
GPACR   EQU     $C00B           ; Auxilary Command Register ( R/W )
GPASWR  EQU     $C00C           ; Address Switch Register ( R )
GPAR    EQU     $C00C           ; Address Register ( W )
GPSPR   EQU     $C00D           ; Serial Poll Register ( R/W )
GPCPTR  EQU     $C00E           ; Command Pass Through Register ( R )
GPPPR   EQU     $C00E           ; Parallel Poll Register ( W )
GPDIR   EQU     $C00F           ; Data In Register ( R )
GPDOR   EQU     $C00F           ; Data Out Register ( W )
```

```
;*****************************************************************************
;
; PIM Equates
;
;*****************************************************************************
PIMCR1  EQU     $C010           ; Control Register 1 ( Write )
PIMCR3  EQU     $C010           ; Control Register 3 ( Write )
PIMCR2  EQU     $C011           ; Control Register 2 ( Write )
PIMSR   EQU     $C011           ; Status Register ( Read )
PIMT1   EQU     $C012           ; Timer 1
PIMT2   EQU     $C014           ; Timer 2
PIMT3   EQU     $C016           ; Timer 3

;*****************************************************************************
;
; PADIM Equates
;
;*****************************************************************************
PDMVCR1 EQU     $C000           ; Video Control Register 1 ( Write )
PDMSR   EQU     $C000           ; Status Register ( Read )
PDMVCR2 EQU     $C001           ; Video Control Register 2 ( Write )
PDMBARL EQU     $C002           ; Base Address Register - Low byte ( Write )
PDMBARM EQU     $C003           ; Base Address Register - Middle byte ( Write )
PDMBARH EQU     $C004           ; Base Address Register - High byte ( Write )
PDMCRL  EQU     $C005           ; Count Register - Low byte ( Write )
PDMCRH  EQU     $C006           ; Count Register - High byte ( Write )
PDMPDR  EQU     $C007           ; Programmable Delay Register ( Write )

;*****************************************************************************
;
; PORT Equates
;
;*****************************************************************************
PORTA   EQU     $C018           ; Blanking Circuit


;*****************************************************************************
;
; RESET Interrupt Vectored Here
;
;*****************************************************************************
        ORG     $E000
RESET   LDS     #SYSSTACK       ; Set up System Stack Pointer

;*****************************************************************************
;
; MPU RAM Test
;
;*****************************************************************************
RAMTEST LDX     #$1FFF          ; 8K of User Ram
```

```
TSTBYTE LDA     #0
        STA     MEMRAM,X
        CMPA    MEMRAM,X        ; Test byte for 0
        BNE     TB2             ; If not zero, error
        LDA     #1
TB1     STA     MEMRAM,X        ; 'walk' a one through byte
        CMPA    MEMRAM,X
        BNE     TB2
        LSLA
        BCC     TB1
        LEAX    ,-X             ; check next byte
        CMPX    #0
        BPL     TSTBYTE
        BRA     TB3

TB2     LDA     $DEAD           ; report error
        BRA     TB2

TB3     EQU     $


;*******************************************************************************
;
;  Initialize GPIA
;
;*******************************************************************************
INITGPIA LDA    #%10000000
        STA     GPACR           ; Reset the GPIA
        LDA     GPASWR          ; Read DIP switches
        ANDA    #$1F            ; Mask off upper 3 bits
        STA     GPAR
        CLRA
        STA     GPACR           ; Clear Reset
        STA     GPPPR           ; Set parallel pole register to zero
        STA     GPAMR           ; Select no options

        JSR     INIT            ; Initialize Device


;*******************************************************************************
;
;  Main Program Loop
;
;*******************************************************************************
LP1     LDX     #$00FF          ; Max. allowable input chars.
        STX     INPLEN
        LDX     #CMDBUF         ; Put input string in command buffer
        STX     INPBUF
        LDA     #%10000101      ; Enable IRQ, CMD and BI interrupts
        STA     GPIMR
        LDA     #1              ; Start listening
        STA     LSNFLG
```

```
IP2     LDA     LSNFLG          ; wait for command
        BNE     IP2
        LDA     CMDBUF          ; get command
        CMPA    #MAXCMD         ; is command valid?
        BHI     INVALID
        ASLA
        LDX     #CMDTBL
        JSR     [A,X]           ; go execute command
        BRA     IP1

INVALID LDA     #80             ; Illegal Command Response byte
        STA     RESBUF
        LDX     #0              ; Length = 1
        JSR     SNDRES          ; Send Response
        BRA     IP1

MAXCMD  EQU     3               ; Jump Table
CMDTBL  DW      TESTMODE, DIAG, UPLOAD, SETUP


;*******************************************************************************
;
; INITIALIZE DEVICE
;
;*******************************************************************************
INIT    EQU     $

;*** Init PIM
        LDA     #%00000001      ; Select CR1
        STA     PIMCR2
        STA     PIMCR1          ; Put PIM in RESET mode

;*** Init Blanking circuit
        CLR     PORTA           ; Initialize PORTA

;*** Init PADIM
        LDA     #0              ; Release PADIM reset
        STA     PDMCR1
        LDA     #%10000000      ; Program START MASK
        STA     PDMCR2
        LDA     #ADDLY          ; Program Delay Register
        STA     PDMDR
        LDX     #512            ; Init line length
        STX     LTH
        LDA     LTL             ; Write Length of line to PADIM
        STA     PDMCRL
        LDA     LTH
        STA     PDMCRH

;*** Init System
```

```
        LDU     #USRSTACK       ; Initialize User Stack
        LDA     #DPA            ; get Direct Page
        TFR     A,DP
        CLR     SYSBYTE         ; Save System Byte
        ANDCC   #%10101111      ; Turn on FIRQ and IRQ

        LDX     #242            ; Init line count
        STX     LCNT

;       JSR     VRAMTST         ; Test Video Ram
;       LDA     SYSBYTE         ; Update System Byte
;       STA     SYSBYTE

        RTS


;*****************************************************************************
;
; TESTMODE
;       Description:  Test Video Ram
;               INPUT:  none
;               OUTPUT: none
;               ALTERS: Acc A, B, X, CC
;
;*****************************************************************************

TESTMODE LDA CMDBUF+1           ; Read Test data
        CMPA    #TSTSYS         ; Which Test ?
        BEQ     SYSTST
        CMPA    #TSTRAM
        BEQ     RAMTST
        CMPA    #TSTVID
        BEQ     VIDTST
        LDA     #$80
        STA     RESBUF
        LDX     #0              ; Length = 1
        JSR     SNDRES
        RTS

;*****************************************************************************

VIDTST  CLR     PORTA           ; Reset PORT A
        LDA     #255
        STA     GPBA            ; Init count
                                ; Now, START DIGITIZING
        LDA     #1              ; Select Timer 1
        STA     PTMCR2
        LDA     #HSDM           ; Set Horizontal Sync Detect Mode
        STA     PTMCR1
        LDA     #HSDTO          ; Set Horizontal Sync Detect Time Out
        STX     PTMT1
        CLR     PTMCR2          ; Select Timer 3
```

```
        LDA     #FDM            ; Set Frame Detect Mode
        STA     PIMCR3
        LDX     #FDTO           ; Set Frame Detect Time Out
        STX     PIMT3
        JSR     FRMSYNC         ; Synchronize to frame
        LDA     #VSDM           ; Set Vertical Sync Detect Mode
        STA     PIMCR2
        LDX     #VSDTO          ; Set Vertical Sync Detect Time Out
        STX     PIMT2
VIDIST1 CLR     LAH             ; Set up address of first line ( = 0 )
        CLR     LAM
        CLR     LAL
        LDA     #HSBF1          ; Get number of HS before Field 1
        JSR     DIGFLD          ; Digitize Field 1
        LDA     #HSBF2          ; Get number of HS before field 2
        JSR     DIGFLD          ; Digitize Field 2

        DEC     GP8A
        BNE     VIDIST1

        LDA     #1
        STA     PIMCR2
        STA     PIMCR1          ; Reset PIM
        LDA     #0
        JSR     SSB             ; Send System Byte
        RTS

;*******************************************************************************

RAMIST  JSR     VRAMIST
        JSR     SSB             ; Send System Byte
        RTS

;*******************************************************************************

SYSIST  LDB     #0              ; Write Byte Pattern into VIDEO RAM
SYSIST1 STB     PDMVCR1         ; Select video page
        LDA     #0
        LDX     #0
SYSIST2 STD     0,X             ; store value in video RAM
        INCA
        LEAX    2,X
        CMPX    #$8000          ; have we done all 32K of this page ?
        BNE     SYSIST2
        INCB
        CMPB    #8              ; have we done all 8 pages ?
        BNE     SYSIST1

        LDA     #0
        JSR     SSB             ; Send System Byte
        RTS
```

```
;*****************************************************************************
;
; Digitize
;       Description:
;               INPUT: none
;               OUTPUT: none
;               ALTERS: A, B ( D ), X, CC
;
;*****************************************************************************
DIG     CLR     IAH             ; Set up address for frame ( = 0 )
        CLR     IAM
        CLR     IAL
        JSR     DIGFRM          ; Digitize Frame
        LDA     #1              ; Response code
        STA     RESBUF
        LDX     #0              ; Length = 1
        JSR     SNDRES          ; Send Response
        RTS

;*****************************************************************************
;
; UPLOAD
;       Description: Upload Video Memory to Host computer
;               INPUT: none
;               OUTPUT: none
;               ALTERS: Acc A, B, X, Y, CC
;
;*****************************************************************************
UPLOAD  LDA     #2
        STA     RESBUF
        LDX     CMDBUF+1        ; Get line number
        LDB     #6              ; compute video page
        JSR     LSR16
        TFR     X,D
        STB     PDMVCR1         ; select video page
        LDX     CMDBUF+1        ; Get line number
        LDB     #9              ; compute line address
        JSR     LSL16
        TFR     X,D
        ANDA    #%01111111      ; Mask off video page
        LDX     #0
UL1     LDY     D,X             ; Get Video Memory
        STY     RESBUF+1,X      ; Store Video Memory in Response Buffer
        LEAX    2,X             ; Inc x by 2
        CMPX    #512
        BNE     UL1
        LDX     #512            ; length = 513
        JSR     SNDRES          ; Send Response ( and line of data )
        RTS
```

```
;*****************************************************************************
;
; SETUP
;       Description:  Set Digitize Parameters
;                     INPUT:  none
;                     OUTPUT: none
;                     ALTERS: Acc A, B, X, Y, CC
;
;*****************************************************************************
SETUP   LDX     CMDBUF+1        ; Get Line Count Parameter
        STX     LCNT
        LDX     CMDBUF+3        ; Get # of conversions per line
        STX     LCH
        LDA     LCL
        STA     PDMCRL
        LDA     LCH
        STA     PDMCRH

        LDA     #3              ; Response
        STA     RESBUF
        LDX     #0              ; Length = 1
        JSR     SNDRES          ; Send Response
        RTS


;*****************************************************************************
;
; Digitize Frame Subroutine
;       Description:  Digitize a video frame
;                     INPUT:  none
;                     OUTPUT: none
;                     ALTERS: A, B ( D ), X, CC
;
;*****************************************************************************
DIGFRM  LDA     #1              ; Select Timer 1
        STA     PIMCR2
        LDA     #HSDM           ; Set Horizontal Sync Detect Mode
        STA     PIMCR1
        LDX     #HSDTO          ; Set Horizontal Sync Detect Time Out
        STX     PIMT1
        CLR     PIMCR2          ; Select Timer 3
        LDA     #FDM            ; Set Frame Detect Mode
        STA     PIMCR3
        LDX     #FDTO           ; Set Frame Detect Time Out
        STX     PIMT3
        JSR     FRMSYNC         ; Synchronize to frame
        LDA     #VSDM           ; Set Vertical Sync Detect Mode
        STA     PIMCR2
        LDX     #VSDTO          ; Set Vertical Sync Detect Time Out
        STX     PIMT2
        LDA     #HSBF1          ; Get number of HS before Field 1
        JSR     DIGFLD          ; Digitize Field 1
```

```
        LDA     #HSBF2          ; Get number of HS before field 2
        JSR     DIGFLD          ; Digitize Field 2

        LDA     #1              ; Select Timer 1
        STA     PIMCR2
        STA     PIMCR1          ; Reset PIM
        RTS

;*****************************************************************************
;
;  Frame Synchronize Subroutine
;       Description:  Sync's the computer to the beginning of a video frame
;               INPUT:  none
;               OUTPUT: none
;               ALTERS: A, X, CC
;
;*****************************************************************************
FRMSYNC LDA     #VSDM           ; Set Vertical Sync Detect Mode
        STA     PIMCR2
        LDX     #VSDTO          ; Set Vertical Sync Detect Time Out
        STX     PIMT2
        CLR     VSDFLG
FSLP2   LDA     VSDFLG          ; Wait for Vertical Sync
        BEQ     FSLP2
        CLR     HSDFLG
FSLP3   LDA     HSDFLG          ; Wait for 12 Horizontal Sync pulses
        CMPA    #12
        BNE     FSLP3
        LDX     #FDTO           ; Clear Timer 3 interrupt flag
        STX     PIMT3
        LDA     #EPDM           ; Set Equalization Pulse Detect Mode
        STA     PIMCR2
        LDX     #EPDTO          ; Set Equalization Pulse Detect Time Out
        STX     PIMT2
        CLR     EPDFLG
FSLP4   LDA     EPDFLG          ; Wait for Equalization Pulse
        BEQ     FSLP4
        LDA     PIMSR           ; Frame Detect ?
        BITA    #%00000100
        BEQ     FSLP5
        LDX     PIMT3           ; Reset Interrupt bit
        BRA     FRMSYNC         ; At end of first field, go back.
FSLP5   RTS

;*****************************************************************************
;
;  Digitize Field Subroutine
;       Description:  The routine Digitizes a complete field
;               INPUT:  Acc A = Number of HS Before Field
;                       IA and IL must be initialized
;               OUTPUT: none
;               ALTERS: A, B ( D ) , X, CC
```

```
;
;*****************************************************************************
DIGFLD  STA     FLD
        LDA     IAL             ; Write Address of Line 1 to PADIM
        STA     PDMBARL
        LDA     IAM
        STA     PDMBARM
        LDA     IAH
        STA     PDMBARH
        CLR     VSDFLG
DFDLP1  LDA     VSDFLG          ; Wait For Vertical Sync
        BEQ     DFDLP1
        CLR     HSDFLG
DFDLP2  LDA     HSDFLG          ; Wait for x Horizontal Sync pulses
        CMPA    FLD
        BNE     DFDLP2
        LDA     #%00000001      ; Enable Blanking Circuit
        STA     PORTA
        LDX     LCNT            ; Get Number of Lines per field
DFDLP3  CLR     HSDFLG
DFDLP4  LDA     HSDFLG          ; Wait for HSD
        BEQ     DFDLP4
        LDD     IAH             ; Update Line Address
        ADDD    #$0002          ; IL * 512
        STD     IAH
        STB     PDMBARM         ; Update PADIM
        STA     PDMBARH
        LEAX    0,-X            ; dec x
        BNE     DFDLP3

        CLR     PORTA           ; Disable Blanking Circuit
        RTS

;*****************************************************************************
;
; Video Ram Test
;       Description:  Test Video Ram
;               INPUT:  none
;               OUTPUT:  Acc A = return code;  0 = ok,  1 = bad memory cell
;                        B = video page of bad memory cell
;                        X = address of bad memory cell
;               ALTERS:  A, B, ( D ), X, CC
;
;*****************************************************************************
VRAMTST LDB     #VPAGE          ; get top video page
VRT1    STB     PDMVCR1
        LDX     #$7FFF          ; 32K of RAM
VRT2    LDA     #0
        STA     VRAM,X
        CMPA    VRAM,X          ; test for 0
        BNE     VRT4
        LDA     #1
```

```
VRT3    STA     VRAM,X          ; 'walk' a one through byte
        CMPA    VRAM,X
        BNE     VRT4
        LSLA
        BCC     VRT3
        LEAX    ,-X             ; check next byte
        CMPX    #0
        BPL     VRT2
        DECB                    ; check next page
        BPL     VRT1
        CLRA                    ; return code = 0
        BRA     VRT5

VRT4    LDA     #1              ; return code = 1

VRT5    RTS

;*****************************************************************************
;
;  Send System Byte
;       Description:  Send the SYSTEM BYTE to the host computer
;               INPUT:  Acc A = SYSTEM BYTE
;               OUTPUT:  none
;               ALTERS:  A, X, CC
;
;*****************************************************************************
SSB     STA     RESBUF+1
        LDA     #3
        STA     RESBUF          ; Response Byte
        LDX     #1              ; Length = 2
        JSR     SNDRES          ; Send Response
        RTS

;*****************************************************************************
;
;  Send Response
;       Description:  Send Response to host computer
;               INPUT:  X = Length of response
;               OUTPUT:  none
;               ALTERS:  A, X, CC
;
;*****************************************************************************
SNDRES  STX     OUTLEN          ; Store Length
        LDX     #RESBUF         ; get address of response
        STX     OUTBUF
        LDA     #1              ; Start talking
        STA     TLKFLG
        LDA     #%11000100      ; Enable IRQ, BO and CMD interrupts
        STA     GPIMR
SR1     LDA     TLKFLG          ; wait for System byte to be sent to host
        BNE     SR1
        RTS
```

```
;*****************************************************************************
;
; LSR16
;       Description:  Shift Right 16 bit
;                     INPUT:  X = 16 bit value to shift, Acc B = # of shifts
;                     OUTPUT: X = shifted result
;                     ALTERS: Acc B, X, CC
;
;*****************************************************************************
LSR16   PSHU    A               ; Save A
        CLRA
        EXG     D,X             ; Put count in X and num in D
LSRLP1  LSRA                    ; Shift num
        RORB
        LEAX    ,-X             ; dec X
        BNE     LSRLP1
        TFR     D,X             ; Put result in X
        PULU    A               ; restore A
        RTS

;*****************************************************************************
;
; LSL16
;       Description:  Shift Left 16 bit
;                     INPUT:  X = 16 bit value to shift, Acc B = # of shifts
;                     OUTPUT: X = shifted number
;                     ALTERS: Acc B, X, CC
;
;*****************************************************************************
LSL16   PSHU    A               ; Save A
        CLRA
        EXG     D,X             ; Put count in X and num in D
LSLLP1  LSLB                    ; Shift num
        ROLA
        LEAX    ,-X             ; dec X
        BNE     LSLLP1
        TFR     D,X             ; Put result in X
        PULU    A               ; restore A
        RTS


;*****************************************************************************
;
; IRQ Interrupt Service Routine(s)
;
;*****************************************************************************
        ORG     $FD00
IRQSR   LDA     GPISR           ; Did GPIA generate interrupt?
        BMI     GPIA
IRQEX   RTI                     ; ignore other interrupts
```

```
GPIA    BITA    #%01000000      ; BO interrupt
        BNE     BO
        BITA    #%00000001      ; BI interrupt
        BNE     BI
        BITA    #%00000100      ; CMD interrupt
        LBNE    CMD
        BRA     IRQEX

;*******************************************************************************
;
; BO  ( Byte Out ) Interrupt Service Routine
;       Description:  Write a string to the GPIA
;               INPUT:  OUTBUF = pointer to an output buffer
;                       OUTLEN = length of the output string
;               OUTPUT: none
;
;*******************************************************************************
BO      LDA     #%10000100      ; Mask out further BO interrupts
        STA     GPIMR
        LDD     OUTBUF          ; Compute end of String pointer
        ADDD    OUTLEN
        STD     TMP16           ; Store value
        LDX     OUTBUF
BOLP1   LDA     GPISR           ; Is GPIA ready for next byte ?
        BITA    #%01000000
        BEQ     BOLP1
        CMPX    TMP16           ; Last byte ?
        BEQ     LASTBYTE
        LDA     0,X             ; Yes, get byte
        STA     GPDOR           ; Output byte
        LEAX    1,X             ; Update pointer
        BRA     BOLP1
LASTBYTE LDA    #%00100010      ; Set feoi and dacd bits
        STA     GPACR
        LDA     0,X             ; Get last byte
        STA     GPDOR           ; output byte

BOLP3   LDA     GPASR           ; Monitor AIN bit
        BITA    #%00010000
        BEQ     BOLP3
        LDA     #%00010000      ; Reset dacd, set dacr
        STA     GPACR

        CLR     TLKFLG          ; Update Talk Flag ( 0 = not talking )
        RTI

;*******************************************************************************
;
; BI ( Byte In ) Interrupt Service Routine
;       Description:  Receive a string from the GPIA
```

```
;               INPUT:  INPBUF = pointer an input buffer
;                       INPLEN = Max. allowable input length
;               OUTPUT: INPLEN = Actual length of received input
;
;*****************************************************************************
BI      LDA     #%10000100      ; Mask out further BI interrupts
        STA     GPIMR
        LDY     #$FFFF          ; Set length = 0
        LDX     INPBUF
BILP1   LDA     GPISR           ; Is there a byte to Read ?
        BITA    #%00000001
        BEQ     BILP1
        BITA    #%00000010      ; Is this the last byte ?
        BNE     BILST
        LDA     GPDIR           ; Read data
        STA     0,X
        LEAX    1,X             ; Inc X, Update pointer
        LEAY    1,Y             ; inc y, increment length
        CMPY    INPLEN          ; Is length too long ?
        BHS     BIOF
        BRA     BILP1
BILST   LDA     GPDIR           ; Read last byte
        STA     0,X
        LEAY    1,Y
        STY     INPLEN
        BRA     BIEND
BIOF    LDA     GPISR           ; wait till END
        BITA    #%00000010
        BEQ     BIOF
        LDA     GPDIR           ; Read data
        STY     INPLEN

BIEND   CLR     LSNFLG          ; Update Listen Flag ( 0 = not listening )
        RTI


;*****************************************************************************
;
; CMD ( Command ) Interrupt Service Routine
;
;*****************************************************************************
CMD     LDA     GPCSR           ; Get CMD status
        BITA    #%00000010      ; Is GPIA in DCAS ?
        BEQ     CMDEND

        LDS     #SYSSTACK       ; Re-Initialize System Stack
        JSR     INIT            ; Initialize Device
        LDA     #%00010000      ; Release handshake that was held off
        STA     GPACR           ; by DCL command ( If there was one )
        JMP     LP1             ; Go to start of main loop

CMDEND  RTI
```

```
;*******************************************************************************
;
; FIRQ Interrupt Service Routine(s)
;               ALTERS:  B, Y
;
;*******************************************************************************
        ORG     $FE00
FIRQSR  LDB     PIMSR           ; Did the PIM cause an interrupt?
        BMI     PIM
FIRQEX  RTI

PIM     BITB    #%00000010      ; Timer 2 ?
        BNE     T2
        BITB    #%00000001      ; Timer 1 ?
        BNE     T1
        BRA     FIRQEX

T1      LDY     PIMT1           ; Reset Interrupt bit
        INC     HSDFLG          ; inc Horizontal Sync Detect Flag
        RTI

T2      LDY     PIMT2           ; Reset Interrupt bit
        INC     VSDFLG          ; inc Vertical Sync / Equal. Pulse Detect Flag
        RTI


;*******************************************************************************
;
; Bad Interrupt
;
;*******************************************************************************
        ORG     $FFE0
BADINT  BRA     BADINT


;*******************************************************************************
;
; Interrupt Vectors
;
;*******************************************************************************
        ORG     $FFF0
        DW      $0000
SWI3    DW      BADINT
SWI2    DW      BADINT
FIRQ    DW      FIRQSR
IRQ     DW      IRQSR
SWI     DW      BADINT
NMI     DW      BADINT
```

```
RSTVEC  DW      RESET

        END
```

Appendix D

Peripheral A/D Interface Module (PADIM)
Specification Sheet

Purpose: This module provides a interface between a high-speed (FLASH) A/D converter and a slow microprocessor.

Description: The PADIM controls the A/D converter and up to 1 Megabyte of RAM memory. Programming the device is similar to programming a DMA device; a start address, the BASE ADDRESS (20 bits), and the number of bytes to be transferred, the COUNT (16 bits), are loaded into internal registers. The transfer can be started either synchronously by the microprocessor or by a low to high transition on the ASYNCSTART input of the device. The memory transfer starts and continues until the transfer is complete or a low to high transition occurs on the ASYNCSTOP input.

Interface to the microprocessor is through a Interrupt Request Strobe and status register. The device may be configured to generate an interrupt on the completion of a memory transfer.

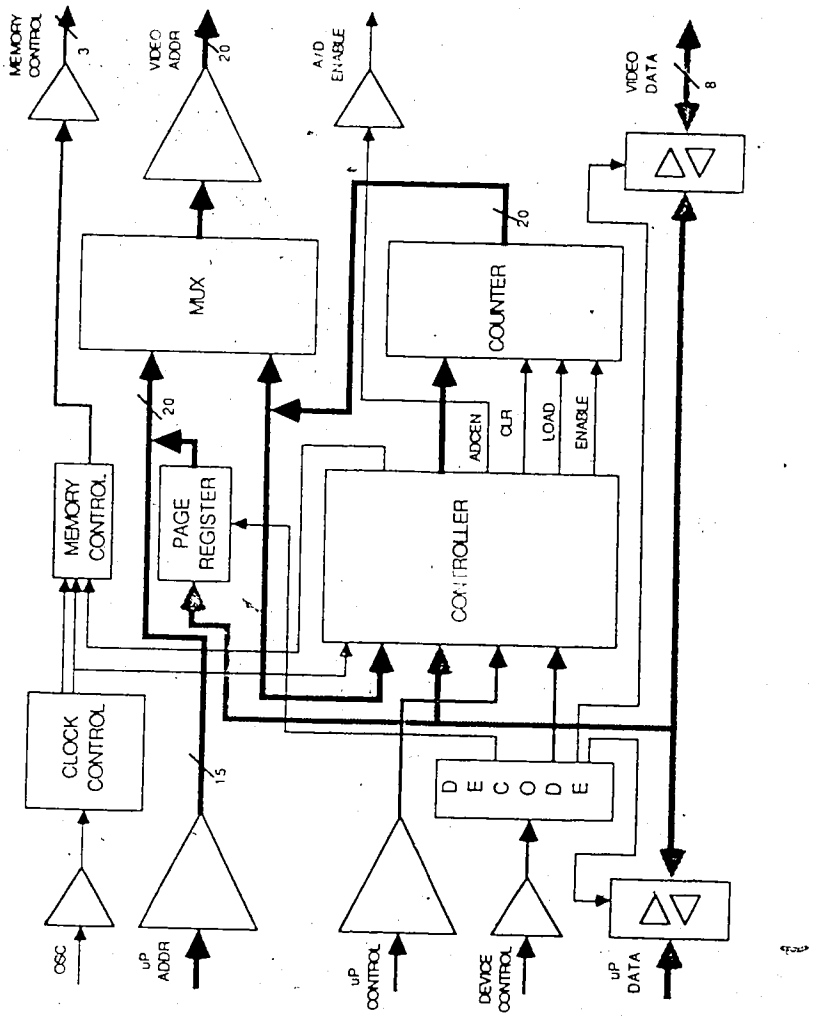


Figure D.1

Block Diagram of PADIM

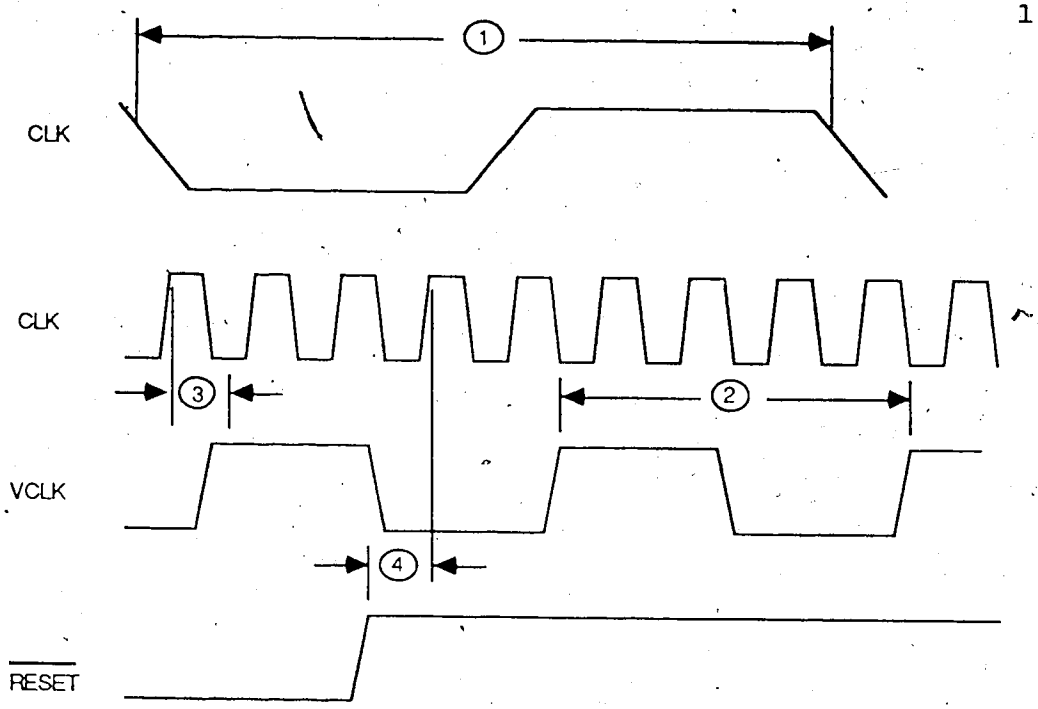


Figure D.2

Reset Timing

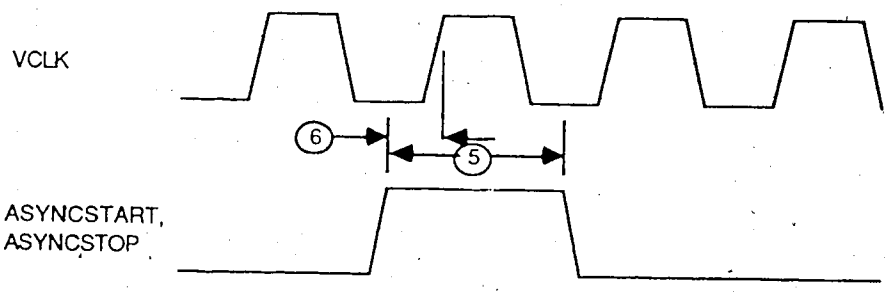


Figure D.3

Memory Transfer Control Timing

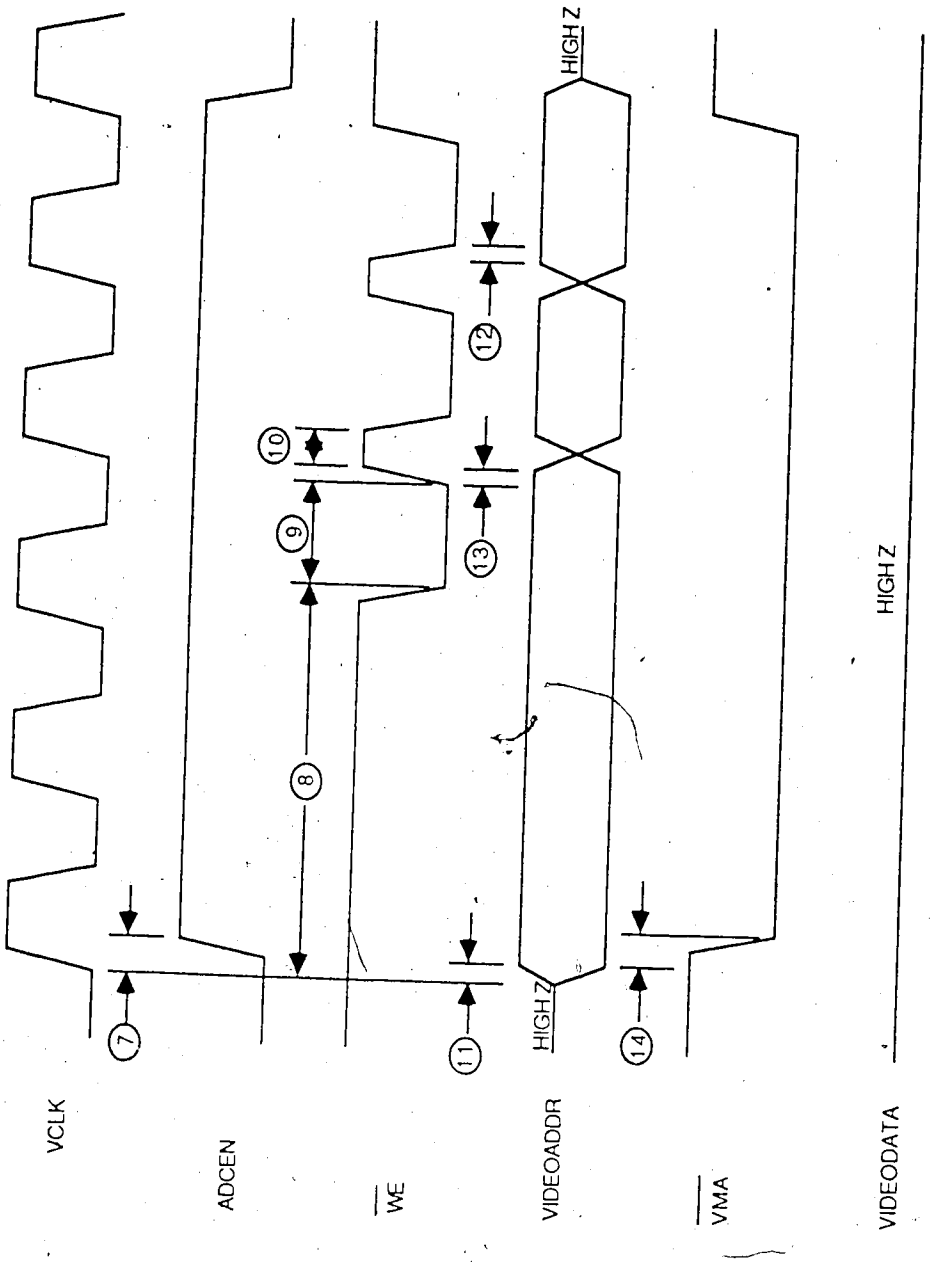


Figure D.4

Memory Transfer Cycle Timing

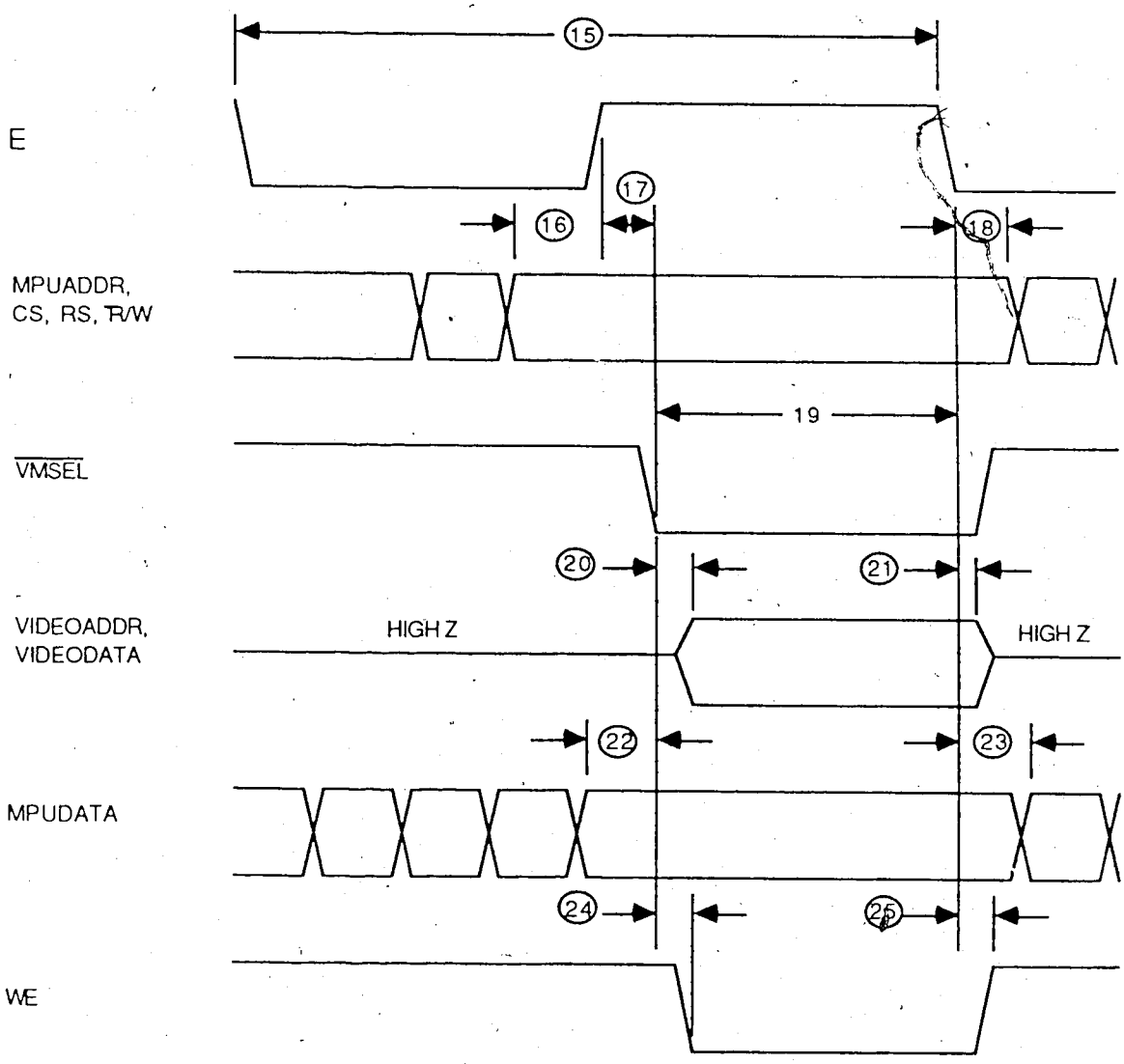


Figure D.5

Microprocessor Write Cycle Timing

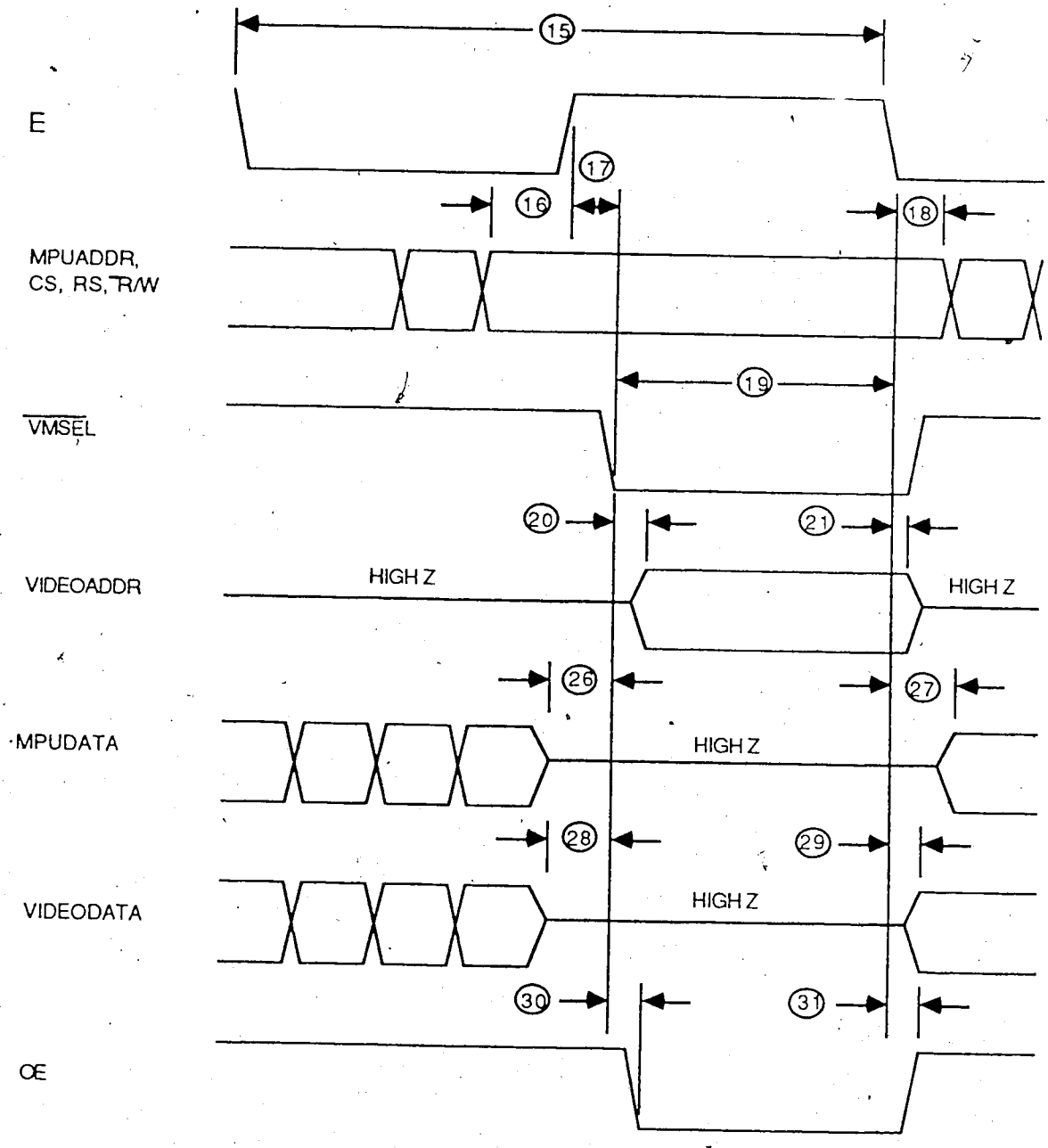


Figure D.6

Microprocessor Read Cycle Timing

#	Characteristics	min	max	unit
1	CLK cycle time	25		ns
2	VCLK cycle time	100		ns
3	Delay, CLK to VCLK	20		ns
4	Reset setup time	5		ns
5	Pulse width, ASYNCSTART or ASYNCSTOP high	100		ns
6	ASYNCSTART or ASYNCSTOP	100		ns
	setup time	10		ns
7	Delay, VCLK to ADCEN	25		ns
8	Delay, VCLK to WE	35		ns
9	Pulse width, WE low	70	80	ns
10	Pulse width, WE high	20	30	ns
11	Delay, VCLK to VIDEOADDR		40	ns
12	VIDEOADDR setup time	10		ns
13	VIDEOADDR hold time	10		ns
14	VCLK to VMA	40		ns
15	E cycle time	250		ns
16	MPUADDR setup time	0		ns
17	VMSEL delay time	0		ns
18	MPUADDR hold time	10		ns
19	Pulse width, VMSEL low	100		ns
20	VIDEOADDR delay time	10		ns
21	VIDEOADDR hold time	10		ns
22	Write MPUDATA setup time	0		ns
23	Write MPUDATA hold time	10		ns
24	Delay, VMSEL to WE	10		ns
25	Delay, E to WE	10		ns
26	Read MPUDATA High Z setup time	0		ns
27	Read MPUDATA High Z hold time	10		ns
28	Read VIDEODATA High Z to OE setup time	10		ns
29	Read VIDEODATA High Z hold time	10		ns
30	Delay, VMSEL to OE	10		ns
31	Delay, E to OE	10		ns

Notes: Specifications for 25°C ambient temperature and statistical wire lengths for 6000 gates.

Table D.1
PADIM Timing

Register Description:

R/W	RS2	RS1	RS0	Name
0	0	0	0	Video Control Register 1
0	0	0	1	Video Control Register 2
0	0	1	0	Base Address Register Low
0	0	1	1	Base Address Register Mid
0	1	0	0	Base Address Register High
0	1	0	1	Count Register Low
0	1	1	0	Count Register High
0	1	1	1	Peripheral Delay Register
1	0	0	0	Status Register
1	0	0	1	Undefined
1	0	1	0	Undefined
1	0	1	1	Undefined
1	1	0	0	Undefined
1	1	0	1	Undefined
1	1	1	0	Undefined
1	1	1	1	Undefined

Table D.2
Internal Registers

Video Control Register 1 (VCR1, Write only):

Bits	Description
0-4	Select 32K deo Memory Page
5	Reset
6	Clear Counter
7	Start

Table D.3
Video Control Register 1

Video Control Register 2 (VCR2, Write Only):

Bits	Description
7	Start Mask

Table D.4
Video Control Register 2

Base Address Register (BAR):

This write only register must be programmed with the start address for the memory transfer. This register is 20 bits. A write to the MSB of the BAR (BARH) latches the Base Address Register.

Count Register (CR):

The count register is a write only register. This register contains the number of bytes to be transferred during a memory transfer. This register is 16 bits wide and a write to the MSB (CRH) latches the Count Register.

Programmable Delay Register (PDR):

The programmable delay register specifies (in the number of video clock cycles) the delay between A/D enable and memory enable. This is a write only register.

Status Register (SR, Read only):

Bits	Description
7	IRQ Pending
0	Memory Transfer In Progress (MTIP)

Table D.5
Status Register

Pin Description:

POWER (V_{DD} and V_{SS}):

These pins supply power to the device.

MICROPROCESSOR ADDRESS BUS (MPUADDR0 - MPUADDR14):

These input pins are connected to the microprocessor address bus.

MICROPROCESSOR DATA BUS (MPUDATA0 - MPUDATA7):

The eight bidirectional pins are connected to the microprocessor data bus. These provide communication between the device and the microprocessor.

VIDEO ADDRESS BUS (VIDEOADDR0 - VIDEOADDR19):

These twenty tri-state address pins supply the memory address information for the video memory system. When the

device is idle, these pins are placed in the high impedance state.

VIDEO DATA BUS (VIDEODATA0 - VIDEODATA7):

The eight signal pins are used to interface the microprocessor bus to the video memory bus. These signal pins are bidirectional.

RESET:

The reset line is active low. A low level on this input causes the device to enter a reset state and clear all internal registers.

Video Memory Select (VMSEL):

A low level input on this pin causes the peripheral to interpret the microprocessor address as a video memory address. These address lines, along with the page register address lines combine to make a video memory address.

Chip Select (CS0 - CS2):

A low level on all three select lines enables the peripheral.

Register Select (RS0 - RS2):

The register select along with the R/W control line combine to generate internal enable signals for the internal

registers.

ASYNCSTART:

This signal input causes the device to start a memory transfer based on the status of the internal start mask. This signal is internally synchronized to VCLK.

ASYNCSTOP:

This input stops a memory transfer. This signal is internally synchronized to VCLK.

Valid Memory Address (VMA):

A low level on this output signal indicates that the address appearing on the video address lines is valid.

Write Enable (WE):

This active low output indicates a write cycle.

Output Enable (OE):

This output pin indicates a read cycle. This signal is active low.

Interrupt Request (IRQ):

This is an open drain output. This pin goes low to indicate a memory transfer has been completed.

CLK:

This input is the base peripheral clock input. This must be four times the desired video clock cycle time.

VCLK:

This output is internally derived by dividing the base clock (CLK) by four.

E:

This input is connected to the microprocessor enable clock signal.

Analog to Digital Converter Enable (ADCEN):

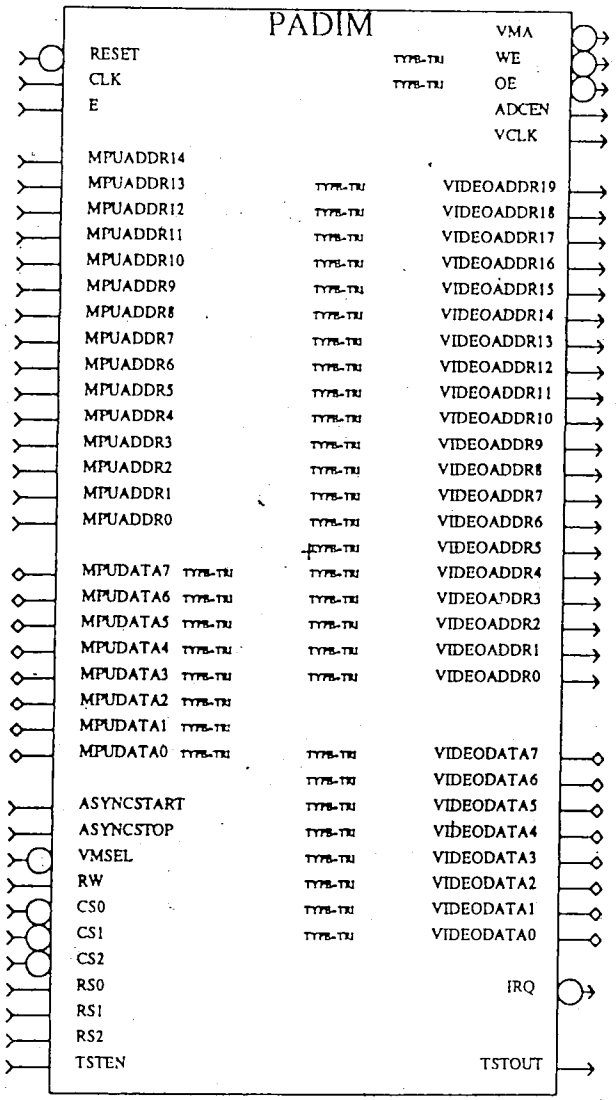
Active high, this signal enables and disables the A/D converter during a memory transfer operation.

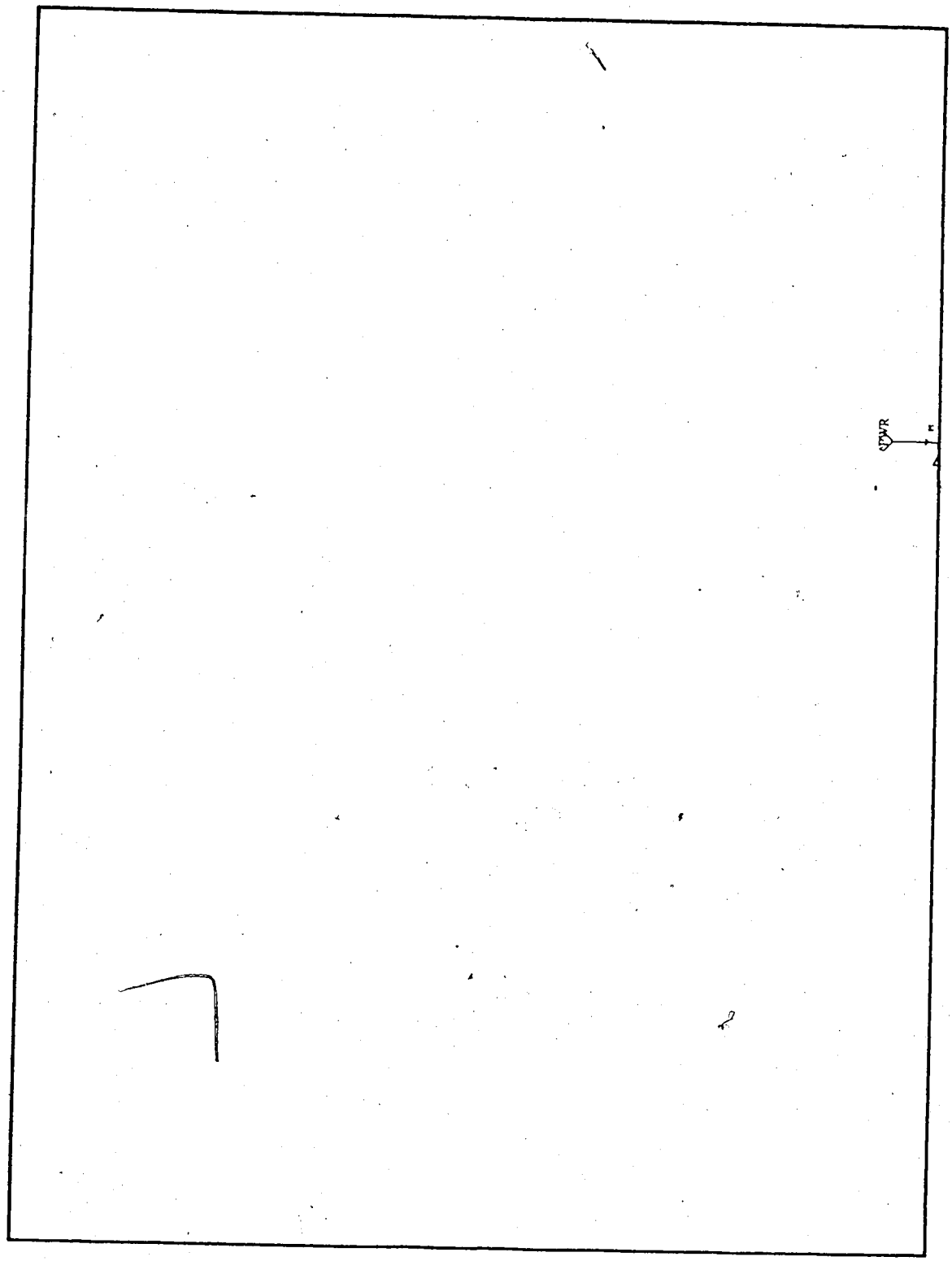
Test Enable (TESTEN):

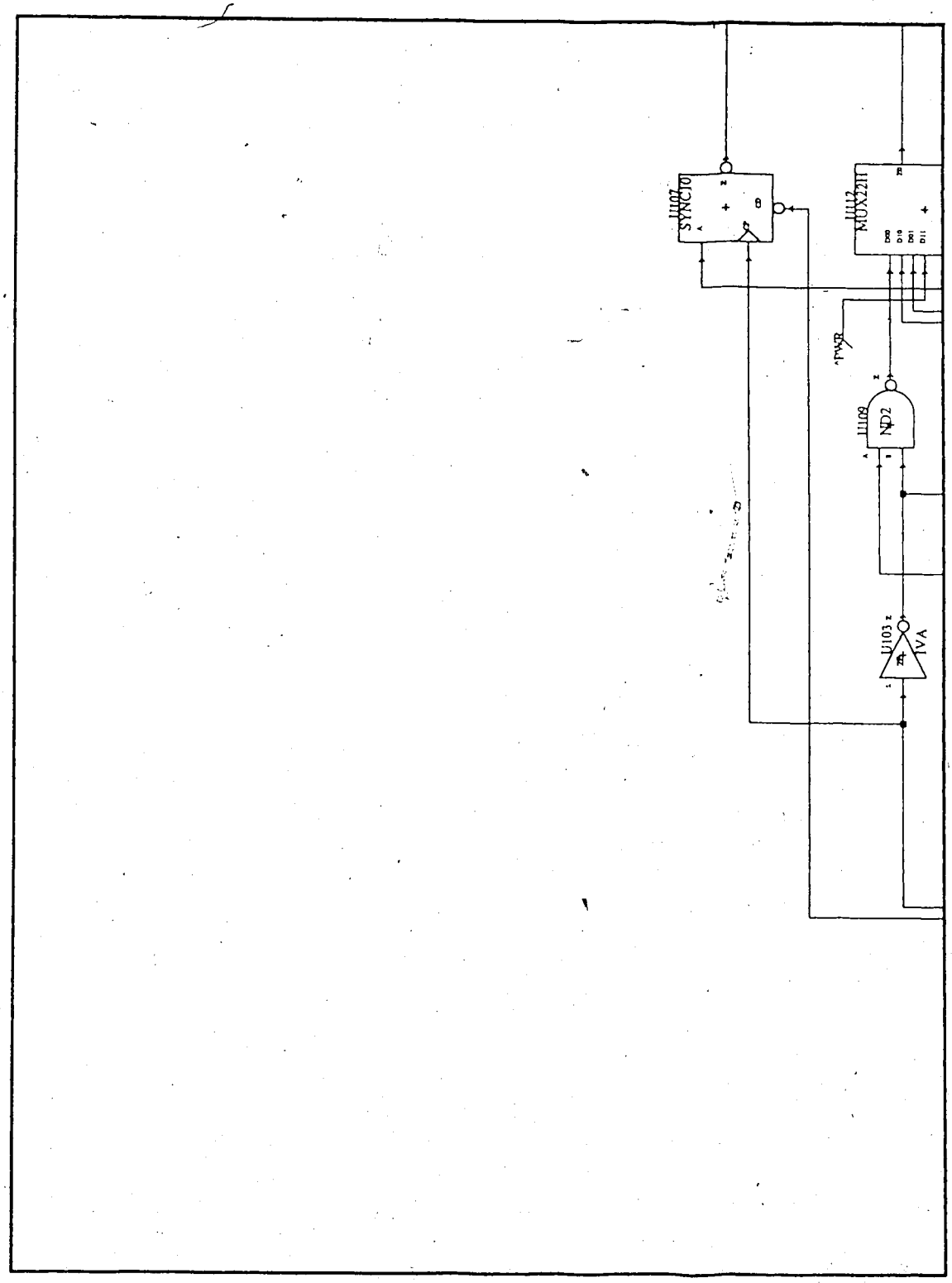
This input must be tied low. It is used during post-fabrication testing and serves no other purpose.

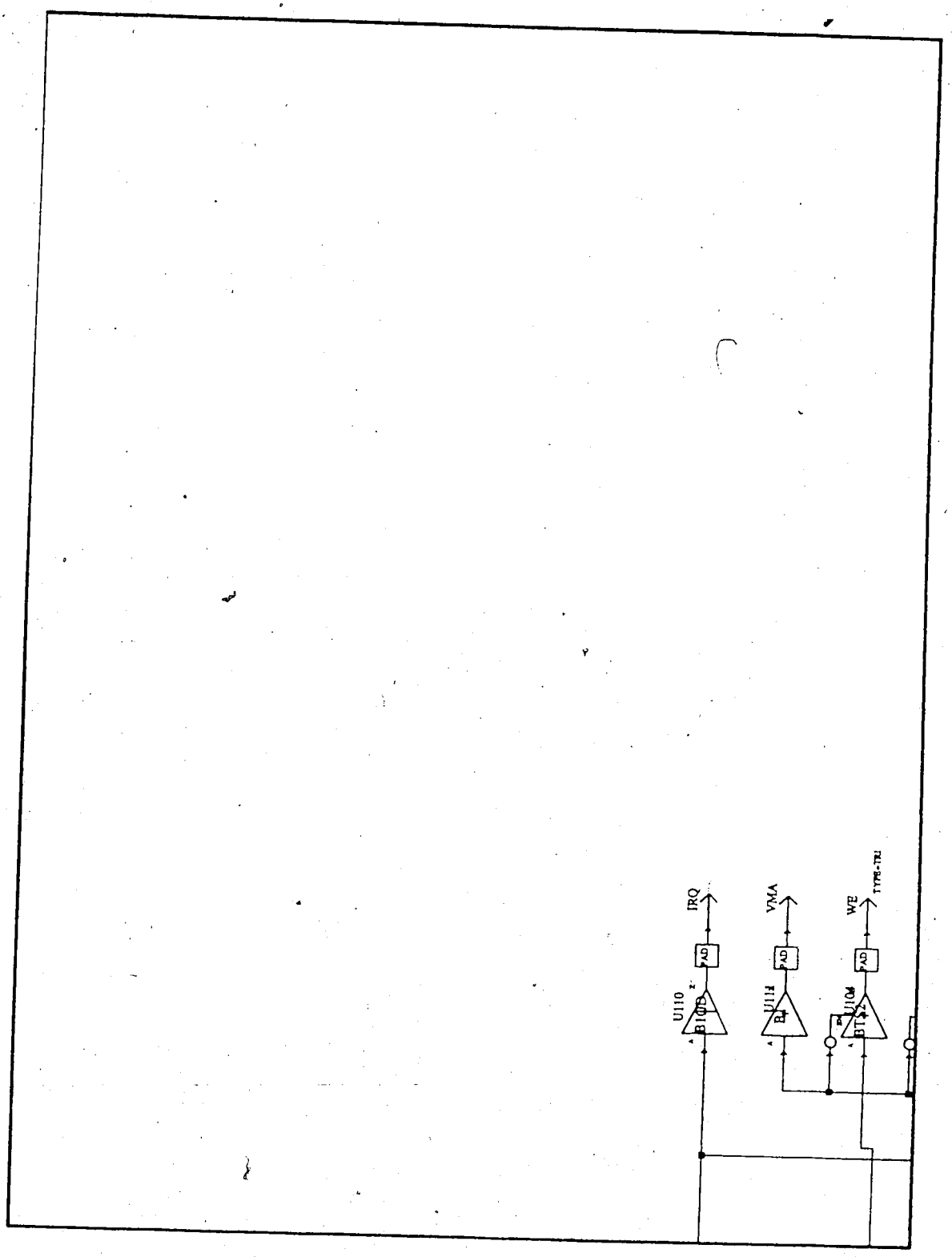
Appendix E

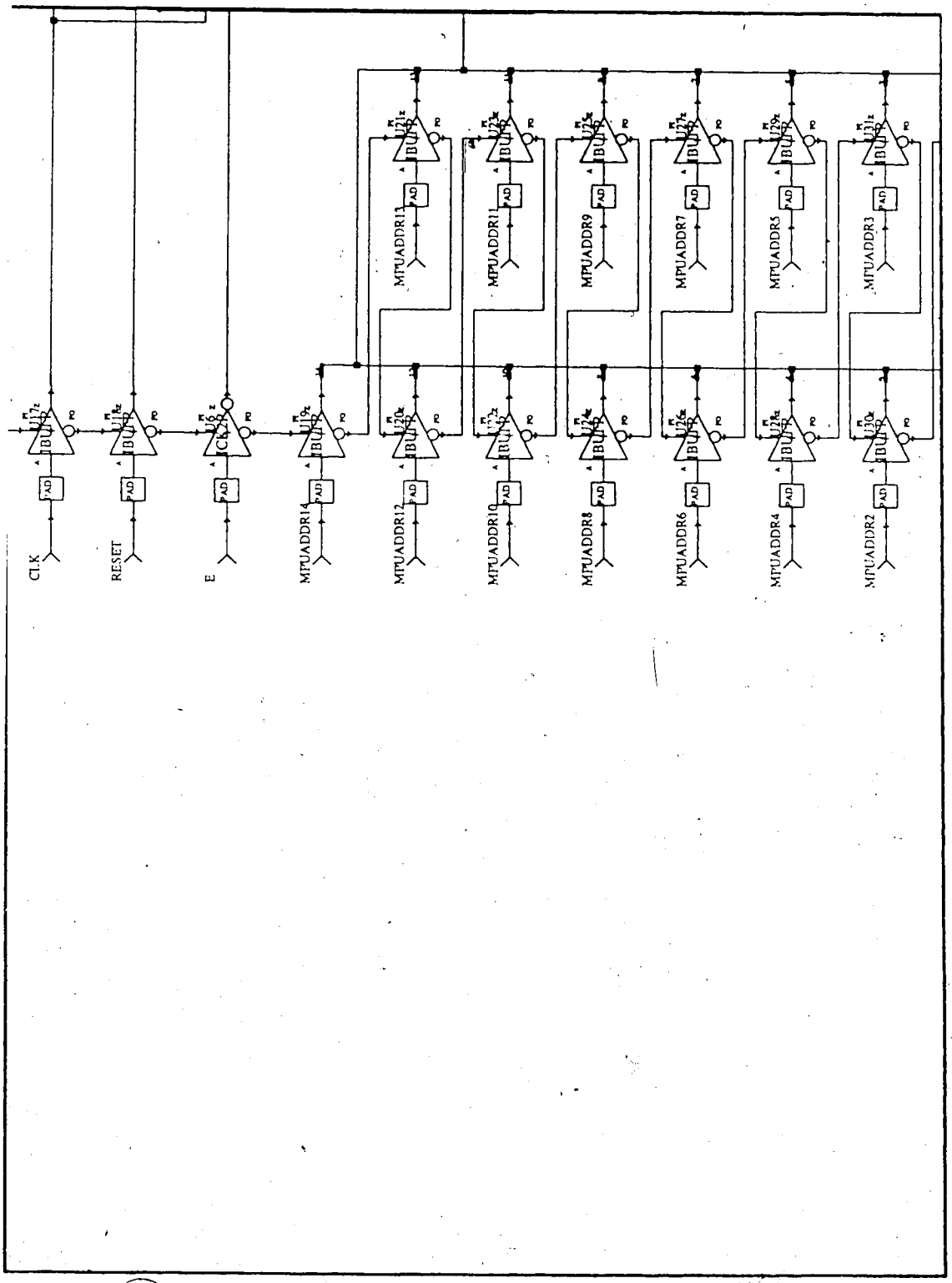
Gate Array Schematic Diagrams

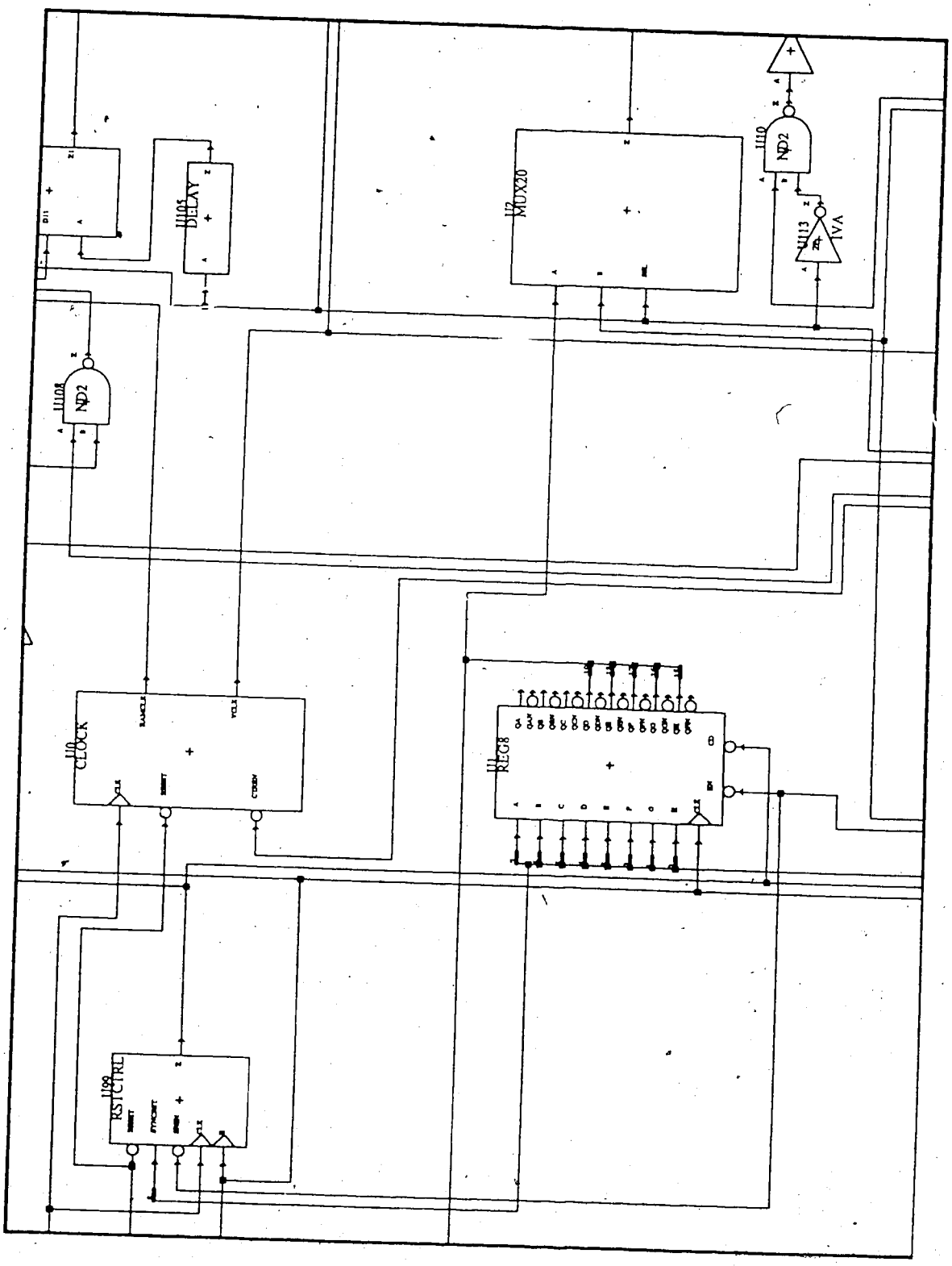

PADIM
RESET
CLK
E
MPUADDR14
MPUADDR13
MPUADDR12
MPUADDR11
MPUADDR10
MPUADDR9
MPUADDR8
MPUADDR7
MPUADDR6
MPUADDR5
MPUADDR4
MPUADDR3
MPUADDR2
MPUADDR1
MPUADDR0
MPUDATA7
MPUDATA6
MPUDATA5
MPUDATA4
MPUDATA3
MPUDATA2
MPUDATA1
MPUDATA0
ASYNCSTART
ASYNCSTOP
VMSEL
RW
CS0
CS1
CS2
RS0
RS1
RS2
TSTEN
VMA
WE
OE
ADCEN
VCLK
VIDEOADDR19
VIDEOADDR18
VIDEOADDR17
VIDEOADDR16
VIDEOADDR15
VIDEOADDR14
VIDEOADDR13
VIDEOADDR12
VIDEOADDR11
VIDEOADDR10
VIDEOADDR9
VIDEOADDR8
VIDEOADDR7
VIDEOADDR6
VIDEOADDR5
VIDEOADDR4
VIDEOADDR3
VIDEOADDR2
VIDEOADDR1
VIDEOADDR0
VIDEODATA7
VIDEODATA6
VIDEODATA5
VIDEODATA4
VIDEODATA3
VIDEODATA2
VIDEODATA1
VIDEODATA0
IRQ
TSTOUT

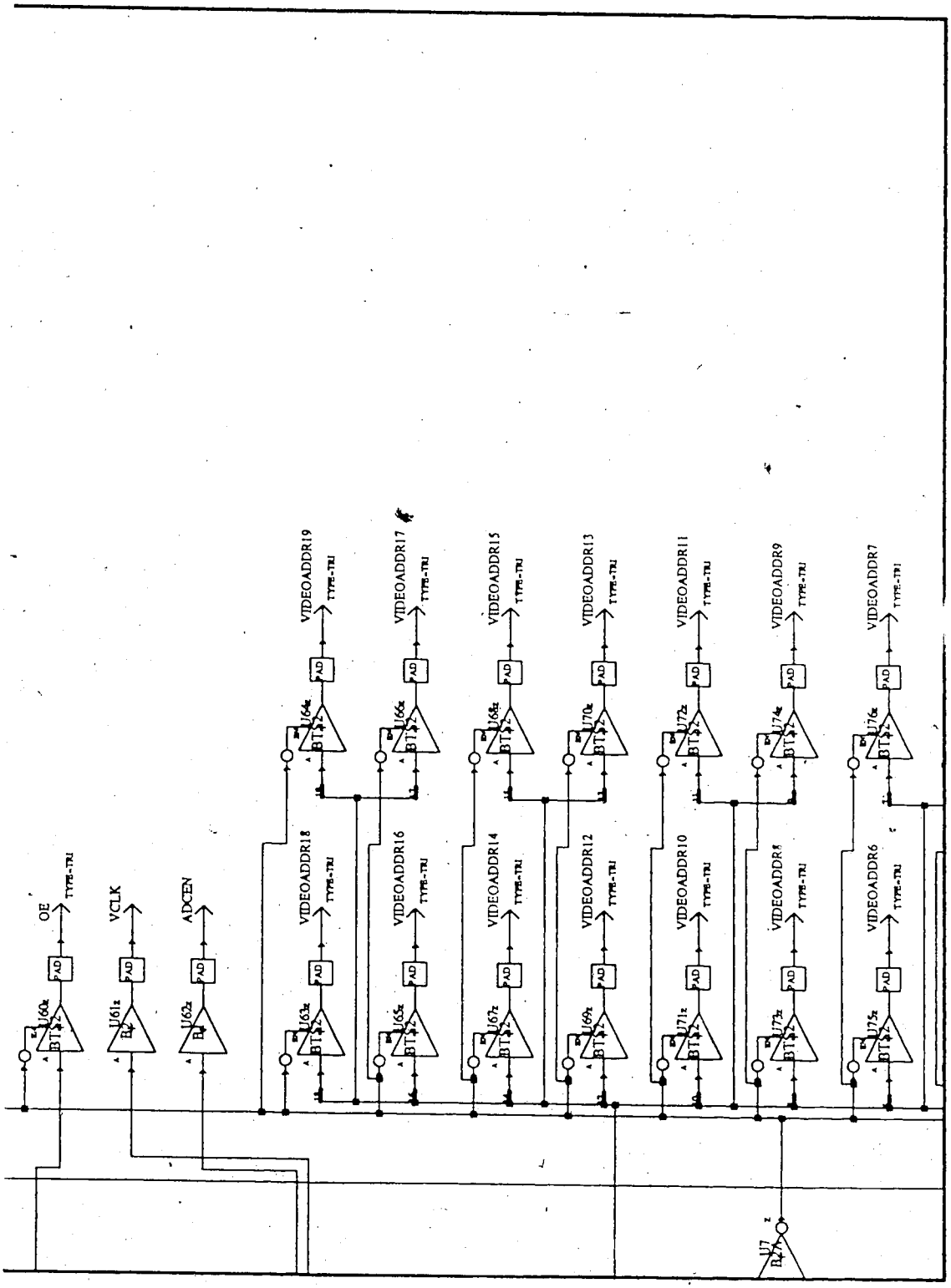


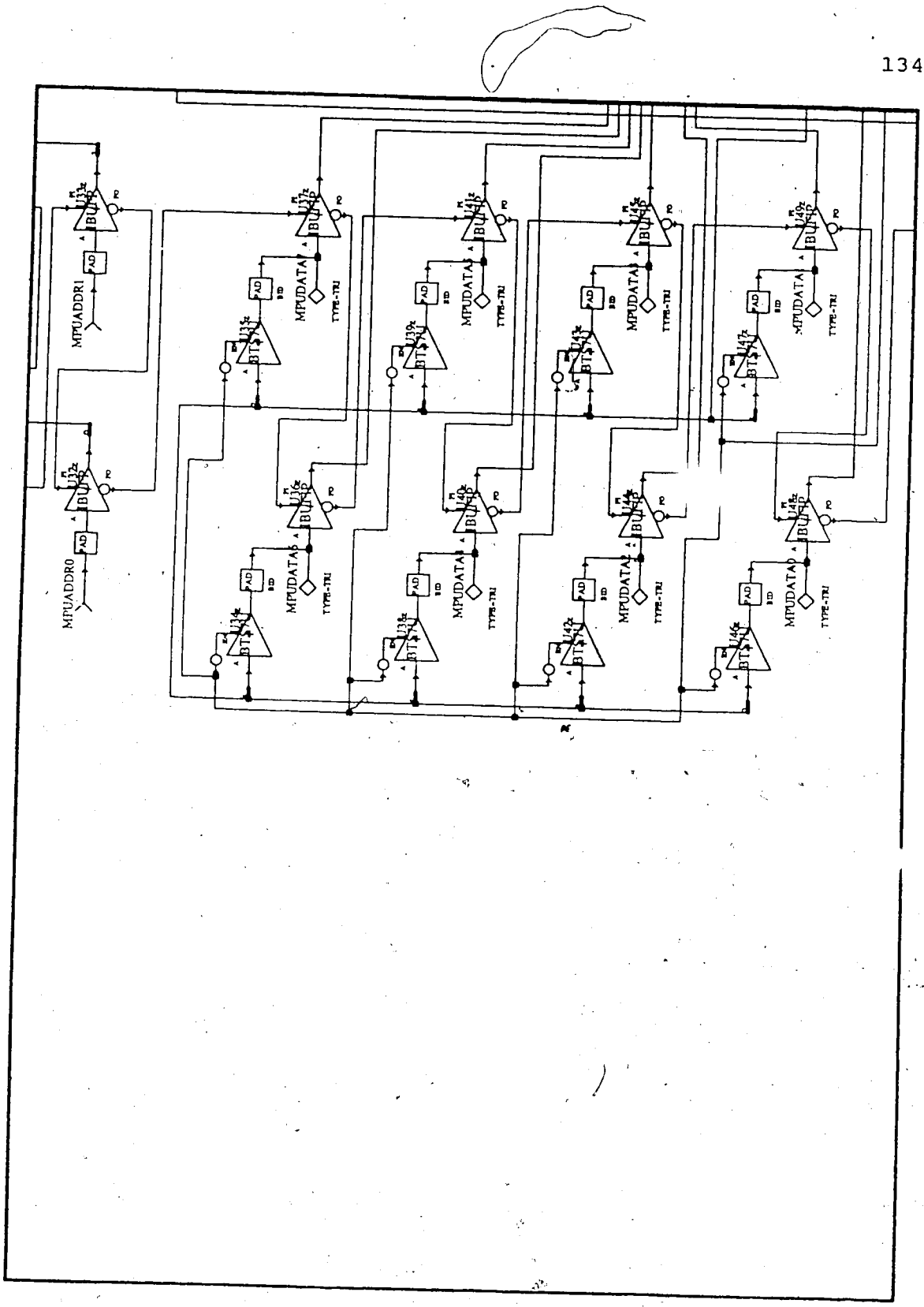

U107
SYNCTO
U112
MUX2111
*PWR
U109
ND2
U103
IVA

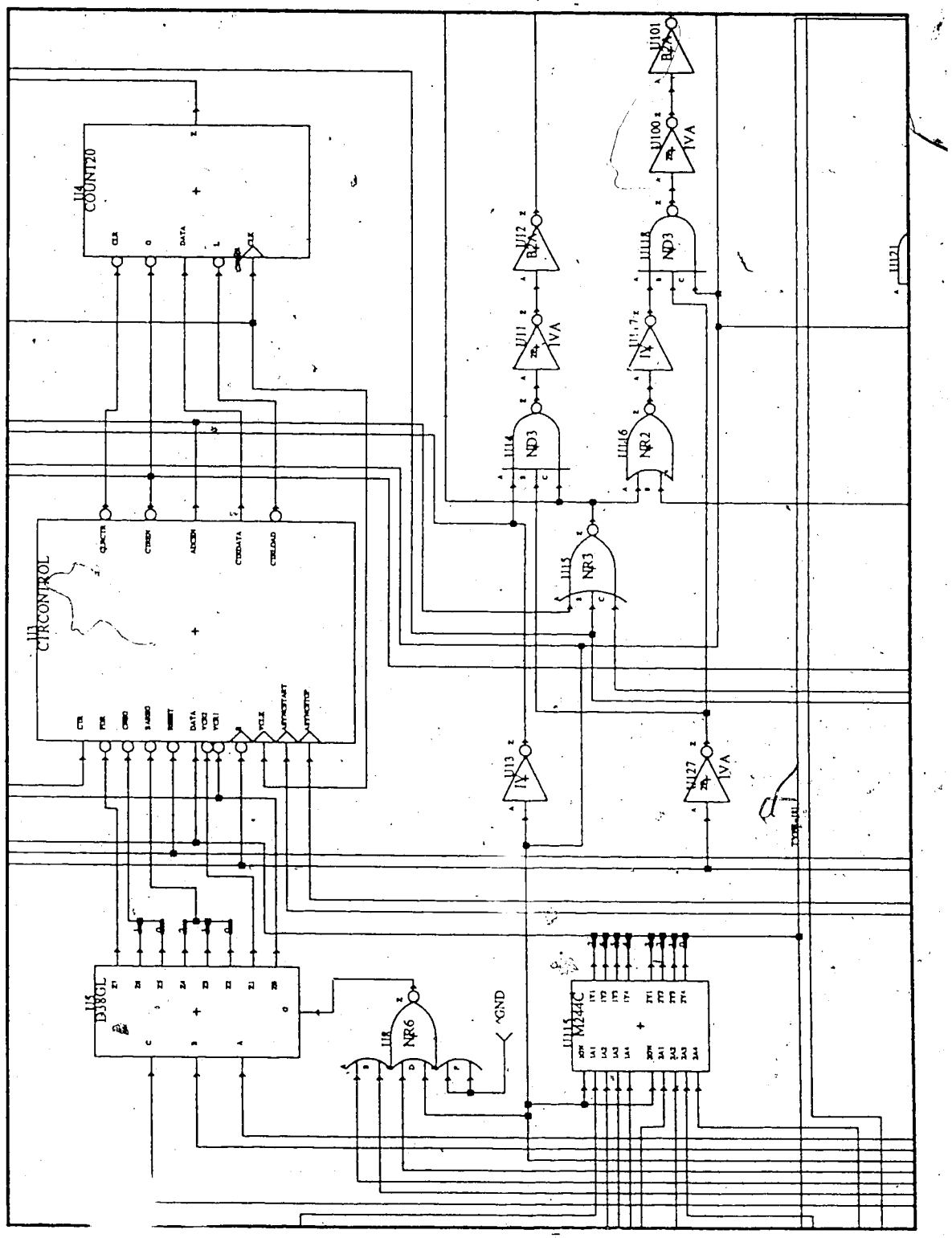

IRQ
VMA
WE
U110
U111
U104

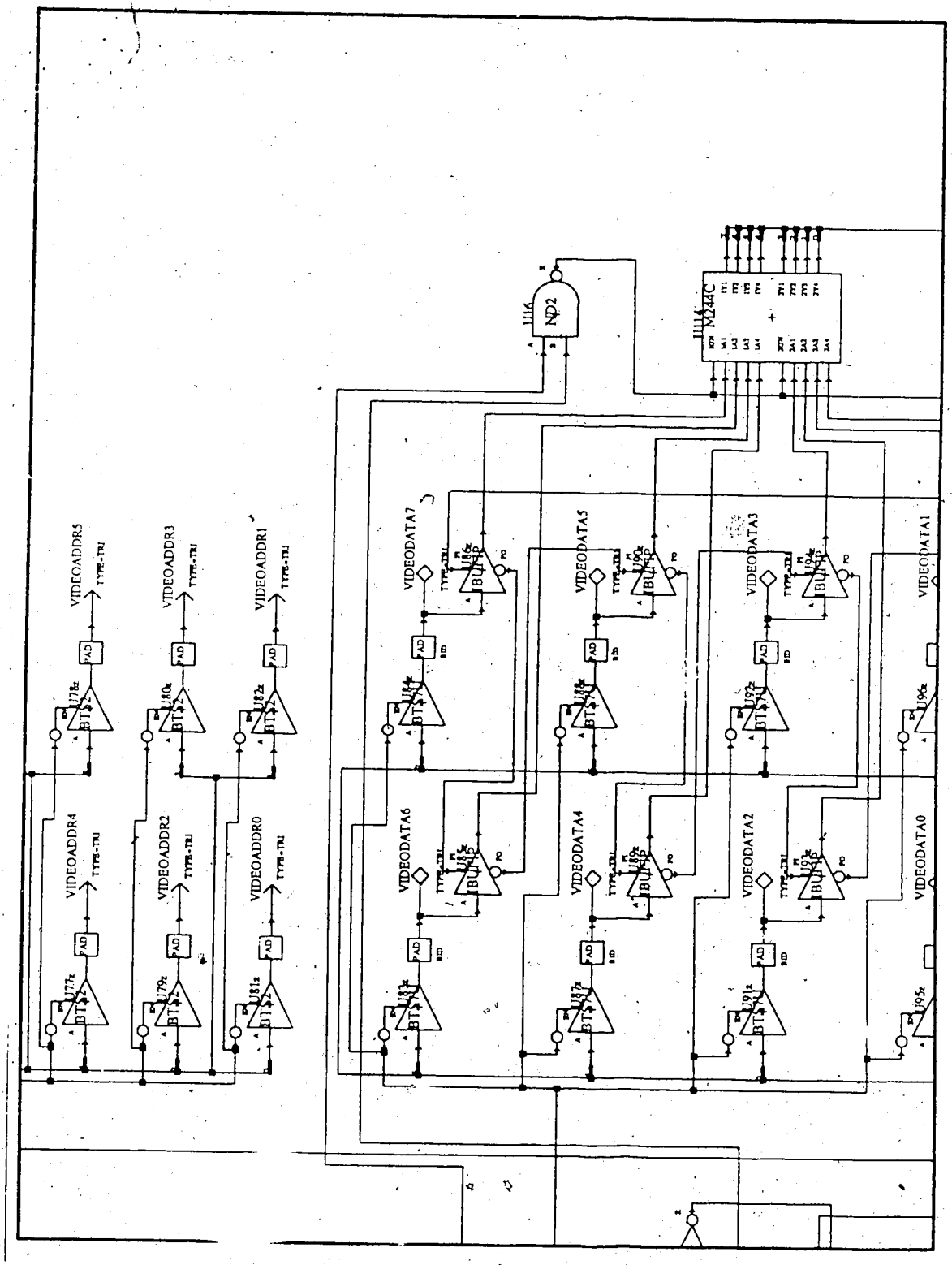

CLK
RESET
E
MPUADDR14
MPUADDR12
MPUADDR10
MPUADDR8
MPUADDR6
MPUADDR4
MPUADDR2
MPUADDR13
MPUADDR11
MPUADDR9
MPUADDR7
MPUADDR5
MPUADDR3
BUF
PAD

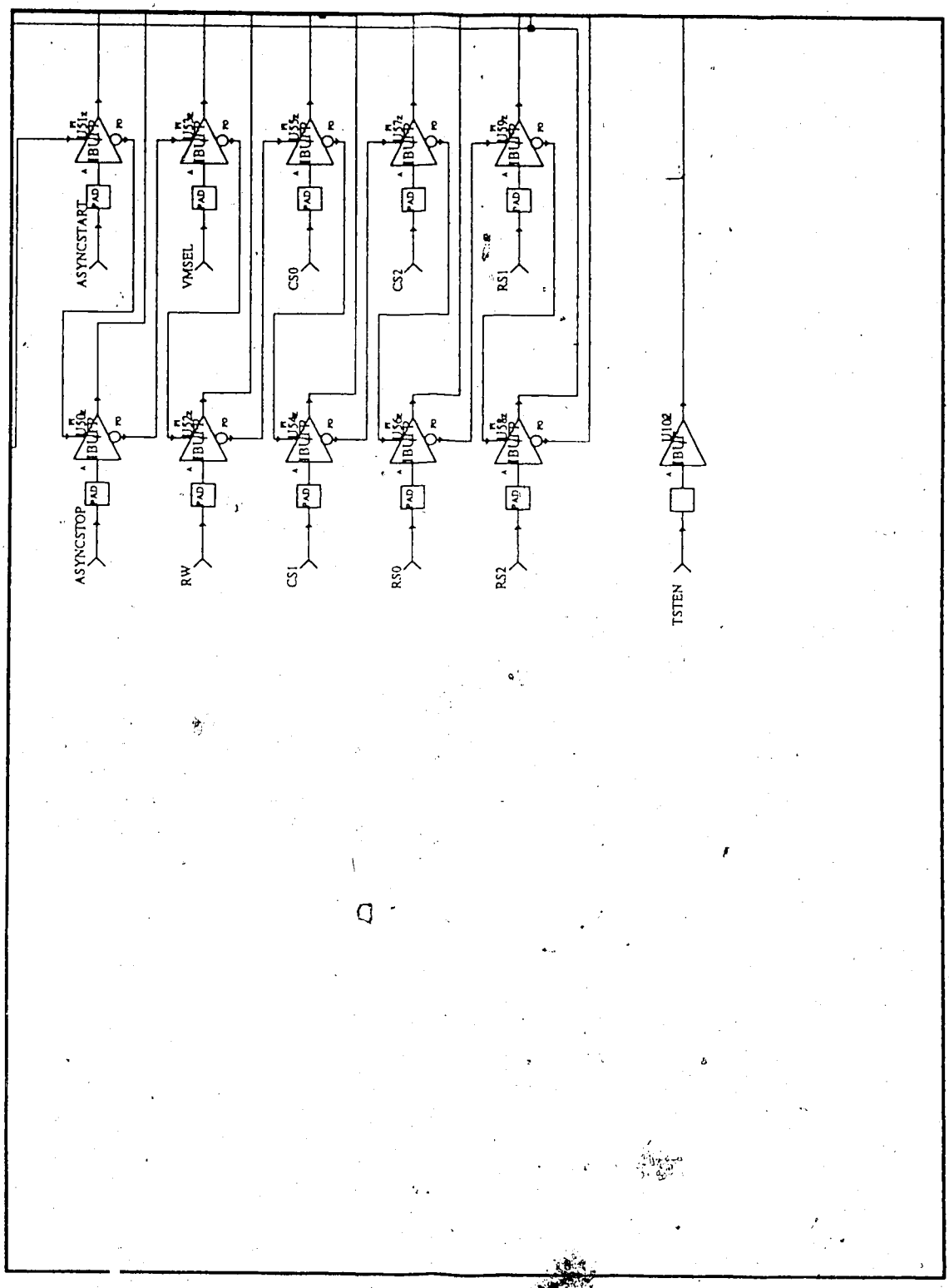

U105
DELAY
U7
MUX20
U110
ND2
U113
IVA
U108
ND2
U0
CLOCK
U1
REG8
U88
RSTCTRL

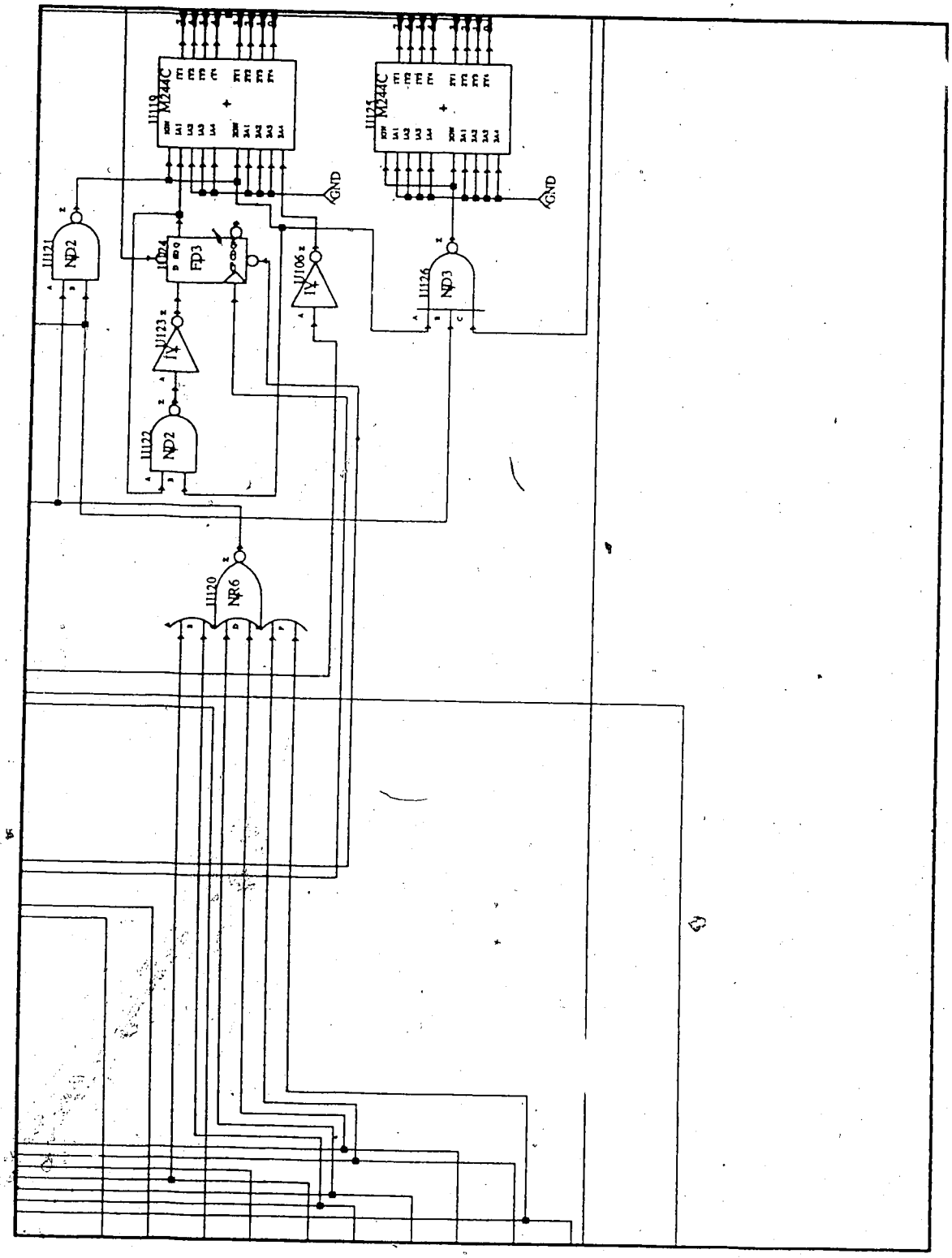

OE
VCLK
ADCEN
VIDEOADDR19
VIDEOADDR18
VIDEOADDR17
VIDEOADDR16
VIDEOADDR15
VIDEOADDR14
VIDEOADDR13
VIDEOADDR12
VIDEOADDR11
VIDEOADDR10
VIDEOADDR9
VIDEOADDR8
VIDEOADDR7
VIDEOADDR6
U60
U61
U62
U63
U64
U65
U66
U67
U68
U69
U70
U71
U72
U73
U74
U75
U76
U7

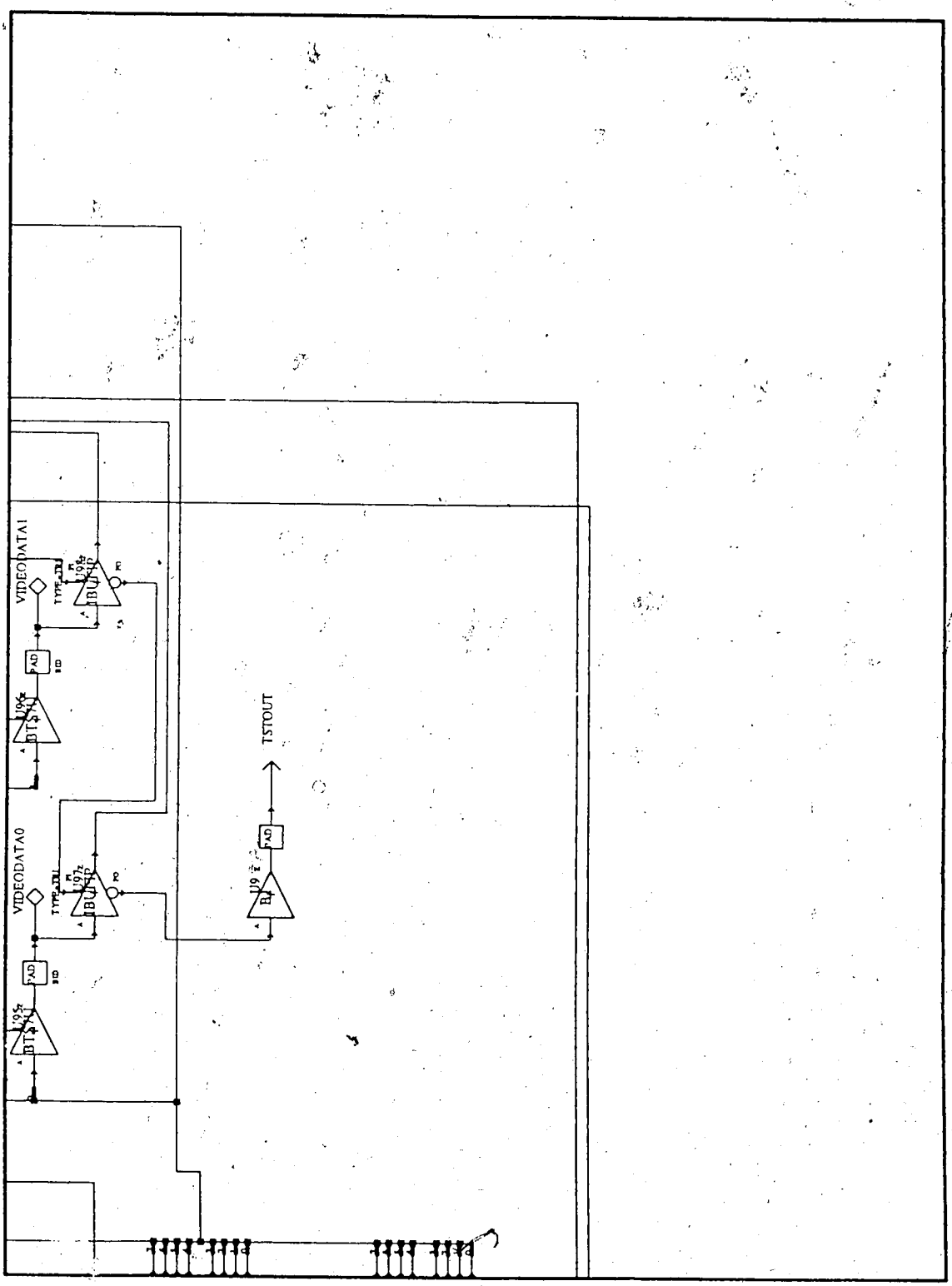

MPUADDR1
MPUADDR0
MPUDATA7
MPUDATA6
MPUDATA5
MPUDATA4
MPUDATA3
MPUDATA2
MPUDATA1
MPUDATA0
BUF8
BTS8
AD
BID
TYPE-TRI

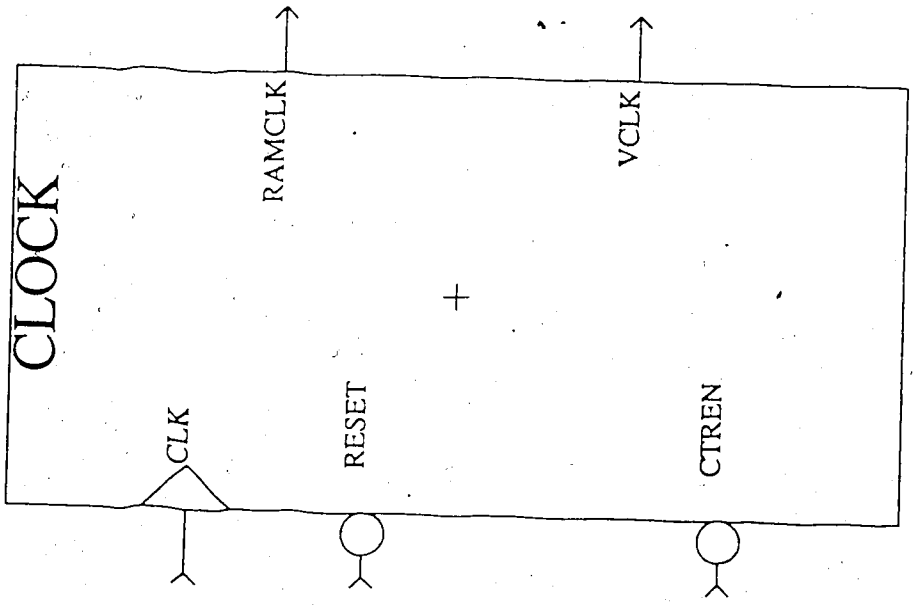

U4
COUNT20
U3
CTRCONTROL
U5
D38CL
U118
NR6
GND
U115
M244C
U101
U100
IVA
U112
U111
ND3
U114
U117
U116
NR2
U115
NR3
U113
U127
U121

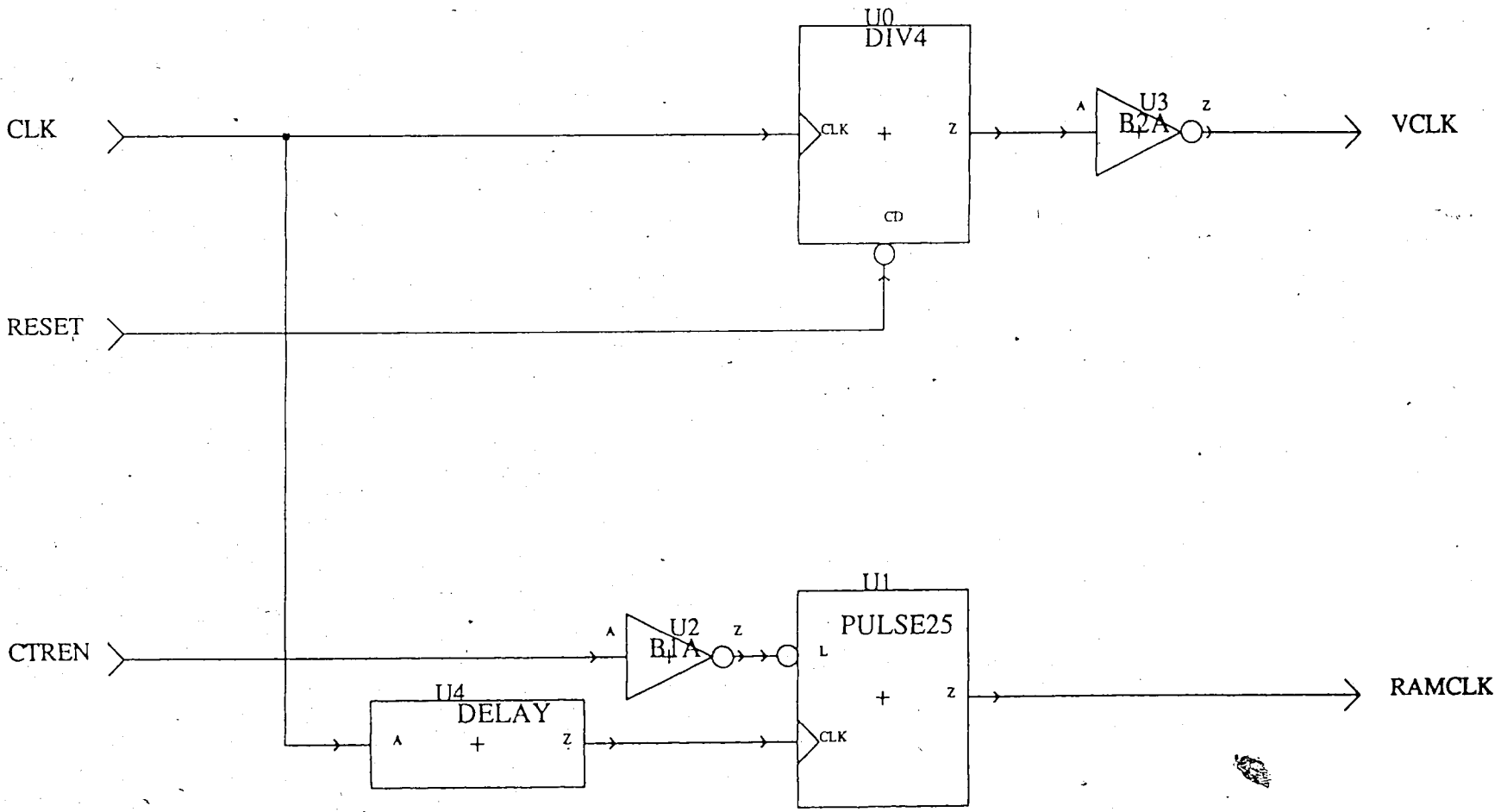

VIDEOADDR5
VIDEOADDR4
VIDEOADDR3
VIDEOADDR2
VIDEOADDR1
VIDEOADDR0
VIDEODATA7
VIDEODATA6
VIDEODATA5
VIDEODATA4
VIDEODATA3
VIDEODATA2
VIDEODATA1
VIDEODATA0
ND2
M244C

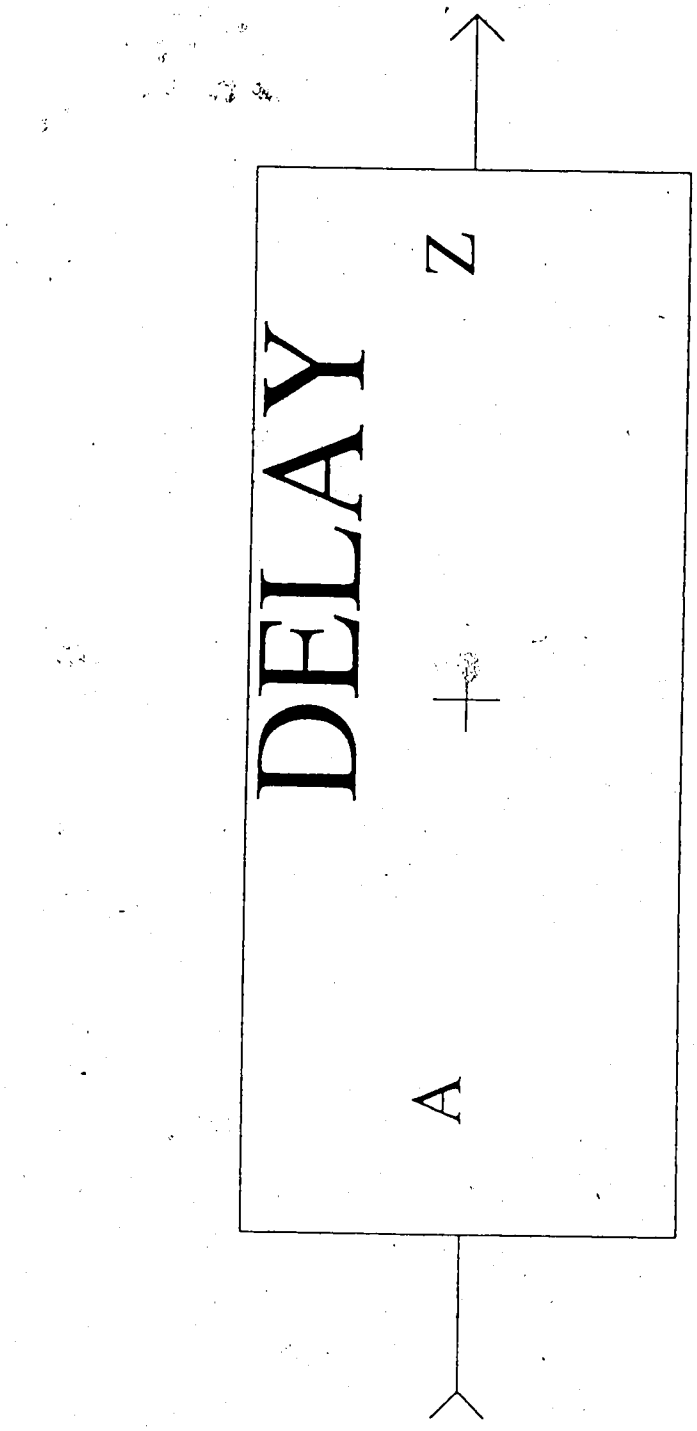

ASYNCSTART
VMSEL
CS0
CS2
RS1
ASYNCSTOP
RW
CS1
RS0
RS2
TSTEN

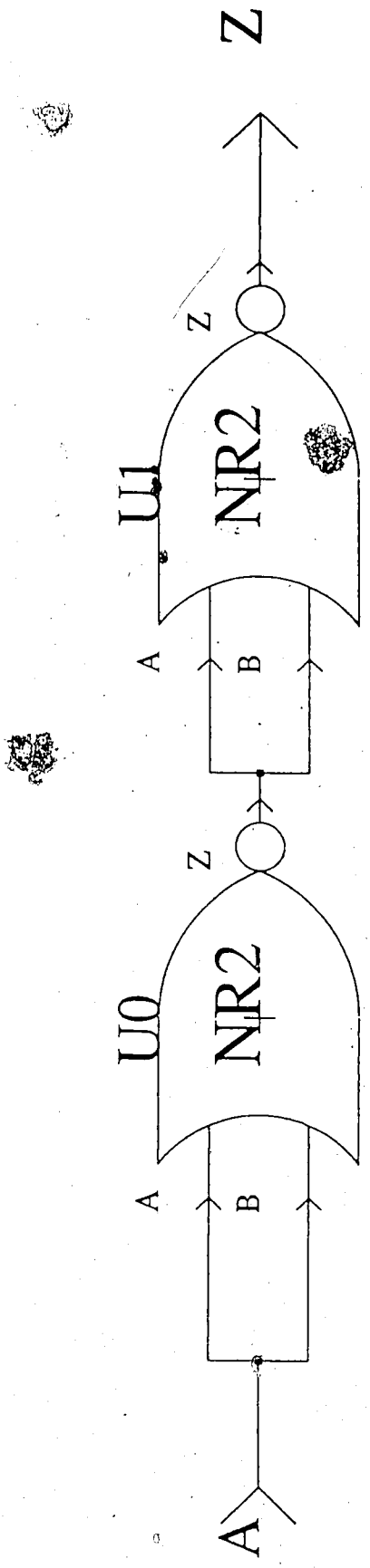

U119
M244C
U125
M244C
GND
GND
U121
ND2
U124
FD3
U106
IV
U126
ND3
U123
IV
U122
ND2
U120
NR6

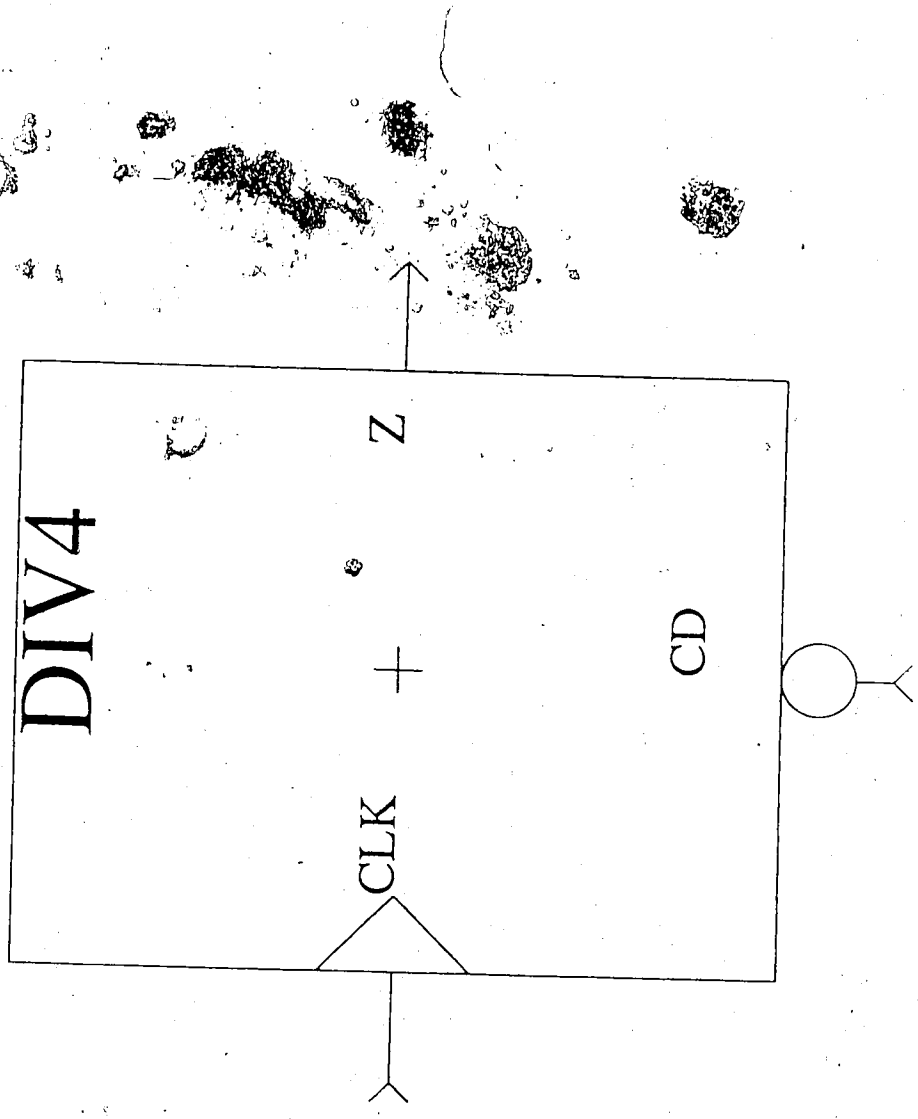

VIDEODATA1
VIDEODATA0
TSTOUT

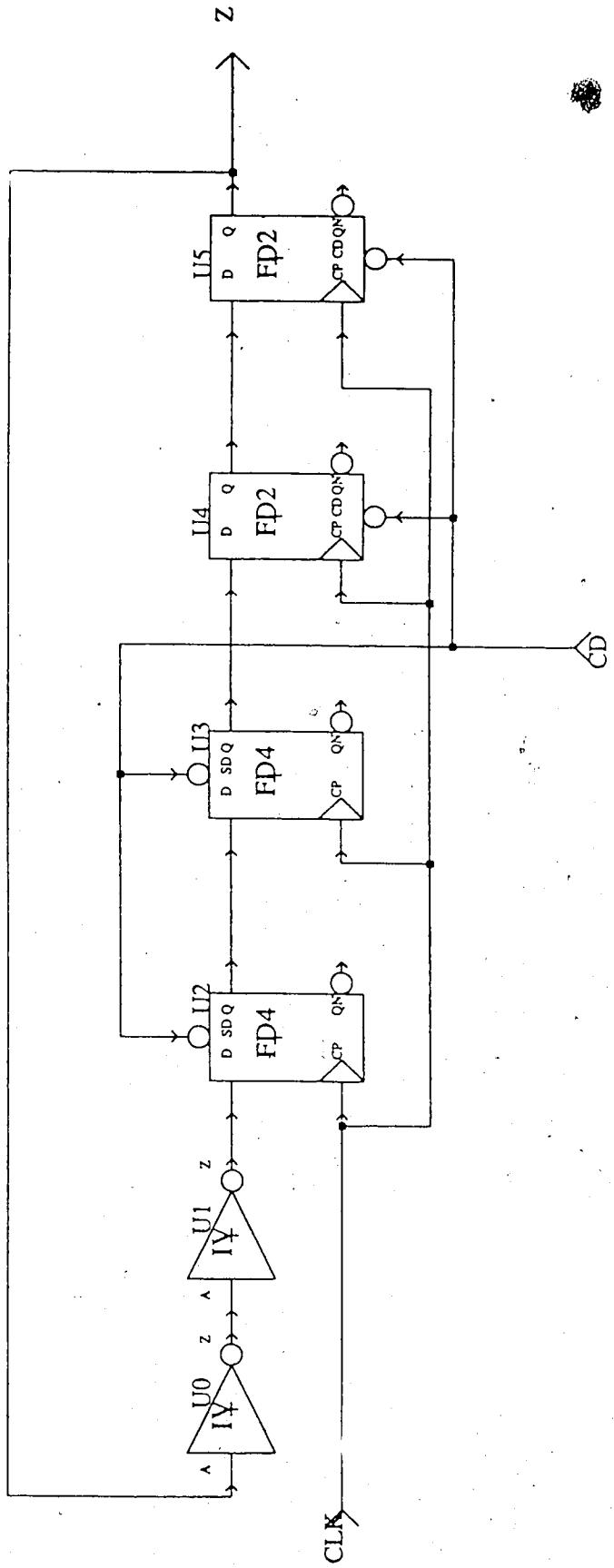

CLOCK
CLK
RESET
RAMCLK
VCLK
CTREN

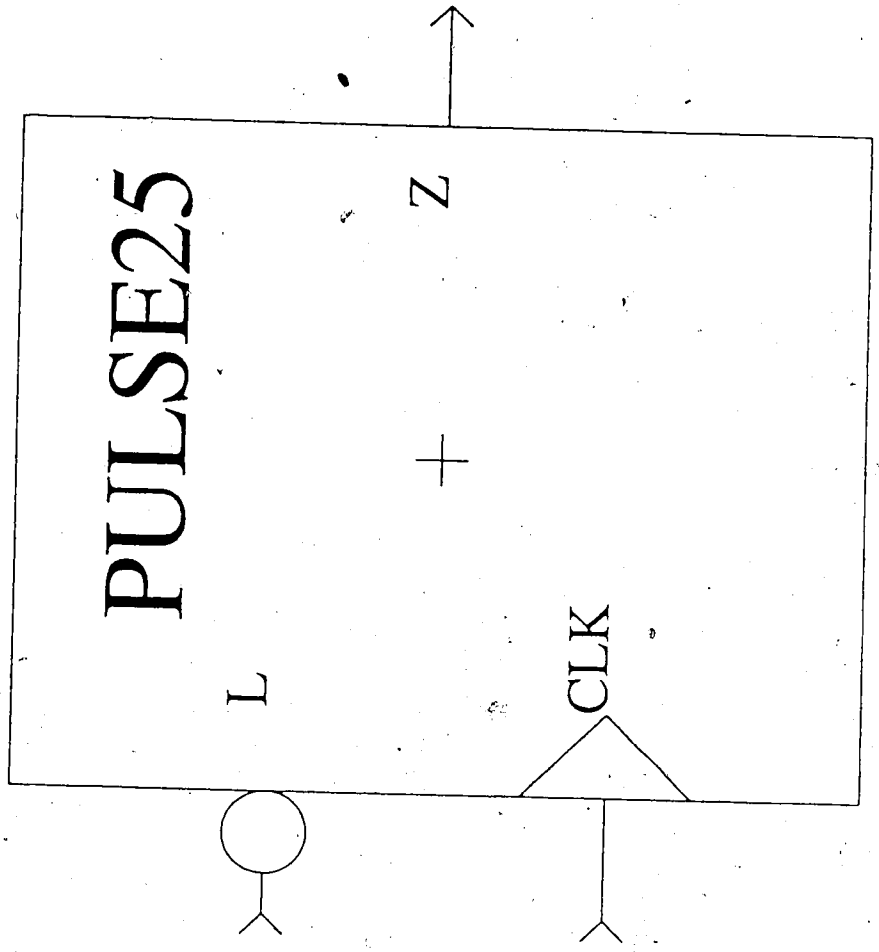

U0
DIV4
CLK
+
Z
CD
U3
B2A
A
Z
CLK
VCLK
RESET
U1
PULSE25
U2
B1A
A
Z
L
CTREN
U4
DELAY
A
+
Z
CLK
RAMCLK

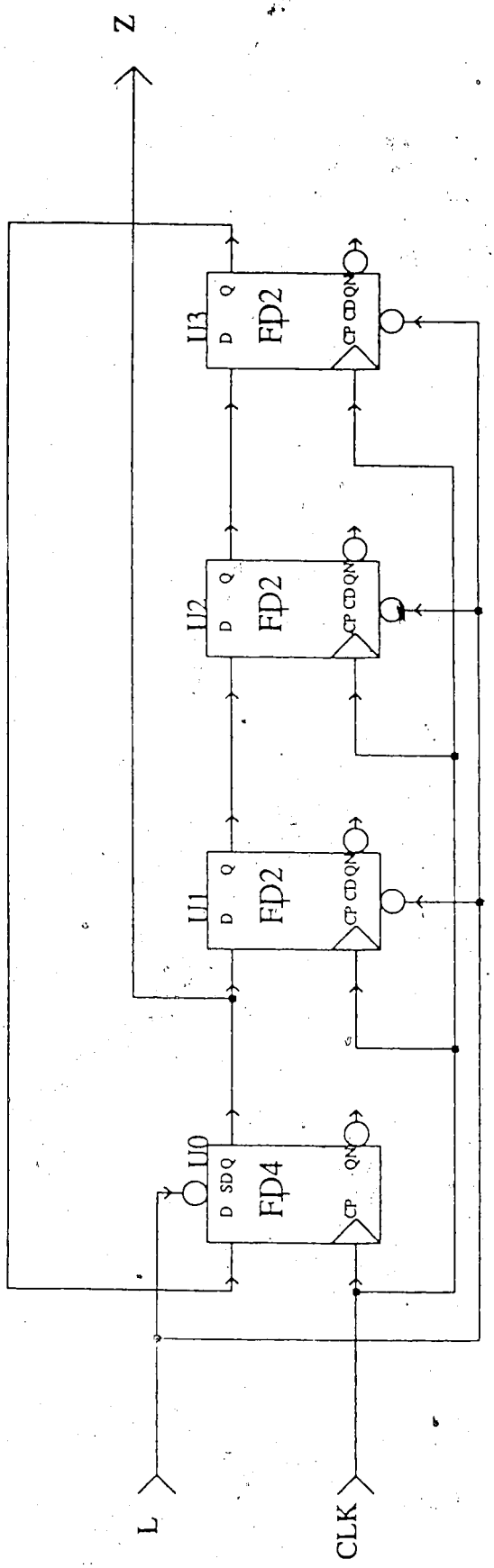

DELAY
Z
A

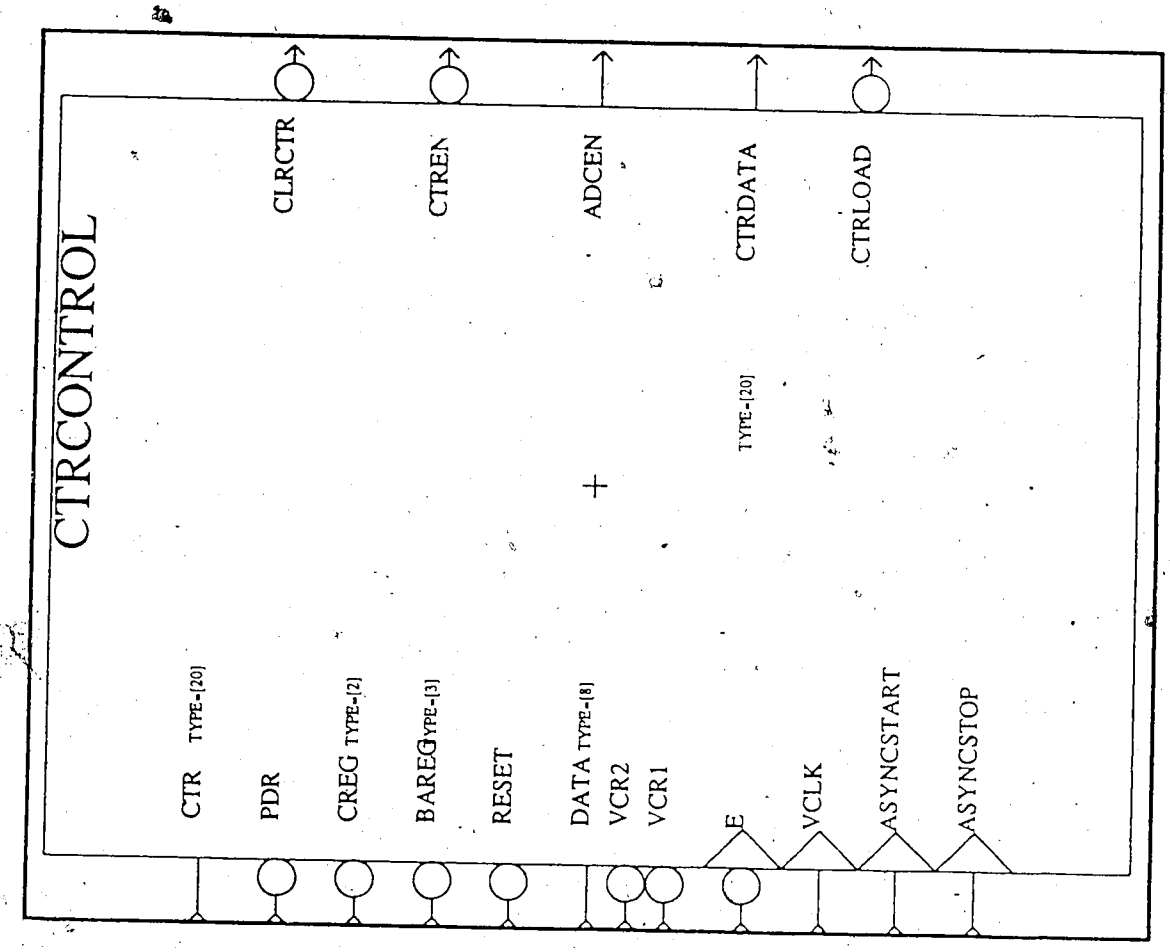

Z
Z
U1
NR2
A
B
Z
U0
NR2
A
B
A

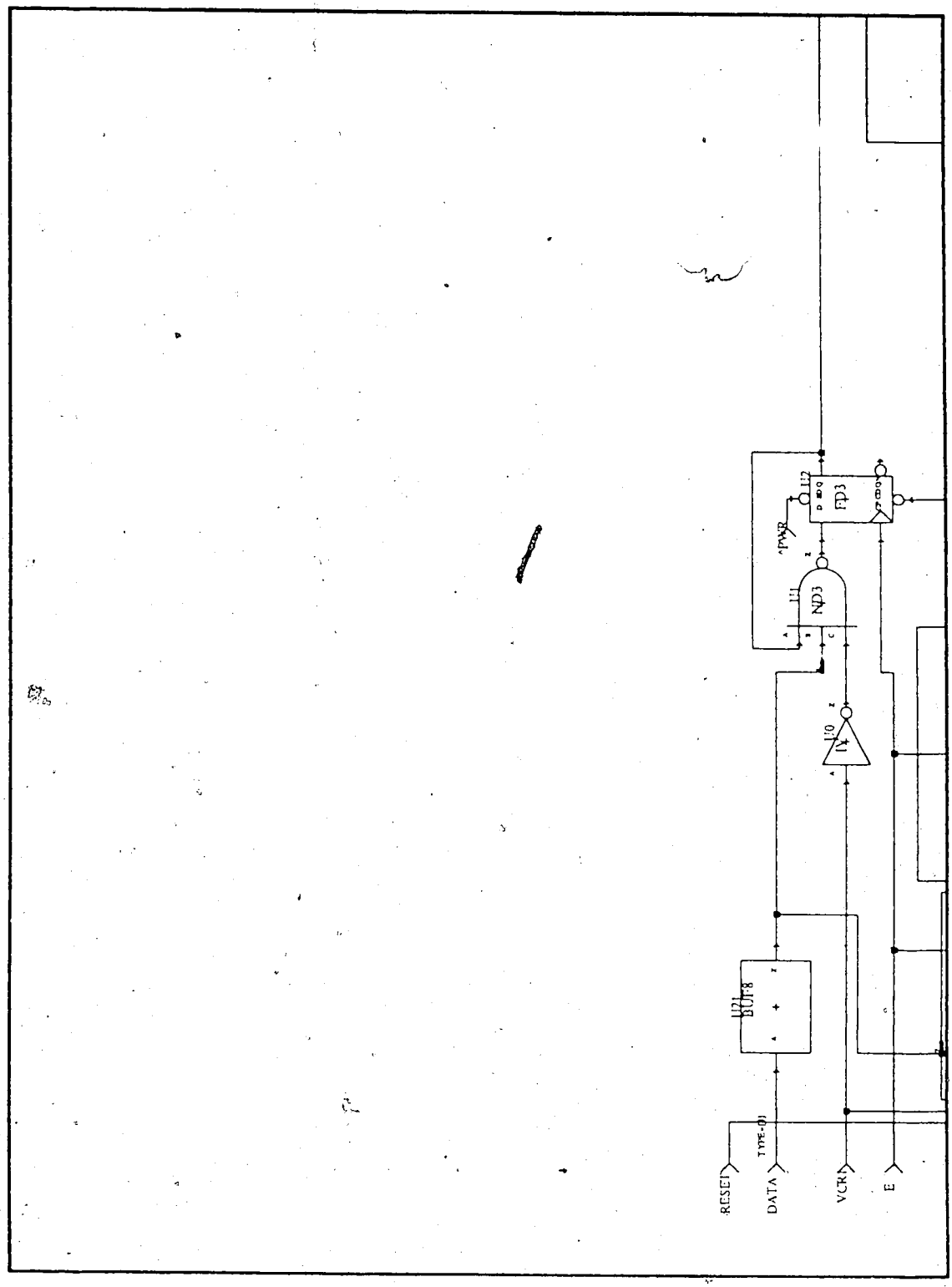

DIV4
Z
CD
CLK

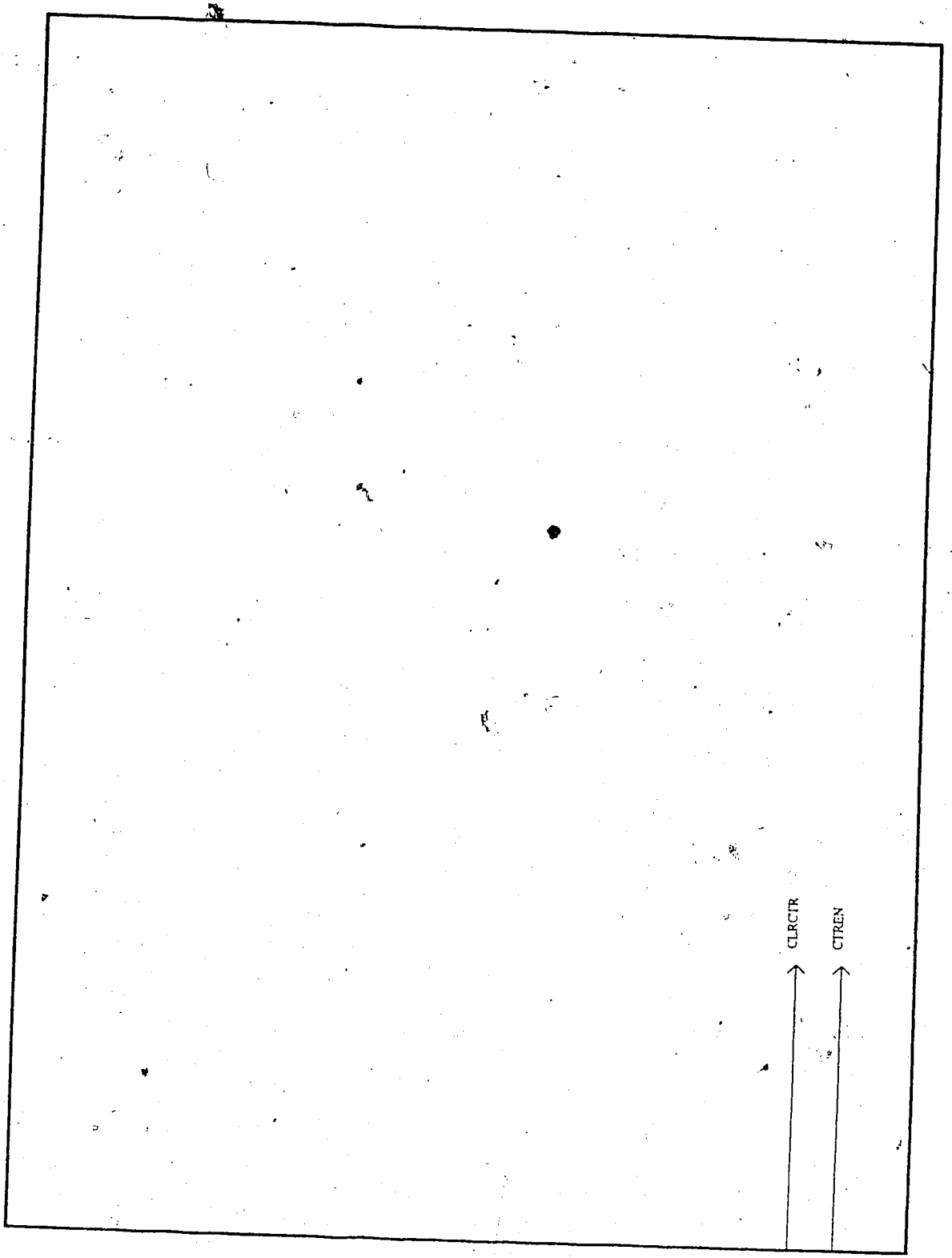

Z
U5
FD2
D Q
CP CD QN
U4
FD2
D Q
CP CD QN
CD
U3
FD4
D SD Q
CP QN
U2
FD4
D SD Q
CP QN
U1
IV
Z
A
U0
IV
Z
A
CLK

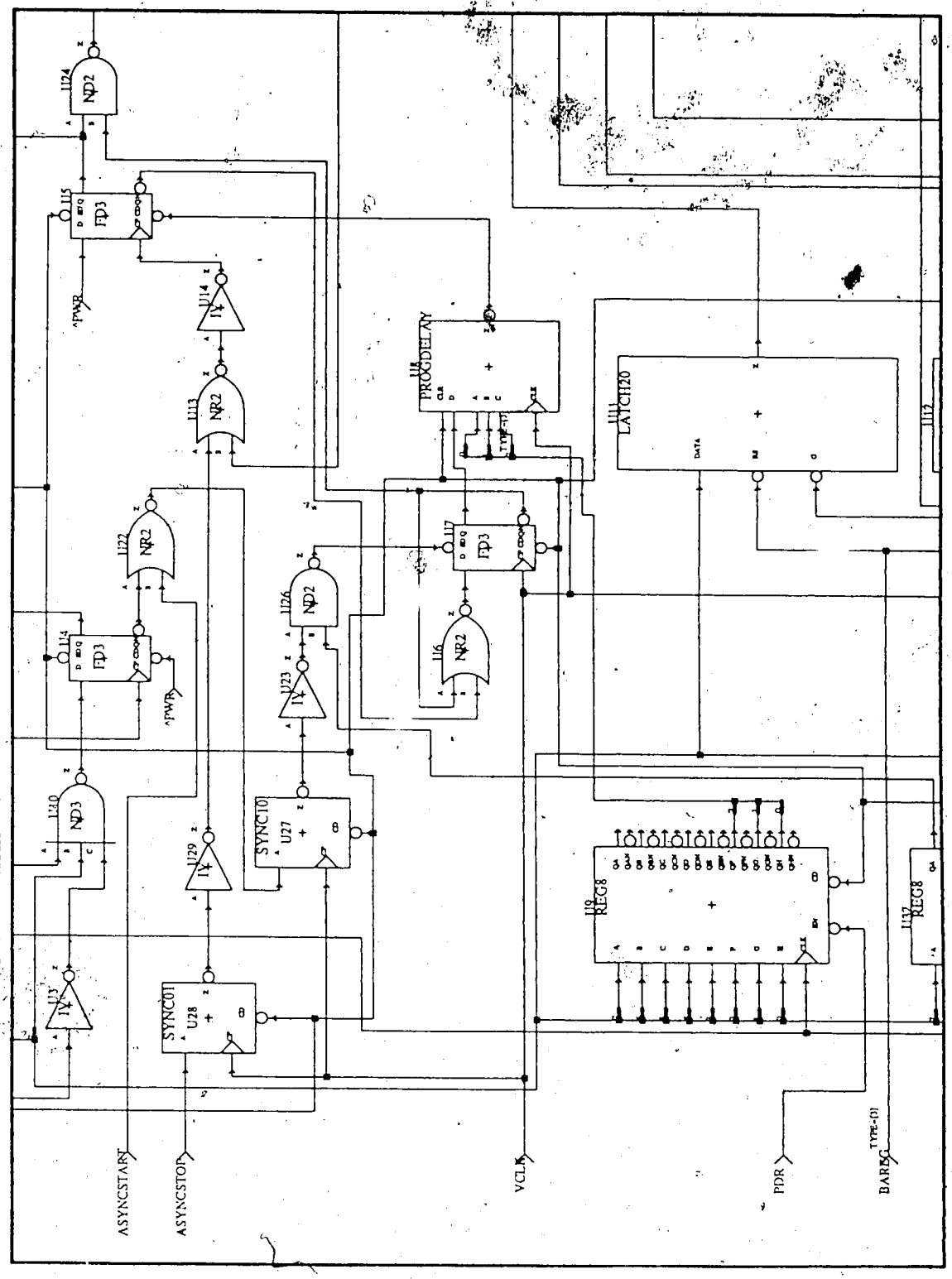

PULSE25
Z
L
CLK

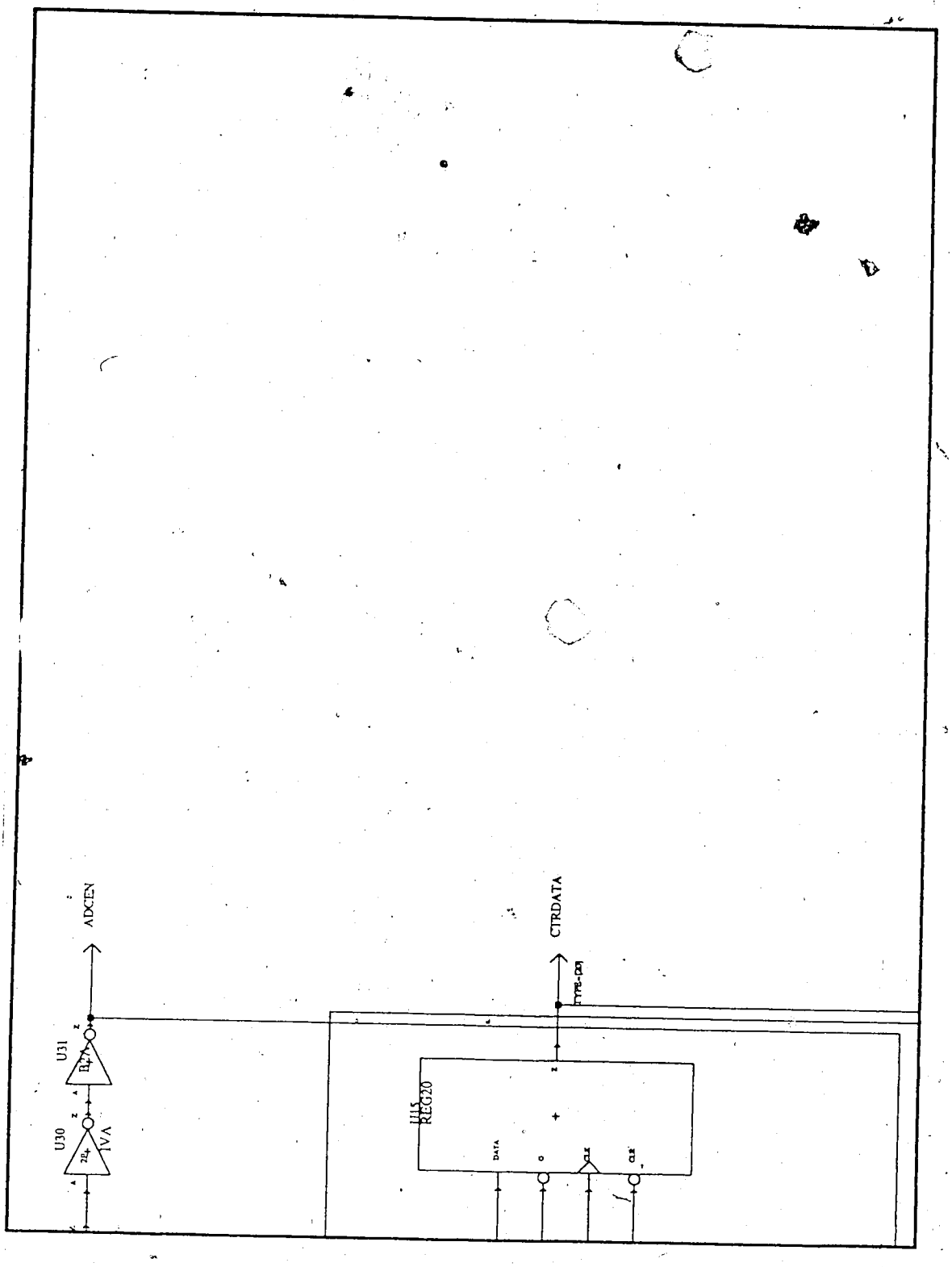

Z
U3
FD2
D Q
CP CDQN
U2
FD2
D Q
CP CDQN
U1
FD2
D Q
CP CDQN
U0
FD4
D SD Q
CP QN
L
CLK

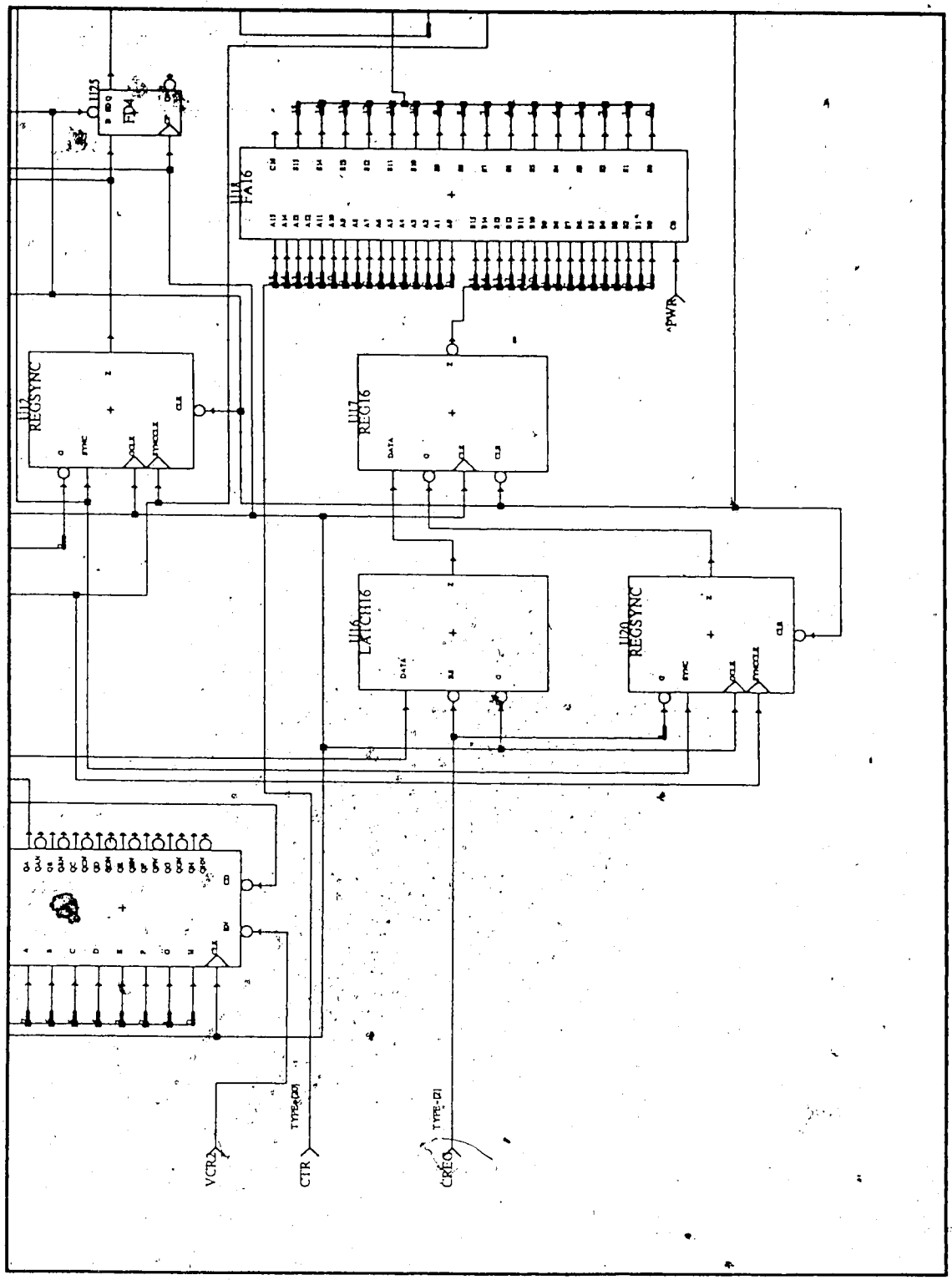

CTRCONTROL
CLRCTR
CTREN
ADCEN
CTRDATA
CTRLOAD
CTR TYPE-[20]
PDR
CREG TYPE-[2]
BAREG TYPE-[3]
RESET
DATA TYPE-[8]
VCR2
VCR1
E
TYPE-[20]
VCLK
ASYNCSTART
ASYNCSTOP

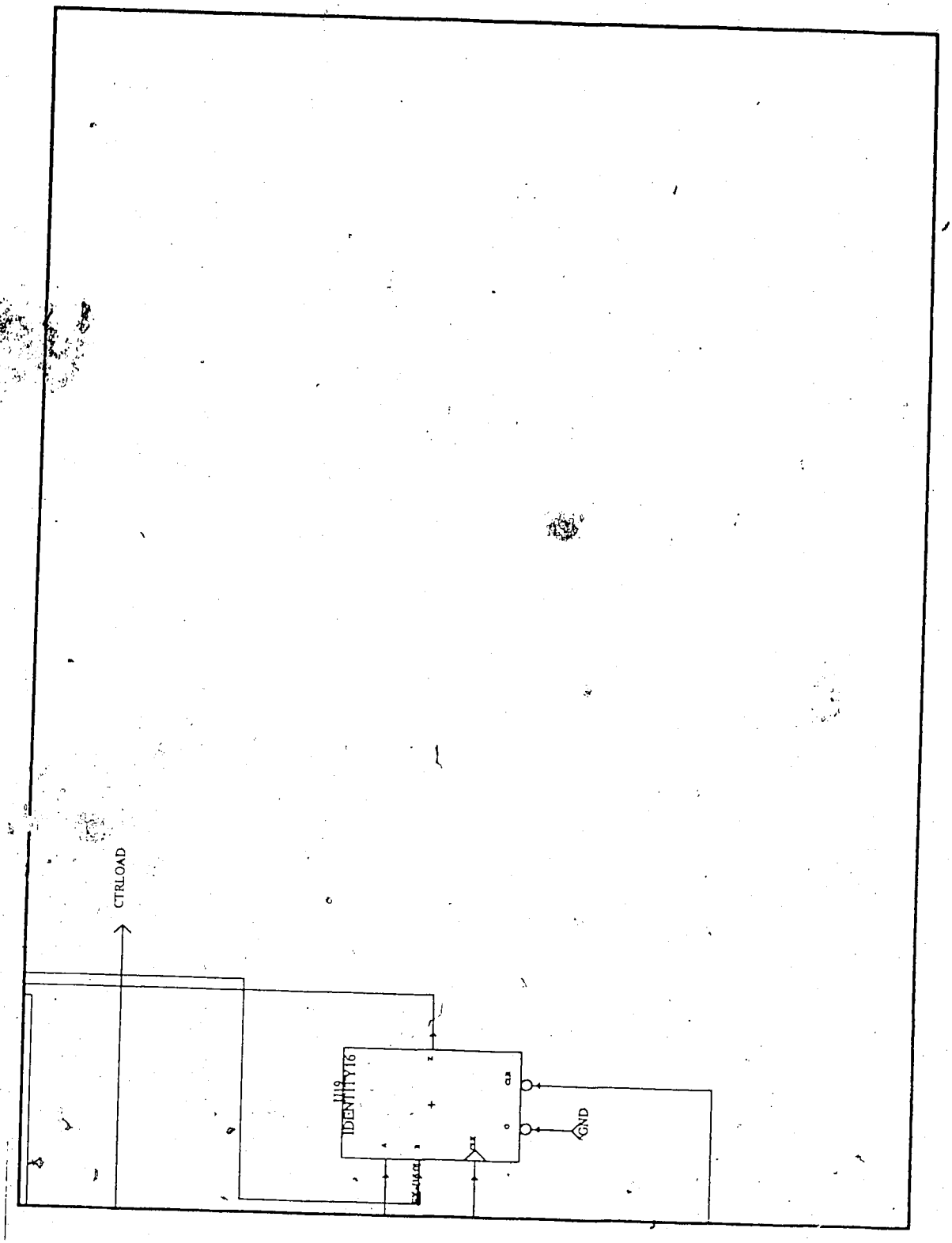

U12
FD3
^PWR
U11
ND3
U10
U21
BUF8
TYPE-01
RESET
DATA
VCR
E

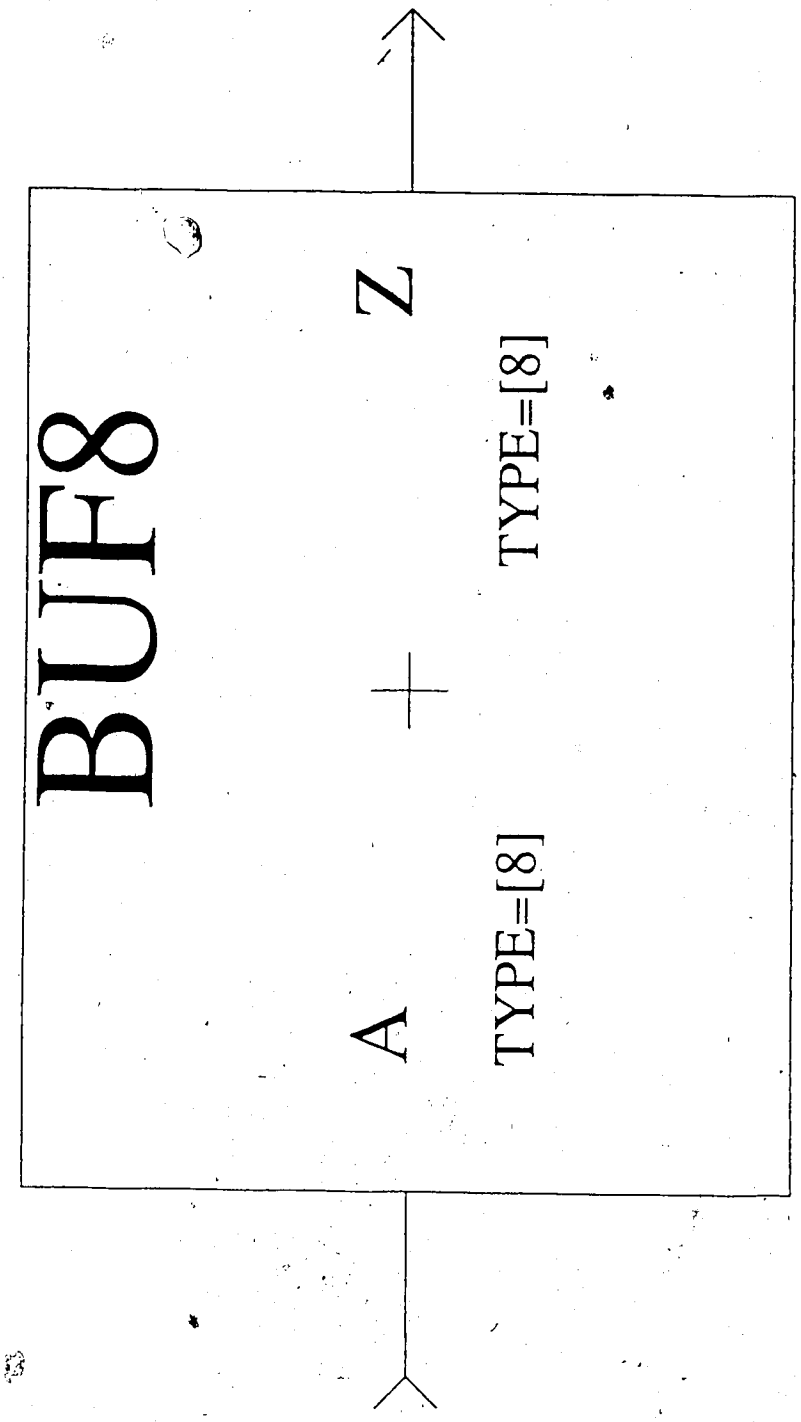

CLRCTR
CTREN

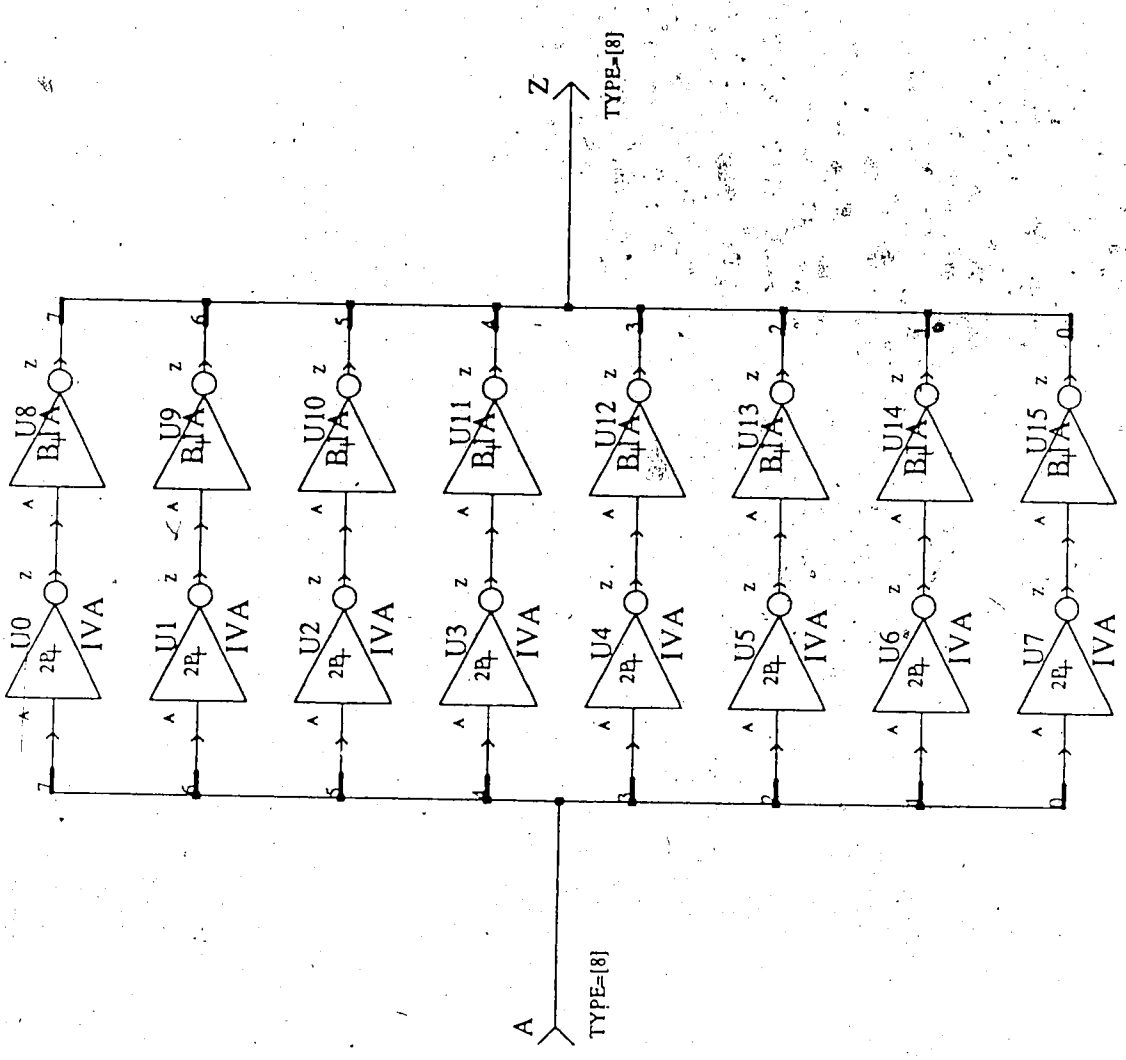

U24 ND2
U15 FD3
^PWR
U14 IV
U13 NR2
U22 NR2
U14 FD3
^PWR
U30 ND3
U13 IV
U29 IV
SYNC01 U28
U26 ND2
U23 IV
SYNC10 U27
U16 NR2
U17 FD3
U6 PROGDELAY
U11 LATCH20
U12
U9 REG8
U37 REG8
ASYNCSTART
ASYNCSTOP
VCLK
PDR
BARE

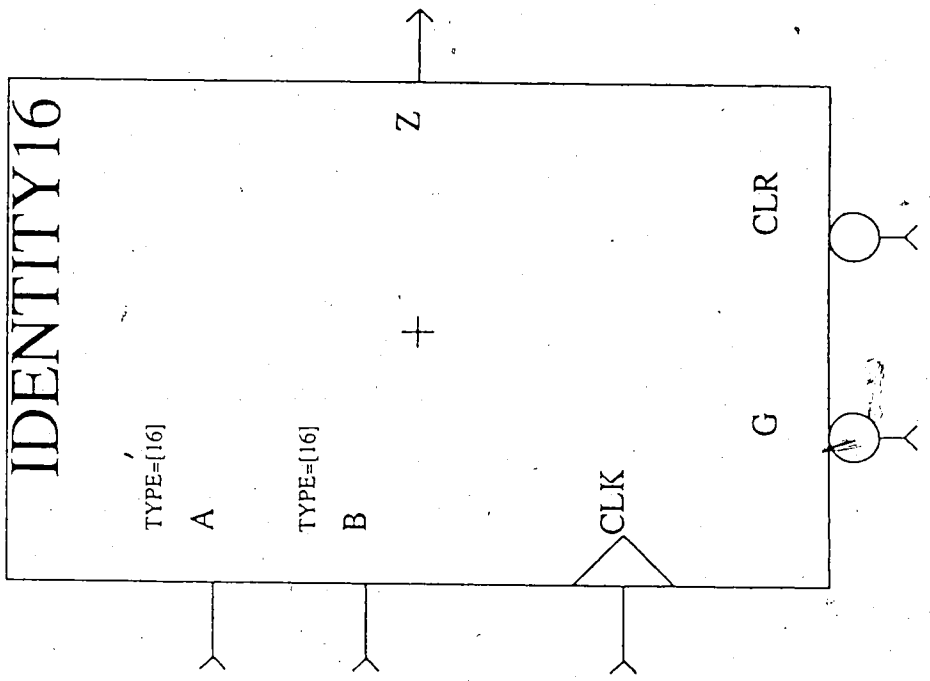

ADCEN
CTRDATA
U31
U30
U15
REG20
DATA

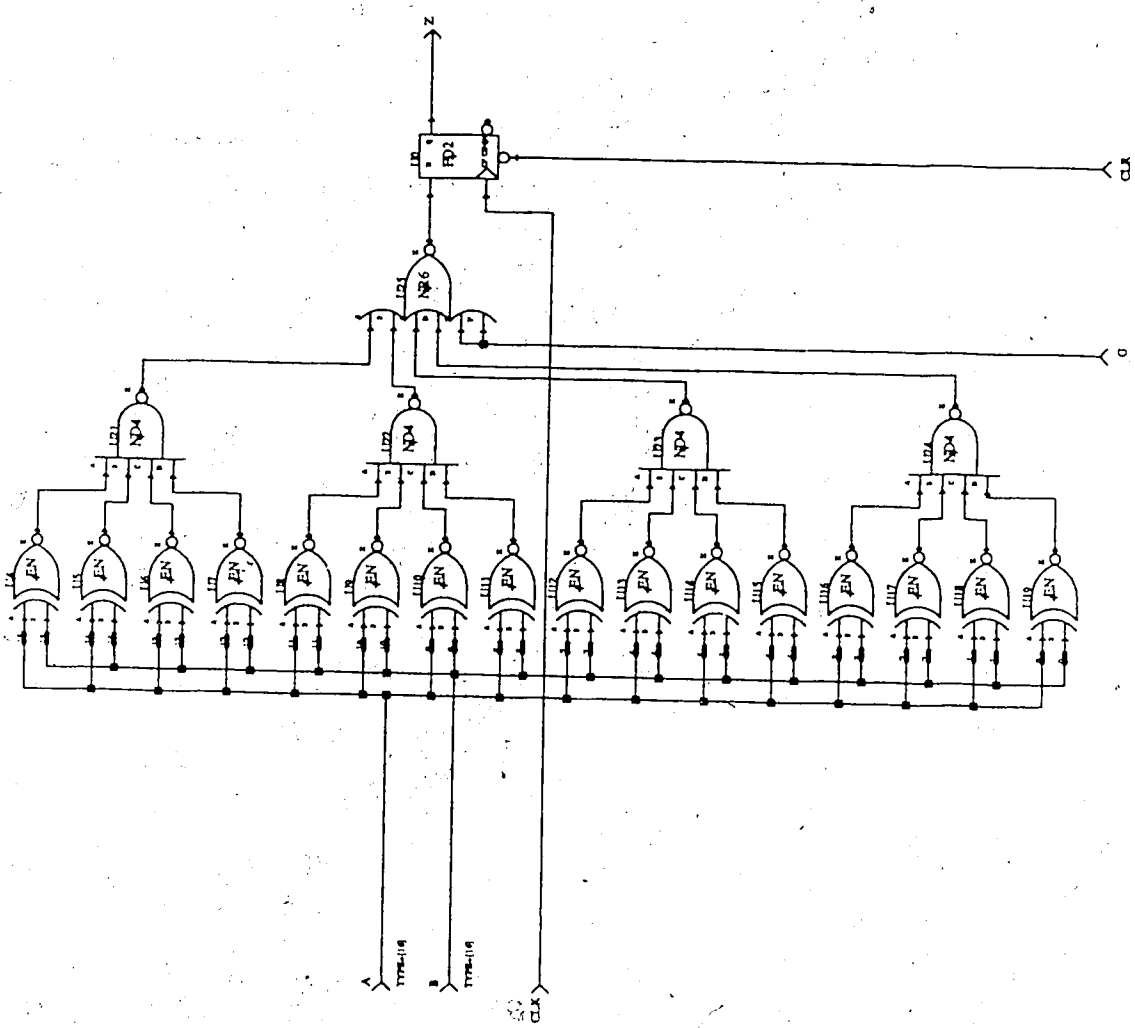

U125
FDM
U18
FA16
^PWR
U12
REGSYNC
U17
REG16
U16
LATCH16
U20
REGSYNC
VCR
CTR
CREC

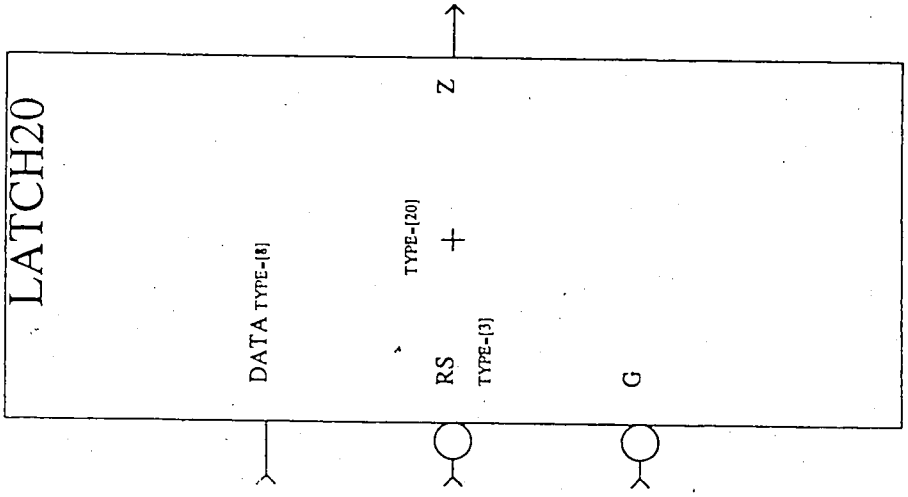

CTRLOAD
U19
IDENTITY16
GND

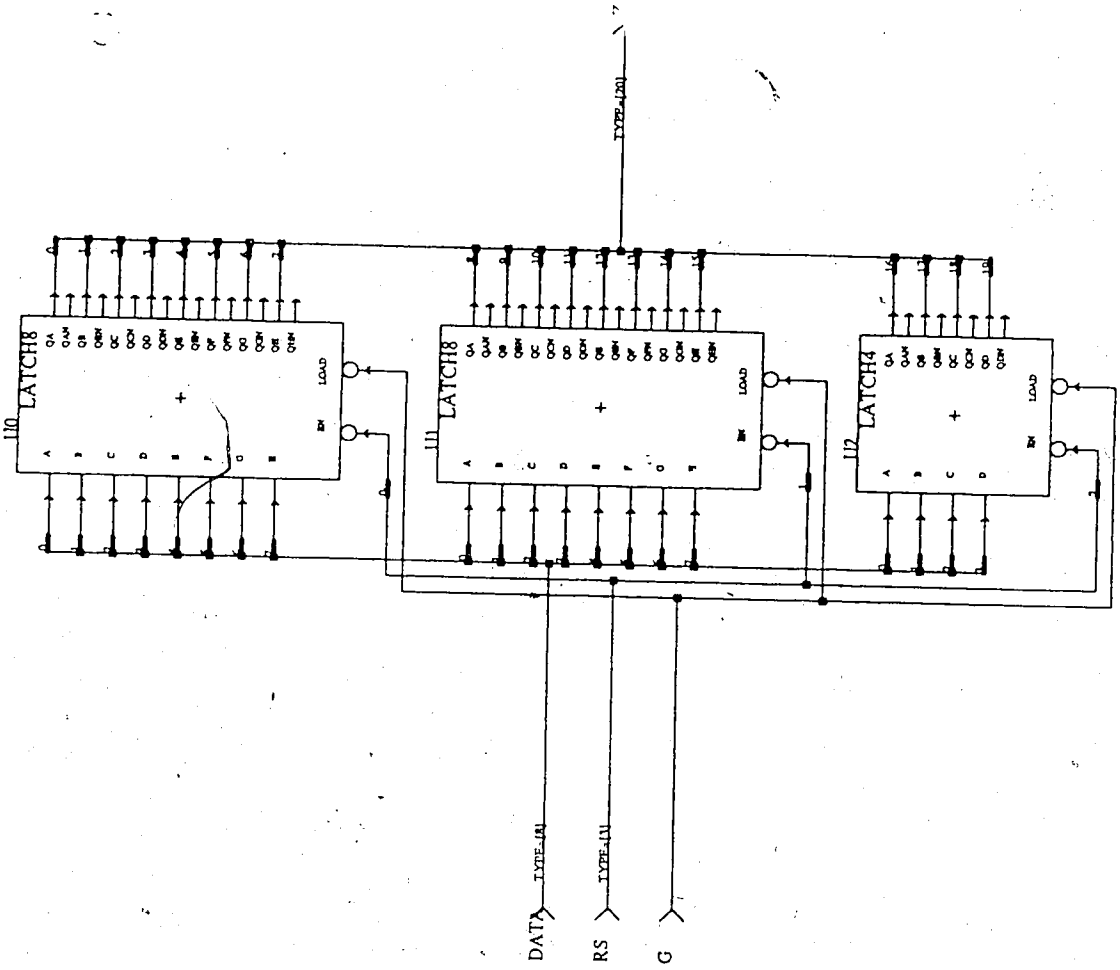

BUF8
A
TYPE=[8]
Z
TYPE=[8]

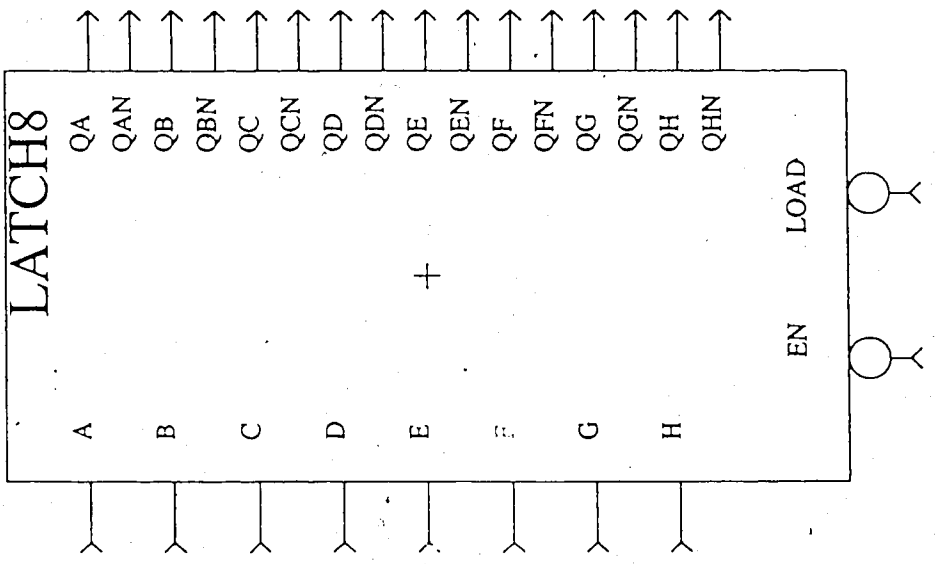

Z
TYPE=[8]
A
TYPE=[8]
U0
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U12
U13
U14
U15
IVA
BUFA
2B
A
Z

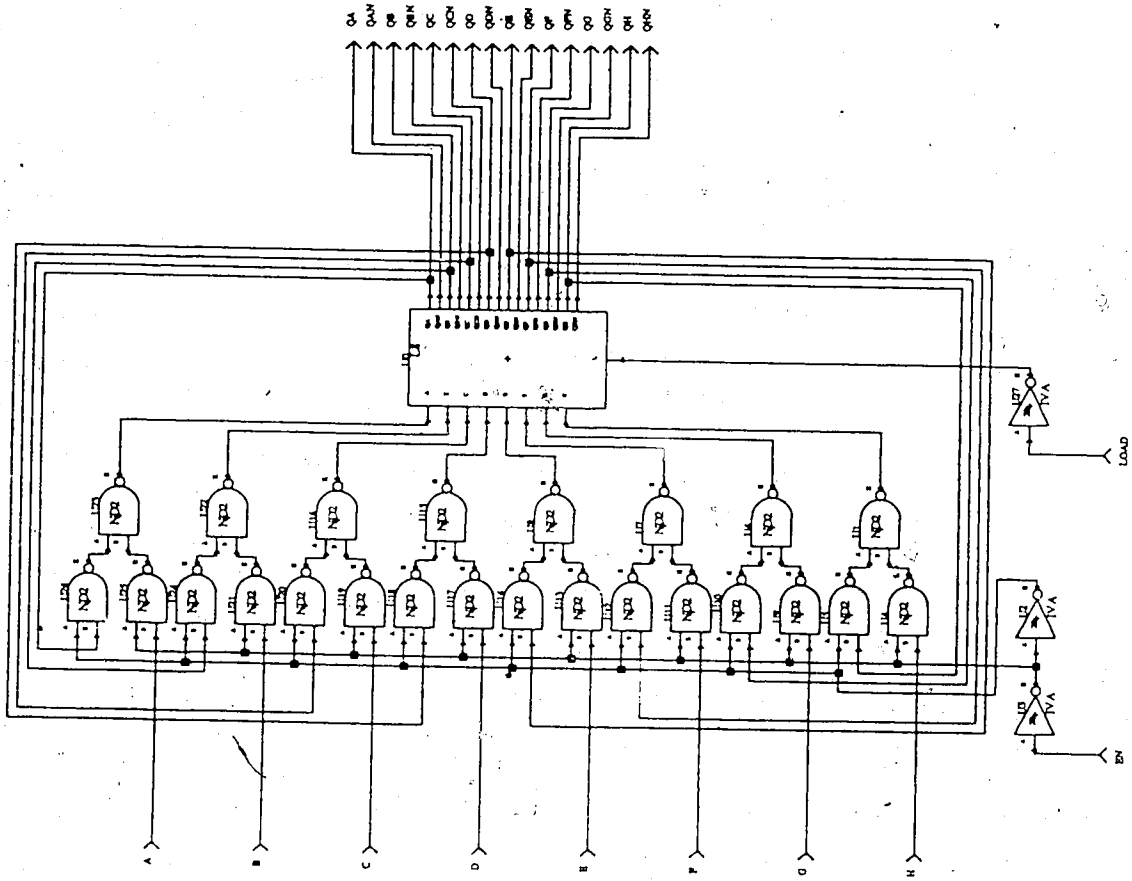

IDENTITY16
TYPE=[16]
A
TYPE=[16]
B
Z
+
CLK
G
CLR

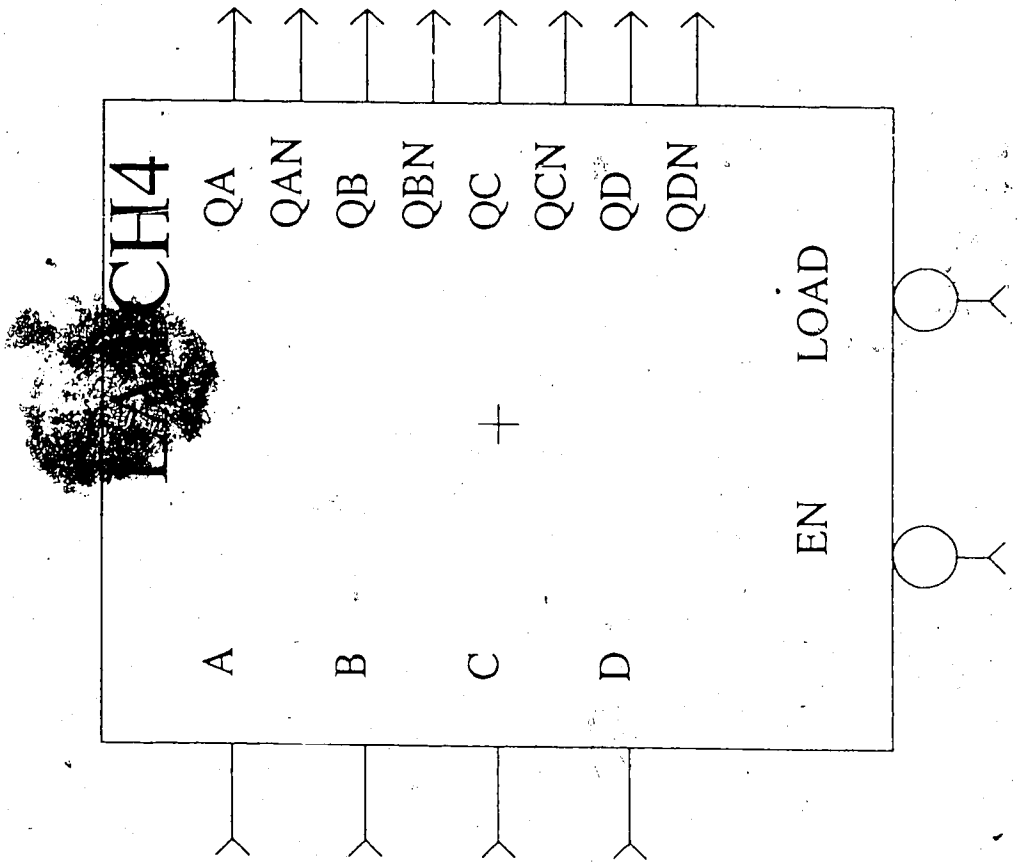


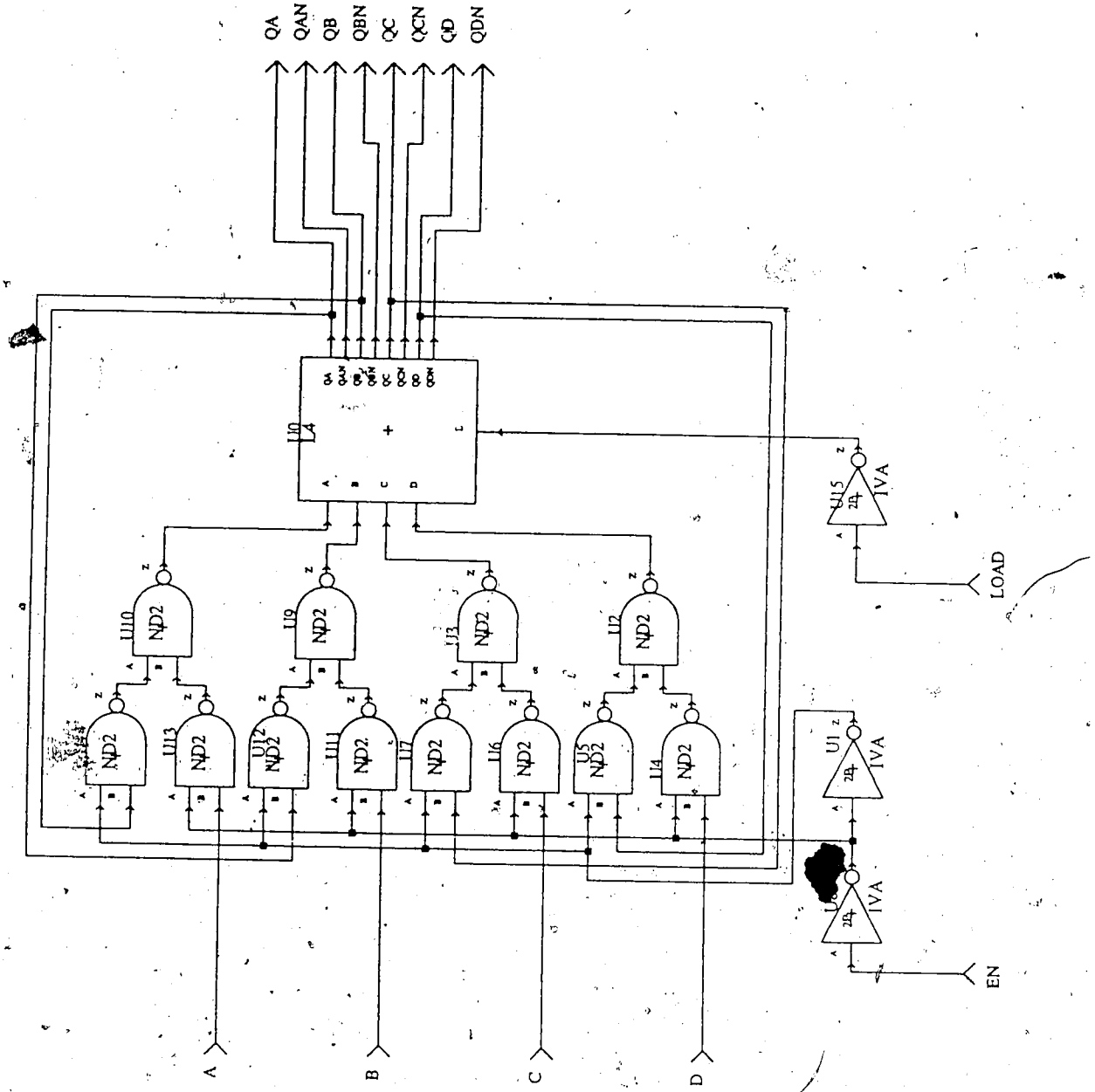

LATCH20
DATA TYPE=[8]
RS
TYPE=[3]
TYPE=[20]
Z
G

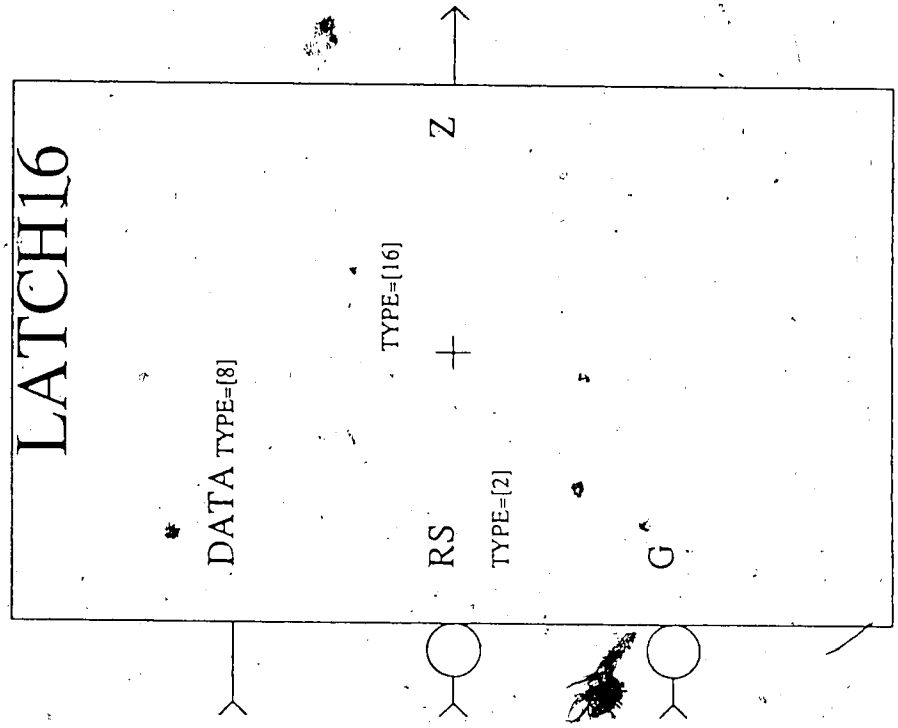

TYPE=D[0]
LATCH8
U10
LATCH8
U11
LATCH4
U12
LOAD
DATA
RS
G
TYPE=J[0]
TYPE=J[1]

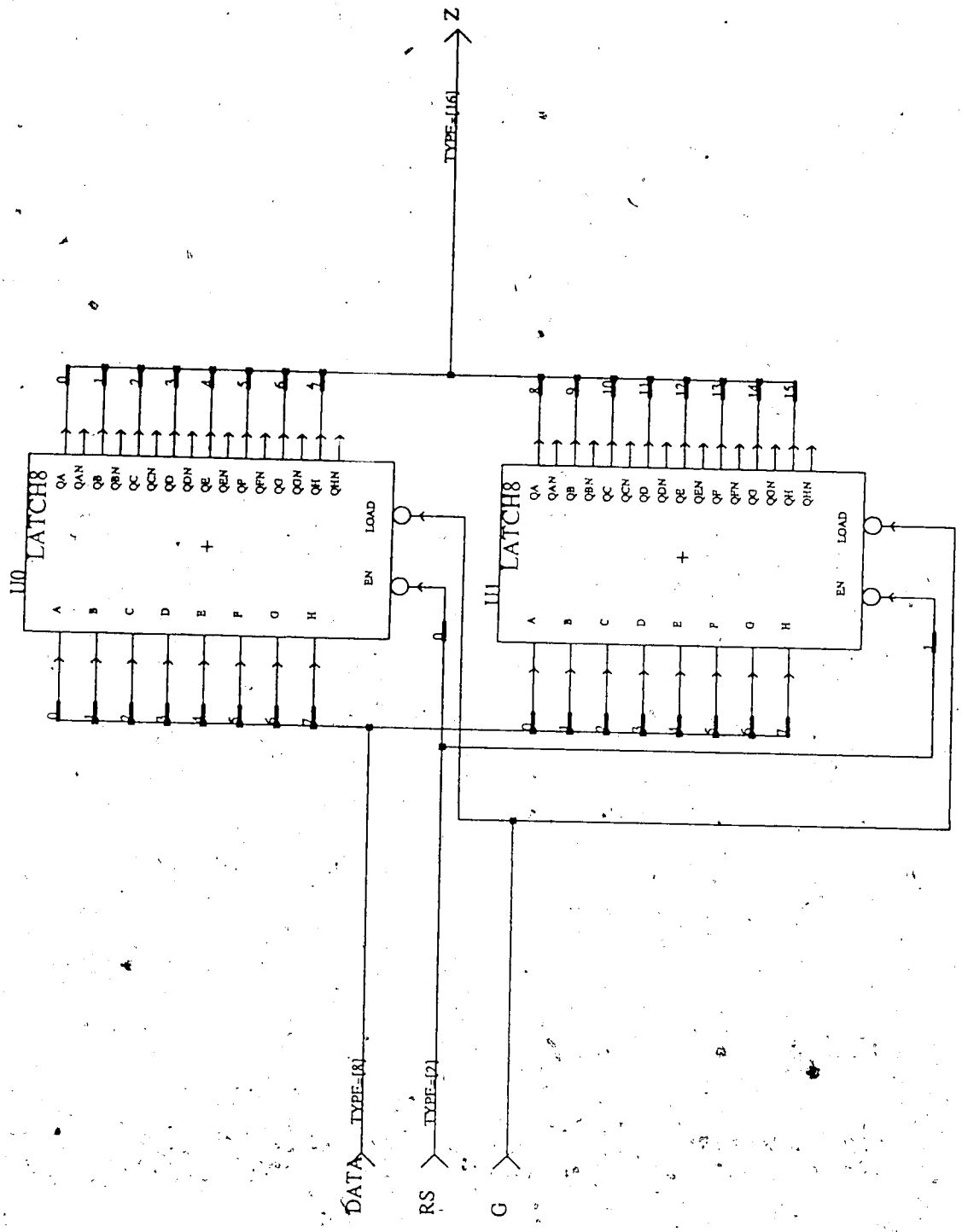

LATCH8
QA
QAN
QB
QBN
QC
QCN
QD
QDN
QE
QEN
QF
QFN
QG
QGN
QH
QHN
A
B
C
D
E
F
G
H
EN
LOAD

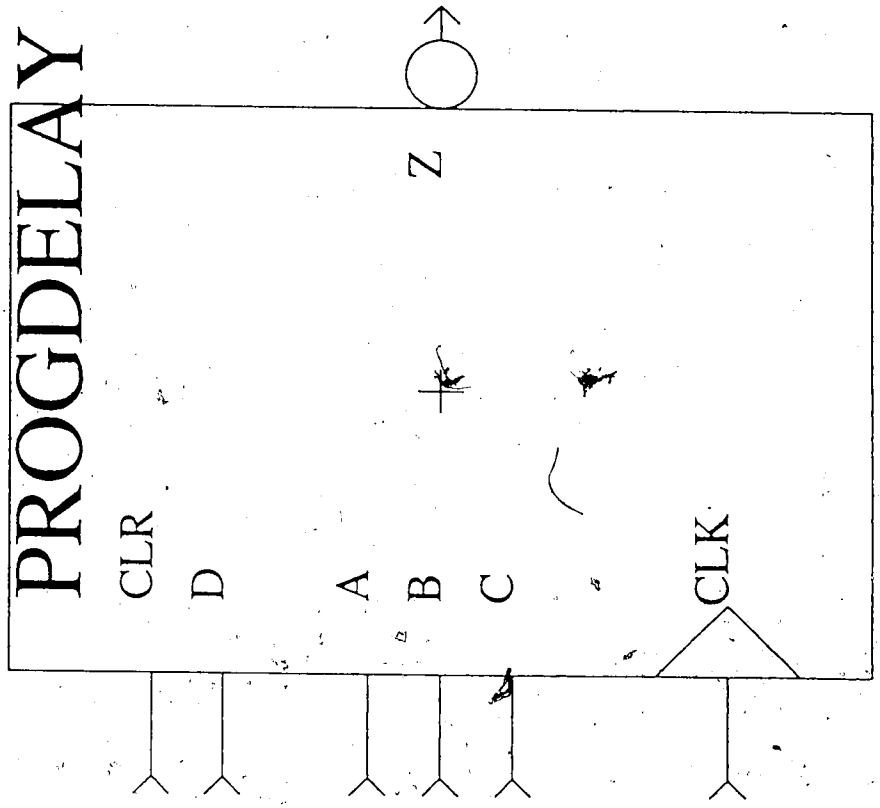


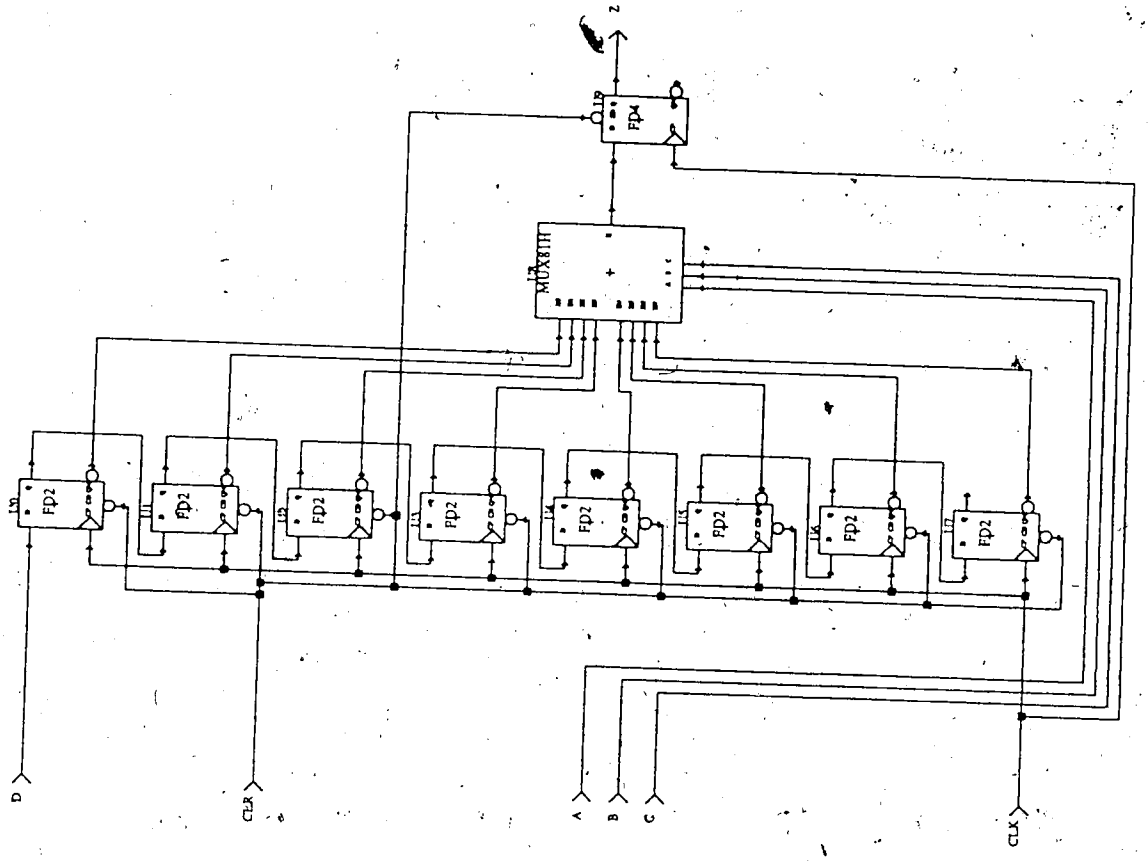

LCH4
QA
QAN
QB
QBN
QC
QCN
QD
QDN
A
B
C
D
EN
LOAD

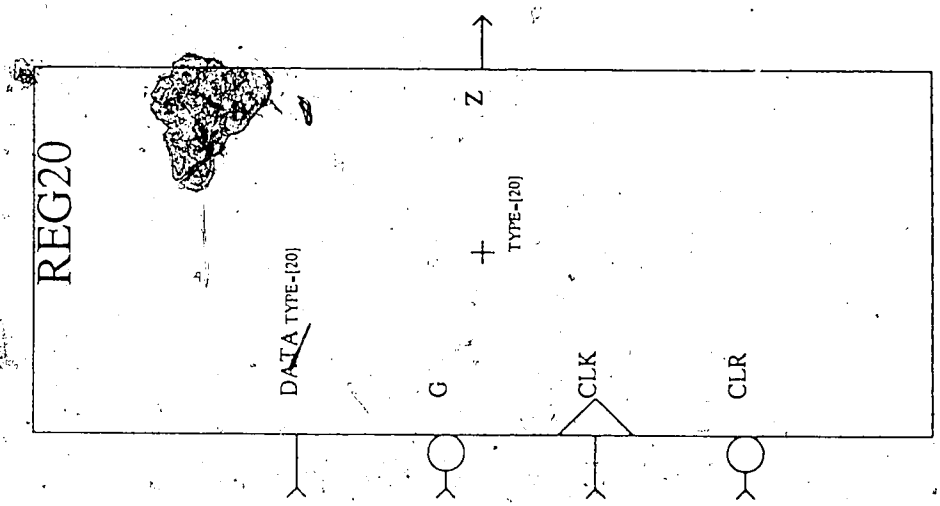

QA
QAN
QB
QBN
QC
QCN
QD
QDN
U10
U15
IVA
LOAD
U1
EN
A
B
C
D

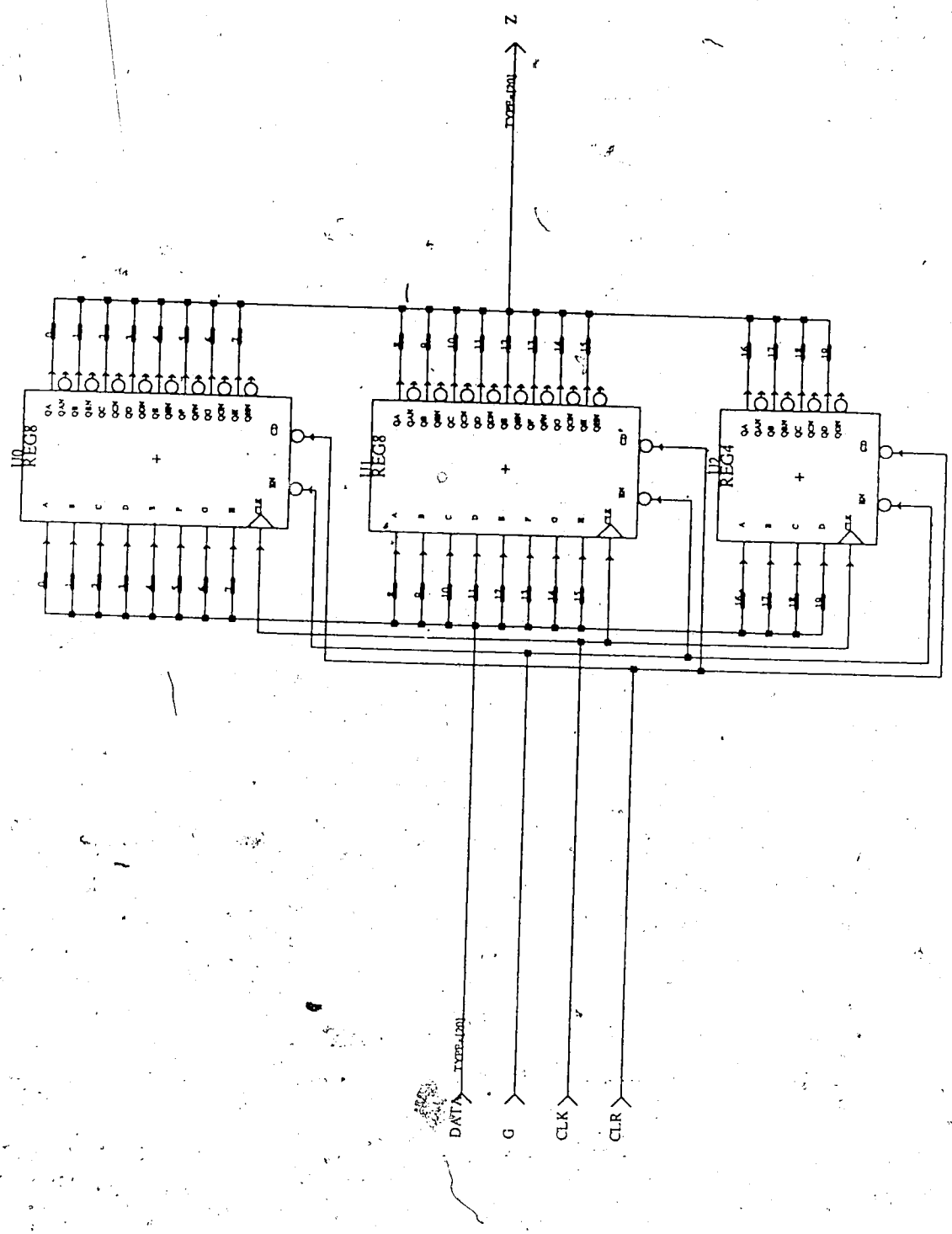

LATCH16
DATA TYPE=[8]
RS
TYPE=[2]
TYPE=[16]
Z
G

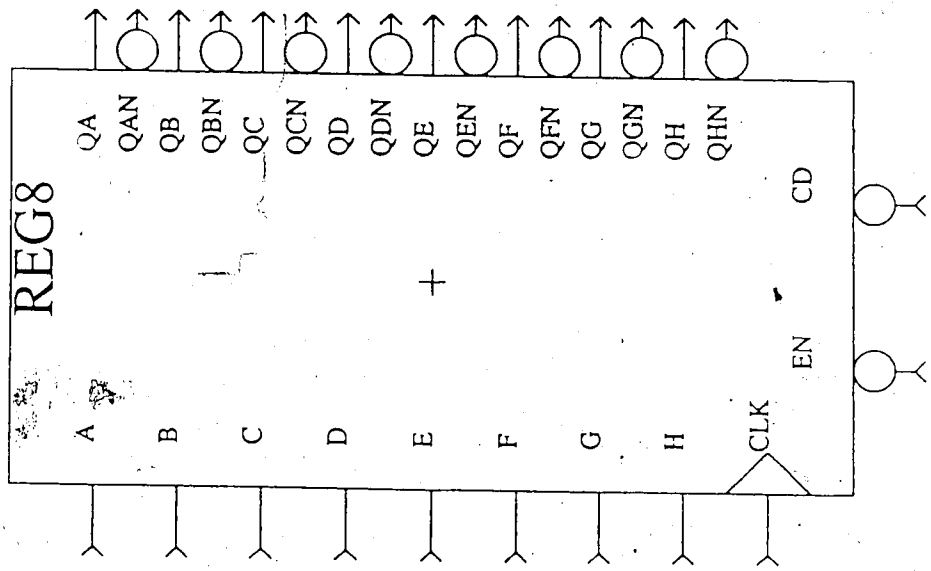

Z
TYPE=[16]
U0
LATCH8
U1
LATCH8
LOAD
EN
DATA
TYPE=[8]
RS
TYPE=[2]
G

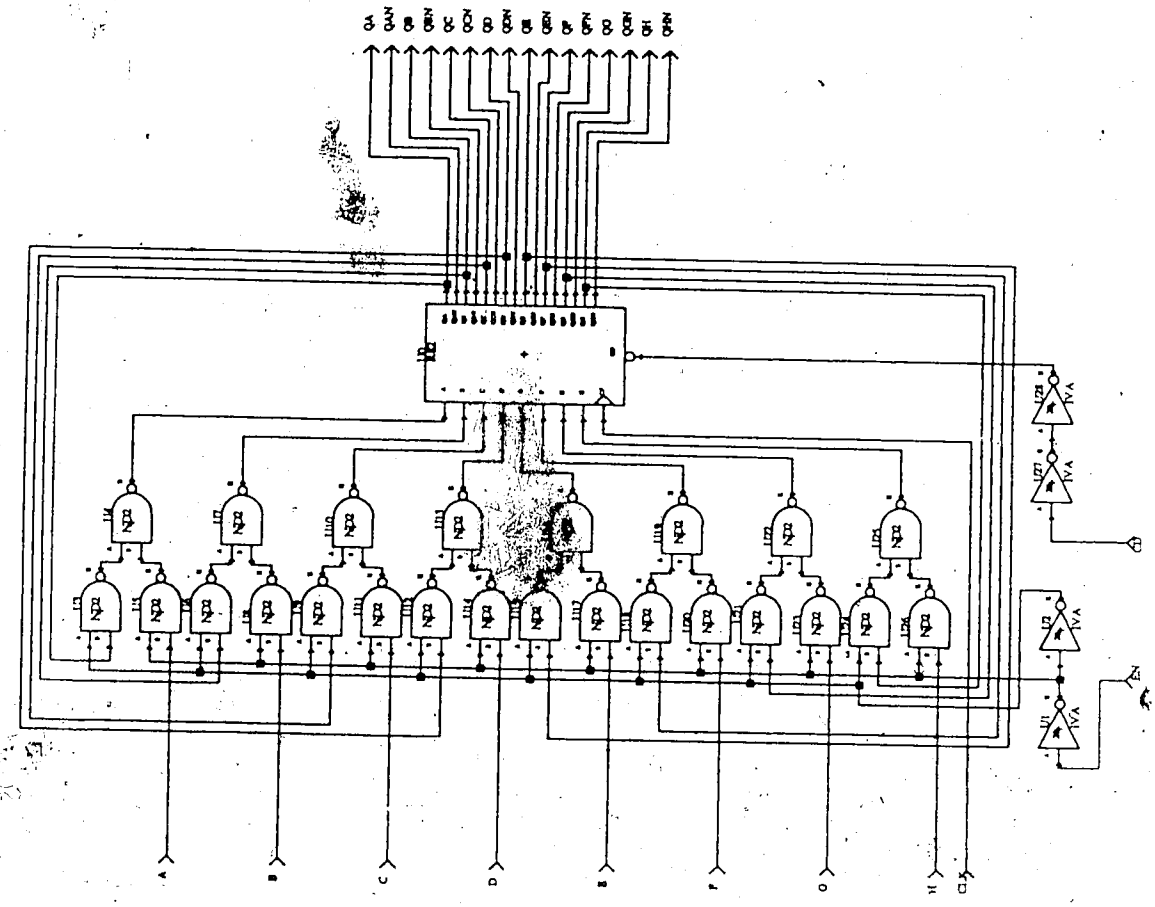

PROGDELAY
CLR
D
A
B
C
CLK
Z

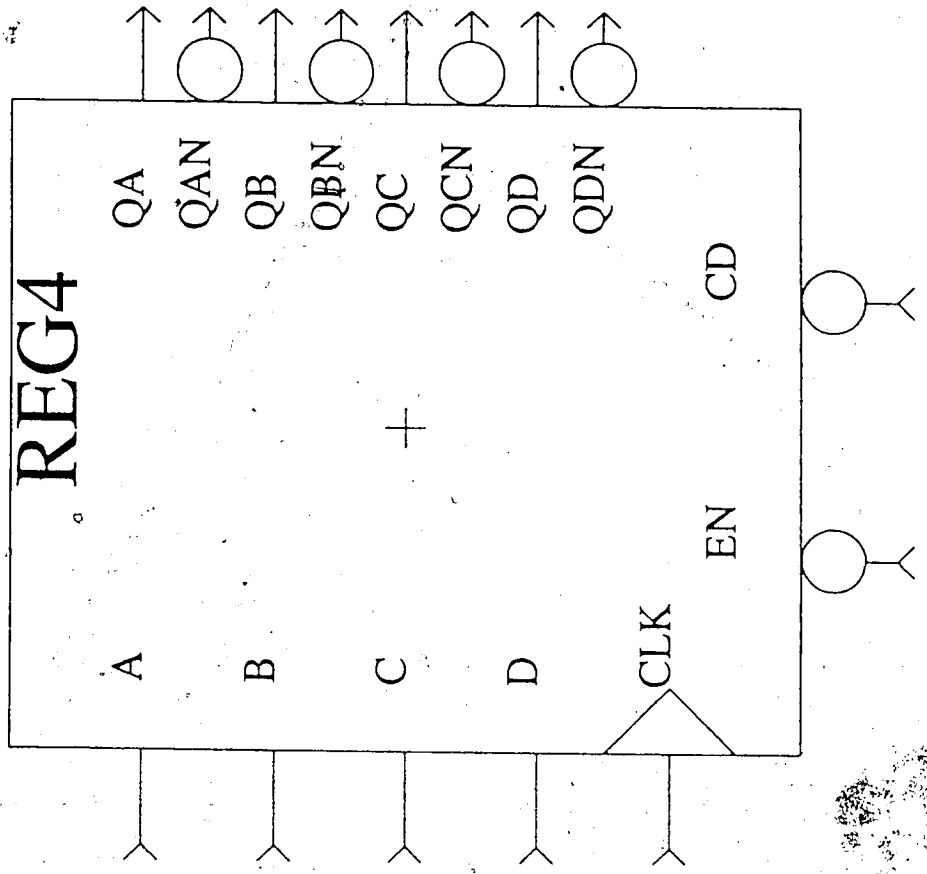

Z
U8
Fp4
U3
MUX8IN
U0
Fp2
U1
Fp2
U2
Fp2
U3
Fp2
U4
Fp2
U5
Fp2
U6
Fp2
U7
Fp2
D
CLR
A
B
C
CLK

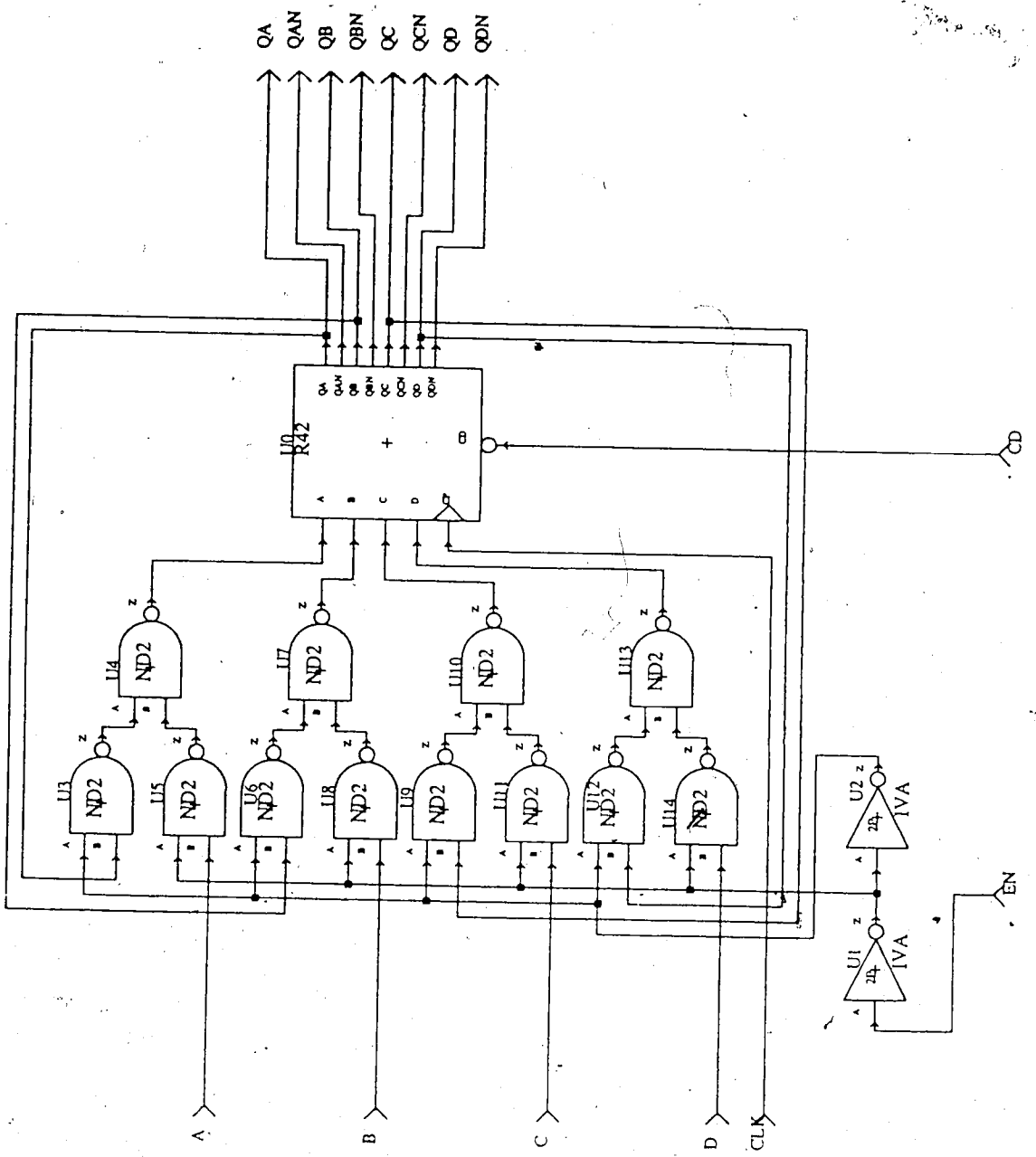

REG20
DATA TYPE=[20]
G
+
TYPE=[20]
Z
CLK
CLR

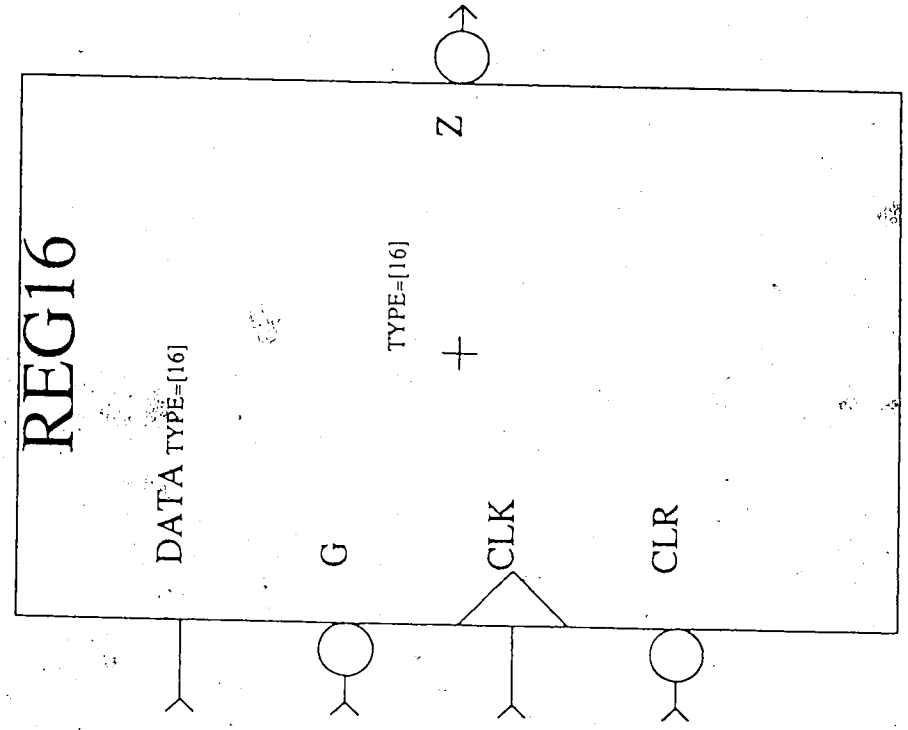

Z
TYPE=IO01
U0
REG8
U1
REG8
U2
REG4
DATA
TYPE=IO01
G
CLK
CLR

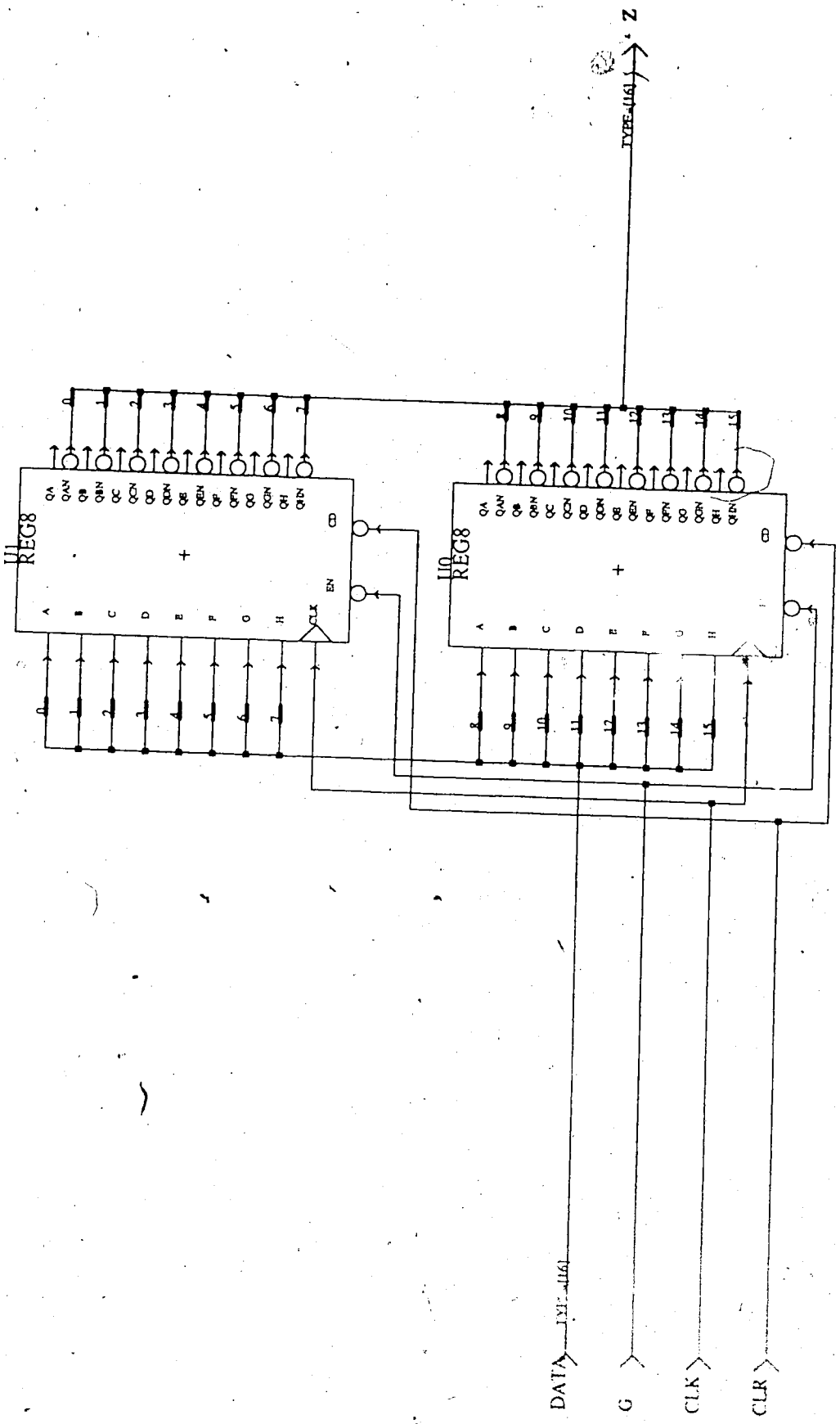

REG8
QA
QAN
QB
QBN
QC
QCN
QD
QDN
QE
QEN
QF
QFN
QG
QGN
QH
QHN
CD
EN
A
B
C
D
E
F
G
H
CLK

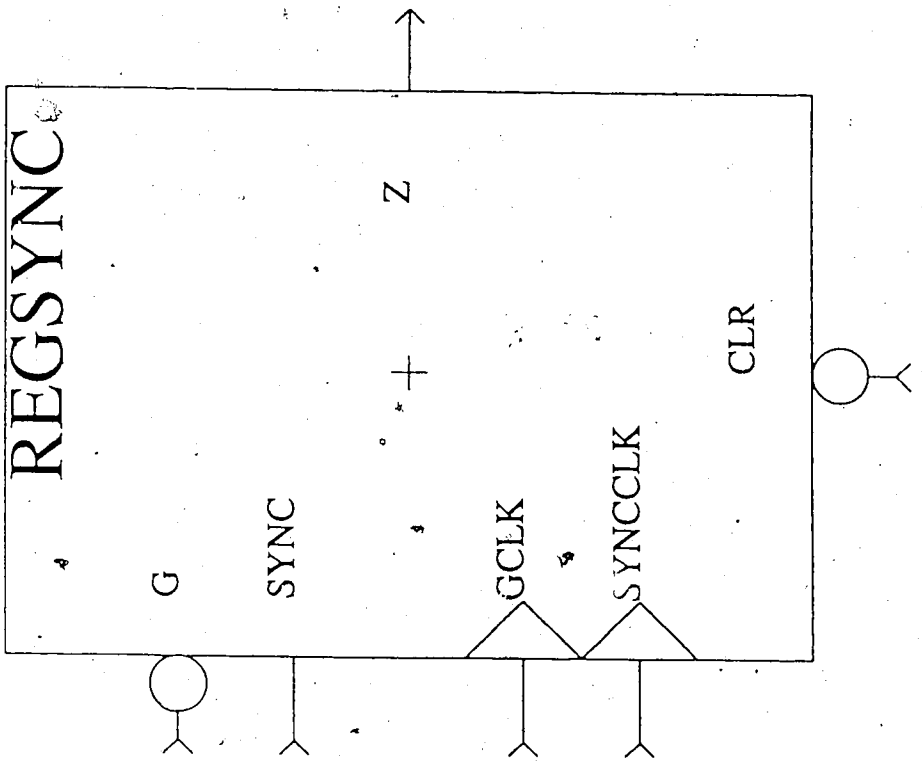


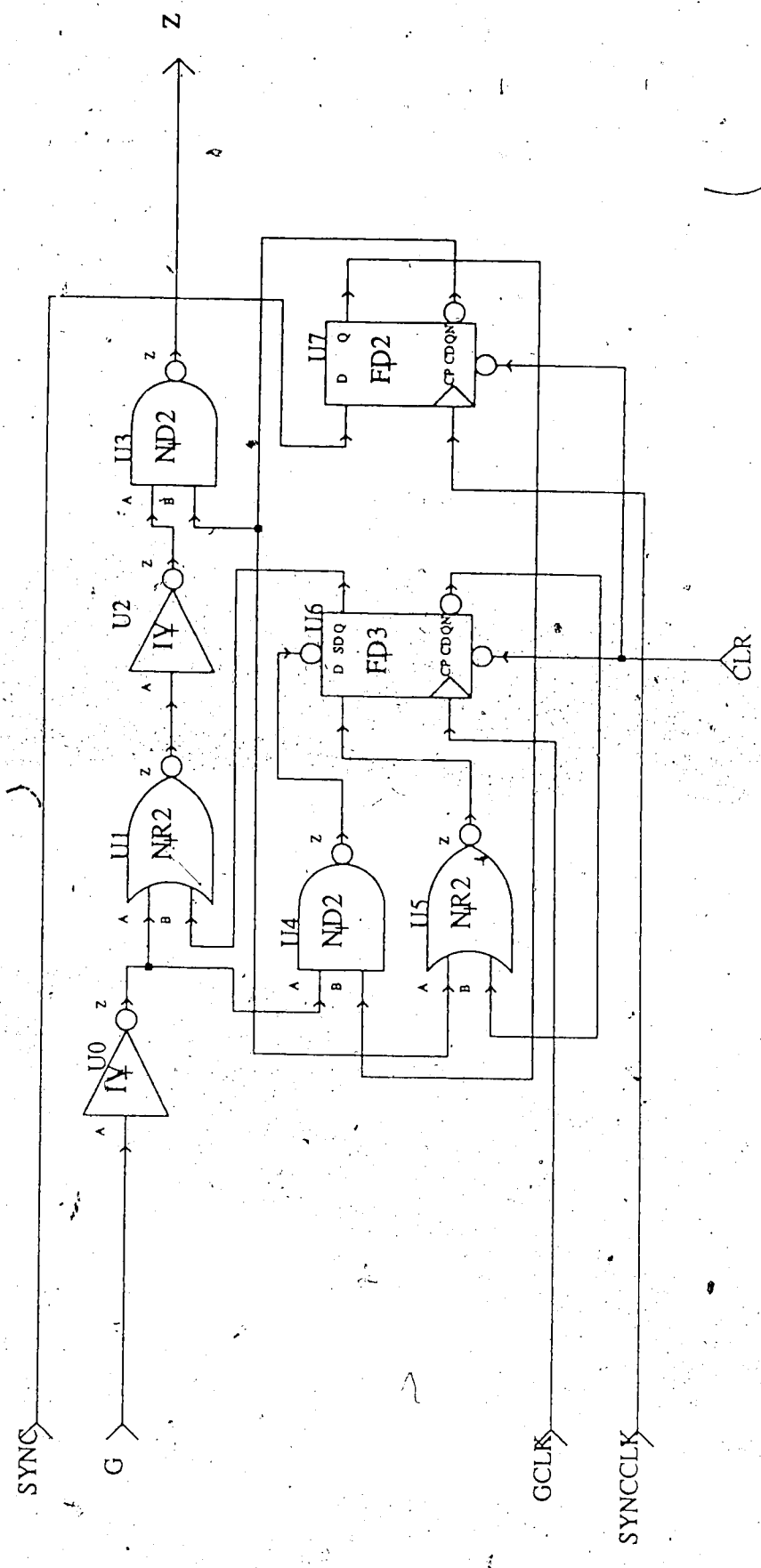

REG4
QA
QAN
QB
QBN
QC
QCN
QD
QDN
CD
EN
A
B
C
D
CLK

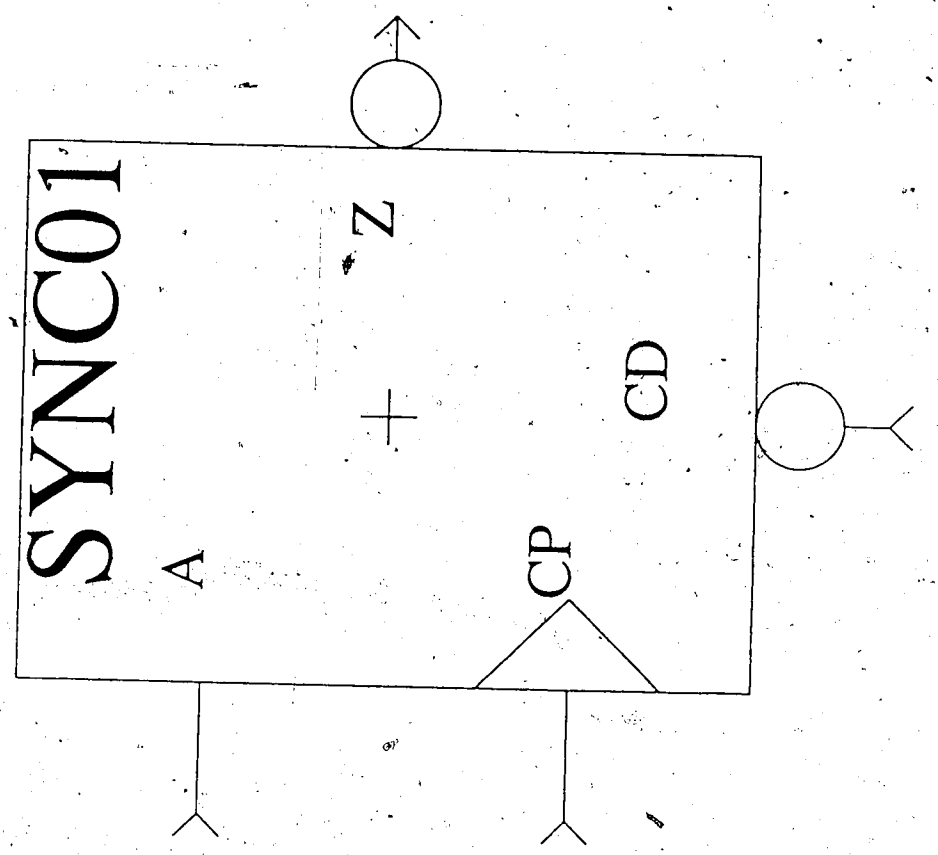

QA
QAN
QB
QBN
QC
QCN
QD
QDN
U0
R42
CD
U4
ND2
U7
ND2
U10
ND2
U13
ND2
U3
ND2
U5
ND2
U6
ND2
U8
ND2
U9
ND2
U11
ND2
U12
ND2
U14
ND2
U2
IVA
U1
IVA
EN
A
B
C
D
CLK

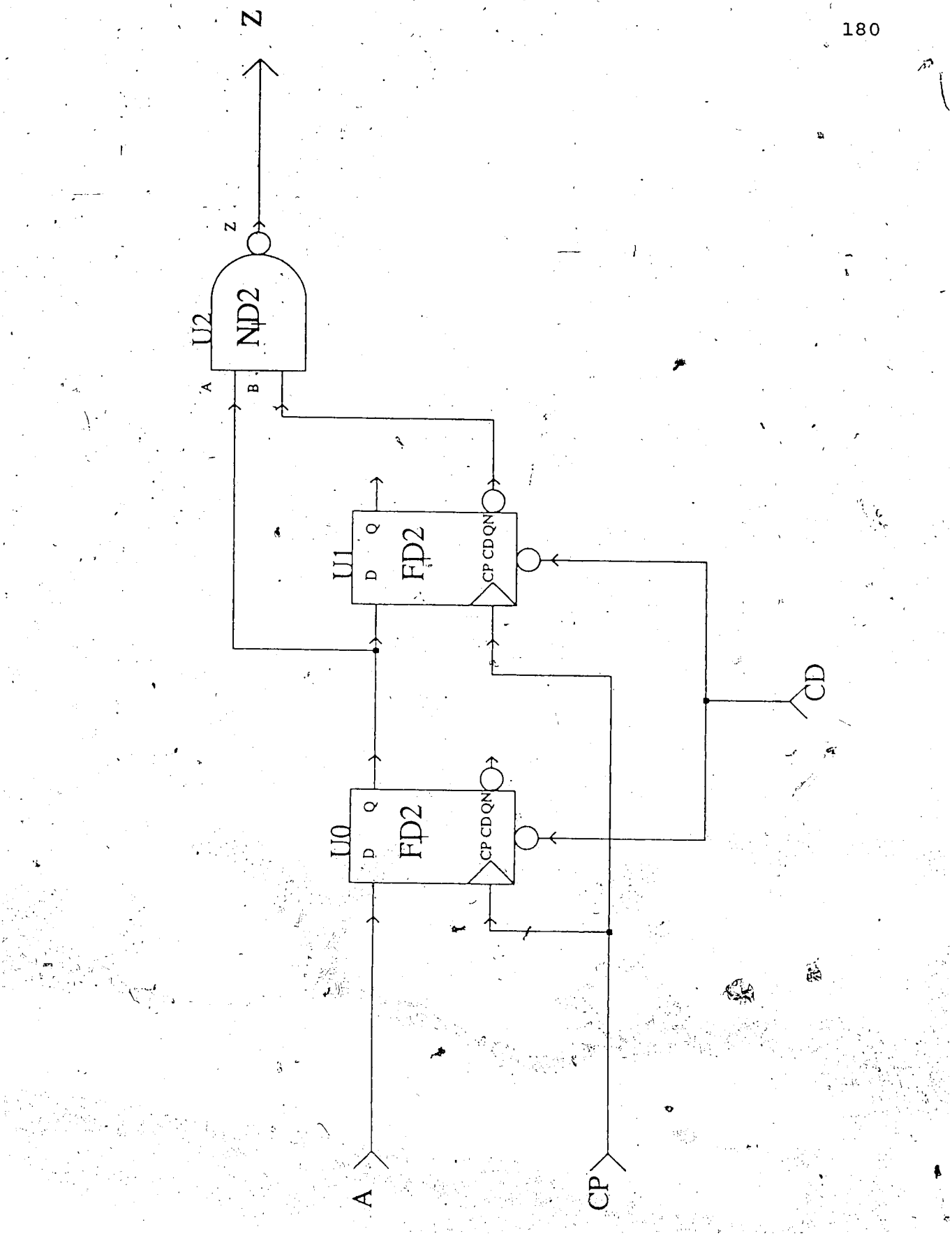

REG16
DATA TYPE=[16]
G
TYPE=[16]
CLK
CLR
Z

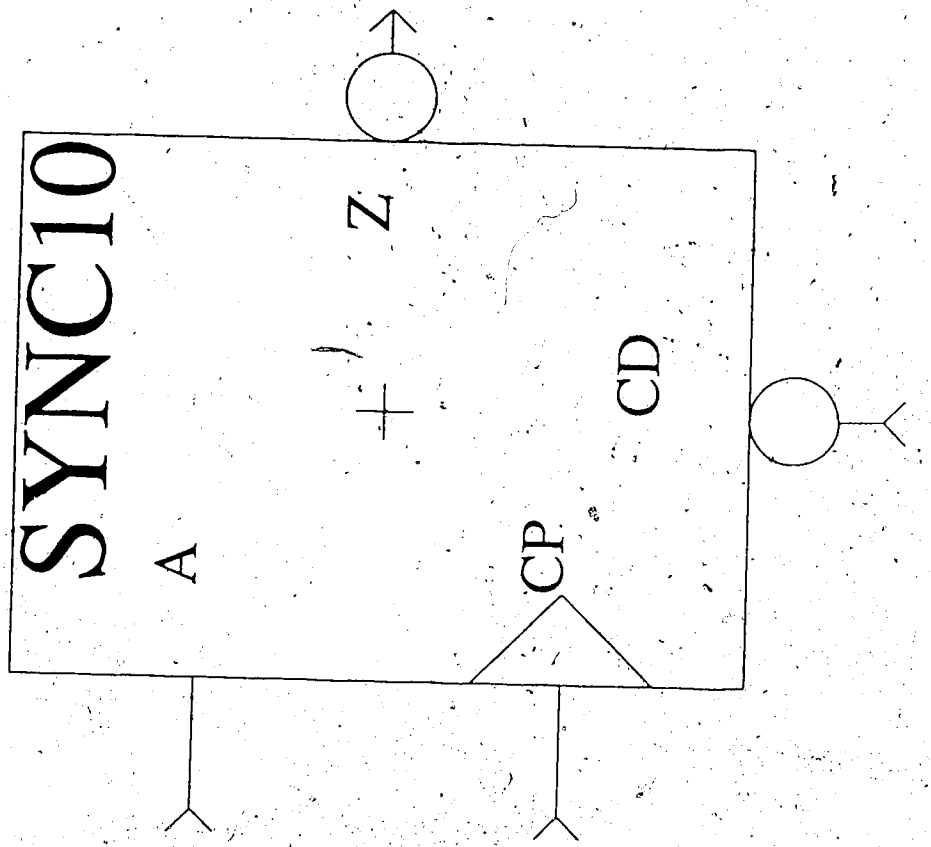

Z
TYPE 4161
U1
REG8
U0
REG8
A
B
C
D
E
F
G
H
CLK
EN
DATA
G
CLK
CLR

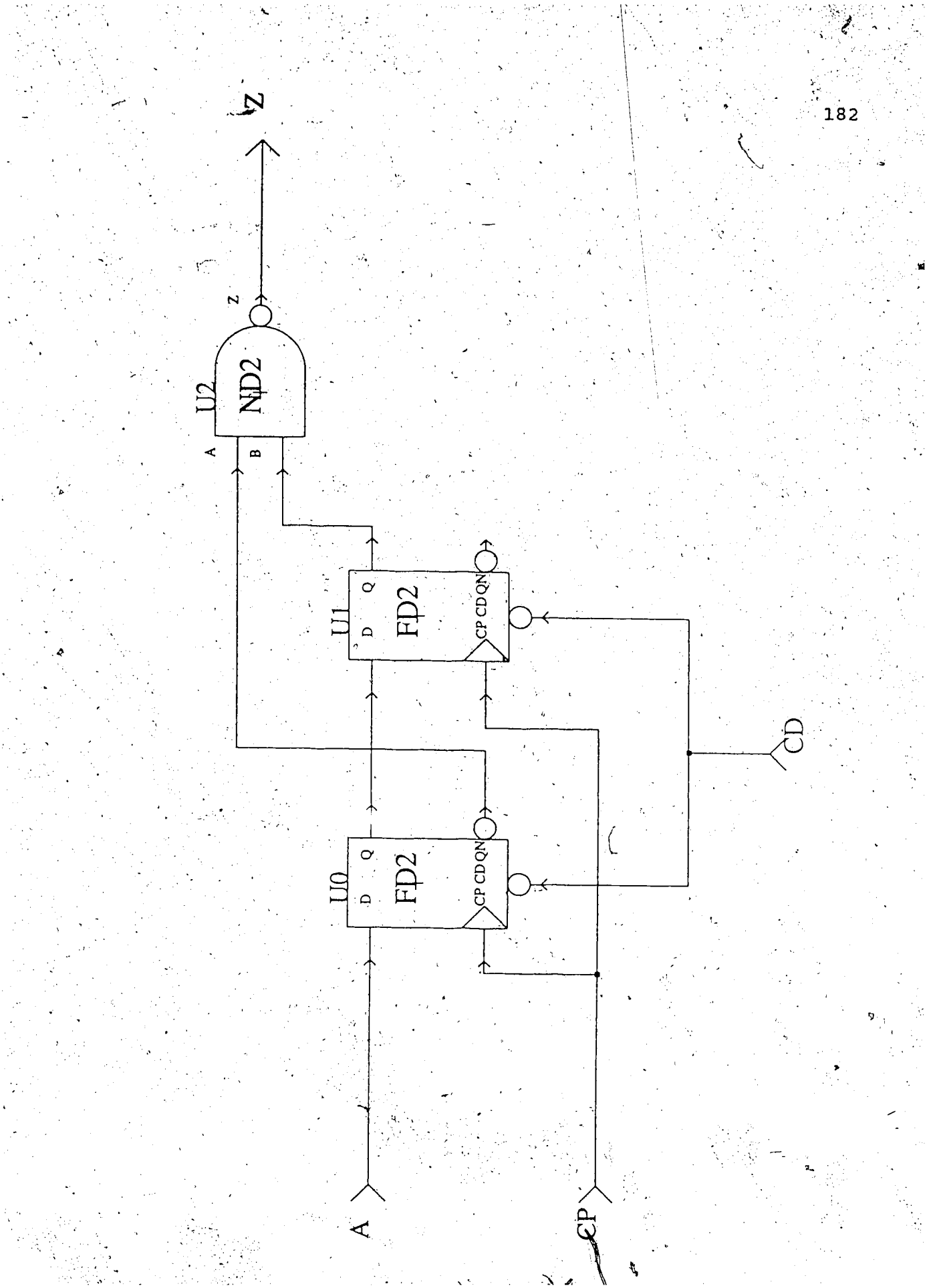

REGSYNC
G
SYNC
Z
GCLK
SYNCCLK
CLR

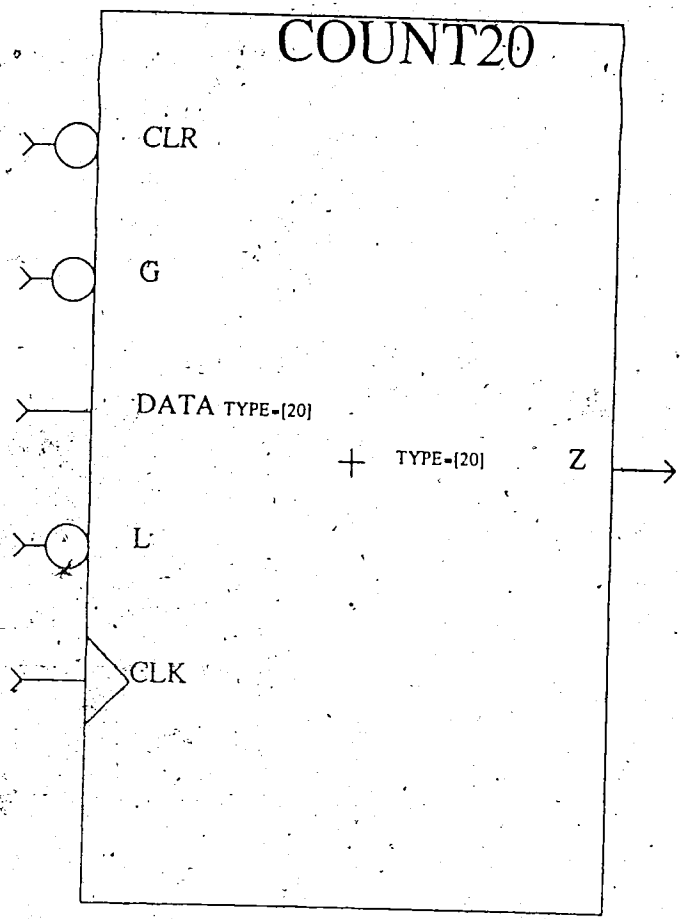

Z
U7
FD2
D Q
CP CDQN
U3
ND2
A B Z
U2
IV
A Z
U6
FD3
D SDQ
CP CDQN
CLR
U1
NR2
A B Z
U4
ND2
A B Z
U5
NR2
A B Z
U0
IV
A Z
SYNC
G
GCLK
SYNCCLK

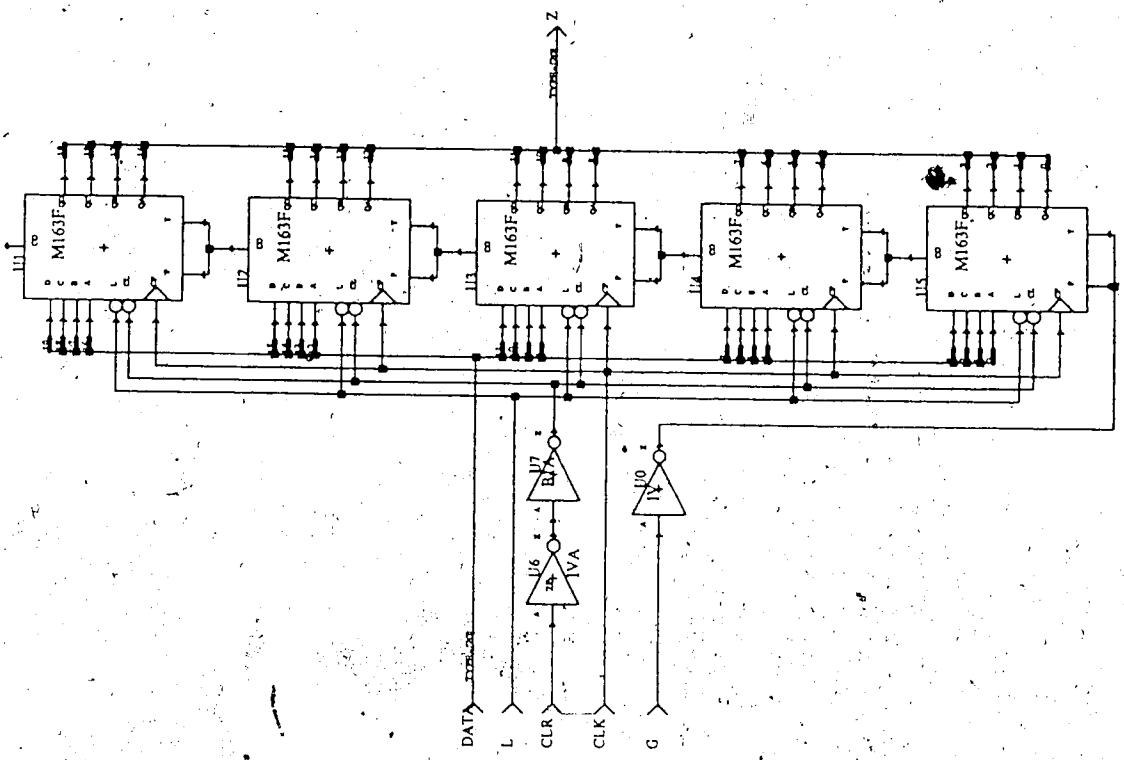

SYNC01
A
Z
CP
CD

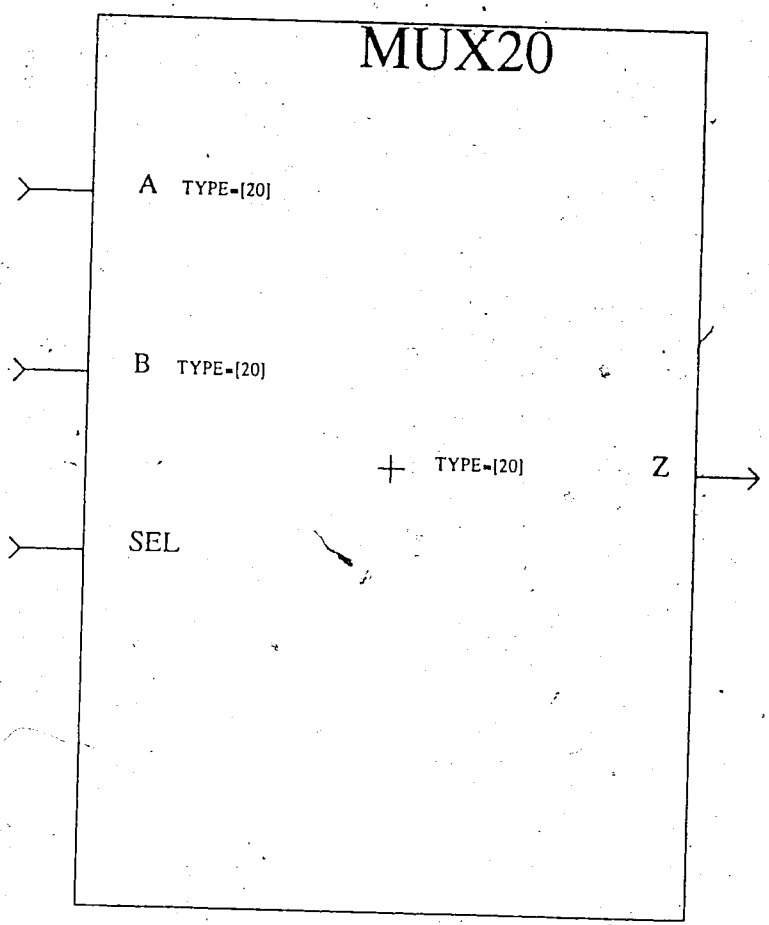

Z
Z
U2
ND2
A
B
U1
FD2
D
Q
CP CD QN
U0
FD2
D
Q
CP CD QN
CD
A
CP

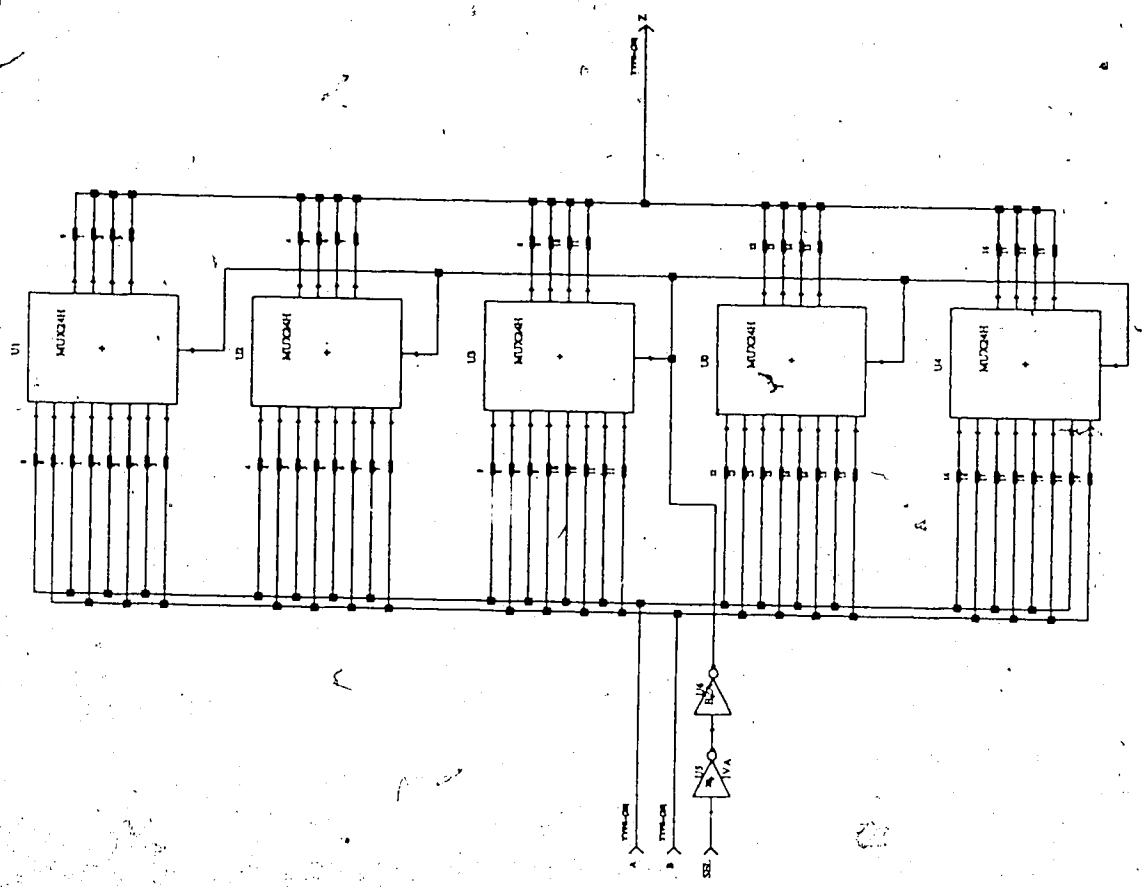

SYNC10
A
CP
CD
Z

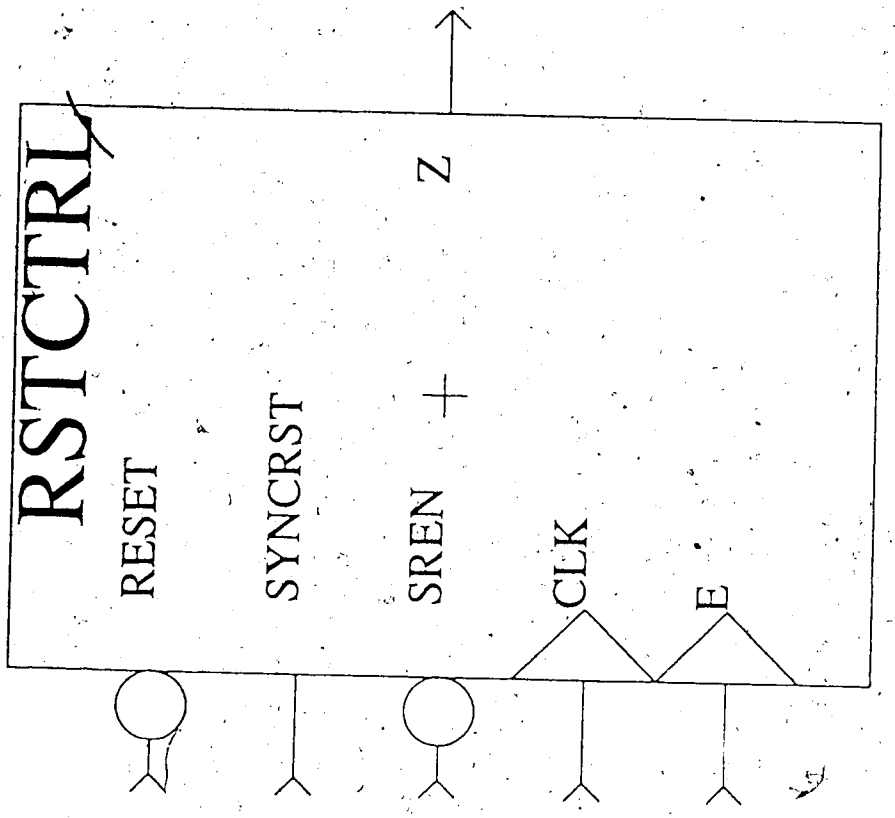

Z
U2
ND2
A
B
Z
U1
FD2
D
Q
CP CDQN
U0
FD2
D
Q
CP CDQN
CD
A
CP

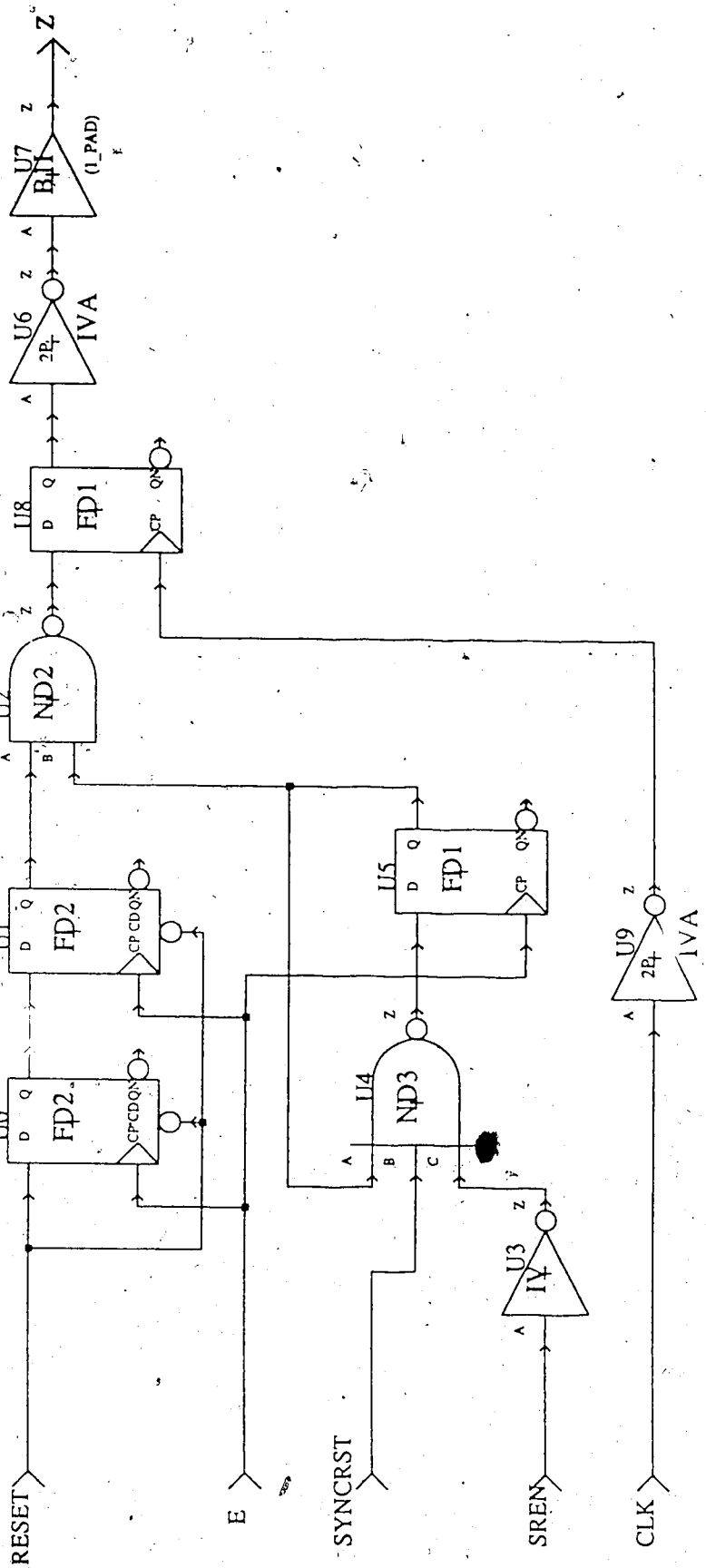

COUNT20
CLR
G
DATA TYPE-[20]
TYPE-[20]
Z
L
CLK


Z
M163F
U1
U2
U3
U4
U5
U6
U7
U10
VA
DATA
L
CLR
CLK
G


MUX20
A TYPE=[20]
B TYPE=[20]
TYPE=[20]
Z
SEL

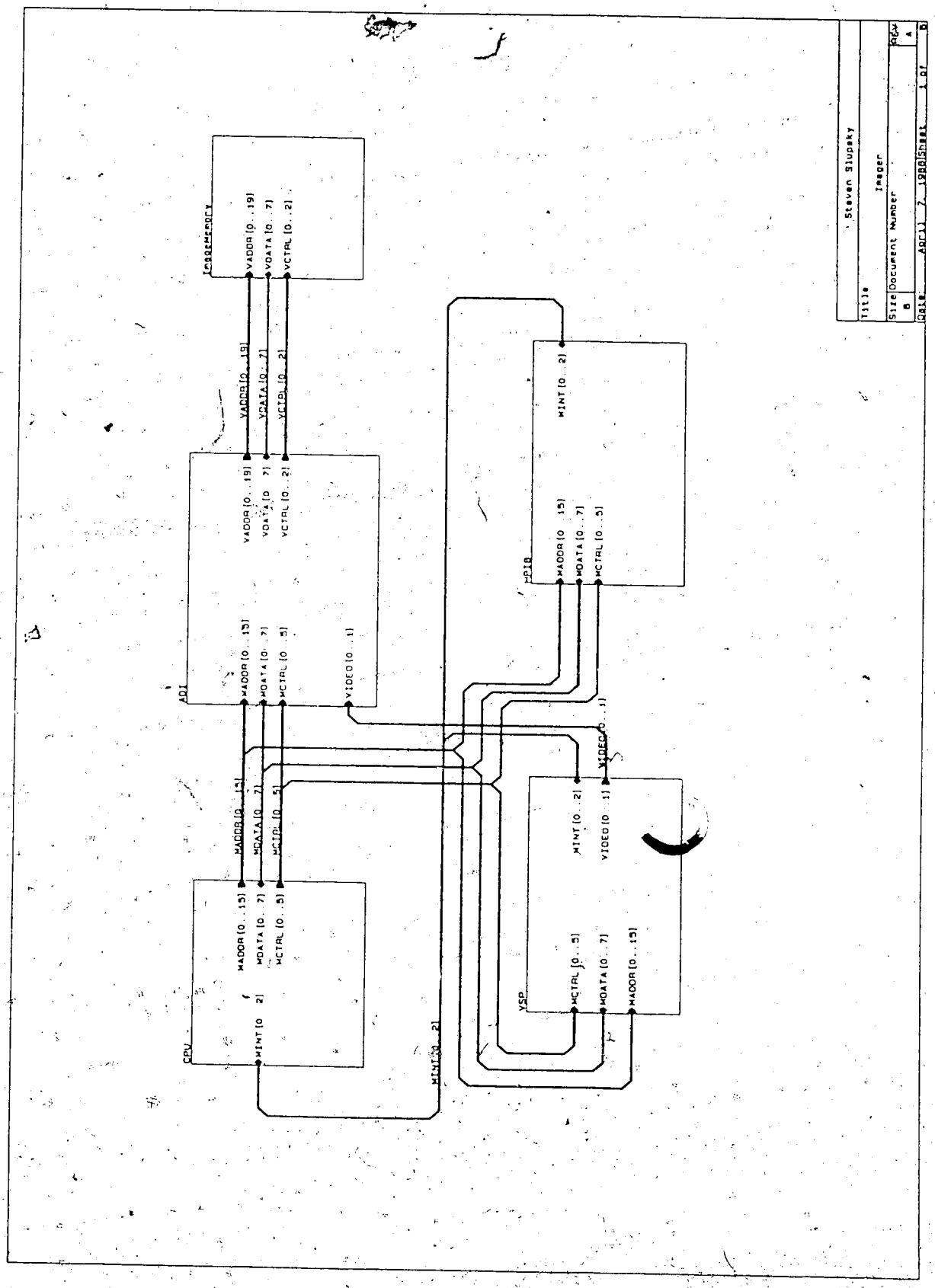


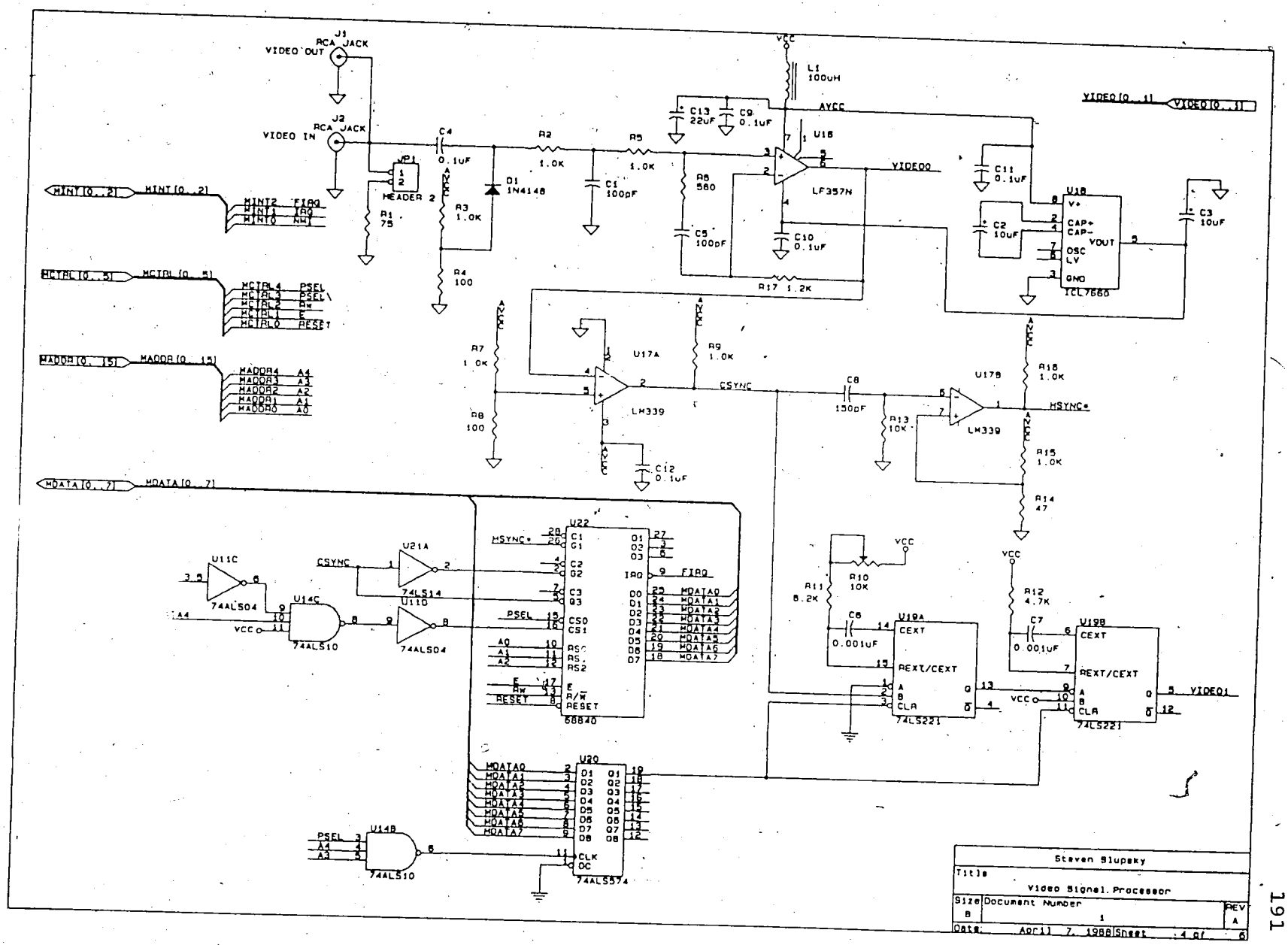

RSTCTRL
RESET
SYNCRST
SREN
CLK
E
Z

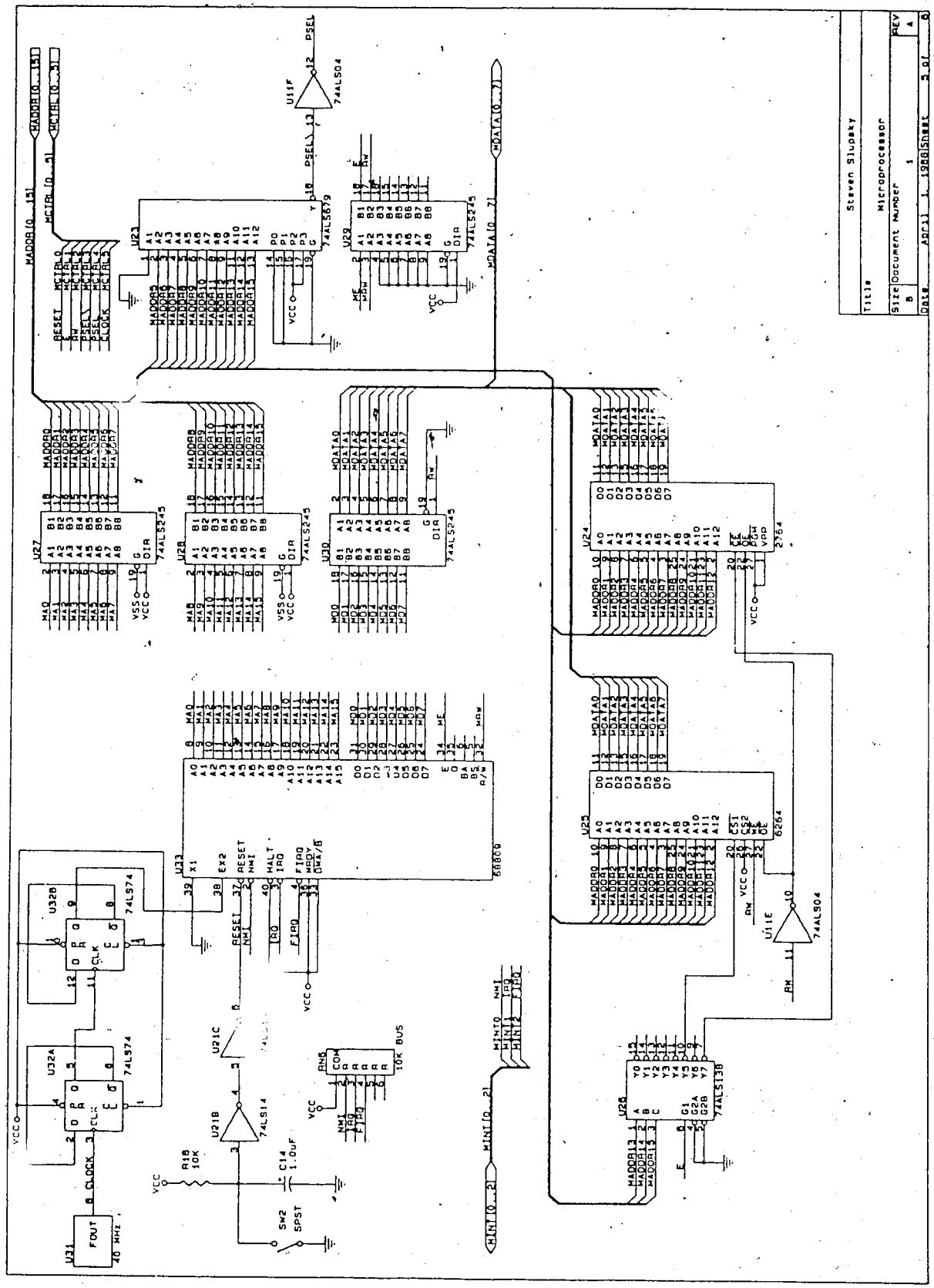

Z
U7
BUT
(I_PAD)
U6
2IVA
U8
FD1
U2
ND2
U1
FD2
U0
FD2
U5
FD1
U4
ND3
U3
IV
U9
2IVA
RESET
E
SYNCRST
SREN
CLK

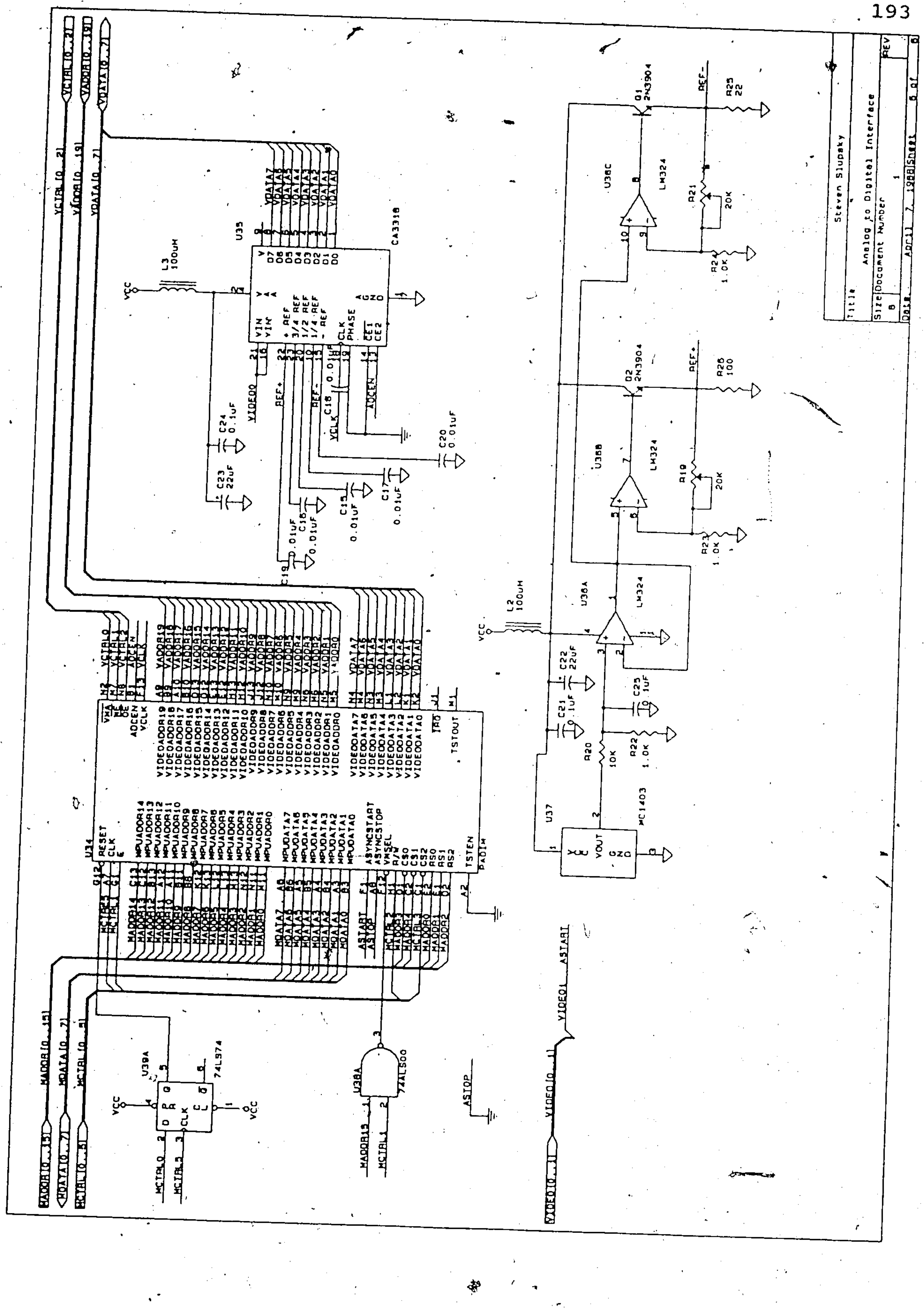
Appendix F

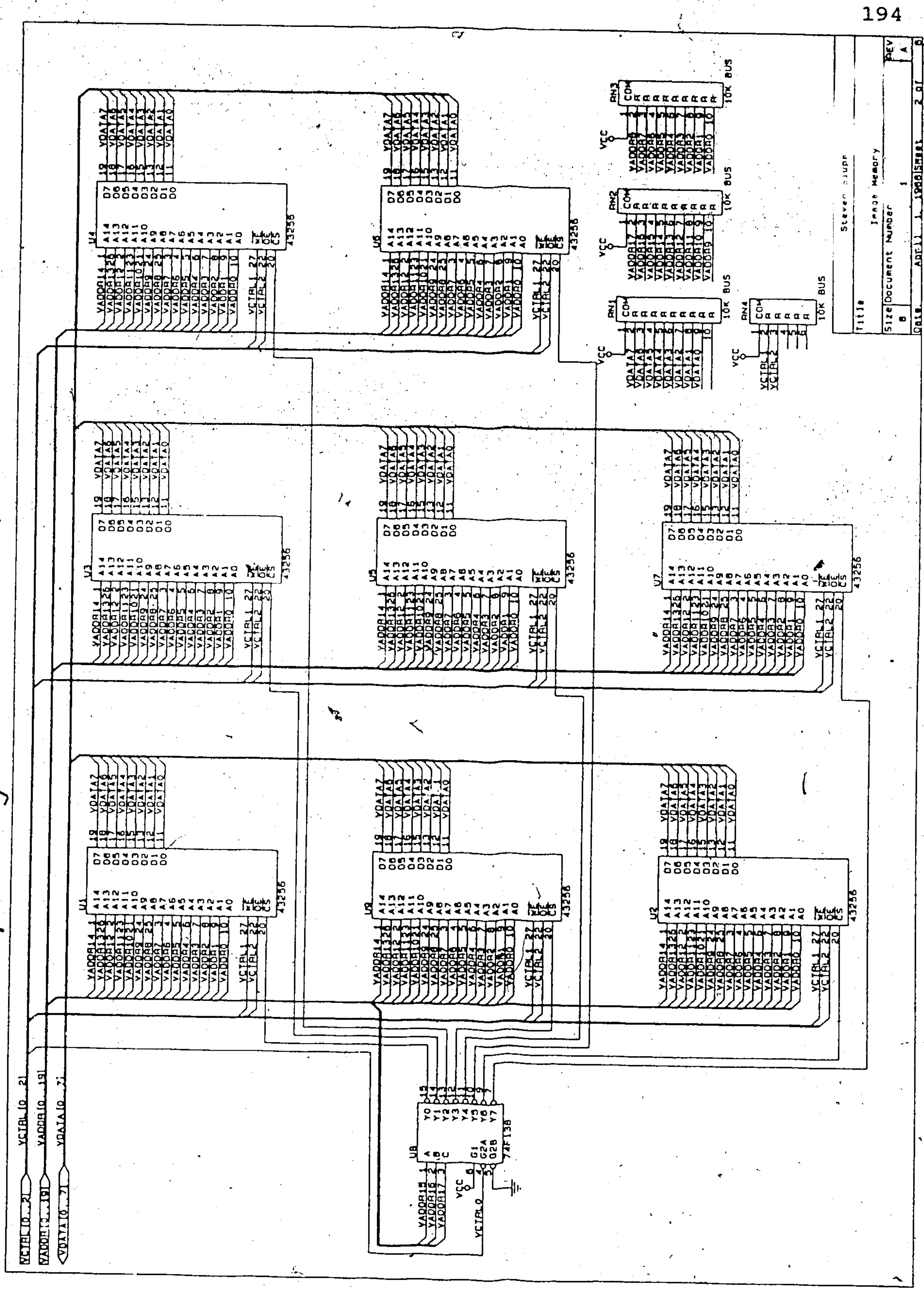
Hardware Schematic Diagrams

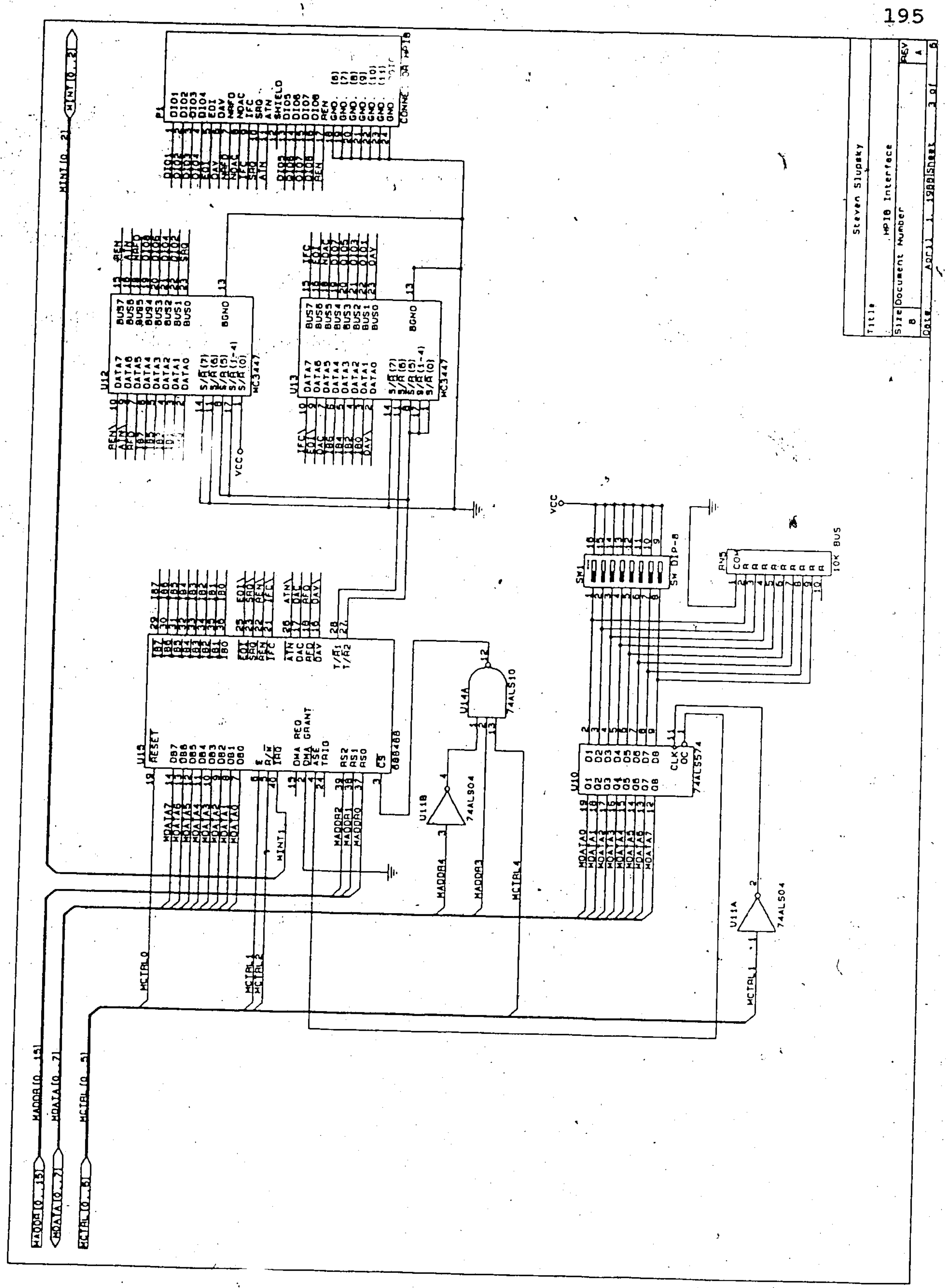

Steven Slupsky
Title
Imager
Size
B
Document Number
REV
A
Date
April 7, 1988
Sheet
1 of
IMAGEMEMORY
VADDR[0..19]
VDATA[0..7]
VCTRL[0..2]
AOI
MADDR[0..15]
MDATA[0..7]
MCTRL[0..5]
VIDEO[0..1]
MINT[0..2]
MPIB
CPU
VSP


VIDEO OUT
VIDEO IN
RCA JACK
HEADER 2
LF357N
ICL7660
LM339
74ALS04
74ALS10
74ALS14
74ALS221
74ALS574
68840
VCC
AVCC
CSYNC
HSYNC*
PSEL
FIRQ
RESET
MINT[0..2]
MCTRL[0..5]
MADDR[0..15]
MDATA[0..7]
VIDEO[0..1]
Steven Slupsky
Video Signal Processor
Size B
Document Number 1
REV A
Date: April 7, 1988 Sheet 4 of 6


Steven Slupsky
Title
Microprocessor
Size
B
Document Number
1
REV
A
Date
April 1, 1988
Sheet
5 of
MADDR[0..15]
MCTRL[0..5]
MDATA[0..7]
MINT[0..2]
U11F
74ALS04
U23
74ALS679
U29
74ALS245
U27
74ALS245
U28
74ALS245
U30
74ALS245
U24
2764
U25
6264
U33
68B09
U32B
74LS74
U32A
74LS74
U31
FOUT
16 MHz
U21C
U21B
74LS14
U11E
74ALS04
U26
74ALS138
RN5
10K BUS
R18
10K
C14
1.0uF
SW2
SPST
VCC


Steven Slupsky
Title Analog to Digital Interface
Size Document Number
B
Date April 7, 1988 Sheet
REV
CA3318
LM324
MC1403
74LS74
74ALS00
2N3904




Steven Slupsky
Title
HPIB Interface
Size
B
Document Number
REV
A
Date
April 1, 1988
Sheet
3 of 5
MC3447
74ALS10
74ALS04
74ALS574
SW DIP-8
10K BUS
MADDR[0..15]
MDATA[0..7]
MCTRL[0..5]
MINT[0..2]