To the Graduate Council:

I am submitting herewith a thesis written by Robert M. Bass entitled "Development of a Microcomputer Based Video Analysis System." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Electrical Engineering.

D. W. Bouldin
Major Professor

We have read this thesis
and recommend its acceptance:

Robert E. Bodenheimer

Daniel B. Koch

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Robert M. Bass

Date 4-20-92

Design and Implementation of a Microcomputer-Based Video Analysis System

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Robert M. Bass

May 1992

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr Donald W. Bouldin, for his guidance and patience. I would also like to thank the other committee members, Dr Robert E. Bodenheimer and Daniel B. Koch, for their comments and educational assistance. I would like to express my thanks to the members of my family, co-workers and friends for their patience and support: Robert and Evelyn Bass in Knoxville.

ABSTRACT

The design and implementation of a microcomputer-based video analysis system is described in this thesis. Both hardware and software development were required to produce a system capable of testing various types of video cards. The hardware design included not only a microprocessor but also programmable digital logic and analog circuitry. The software design included the use of CUPL to develop the programmable logic, MCS BASIC-52 for internal control of the microcomputer, and QuickBASIC programs for the BASIC-52 development environment. QuickBASIC was also used as trial development software for the test station. The test system objective is to replace all forms of operator-assisted video testing currently performed on the production floor. This video testing technique results in improved product quality and higher test coverage.

TABLE OF CONTENTS

SECTION	PAGE
1. INTRODUCTION	1
Current Philosophy of Testing	1
Design Goals	2
Hardware Communication Requirements	4
2. HARDWARE SUBSYSTEM REQUIREMENTS	7
Microprocessor Control and Interface	7
Discrete Address Decode and Analog DAC Board	13
Analysis System Clock Generator	14
Dot Clock Synchronization	16
Analog Video Signal Detection Circuitry	19
Digital Video Routing and Pipeline	22
Vertical Sync Detector	24
Signature Start/Stop Control Logic	27
Signature Analysis/Pixel Counters	30
3. SOFTWARE DEVELOPMENT SUPPORT	34
Implementation of Pseudo-Structured Rules for BASIC-52 Programming	34
BASIC-52 Software Development Environment	37
4. BASIC-52 VIDEO ANALYSIS PROGRAM STRUCTURE	42
Non-Intelligent Terminal Protocol Definition and Interpreter	42
Test and Subtest Sequencer	45
Test Program Structure	50
Error Handling & Reporting	51
System Programmer Model	53
Analysis System BASIC-52 Framework	56
Video Attributes Hardware Interface Utilities	59
Hardware Self-test Support Software	61
Video Frame Acquisition Software	63
5. CONCLUSIONS AND FUTURE DEVELOPMENT	65
LIST OF REFERENCES	71
APPENDIX	73
Analysis Board Schematic Diagrams	74
Decoder Board Schematic Diagrams	83
BASIC-52 CPU Board Schematic Diagrams	90
Analysis System Backplane	95
BASIC-52 Video Analysis Program Listing	97
BASIC-52 Software Development Environment Listing	126
CUPL PAL Listings	141
VITA	187

LIST OF FIGURES

Figure 1 Communication Paths 6

Figure 2 Video Analysis System Block Diagram 8

Figure 3 Frequency Doubling Function 16

Figure 4 Video Dot Clock Sampling and Meta-stability . 18

Figure 5 Horizonal and Vertical Sync Waveforms 26

Figure 6 Frame Acquire State Diagram 29

Figure 7 NITP Communication Packet 44

Figure 8 Diagnostic and Control Commands 46

Figure 9 Video Test Commands 47

Figure 10 Test Processing Flowchart 49

LIST OF TABLES

Table 1 MCS BASIC-52 Defined Memory Map 11

Table 2 Video Format Characteristics 19

Table 3 Documented and Extracted BASIC-52 Program. . . 39

Table 4 Video Analysis System Memory Map 54

1. INTRODUCTION

Current Philosophy of Testing

The current test philosophy at Siemens Industrial Automation Inc (formerly Texas Instruments) is to provide automated testing with a minimum amount of test operator intervention. The goal is to prevent unnecessary use of technical personnel to operate production test equipment.

Small scale product production may requires assistance of a test operator to provide mechanical handling and a limited amount of human sensory feedback. This assistance may include audible and visual confirmation of the test results or electro-mechanical component adjustments. Simple sensory testing can be accurately performed by test operators if the test controller integrates pseudo-random variations into the test sequence and output patterns. These variations avoid repetition, which decreases the effectiveness and quality of the operator's assistance. The effectiveness of a test operator becomes questionable when multi-variable decisions must be made, instead of simple true/false decisions. Multi-variable decisions occur when operator's decisions include quality, quantity or dimensions.

Large scale production can justify the cost of higher levels of automation to replace human sensory tasks. An alternate cost justification for low production volumes must be based on weighing product complexity against product quality goals.

Design Goals

The design goal of this project was to provide a low cost video signal analysis system that provides true/false decisions to complex video images. The fundamental idea underlying this design was to accumulate cyclic redundancy check (CRC) signatures of video signal components during a single video frame. Video patterns generated by the unit under test (UUT) must be designed to detect and isolate potential video problems. In addition, a method to quantify test results was made available. This method reveals a level of failure severity from which the analysis system may isolate and diagnose the fault source. Hence, a correlation between fault classification and test signature failure can be established.

Previous production video signal testing required a video monitor and a series of stimulus pattern screens. The operator was asked to make decisions on the following video attribute categories.

1. Color spectrum and color purity.
2. Color intensity ranges.
3. Dot-to-dot and line-to-line alignment.
4. Dot addressability of digital memory image area.
5. Consistent dot-to-dot electronic timing to prevent dot sparkling or screen flickering.
6. Generation of proper fonts and character sets.
7. Proper screen paging.

Tests based on operator vision required a significant level of test time. Where test time can be sub-divided into computer pattern generation time, operator pattern observation time and operator result reporting time. All video attributes are subject to interpretation based on the operator's experience, understanding of judgment criteria and attentiveness.

The other video testing alternative considered was to purchase an over-the-counter video acquisition board for a PC-AT computer. The major design obstacle of a video analysis system is meeting the various video format requirements needed to capture information about a video signal. Most consumer video acquisition boards host only camera video output formats and do not meet computer video monitor electrical specifications. An additional problem preventing modifications of video boards is that synchronization is controlled by the board not the camera. Therefore the board does not function as a slave similar to a video monitor. Specifications mismatch since the vision boards do not operate at computer generated video frequencies or formats, associated with Video Graphic Adapters (VGA), Color Graphic Adapters (CGA) or Hercules Monochrome, etc. This is not intended to degrade the valuable nature of a PC-compatible as a good test platform, but its use in meeting the multiple video specifications of the video analysis system has not been promising.

Hardware Communication Requirements

The hardware design goal was to design a stand-alone video analysis instrument that could be controlled by several different test hosts. This design must provide video signal analysis circuitry interfaced to the test host using a high level communications protocol. A microcontroller was chosen to provide intelligent isolation between the test host and the video analysis circuitry, thereby minimizing direct host-to-hardware control.

A RS-232 serial communication channel provides an easily maintainable command interface between the test host and the BASIC-52. The serial line program command information exchange must conform to a subset of the *Non-Intelligent Terminal Protocol* (NITP). This information exchange protocol is currently being used by the test station for communication to the unit under test (UUT). Implementation of this protocol allows the test station to communicate using existing software drivers and minimal software overhead.

The second form of BASIC-52 communication protocol required is the ASCII dumb terminal communications, which is used to transfer or modify the analysis system test program. A special use of this ASCII communication would allow the test host to reprogram the BASIC-52 test program for special video tests or exhaustive diagnostic tests.

The typical video UUT will generally be a sub-component of a Programmable Logic Controller system (PLC). This type of

controller system may be manufactured in several different configurations from a single board to a large multiple-board VME PLC system. The multiple-board PLC is the initial UUT target setting where the video annex UUT can be located on any slot in the system card cage. Communication paths to the video annex UUT will consist of both RS-232 serial line and VME word serial communication. All test communication paths will distribute *Non-intelligent terminal protocol* information packets as a standard communication philosophy.

Internal data bus communication between the BASIC-52 analysis CPU and the analysis video hardware consist of byte-serial transfers. Information on this data bus will program the video hardware configurations, trigger measurement events, retrieve video analysis results, and perform diagnostic testing.

The test host will use the NITP communications to send commands and to interpret responses from both the UUT and the video analysis system. The communication paths for the test system are shown in Figure 1 with the following communication cycle description.

The communication paths and procedures that will be followed during a video test acquisition cycle.

1. Test host to PLC UUT, a RS-232 transmission of NITP video pattern command packet.
2. PLC UUT to slave UUT, a VME transfer of NITP video pattern command packet.

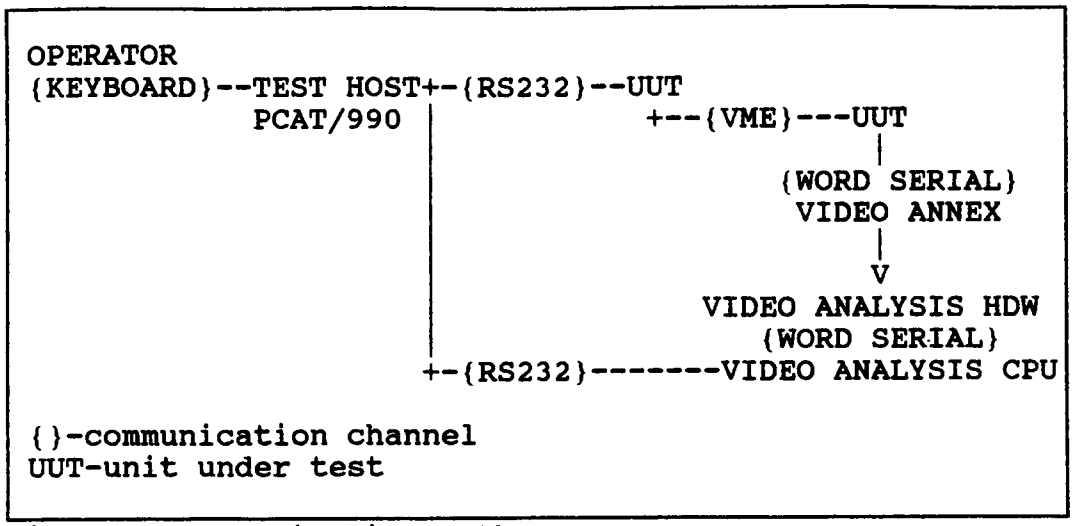


Figure 1 Communication Paths

3. Slave UUT returns a previous NITP response packet then begins execution of video pattern command.

4. PLC UUT to Test host, a RS-232 transmission of NITP previous response packet.

5. Test host to video analysis system, a RS-232 transmission of NITP signature acquisition command packet system.

6. Video analysis system returns a previous response packet then begins execution of a frame acquisition cycle.

7. Test host to Video analysis system, a RS-232 polling transmission for response packets.

8. Video analysis system completes acquisition command then waits for response poll before returning the signature response packet.

2. HARDWARE SUBSYSTEM REQUIREMENTS

The block diagram shown in Figure 2 subdivides the video analysis system into functional blocks, each block will be discussed with a brief description of its function.

Microprocessor Control and Interface

The goal in developing the test BASIC-52 CPU was to provide a low cost stand-alone test control system. The test system was targeted primarily for high production volume testing with a small product mix. The complete test system originally consisted of the BASIC-52 CPU card, analog input/output card, discrete input/output and a project specific interface. The test system uses a custom-wired backplane and small footprint printed circuit boards to create a versatile test system. The modularity of the test system results in a secondary benefit of being recyclable when products end.

Due to the specific input/output requirements of the video analysis system, the original analog and discrete cards were not used. A new interface card was designed to incorporate both analog and discrete functions onto the same card. The interface card reduced the backplane board count and improved performance characteristics of the analysis board. Specific benefits of this design are discussed in the board sections of this document. The resulting video analysis system is composed of a BASIC-52 CPU card, an analog/discrete interface card (Decode), a video analysis card (Analysis) and

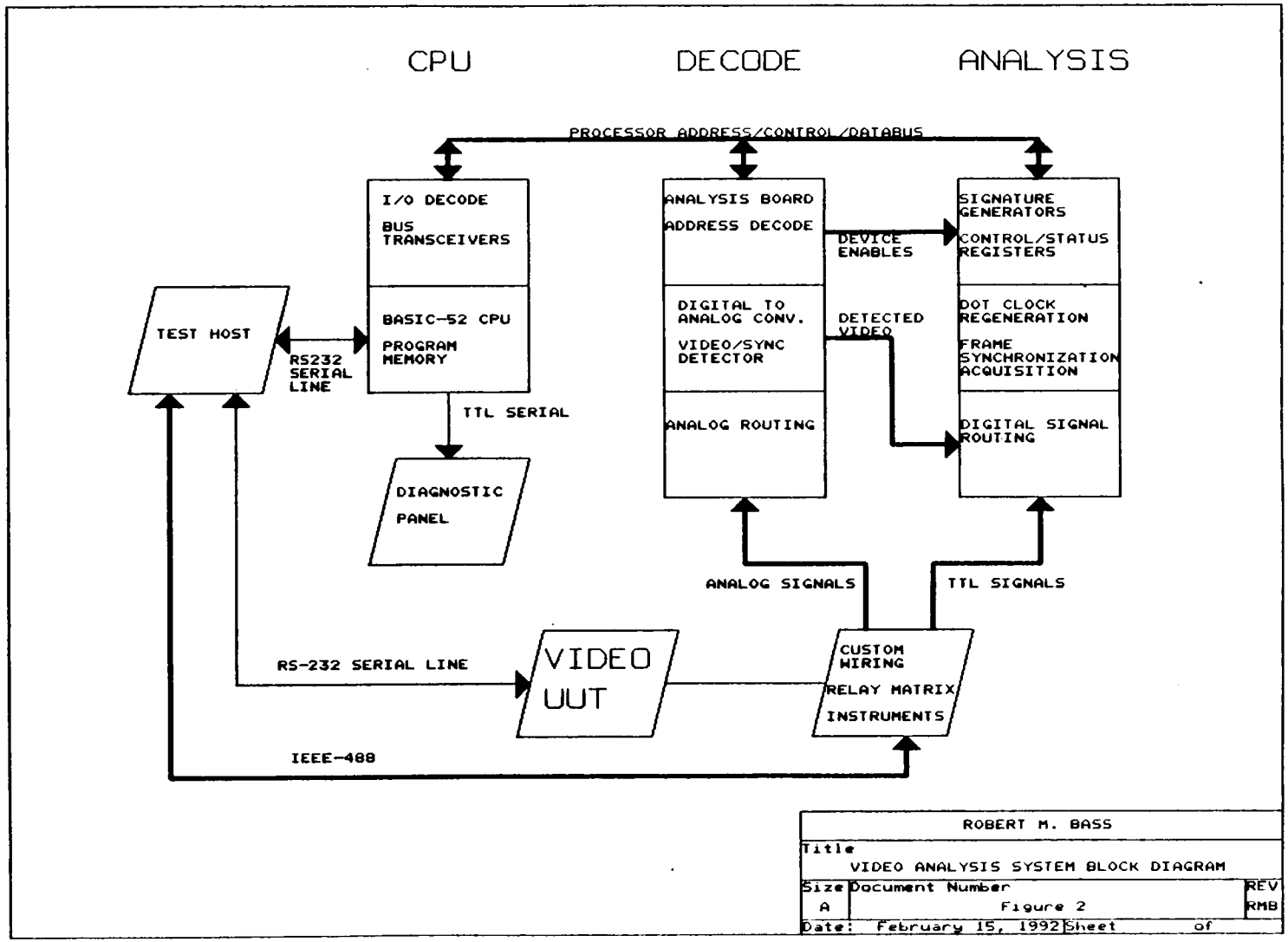


Figure 2 Video Analysis System Block Diagram

a custom backplane. The electrical schematics for each card are in the appendix. In addition the complete program listing and output simulations for all PAL's used in the video analysis system appear in the appendix.

The CPU card was implemented using Intel's 8052 microcontroller with an internal BASIC interpreter, Intel's MCS BASIC-52. This microcontroller was selected for the imbedded functionality that minimized the hardware and software required for communications and language development. The microcontroller features memory addressing of 128 k bytes, three built-in interval timers, dual serial communication ports and a BASIC language interpreter. The internal BASIC interpreter provides BASIC language control of the imbedded serial terminal port, the serial printer port, the baud rate generators, the real-time clock and software interrupts.

The memory addressing of Intel's 8052 is not organized as the typical Von-Neumann memory architecture. This processor's memory architecture is partitioned into distinct regions labeled as data and program/code areas. [1] This implies that data and program memory are mapped into separate physical address spaces. This memory concept is very important in understanding the system memory maps. The code area defines the region where machine language instructions are executed. This code area is typically defined in hardware as ROM. The INTEL's 8052 hardware restricts machine language execution to the code area only. Thus, the data area is only accessible by

8052 specific data access instructions. Definitions of both regions are important in understanding the overlap and duality of the memory maps in Table 1.

Special considerations and restrictions on hardware development for the BASIC-52 language are revealed in the MCS BASIC-52 user manual. Memory architecture requires that the internal BASIC interpreter ROM must reside in program/code area. Since the BASIC language program is an interpreted language, the program must be stored in the data area. BASIC-52 RAM stack area and system pointers must reside in the data areas. Any memory map definition beyond these basic requirements is up to the system hardware designer. Memory map design options are listed in Table 1.

The first design option shows dual mapping of external user EPROM into data and code areas. This mapping allows the EPROM to store both BASIC language programs (data) and assembly language programs (code). This design allows both interpreted basic execution and user assembly language execution.

A second design option maps the EPROM and RAM into the data area in consecutive blocks. This allows the BASIC language interpreter to process tokenized BASIC from either RAM or ROM. This mapping allows BASIC language program development in RAM. Later the completed program developed in RAM can be transfer into EPROM for stand-alone execution. The inverse transfer from EPROM to RAM is also supported by the

Table 1 MCS BASIC-52 Defined Memory Map

BASIC-52 DEFINED MEMORY MAP

ADDRESS	PROGRAM AREA	DATA AREA
0000-1FFF	INTERNAL ROM INTERPRETER	EXTERNAL RAM
2000-7FFF	EXTERNAL ROM	EXTERNAL RAM
8000-DFFF	EXTERNAL ROM	EXTERNAL ROM/RAM
E000-FFFF	USER IO/ROM	USER IO/ROM

ACTUAL BASIC-52 CPU MEMORY MAP

ADDRESS	PROGRAM AREA	DATA AREA
0000-1FFF	INTERNAL ROM INTERPRETER	EXTERNAL RAM 8K
2000-5FFF	EMPTY	EXTERNAL RAM 16K
6000-7FFF	EMPTY	EXTERNAL RAM 8K
8000-BFFF	EXTERNAL ROM 16K	EXTERNAL ROM 16K DUAL MAP
C000-DFFF	EXTERNAL ROM 0K	EXTERNAL ROM 0K DUAL MAP
E00X-E1FX	EMPTY	INTERVAL TIMER
E200-E3FF	EMPTY	ANALOG CARDS
E400-E5FF	EMPTY	DISCRETE CARDS
E600-E7FF	EMPTY	EMPTY
E800-EFFF	S.F.MEMORY AREA	S.F. MEMORY AREA
F000-FFFF	TOPPAGE	TOPPAGE MEMORY

BASIC language. This allows the software programmer to make experimental modifications to a copy of the stored program without affecting the EPROM version.

The final CPU design has 32 K bytes of static RAM implemented using a 43256-10 static RAM (U8).[2] The EPROM occupies 16 K bytes using a 27128 EPROM (U10) and an EPROM programming port connector.

The CPU card provides a three-channel interval timer implemented with an Intel 8254 (U12). The interval timers can be used as programmable clock generators for time measurement testing. The interval timers have all clock, gate and output control lines routed externally to the backplane interface.

External clock reference frequencies are generated by a 16 Mhz oscillator and TTL clock dividers (U14). These reference frequencies are available on the user backplane interface. The backplane can be wired to connect the clock reference frequencies to the interval timers to perform the programmable clock generators and frequency measurement functions.

Since the BASIC-52 supports pulse width modulation (PWM), a buffered output signal was routed to the backplane interface. The PWM outputs primary use is for audible tone generation to an external speaker.

A design decision was made early in the CPU development to provide EPROM connections for external EPROM programming voltage sources. The BASIC-52 CPU board is still responsible

for generating the programming control signals and timing. This design allows the test designer various programming and hardware options, such as various EPROMs, flash EPROMs or Non-Volatile RAM. This also allows the test engineer to select EPROMs based on availability instead of fixed programming voltage parameters.

Discrete Address Decode and Analog DAC Board

The second board is occupied by address decode control logic and analog DAC components. Design criteria for this card was that excess logic that did not receive any performance benefit from being located on the high speed analysis card would reside on this board. The intent of preserving the high speed video analysis card allows for selective component placement and small wire lengths, thereby reducing clock and sampling delays.

The first decode function located on this board was address decoding logic for the hardware control ports. The analysis board hardware interface has one input port, four output ports and six data ports. All read/write control ports and data bus lines physically reside on the high speed analysis board. The location of the ports on the analysis board prevented wire length capacitances from slowing down the high speed video acquisition circuitry. The remote location of the address decoder was a tradeoff that places wire length capacitance on the slower non-critical chip enable lines. The address decode logic maps the analysis hardware into the

BASIC-52 test system address space previously reserved for the eight discrete control cards. This broad decoding map allows for future hardware expansion without memory map changes.

The second functional block to occupy this board is the quad-bipolar digital-to-analog converters (DAC) circuits. This circuitry was developed using an 8-bit quad-current output DAC that is converted to voltage by a closed-loop operational amplifier. A second operational amplifier performs a subtractive level-shifting transformation to a bipolar output voltage limit of 1.25 volts. The 2.5-volt DAC reference and the level-shifter's reference were developed from a precision 5.0-volt reference. The voltage reference was skewed to 2.5v by a virtual ground offset driven by an operation amplifier.

The last hardware block is the analog video relay matrix, which provides analog signal routing. The relay matrix was considered the best method to route signals and still preserve the signal timing and voltage characteristics. The off-board wiring length penalty creates very little effect on the analog video signals as compared to the video feed cable length. The analog window detection circuitry is also located on the decode board and will be discussed later in the video detection section.

Analysis System Clock Generator

The analysis system clock generator circuitry was based on video frequency requirements created by the initial target UUT called the CRT-annex board of the VME PLC system. The CRT

annex UUT uses a 28.64 Mhz system oscillator that generates the local video dot clock of 14.32 Mhz. This dot clock establishes the two video bandwidth standards required for the Color Graphic Adapter (CGA), and Hercules' monochrome display. Yet the UUT Hercules' video mode does not replicate the true bandwidth specs of 16.257 Mhz as published by IBM. This design simplification in this UUT creates a side effect of an improperly sized screen that can be easily adjusted on most monitors. The analysis board design uses a single oscillator on the same frequency as the CRT annex UUT. Therefore the video analysis unit is not affected by this compromised UUT oscillator design. This allows testing of Color Graphic Adapters, Enhanced Graphic Adapters, Hercules Graphic Adapter, TI Color and TI Monochrome. A later video analysis board modification would allow electronic switching between two oscillators to meet specifications of IBM Hercules Monochrome and Video Graphic Adapter (VGA).

In any video test system, precise sampling of video pixels will be necessary to capture a repeatable signature. Of the video standards being tested, none provide an external reference clock. Therefore the purpose of the clock generation circuitry is to provide a four-times over-sample (fast) clock to generate a replica of the UUT dot clock. The dot clock is used as a sample clock to drive the video pixel sampling pipeline and signature analyzers. Since frequency multiples of the oscillator timebase of 28.64 Mhz are not readily

available, the alternative of frequency doubling was used. This circuitry was developed using a modulo-2 summer (Exclusive-Or) and a one-quarter clock period time delay element diagrammed in Figure 3.

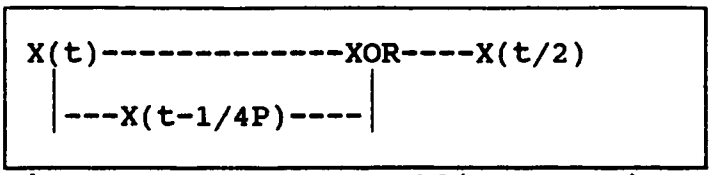


Figure 3 Frequency Doubling Function

In the prototype version the delay element was implemented with a 74ALS14 Schmitt trigger inverter chain. A production version of this circuit would use an accurate delay line to implement the delay. This inverter method allowed access to a series of propagation delay taps to yield a delay of 1/4 clock period of the active oscillator. The resulting 74ALS86 modulo-2 summation of the oscillator clock and the delayed oscillator clock will yield twice the fixed oscillator frequency. The duty cycle can be adjusted by selecting different combinations of delay taps. So, the 57.28 Mhz generated meets the system requirement of four-times the system clock.

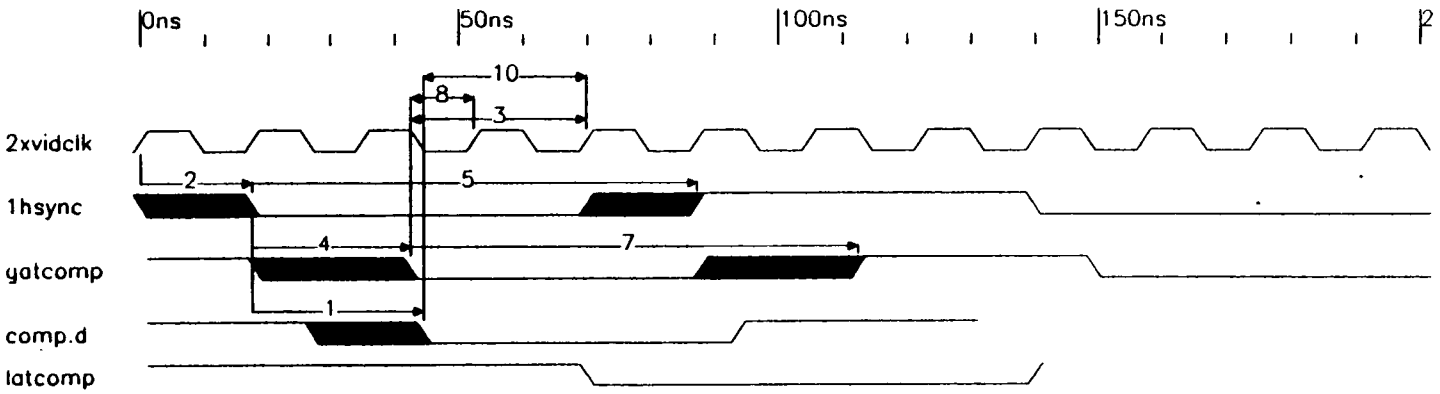
Dot Clock Synchronization

Proper dot clock synchronization was based on the four-times clock and an edge transition of the horizonal sync line. The fast clocked synchronization circuitry aligns the replica dot clock with the pixel midpoint. This mid-pixel alignment prevents asynchronous sampling meta-stability, resulting in

unstable signatures. Timing waveforms in Figure 4 illustrates clock sampling and meta-stability. The horizonal sync was selected as the synchronization signal due to its periodic occurrence rate of once per video line, as compared to the vertical sync rate of once per video frame.

Since synchronization reoccurs every video line, the phase relationships between the UUT and the analysis board oscillators must achieve only short intervals of stationarity. Therefore, absolute accuracy of the fast clock is not critical. Accuracy differences must not exceed one clock period per scan line or 640 active pixels. One clock period per scan results in a worst case allowable error of 17.46 ns over a period of 44.69 us, the relationship yields a mean accuracy of 0.039% between oscillators.

The fast clock rate encountered in the synchronization circuit places tight propagation delay specifications on the registered PAL state machine. The synchronization PAL circuit allows for programmable selection of video clock delays to control the transition point of the video pixel clock (dotclk). This selectable delay allows for compensation of video sampling errors due to propagation delays encountered through signal routing paths. The synchronization circuit speed requirements were met using the Lattice GAL16V8-15 (U4) [3] high speed PAL. This device provides features of 15 ns propagation delay and electrically reprogrammable logic.



Parameter – vidsync.td						
Id	Name	Expression	Min	Max	Margin	Comment
D1	xor _22v10:	27	27	27		74als86 _22v10 input to reg setup
D2	clk skew	[0,17.50]	0	17.50		asynchronous clock misalignment
S3	late hsync	setup16r8	13		14.33	late arrival will miss edge prior to this
D4	22v10 logdl	[18,25]	18	25		
D5		pixel	69.84	69.84		
V6	pixel	69.84	69.84	69.84		
D7		pixel	69.84	69.84		
S8	early hsync	setup16r8	13		-3.13	early arrival will missed 1st clk edge
V9	setup16r8	[13,]	13			
S10		[,]				

Figure 4 Video Dot Clock Sampling and Meta-stability

Analog Video Signal Detection Circuitry

The CRT annex UUT will generate several different analog video formats such as Enhanced Graphic Adapter (EGA), Professional Graphic Adapter (PGA), TI Color, Hercules and TI Monochrome. Each analog format has particular voltage/timing characteristics that will affect the test parameters to be verified. Sample video format characteristics are shown in Table 2 .

Table 2 Video Format Characteristics

	IBM MONOCHROME	IBM COLOR	TI MONOCHROME	TI COLOR
VIDEO LO	0.0-0.6V	0.0-0.6V	0.0-0.6V	0.0-0.6V
VIDEO HI	2.4-3.5V	2.4-3.5V	2.4-3.5V	2.4-3.5V
SYNC COMP	N/A	N/A	0.0-0.2V	N/A
LOW COMP	N/A	N/A	0.5-0.9V	N/A
HIGH COMP	N/A	N/A	1.3-2.2V	N/A
FRAME RATE	49.82 hz	59.92 hz	50.00 hz	59.92 hz
LINE RATE	18.43 Khz	15.70 Khz	18.45 Khz	15.70 Khz
VIDEO BW	16.26 Khz	14.32 Khz	14.32 Khz	14.32 Khz
LINES/CHAR	14	8	13	9
V INTERVAL	868 us	1019 us	867 us	1019 us
H INTERVAL	8.30 us	5.59 us	7.82 us	5.59 us
PIXEL WIDTH	54 us	69 us	69 us	69 us

Analog video is sub-divided into voltage regions that contain a synchronization component, a color component and a

superimposed intensity component. Analog signals require a voltage window detector to isolate a particular video signal intensity level and to reject synchronization information. To operate as a video detector the upper and lower voltage window limits are independently programmed. This will allow a true/false decision to be made based on the observed region of the video signal. The window levels will be generated by an 8-bit digital-to-analog converter (DAC) configured to yield a bipolar output. The prototype version contains a quad DAC identified as a DAC8408 (U6) [5] that provides the video intensity resolution required to test the current production boards. Higher video resolution testing may require 12 to 14-bit DAC's to measure linearity and resolution.

The analog window detector consists of two high speed analog comparators NE522 (U12) [6] with independent programmable thresholds. The comparators must be high speed differential comparators similar to those used in sense amp applications, and in disk drive read-head circuits. Since the nominal video pixel duration is only 69 ns, all logic propagation delays must be minimized to meet sampling setup margins. A high speed comparator must generate propagation delays less than 15 ns. Upper and lower window comparator outputs are configured into a logical wired-OR form, to avoid additional logic delays. Video levels outside the window, either high or low, result in a false or undetected condition.

The resulting wired-OR output (detected video) is processed by the analysis board TTL video routing PAL (U8). Delay skews between video and sync signals are within 8 ns for the analog and digital paths, with an average path propagation delay of 15 ns. This fast settling time and small signal skew reduces the chance of meta-stability at the sampling pipeline register. These mean and skew delay times include the following video signal routing combinations of analog video verses analog sync, analog video verses TTL sync, analog sync video verses analog sync, TTL video verses TTL sync and TTL sync video verses TTL sync.

The analog video signal to be tested is routed from the video connectors through a DIP relay matrix (K1-K4) to the window circuitry. The mechanical relays have advantages over electronic switching in this video application, due to low insertion loss, high off isolation, minor waveform distortion and small propagation delay. The video signals to be routed to the video window circuitry are *red video*, *green video*, *blue video*, *composite video* and *test reference level*. The *test reference level* provides an independent voltage level to self-test the analog window comparators, DAC's and precision voltage reference.

The sync analog window detection circuit (U11) separates *analog-composite-sync* from the video signal *sync-on-green*. The sync signals routed by the relay matrix are *sync-on-green video* and *test reference level*. The test reference level

supports sync window self-test as stated in the video section. *Sync-on-green video* is the only analog video mode composed entirely of analog signals for video and control signals. All other analog video modes are a hybrid mix of analog video and TTL control. *Sync-on-green video* is divided into regions where composite sync occupies the bottom 1/3 and the green video is the upper 2/3 of the signal. In detecting sync from the *sync-on-green* signal, only the lower 1/3 of the video requires observation; therefore, a single-ended window comparator simplified the circuit design. The comparator's output will be detected composite sync that is processed by the digital video routing PAL. The single-ended window detector requires only the upper analog threshold to be independently programmable. The programmable window level is generated by an 8-bit DAC configured as a bipolar DAC. Special conditions defined by *sync-on-green* specifications require positive or negative active sync levels to be supported by video monitors. To satisfy this requirement software controllable sync polarity inverters were implemented (U1a,b).

Digital Video Routing and Pipeline

Supporting different video formats creates a considerable signal management task of routing proper video signal to the analysis circuitry. An early development decision was to make the hardware respond to programmable selection of established video modes, EGA, CGA, VGA, etc. This reduced control requirements between software and hardware by forcing a rigid

definition for each video mode. Satisfying this definition required a set of three video mode lines (*VIDMODE2..0*) to control signal routing hardware (U6,U7).

Digital video signal routing is performed by two blocks of digital logic. The first block routes TTL video information through a TTL multiplexer (U2) under control of three programmable source select lines (*SRCSEL2..0*). Source selecting routes any video or sync signal component through the video signature channel.

Video signals next enter the second block of routing known as the pipeline/combiner/routing PAL (U8). All signals enter this PAL under a grouping of either video, sync, mode control or self-test control. The PAL's combiner function is to combine available sync signals to generate a fixed form of composite sync for the synchronization circuits. As an example, to get composite sync from separate sync video modes, the PAL must combine horizonal sync and vertical sync using an Exclusive-OR logic function. The dot clock synchronizer (U4) requirements for a fixed form of composite sync simplifies downstream hardware for all operating modes.

The second PAL function is signal routing, which allows for optimum routing of analog and TTL signals to minimize propagation delays. This internal routing allows the high speed clock synchronizer circuitry and pipeline register improved setup timing margins. Additional self-test routing is provided by the PAL to support analog window reference testing

and video hardware self-test. During the video self-test mode the PAL will generate a *self-test* signal that informs external circuitry to enter diagnostic operations.

The final section of this PAL is the pipeline register which holds discrete samples of video and sync signals on the rising edge of synchronized dot clock (*SDOTCLK*). The video signal pipeline function is to sample video to maximize the setup time available to the signature analyzer counters. Given this sample-and-hold method, the video information is stable for a dot clock cycle minus small PAL output and setup delays.

Vertical Sync Detector

The vertical sync interval portion of a video frame is the only uniquely detectable interval that establishes frame cycle timing. From the vertical interval a reliable timing point exists indicating the end of the current video frame. The vertical sync detector must detect vertical sync from within a composite sync signal containing both horizonal and vertical sync intervals. Therefore timing obtained from the vertical sync interval is used to generate the signature circuitry start/stop control signals. The vertical sync detector circuitry acknowledges when the vertical interval has started by activating signal *VRETTO*. This signal informs the frame acquire circuitry to video lock when the last frame is ending and the next frame is about to begin. This signal can be combined with horizonal sync to determine the actual start of the next frame, thus triggering acquisition of a video

frame signature. Vertical sync is always of significantly longer duration than horizonal sync thus can be digitally timed to detect the vertical interval. The vertical sync detector was developed from quad binary counters (U21-U24) implementing a high speed 16-bit interval timer. This circuit measures the vertical blanking interval by up counting until counter overflow. The overflow condition signifies that a true vertical sync interval started, with a secondary use of inhibiting further counting. Since the rejection of horizonal sync is based on the short interval duration. The occurrence of horizonal sync will start the timers while the end of horizonal sync will reload the timer prior to count overflow condition. The four 4-bit counters used are microprocessor programmable by octal latches (U18-U19), these latches allow for various vertical interval timeout settings for each video mode. This method of frame synchronization requires composite sync for all video formats. The previously discussed routing PAL satisfies the generation of composite sync for all video modes. This simplification of handling all video modes using the same sync detector and signature acquisition logic reduces the system hardware complexity.

To improve video monitors signal lock, several video modes defined a form of composite sync that superimposes horizonal sync pulses during the vertical interval, as shown in Figure 5 waveform set 2. [7] These concurrent sync signals provide continuous synchronization to video monitors to

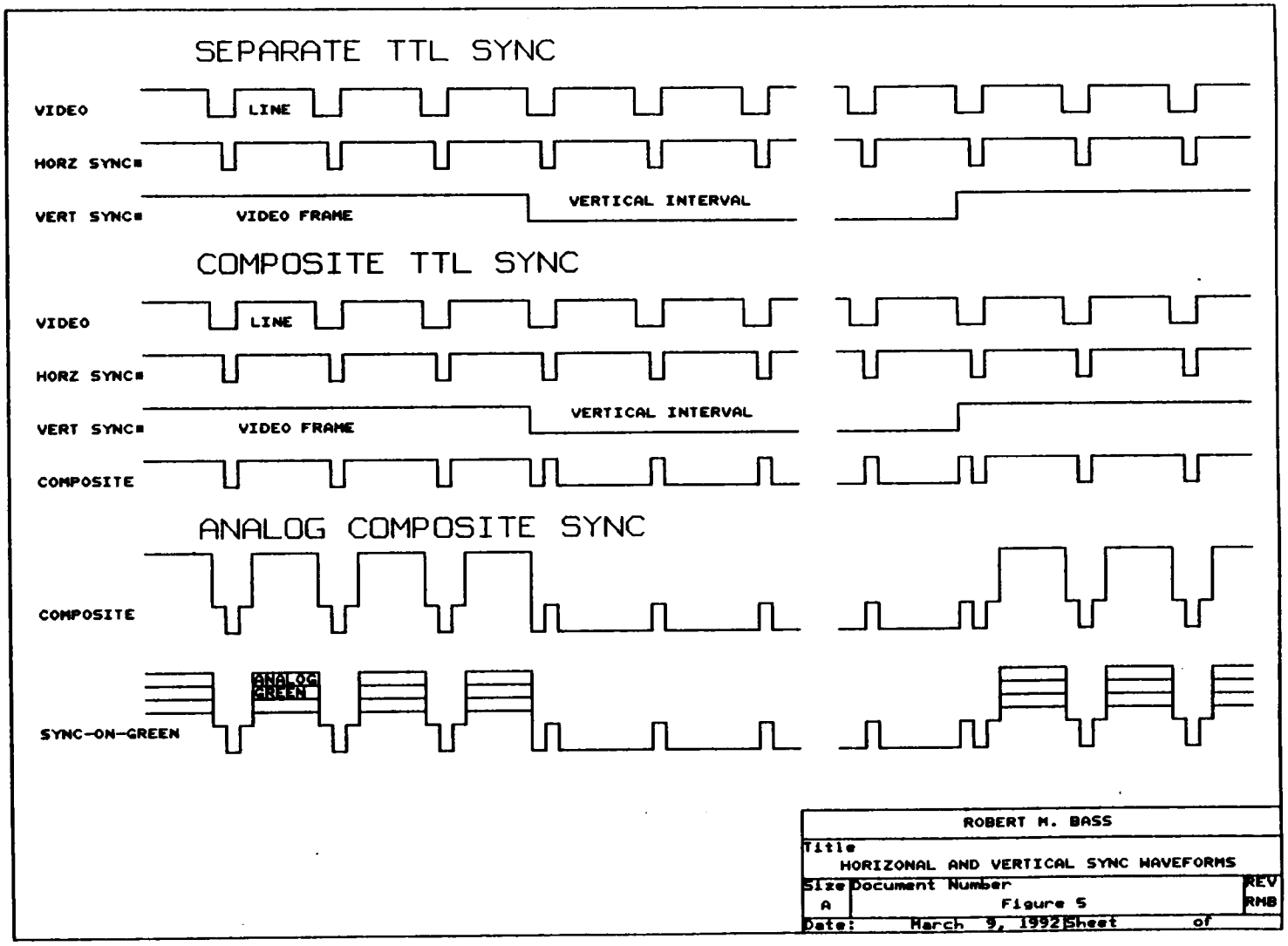


Figure 5 Horizonal and Vertical Sync Waveforms

prevent loss of signal lock of phase locked loops (PLL's). Notice that this concurrent composite sync requires inverted horizonal sync during the vertical interval. This polarity inversion does create a phase shift penalty, causing a momentary unlocking of phase lock loops (PLL) at the transitions between vertical and video intervals. Notice on waveform that this longer sync interval due to PLL unlocking may cause video monitor to tear the first scan line. In many video boards the first scan line is blanked to hide this monitor tearing problem. For the video analysis system the small oscillator drifts during this phase shift will not be large enough to create miss-sampling of pixels.

Signature Start/Stop Control Logic

In order for the video analysis board to acquire stable signatures, the control circuitry must provide a repeatable signature interval. The acquisition interval chosen for all signature sampling is the entire video frame including the vertical sync interval. The frame acquisition PAL starts signature acquisition on the first vertical sync of the video frame. Acquisition continues through the video frame including horizonal sync intervals, until signature acquisition on the start of the following vertical interval.

The acquisition control logic must be careful to determine when to lock the video frame to acquire signatures. The acquisition cycle starts when the microprocessor triggers the *acquire frame* signal but this event is asynchronous with

respect to the current video frame. If the frame is locked improperly this creates unpredictable frame length problems. These frame length errors are known as frame overrun or underrun and the results are bad or unstable signatures. Effects of this frame length misalignment such as a shortened vertical sync, may be misinterpreted as a horizonal sync. This problem was alleviated by starting lock-on-process during a prior video interval and requiring redundant video states in the control PAL to reject the possibility of misalignments. This forces the synchronization logic to find proper frame lock or skip a video frame, if necessary to satisfy the lock criteria. The current control PAL (U4) has used eleven of sixteen possible states for lock-on and video frame acquisition. The following Figure 6 shows the complete state diagram required to lock-on and acquire a complete video frame. Also the complete program listing and output simulations of this PAL can be found in the appendix.

The PAL makes all state change decisions based on three signal lines the microprocessor *Acquire Frame (ACQFRM)* trigger signal, composite sync signal to find transition edges and *vertical retrace timeout (VRETTO)* to separates vertical interval from horizonal interval. Interpretation of the state diagram defines State 0-6 as used for video interval lock-on process, ending this state sequence on occurrence of the vertical interval. State 8-11 accumulates signature from vertical interval through the video frame. Then state 7 ends

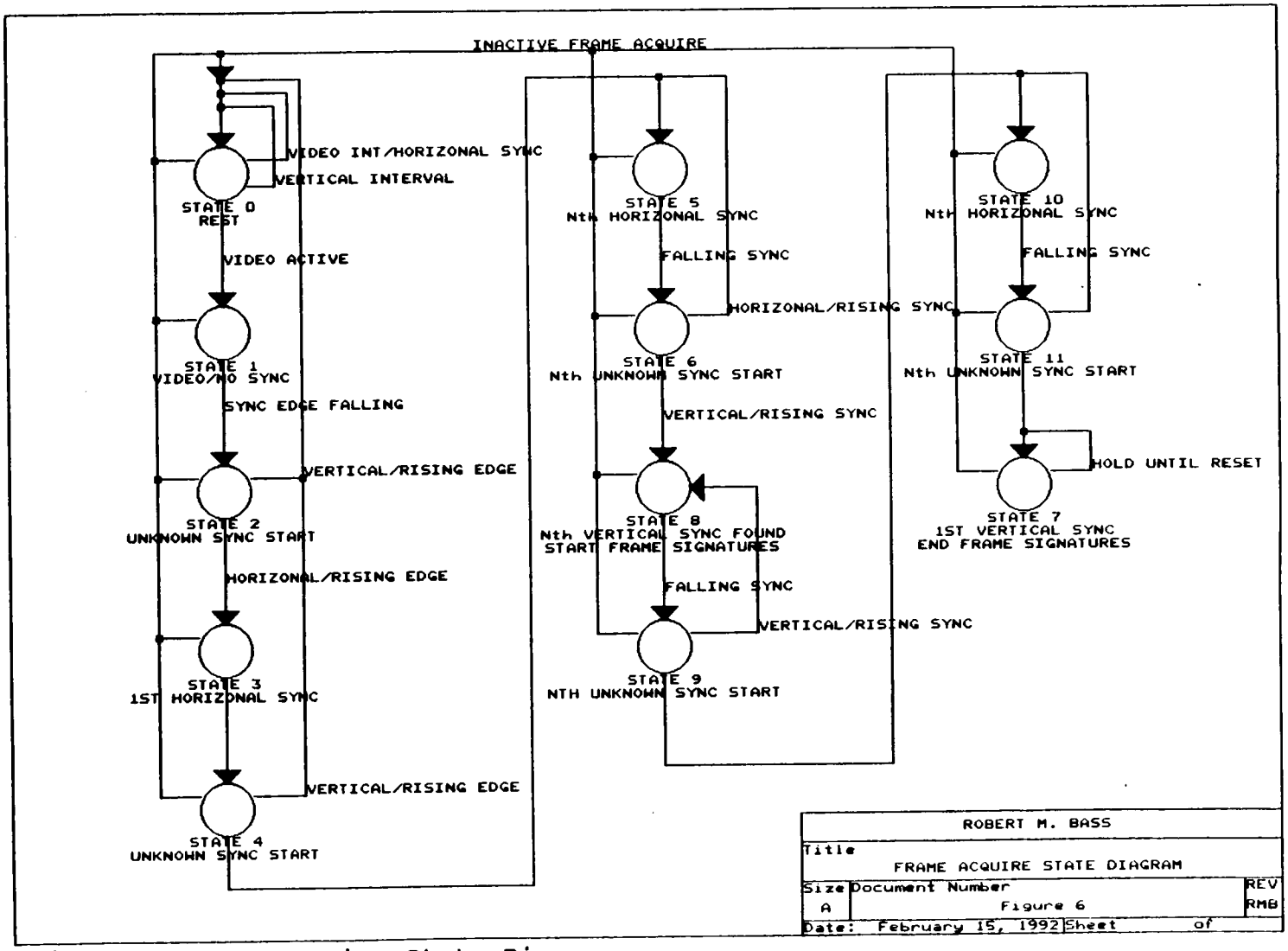


Figure 6 Frame Acquire State Diagram

the video frame on the next vertical interval, this in turn stops signature accumulation. This state machine lock-on sequence provides a very stable signature start/stop cycle. Special decoding of acquisition State 8-11 can be used to create acquisition cycles where horizonal and vertical intervals may be blocked from video signatures. This decoding may be required for video cards that do not guarantee the video pattern during either blanking interval.

Signature Analysis/Pixel Counters

The purpose of the signature analysis counters is to capture the video frame and composite sync signatures. The accumulated results available are Cyclic Redundancy Checks (CRC) signatures or pixel counts. The analysis circuitry (U12-U17) consisting of video signature hardware and composite sync signature hardware. This dual analysis circuitry was implemented to reduce the hardware overhead as a trade-off to more testing. The original design considered using three video channels to get a complete real-time video characterization. This design increased hardware redundancy by adding two video signature channels over the current two channels. Since the composite sync signature was added to qualify acquisition timing integrity, the additional hardware for three video channel signatures was no longer necessary. This tradeoff does require that the microprocessor software to perform individual frame acquisition cycles for each video signal to be characterized. Therefore a tradeoff occurred between

acquisition channels and number of test cycles performed, thus hardware was sacrificed for increased test time.

A single channel of the signature analysis circuitry contains a 24-bit multi-function register (U12-U14) providing signatures, pixel counts and self-test functions. The register is microprocessor memory map accessible in three byte locations to support reading of acquired results. The 24-bit register was developed using three cascaded stages of PALs. Synchronous clocking and internal look ahead carry logic were required to meet the speed requirements imposed by the dot clock. Several PAL technologies that achieve the required logic functionality and speed performance were considered. In addition combinations of programmable array logic (PAL) and Very Large Scale Integration (VLSI) devices were considered, but optimal timing margins were met using an all-PAL implementation. The final PAL selection of an AMD PALCE22V10-25 to implement all accumulator stages provided sufficient product terms, sum terms and output configurations for the design. The PAL's were designed to implement four different functional operations to meet current and future design requirements. Internal functionality as a binary up/down counter is included to provide both pixel counting and event capture capability. Logical shifting function provides a mechanism for self testing the register connection paths. The Cyclic redundancy checking function combines the logical shifting and Exclusive Or logic to create a maximal length

pseudorandom shift register. The first version of the signature circuitry only supported CRC-16 signatures with the upper stage 8-bits only receiving carry output of the CRC generator. The necessity for a 24-bit register length was derived from pixel counting requirements, where the count was determined by the number of pixel per video frame. Later PAL versions were developed using the 24-bit register length for maximal length CRC-24 signatures.

The sync signature channel (U15-U17) generates a sync reference CRC signature that will provide the same signature for every video frame acquisition cycle, despite the changes in video pixel information. This circuit monitors the repeatability of the composite synchronization signal, which in turn controls the acquisition control circuitry. The repeatability of the sync signature determines the timing accuracy of the observed video channel signature. This prevents improper diagnosis of video faults caused by sync signal faults or instability.

The video signature circuitry is identical in every aspect to the synchronization signature circuitry. The major functional difference is that signals routed to the video signature circuitry may be any video or sync signal. The complete video test cycle requires several frame acquisition test cycles that monitors different signal components under various video pattern conditions. For each test acquisition cycle a source and sync signature will be generated and

readable by the microprocessor. The microprocessor must validate the sync signature prior to interpreting the results obtained from the video frame signature.

3. SOFTWARE DEVELOPMENT SUPPORT

Implementation of Pseudo-Structured Rules for BASIC-52 Programming

In any industrial or educational software development project the requirements for documented and maintainable software is of primary concern. Often the software designer considers the project is to trivial to maintain good documentation. The impact occurs in future software maintainability when engineering expertise is lossed, or elapsed time decays software understanding. The most severe documentation impact occurs when the programming language does not promote structured programming, such that the inherent documentation provided by the programming environment does not exist.

The BASIC-52 environment is typical of line numbered non-structured languages that provides only embedded line remark "REM" statements for limited documentation use. This BASIC language is based on simple variable names, on line numbered branches, and global variable subroutine calls. These restrictions prevent inherent self-documentation similar to structured languages such as PASCAL.

The creation of a pseudo-development environment was required to provide the programmer the visual illusion of a structured language. The illusion occurs because a major compiler and linker would be required to provide private variable usage and branch label linkage similar to a

structured language. The pseudo-environment does provide structured program documentation in a unique free-form programming language.

The pseudo-environment is a rule-based program that extracts BASIC-52 strict program code from the free-form documented program. This rule-based program must understand generic constructs and punctuation of the BASIC-52 language, plus private documentation constructs of the pseudo-environment. The rule processing portion of the environment is known as the extractor.

The following sections present a brief review of the rules and examples used by the pseudo-environment program. The first set of rules interpret the usage of BASIC language symbols, an actions caused by each symbol. All symbols have been bounded by {} for text readability and separation only.

{" xx xx "} - Text bounded by double quotes is extracted verbatim to retain programmers intended character string.

Example:
10050 PRINT "P.O.F=",X4," C.F.=",X Input & extracted

{ } -Excess spaces are removed by the extractor to minimize the BASIC line length downloaded and maximize character content. A BASIC program line received in input buffer is limited to a maximum of 72 characters. The BASIC-52 interpreter will post process out unneeded spaces after reception into the input buffer. Yet line length error can occur without reaching the maximum character content.

Example:

```
   100 FOR    X = 0   TO  12 STEP4          INPUT STRING
   100 FOR X = 0 TO 12 STEP4                EXTRACTED STRING
```

{,}{:} -The comma in BASIC-52 statements provides grouping separator for commands consisting of multiple entities such as multiple line numbers, multiple data fields. Where the colon is a command separator for multiple commands assigned to the same line number. The comma or colon sets the extractor's extended line flag. This flag allows any BASIC-52 statement the option to be split across multiple text lines for improved documentation purposes.

Example:

```
  230 ON W-10 GOSUB 5000,      [SUBROUTINE 10: Power-up
                    5100,      [SUBROUTINE 11: Wait
                    6000,     [SUBROUTINE 14: Recognition
                   12500       [SUBROUTINE 15: Self-test
   98  X=0 :     [CONTINUE ON FAIL FLAG
      X1=0 :     ["LOOP TEST" FLAG
      X3=0 :     ["PRINT PARAMETERS" FLAG
      X4=0       ["PAUSE ON FAIL" FLAG

 230 ON W-10 GOSUB 5000,5100,6000,12500   EXTRACTED STRING
  98 X=0 :X1=0 :X3=0 :X4=0                EXTRACTED STRING
```

The following symbols are non-BASIC symbols that are private to the extractor program and do not appear in the BASIC-52 output program.

{[} -Begin documentation symbol, this symbol signals the extraction program to ignore all characters to the right of this symbol. This comment occurrence does not change internal extraction control flags, but forces extraction to resume on next line of program. This comment symbol is the preferred method of documentation compared to the BASIC language "REM" statement that uses valuable program memory to store comments.

Example:

```
     [TITLE SCREEN POSITIONING ROUTINE
   100 X=X+1      [INCREMENT COLUMN INDEX
```

(@) -This private symbol forcefully informs the extractor that the completion of the BASIC Instruction will exist on the next line of the documented file. This symbol allows program lines to be broken apart for separate documentation requirements.

Example:

```
  8060  IF F9 THEN @              [IF FAIL FLAG SET THEN
          PUSH E9,5,F3,F4,F2,0   [PUSH TEST SYSTEM FAIL

  8060 IF F9 THEN PUSH E9,5,F3,F4,F2,0  EXTRACTED STRING
```

(!) -This symbol informs the extractor to cancel any line extension flags set by previous encountered symbols such as (,)(:)(@). The symbol primary use is for a special BASIC-52 command that uses a trailing comma to suppress carriage return.

Example:

```
    10 PRINT "TIME=",!
    20 INPUT A

    10 PRINT "TIME=",   EXTRACTED STRING
    20 INPUT A
```

BASIC-52 Software Development Environment

The BASIC-52 software development environment (VTB52) was created to provide the functions of program language extraction, to host downloading of program into BASIC-52 CPU, perform ASCII dumb terminal emulation and text editor shell. These functions were considered a necessity when one realizes the non-existence of a simple line editor or any formal documentation mechanism. This software platform provides a

disk supported method of editing and maintaining the documented source program used by the video analyzer. The development language used for this environment program is Microsoft QuickBASIC 4.5.[8] The environment program was developed to executed on an IBM PC-compatible with the program listing appearing in the appendix.

The program extraction function is based on concise use of the pseudo-structured documentation rules developed in the previous section. This program flow is too complex to be specific but a brief overview will be presented. The pseudo-structured documentation file is read one line at a time by the extraction program. The program scans each character left to right searching for extraction control characters. Extraction characters encountered set specific control flags that modify the rules for reassembly of the BASIC-52 language program lines. This process continues using these extraction rules until the end of the file is reached. This process creates a pure program output file that will be used for BASIC-52 CPU downloading. If documentation rules are violated, then the extraction program will stop processing when a problem is detected and log the results to an error file.

The following Table 3 is a comprehensive example of several documentation features and the resulting extracted output program in final form ready for download to BASIC-52.

The second function of the software environment is downloading of the BASIC-52 program to the microprocessor via

Table 3 Documented and Extracted BASIC-52 Program.

Example: Documented program

```
150 IF X5=0 THEN GOTO 190     [X5=0 RUN PRODUCTION
160   DO:
        READ X7:              [GET NEXT SUBTEST NUMBER
      UNTIL X7=0              [FIND END OF TEST
165   Y2=0
170   DO : READ X7            [LOOK-UP TEST TABLE
175     IF X7=0 THEN @        [CHECK FOR SEQUENCE END
          Y2=Y2+1             [INCREMENT TEST NUMBER
180   UNTIL Y2=X6             [FIND REQUESTED TEST
190 TIME=0:                   [RESET TEST TIME
     DBY(71)=0                [CLEAR MODE FLAG
200 READ X7:                  [READ SUBTEST SEQUENCE
     PRINT " LOOP=",X9,       [DISPLAY TEST LOOP COUNTER
           " TIME=",TIME,     [DISPLAY ELAPSED TIME
           " SUBTEST=",X7     [DISPLAY SUBTEST NUMBER
205 E=X7*1000                 [COMPUTE BASE ERROR CODE
210 ON INT(X7/10) GOTO 220,   [SUBTEST 10:INITIALIZE
                       230,   [SUBTEST 20:CHECK HARDWARE
                       240,   [SUBTEST 30:SET HARDWARE
                       250    [SUBTEST 40:ACQUIRE FRAME
```

Example: Extractor output

```
150 IF X5=0 THEN GOTO 190
160 DO : READ X7 :UNTIL X7=0
165 Y2=0
170 DO : READ X7
175 IF X7=0 THEN Y2=Y2+1
180 UNTIL Y2=X6
190 TIME=0: DBY(71)=0
200 READ X7:PRINT"LOOP=",X9,"TIME=",TIME,"SUBTEST=",X7
205 E=X7*1000
210 ON INT(X7/10) GOTO 220,230,240,250
```

the RS-232 serial line. The establishment of communication is simplified since the BASIC-52 CPU auto-baud's to an ASCII space character at power-up. The software environment sets a baud rate of 9600 bps to provide compatibility with the test host and other equipment. The download program performs a series of operations to transfer a BASIC-52 program line to the CPU. The first step in the download support program is to attempt to auto-baud the BASIC-52 microprocessor. Once BASIC-52 monitor prompt is acknowledged, the environment begins the download program. The download program reads a line from the program disk file then outputs the program line over the serial line. The downloading program listens for the CPU's echo of the program line and the following BASIC-52 monitor prompt. The monitor prompt is the primary handshake signal acknowledging completion of the BASIC-52 line translation to its internal token symbolism. This handshake prevents overrun of the BASIC-52 interpreter by the downloading computer. The downloading program monitors BASIC-52 command line echos to perform transfer verification. Any additional error messages returned during the CPU's command line echo from the CPU board are saved to an error file on the downloading host disk. The occurrence of an echo error is not considered fatal but is tallied and reported to error file. This provision of non-fatal echo errors allows the use of BASIC-52 immediate execution commands that generates responses different from the command line echo's. Examples of immediate execution commands

such as "new" will clear program memory, "RAM" will select program memory as ram, "FPROG" will begin fast programming EPROM algorithm.

The dumb terminal emulation function provides an 80-column by 23-row character screen with direct ASCII keyboard input. The input character translation provides a subset of the character control standard defined under the ADDS 3+ mode.[9]

The output characters sent from the keyboard are not translated, but a few of the keys are blocked due to incompatibility with the BASIC-52 interpreter. Additional echo on/off terminal control can be user selected through program function keys for program development or program execution.

The fourth function provides a text editor shell that allows the programmer to make modifications to the documented program without leaving the QuickBASIC program environment. The software environment configuration file allows the programmer to specify their chosen text editor.

All functions developed for the video terminal program results in an environment that allows programming enhancements beyond the BASIC-52 monitor.

4. BASIC-52 VIDEO ANALYSIS PROGRAM STRUCTURE

Non-Intelligent Terminal Protocol Definition and Interpreter

The Non-intelligent Terminal Protocol (NITP) defines the requirements to communicate between the test station and video analyzer unit. The terminal protocol defines both the communication procedure and the packet format. The communication protocol is a master/slave hierarchy, where a test host (master) sends a command to the analyzer (slave). The slave then immediately returns the response of any previously processed command, the slave is then free to execute the new command. This method of "do not speak until spoken to" communication allows for multiple listeners and small communication buffering requirements. In this communication procedure a busy slave processor does not listen until the current executing command has completed. This requires the host to retry communication reconnection several times before giving up on the next slave command. A benefit of this form of communication allows both master and slave to execute simultaneous program, without the continious need to support the communication channel. This simultaneous processing if used properly can reduce total test time in a well planned test. The primary disadvantage is that incorrectly written channel reconnection software using fixed communication delays instead of communication polling for reconnection of channel will suffer serious inefficiency problems.

Beyond the communication procedures the NITP terminal Protocol also defines the content and format of the communication packet. The packet character set contains the ASCII characters for the hexadecimal numbers 0-9, A-F and the bounding characters of colon, semicolon and carriage return. Packet information is partitioned by two hexadecimal ASCII characters representing an internal byte to the CPU. This simple character set allows ease of communication from a test computer or an ASCII dumb terminal. All packets are bounded by delimiters of a leading colon and a trailing semi-colon, which signifies start and end of pack respectively. Any received characters strings not bounded correctly by these delimiters are ignored. The test host command preamble consists of a message length field, a destination address field, a primary command field and a secondary command field as shown in Figure 7. Fields beyond the preamble are defined to be variable format based on the command and secondary command requested.

The command interpreter portion of the BASIC-52 program dissects the received command to check for proper command format. The command interpreter checks for proper packet bounding, for accuracy of message length and checks destination field for expected address. Then range checks the command field and the subcommand field. The subcommand field is called the test number within the BASIC-52 program. The variable data field of the command will be interpreted by

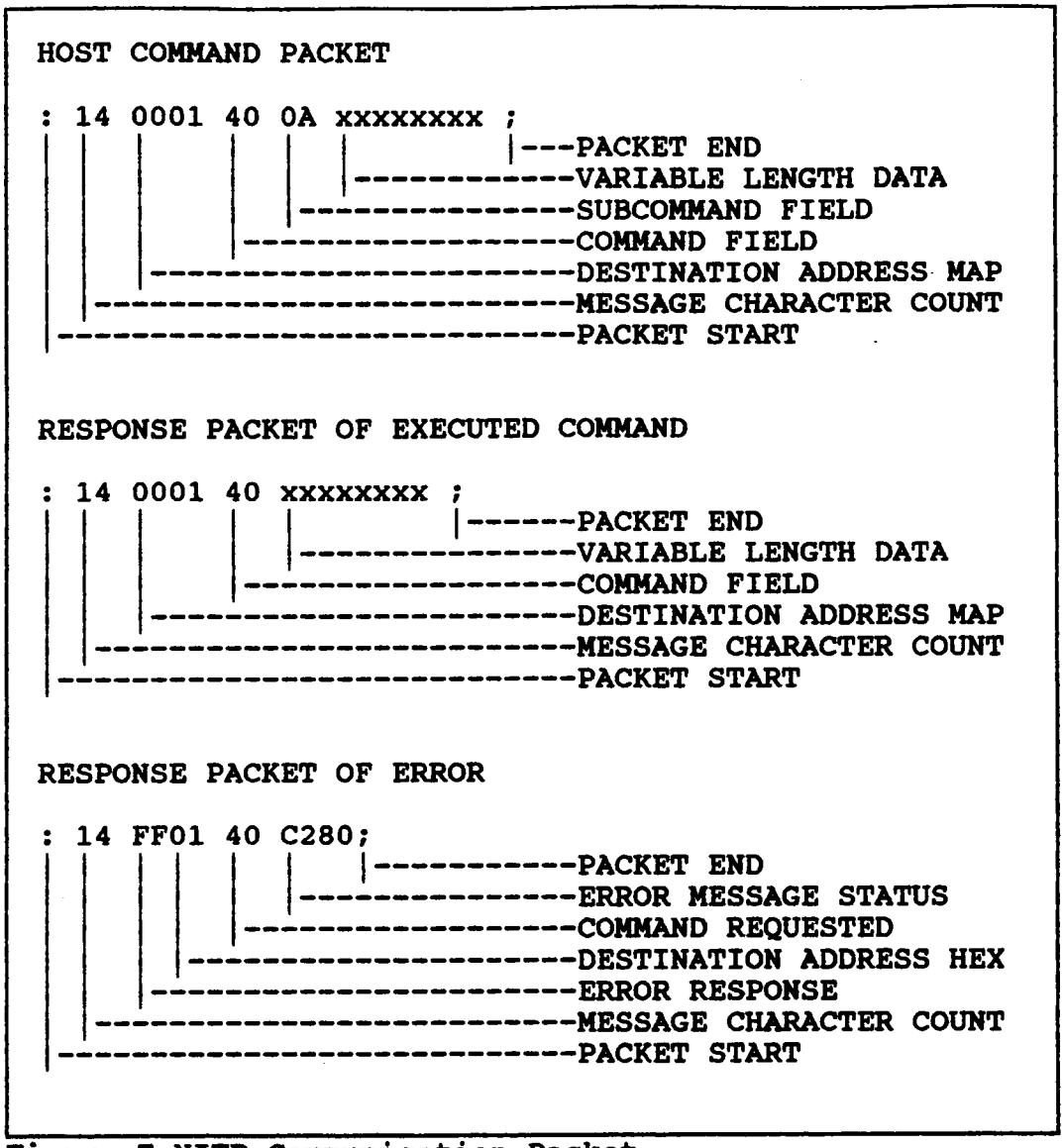


Figure 7 NITP Communication Packet

rules established for the requested test command and subcommand field. Error checking of variable data field is the responsibility of the requested test and not the command interpreter.

The communication packets encountered in the analysis system defined three distinct packet classifications. The first type is the host command packet, which defines the

specific test function to be performed by the video analysis system. The second type is the minimal response packet that generally acknowledges the completion of a previous command request. This packet may contain additional information based on specific function of command. The third type is the error packet, which informs the test host that a major error was encountered in executing a test command on the video analyzer, or improper communication packet format was detected. The error packet provides an error code field to classify the type of fault detected.

Currently supported command and response packets are directed into the four primary areas of offline diagnostic, video configuration, video test and special debug operation. The command packets shown in Figure 8 and Figure 9 are presently being supported by the test system.

Test and Subtest Sequencer

The test sequencer objective is to search for, then execute a requested test routine. The test routine consists of a set of small subtest that perform specific testing functions, required to complete the main test. Therefore the test sequencer is the program controller that schedules sequenced execution of subtests to create the requested test. The use of subtest sequencing promotes modular programming, this simplifies program development and debug of software/hardware problems. The test sequencer also may be called the test manager interchangeably in this document.

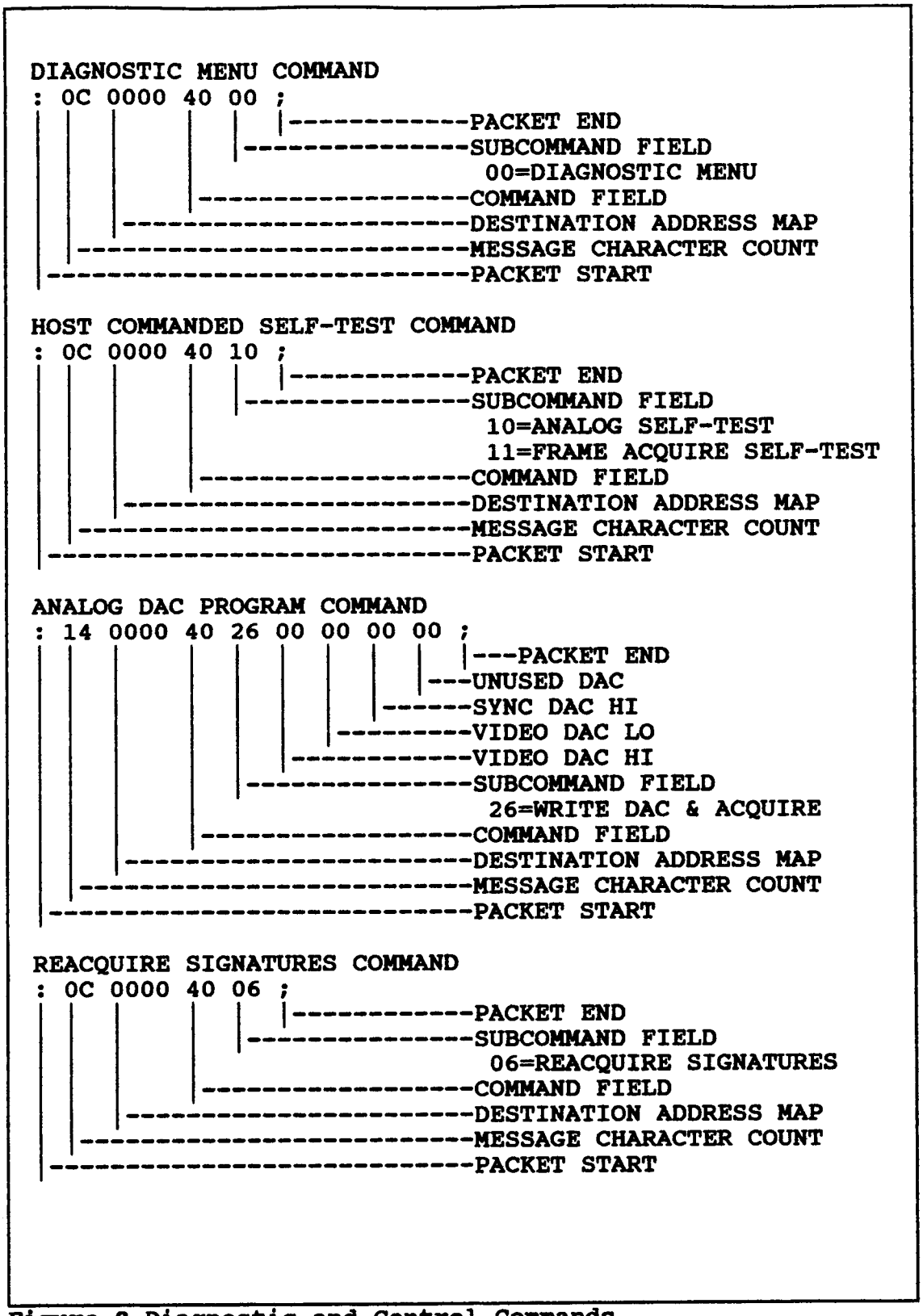


Figure 8 Diagnostic and Control Commands

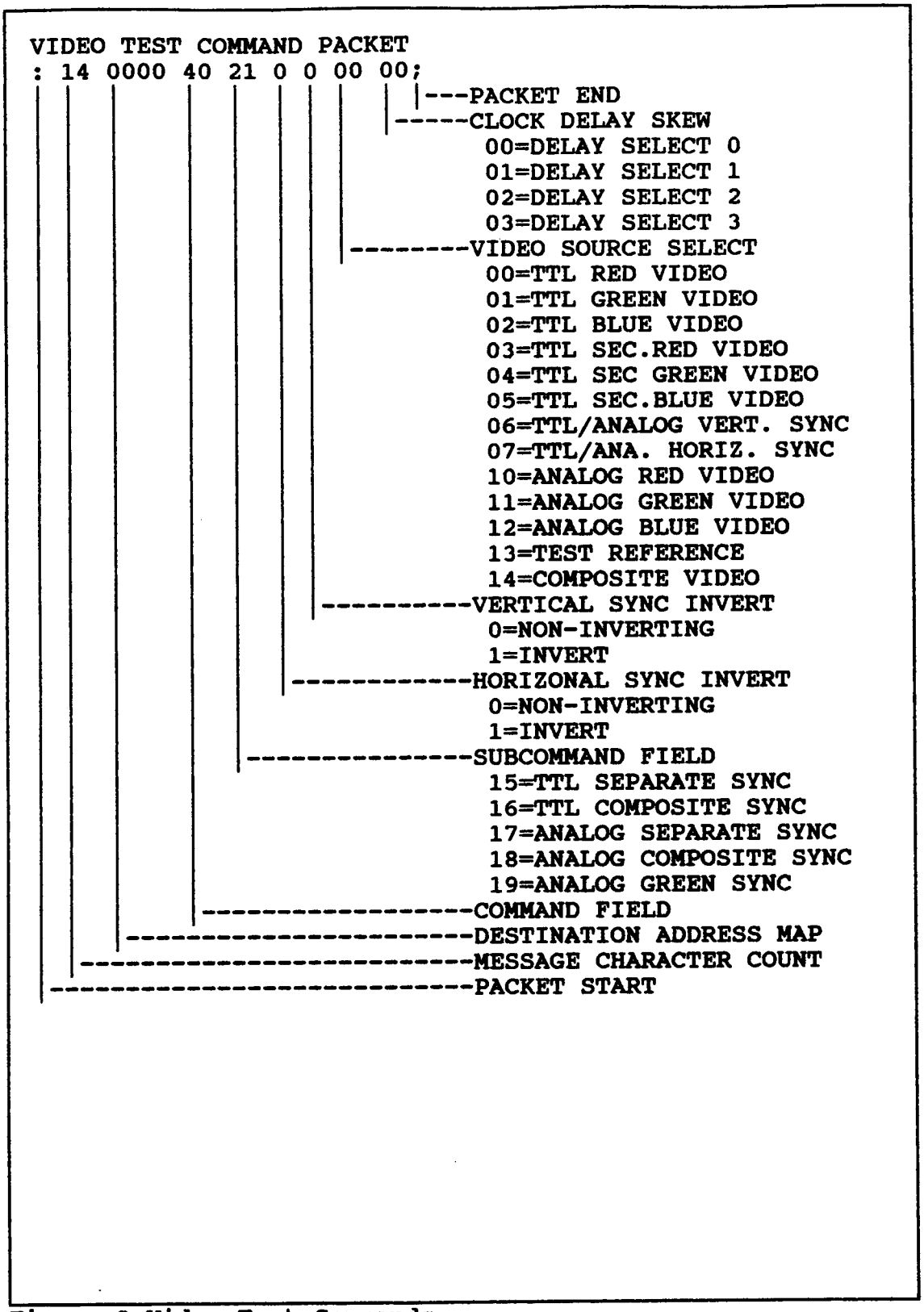


Figure 9 Video Test Commands

The test sequencer software is sub-divided into several specific tasks that directs all test execution. The first operation of the test sequencer is to search a table of test sequences for the requested test. The test sequence tables have a specific structure where each test sequence terminates by a subtest zero.

```
Example: Test Sequence Table
   300  DATA 1,2,3,4,5,0
   310  DATA 1,4,6,12,13,0
```

Next the sequencer then reads and calls each subtest in the sequence found in the table. The subtest numbers are mapped to subtests routines by a BASIC-52 implementation of a case-gosub selector. This sequencing mechanism allows for a controllable environment, where special program control flags can modify test execution to allow special program development options. Examples of these options are loop-test, pause-on-fail, continue-on-fail, reveal intermediate pass/fail results and single-step subtest. The first subtests in the sequence list generally initialize the test system hardware and program variables to a known initial state. The middle subtests generally perform the specific video mode hardware setups and actual testing. The zero subtest ends the sequence but also is the only subtest allowed to generate a "TEST PASS" result in the software. This implies that "TEST FAIL" results will exit testing before the test sequence completes. A graphical overview of subtest sequencing and error handling is shown in Figure 10. Notice for the video analysis unit "TEST FAILS"

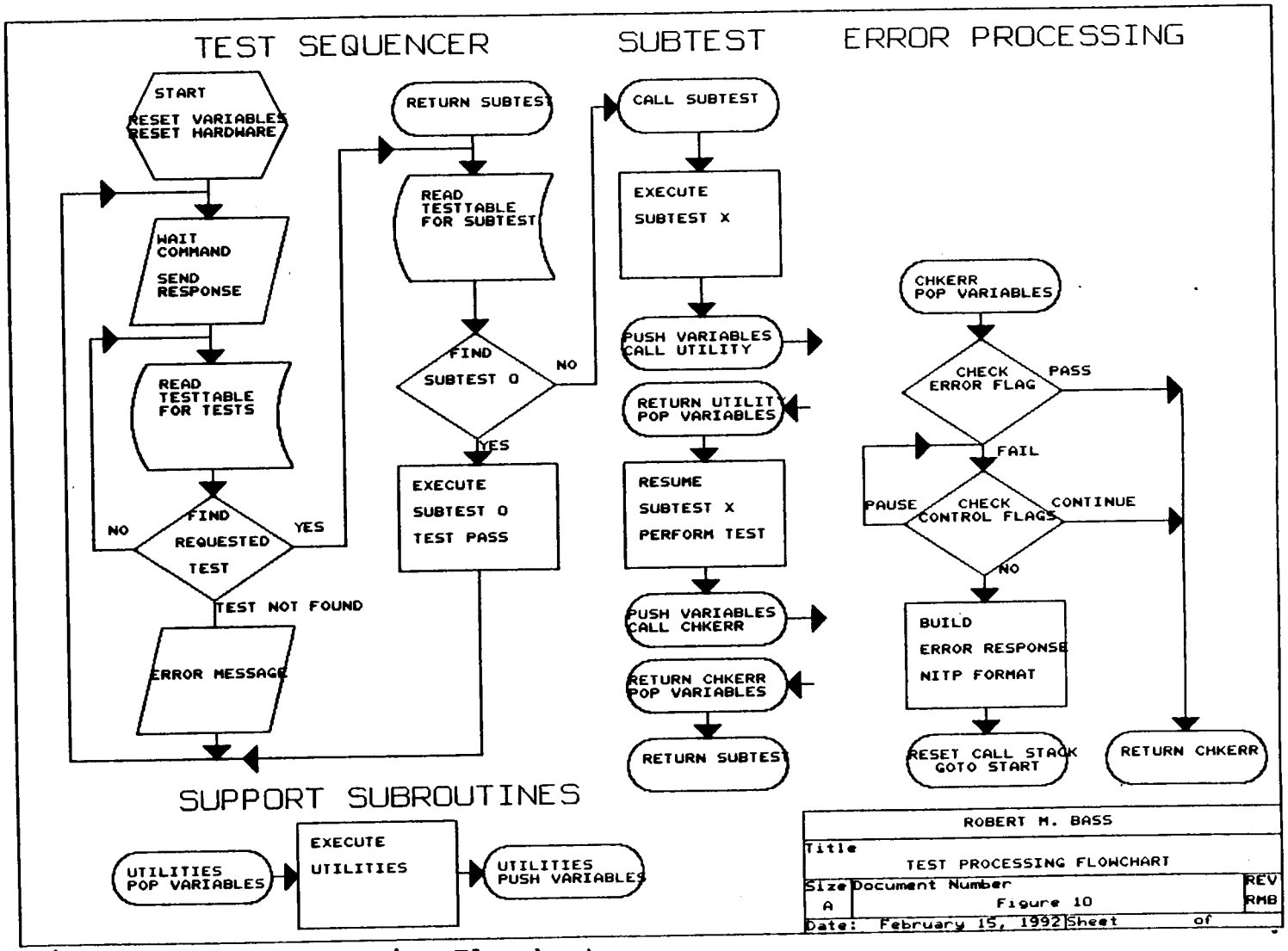


Figure 10 Test Processing Flowchart

only occurs for incomplete frame acquisition cycles or unstable signatures. Since the analysis system acts as a signature measurement instrument the expected video signatures are only available to the test host. Therefore the analysis board cannot fail the UUT for bad signatures. Reporting bad signatures failures is therefore the responsibility of the test host.

Test Program Structure

The video analysis test program structure allows for powerful software designs with minimal effort required to support error logging, error handling, test sequencing and test development. Yet there are basic requirements on program structure that must be satisfied to provide proper operation in the test environment. Since the test routine is a series of subroutines, then only proper control stack usage is allowed. This prevents test manager control stack mismanagement errors. The use of structured programming techniques that minimizes use of "goto" statements and proper visual indentation of the program improves readability.

In addition all variables are considered global in extent, thus variable reuse must be carefully monitored or better yet avoided. A safer practice is to consider all subroutine variables as private, then variable linkage can be performed by using BASIC-52 stacked variable argument passing. Stacked variable passing operates as a push then pop stack manipulated as a first-in last-out stack.

The test routines must avoid direct hardware control, since most hardware control ports have several functions defined to particular bit groups. The possibility of unintentional port hardware mishaps can occur if routines do not manage port writes carefully. The better method is to use variable passing to utility routines that modify only bits related to the utility hardware function. Hardware port setting history must be maintained by using global port variables between all utilities. This method also reduces the software development mistakes of writing redundant hardware control code every time a hardware setup changes. A favorable side effect is that utility routines provide improved functional documentation due to the consolidation of all functions into one routine.

Error Handling & Reporting

Proper error handling is very important when allowing for special test control operations. A subset of the test control operating modes are loop-test, continue-on-fail, pause-on-fail and forced-fail on error code. These special modes are activated by a series of control flags that affect the test sequencer operations. To process potential test errors a single error handling routine was designed into test manager program. This routine properly handles all error check decisions and monitor sequencer control flags.

EXAMPLE:

1.Setup local variables
2.Pass variables on stack, Call utility subroutines
3.Perform test stimulus operation.
4.Check response results then set PASS/FAIL flag.
5.Stack decision parameters and potential error code.
PUSH ERROR,LOW,EXP,HIGH: GOSUB CHKERR
6.Call error handler despite PASS/FAIL condition.
7.If return occurs continue testing normally
8.Restore necessary hardware on subtest completion.
9.Return to test manager.

It is important for a subtest to satisfy the following basic requirement to allow full utilization of the test manager program and features. One basic requirement is that subtests with potential failures must not reset the system hardware or variables prior to execution of error checking routine. This preservation of the system history allows for maximum flexibility in debugging software and hardware by freezing the last setups. The additional history also allows the sequencer to resume modes of continue on fail and forced fail on error code. These diagnostic modes are used to modify sequencer operation or reveal measurement result margins for statistical process control (SPC) monitoring. For flowchart overview of subtest sequencing and error handling see Figure 10.

5. SPECIFIC VIDEO TESTING SOFTWARE

A basic understanding of the video test software structure will be valuable in comprehending the relationship between test host, video UUT and analysis hardware.

System Programmer Model

The previous sections presented an overview of the functional hardware, while this software section will view the hardware sub-system from the point of view of a software programmer's model. Software development proceeded after the hardware was translated into a comprehensive memory map. This memory map defines all hardware ports down to individual addresses and unique port control bit assignments. The full detail memory map is shown in Table 4. The CPU card's memory map has reserved sixteen discrete I/O card locations, each containing sixteen bytes of I/O. Note that all the video analysis functions map into the area reserved for discrete I/O cards, ranging from card 9 to 11. Since the analysis system was never intended to support such a large I/O map partial address decoding minimized the decoder logic resulting in a three card location memory map. The equation below explains the address decoding involved for discrete and analog cards as defined by the original BASIC-52 test system.

Table 4 Video Analysis System Memory Map

ADDRESS	PROGRAM AREA	DATA AREA
0000-1FFF	INTERNAL ROM INTERPRETER	EXTERNAL RAM 8 K
2000-5FFF	EMPTY	EXTERNAL RAM 16 K
6000-7FFF	EMPTY	EXTERNAL RAM 8 K
8000-BFFF	EXTERNAL ROM 16 K	EXTERNAL ROM 16 K
C000-DFFF	EXTERNAL ROM 0 K	EXTERNAL ROM 0 K
E00X-E1FX	EMPTY	INTERVAL TIMER
E200-E3FF	EMPTY	ANALOG CARDS
E400-E5FF	EMPTY	DISCRETE CARDS

ADDRESS		DATA AREA
E400+88	R/-	SIGNATURE DEVICE 0= LOW BYTE
E400+89	R/-	SIGNATURE DEVICE 1= MID BYTE
E400+8A	R/-	SIGNATURE DEVICE 2= HIGH BYTE
E400+8B	R/-	SIGNATURE DEVICE 3= LOW BYTE
E400+8C	R/-	SIGNATURE DEVICE 4= MID BYTE
E400+8D	R/-	SIGNATURE DEVICE 5= HIGH BYTE
E400+9B	R/-	CTRLREG1RD= DB7..0=not defined
E400+9C	R/-	CTRLREG2RD= DB7..0=not defined
E400+9F	R/-	TESTREGRD
		DB3..0=CTL3..0 ACQUIRE FRAME STATES
		DB4=LATHORZ: PIPELINE HORIZONAL TEST
		DB5=LATVIDEO: PIPELINE VIDEO TEST
		DB6=LATCOMP: PIPELINE COMPOSITE TEST
		DB7=LATVERT: PIPELINE VERTICAL TEST
E400+98	-/W	RLYSET RELAY CONTROL SET
		DB0=RLYREF/GRN
		DB1=RLYCMP/RED
		DB2=RLYGRN/BLU
		DB3=RLYRED/GRNBLU
E400+99	-/W	WRTMRLO VERTICAL SYNC TIMER LOW
		DATA= 00..FF
E400+9A	-/W	WRTMRHI VERTICAL SYNC TIMER HIGH
		DATA= 00..FF

Table 4 (Continued)

ADDRESS		DATA AREA	
E400+9B	-/W	CTRLREG1WR	
		DB2..0=SRCSEL2..0	TTL VIDEO MUX CONTROL
		DB5..3=VIDMOD2..0	VIDEO MODE CONTROL
		DB6= INVERT HORIZONAL/COMPOSITE SYNC	
		DB7= INVERT VERTICAL SYNC	
E400+9C	-/W	CTRLREG2WR	
		DB1..0=DLYSEL1..0	TIMER SKEW CONTROL
		DB4..2=MODE2..0	SIG/COUNTER MODE
		DB5=ACQFRM*	ACQUIRE FRAME TRIGGER
		DB6=CMPTST*	SYNC TEST SIGNAL
		DB7=VIDEOTST	VIDEO TEST SIGNAL
E400+9D	-/W	UNUSED 5	
E400+9E	-/W	TSTCLKWR	TEST CLK STROBE
E400+9F	-/W	TESTREQWR	TEST CONTROL REGISTER
		DB0=TSTCMPSYNC	TEST COMPOSITE SYNC
		DB1=VIDTST	TEST VIDEO SIGNAL
E400+A0	-/W	DAC CH.B & CH.D	
E400+A0	R/-	DAC HOLD	
E400+A1	-/W	DAC CH.A & CH.C	
E400+A1	R/-	DAC HOLD	
E400+A2	R/W	DAC CH.D	
E400+A3	R/W	DAC CH.C	
E400+A4	R/W	DAC CH.B	
E400+A5	R/W	DAC CH.A	

ADDRESS	PROGRAM AREA	DATA AREA
E600-E7FF	EMPTY	EMPTY
E800-EFFF	S.F.MEMORY AREA	S.F. MEMORY AREA
F000-FFFF	TOPPAGE	TOPPAGE MEMORY

```
     Address Structure for Discrete Cards
E400 +                          Discrete base address
       000x-00Fx +              Sub-address of card
                   0000-000F    Byte address on card

     Address Structure for Analog Cards
E200 +                          Analog base address
       000x-00Fx +              Sub-address of card
                   0000-000F    Byte address on card
```

Analysis System BASIC-52 Framework

The next sections will discuss the specific BASIC-52 video software functional blocks. The sections were written with intentional avoidance of overwhelming program details to improve readability and comprehension. The functional software blocks are defined to reflect the definitions established in previous sections for communication protocol, test system outline and hardware requirements. Each functional block description includes program line number ranges for parallel study of the BASIC-52 video test program located in the appendix.

Software System Initialization

The system initialization software resides in two distinct regions of the test program, these regions defined memory variables as volatile and non-volatile. Non-volatile variables are those variables that are set only once at power-up and are never changed under program execution thus require careful programmer protection. The typical non-volatile BASIC variable initialization region is in program range lines 0-99. These variables configured the software test sequencer and BASIC-52 CPU hardware resources. Typical system control

variables initialized are the microprocessor clock frequency, the display port baud rate and the clock timebase generation. Included in this non-volatile section is also the test system configuration variables.

Variable classified as volatile are those variables that are modified and shared during program execution, these require careful use and initialization to prevent unwanted side effects. Shared variable usage is commonly used in this test software to preserve history of a hardware device or as a linkage variable between unrelated routines. Typical volatile variables are being reinitialized in lines 270,1700-1799, these program lines show variables to control test execution, to preserve software port history and to reinitialize hardware ports. Most volatile variables are reinitialized at the beginning of each test sequence execution cycle to guarantee test cycle repeatability. All other variables not classified in these two categories are considered free use variables and have no use restrictions.

Message Input Parsing and Verification

The message input parsing and verification region of the program at lines 100-249 begins by receiving the command string from the host computer. The routine then checks the command string packet for proper format based on Non-Intelligent Terminal Protocol (NITP) rules. When packet errors prevent completion of the requested command the parsing routine builds an error response string to return to the host

as the next response. On completion of parsing error checks the program extracts out test and subtest numbers to be used by the test sequencer.

Test Sequencer Program

The Test sequencer section is at program lines 250-499. The first group of program lines checks the sequencer control flags for special operating modes, such as diagnostic menu control or continuous test looping. The next portion of this section scans the test tables for required test sequence starting point. The test sequences frames are counted by searching for the termination subtest zero. The last section beginning at line 300-499 emulates a case-gosub selector, this routine correlates a subroutine to a subtest number then calls the subroutine for execution. When the subroutine execution completes it returns control back to the test sequencer which then executes the next subtest.

Development Menu

The development menu section of the program located between lines 1000-1199. This program provides an interactive display menu to allow the system programmer or repair technician direct menu control of the video analysis test software. This menu interface bypasses the requirements to communicate with the analysis system using the *Non-Intelligent Terminal Protocol*, providing a user friendly development interface. Menu operation is evoked by sending a unique menu control command packet.

The program section from line 1100-1190 is the diagnostic option menu. This option menu allows the user to set special control flags to modify test flow and test sequencing. Examples of programmable control flags are *pause-on-fail*, *continue-on-fail*, *loop-test*, *error-code-trap*, etc. The control flags are set or cleared by a set of IF-THEN-ELSE statements that toggle the flag setting. The error code trap allows the programmer to input an error code into a basic variable. During test execution at pass/fail decisions the test manager error check routine screens the error code against the error code trap variable. On occurrence of an error trap match the error check routine forces the fail condition, then fatal error processing. This allows the repair technician to stop and hold all hardware settings at all critical points in the test cycle. The software designer must decide where the critical test points are located, then place error check calls at these locations.

Video Attributes Hardware Interface Utilities

Video Hardware Configuration Utilities

The video configuration utility subroutines provide the software interface to the video control hardware. These utility programs are located in program lines 7680-7859. The utility subroutines methodology provides specific routines to control functionally related hardware signals. By using only these utilities to configure the video analysis hardware the programmers risk of incorrectly manipulating an I/O port is

minimized. The eleven BASIC-52 subroutines defined in this section applies to analysis board write accessible hardware. As an example the second routine at line 7700 receives the parameter of video source selection from the BASIC-52 argument stack. The utility then range checks the value to decide if the mode grouping is TTL or analog video. If the selection is TTL video then the TTL multiplexer routes the requested signal to the pipeline pal. If the selection is analog video then the analog video signal is relay routed through the window detectors to the pipeline pal.

Video Hardware Inquiry Utilities

Subroutines that read or monitor the analysis hardware are called the video hardware inquiry utilities. These utilities also prevent accidental misuse of hardware and simplify hardware monitoring. These subroutines perform read accesses to accumulate, separate, and format results. All inquiry utilities functions are shown in program lines 7860-7889. An example subroutine in line number 7880 accumulates signature from sync or video signal accumulators. The utility first decodes the argument stack variable into the hardware base address of either the sync or video signatures accumulators. The routine reads each hardware port using a displacement offset to align with one of three accumulators. The three readings are then returned to the calling routine.

Video Hardware Self-test Utilities

Special self-test utilities throughout this program provide direct hardware simulation of video and sync signals. The routines in lines 7900-7999 provide the software simulation of the video patterns or sync signal generation for a requested number of video clock cycles.

Hardware Self-test Support Software

Analog Circuitry Self-test Subroutine

The analog detection circuitry self-test shown in Program lines 8000-8499 performs the digital to analog converter (DAC) and analog comparator self-tests. The self-test begins by relay routing the test reference signal into each comparator. Then the test software loop programs the DAC until the comparator output acknowledges crossing the trigger point. The DAC trigger point value is checked to be within the specification range. All comparators' channels are checked one-by-one using this method. Since the test reference signal is uncorrelated to the DAC reference a reasonable analog self-test is accomplished without the possibility of a reference tracking error.

Digital Video Acquire Self-test Subroutine

The digital video frame acquire self-test shown in program lines 8500-9019 perform the most complex part of the digital self-test. This test makes use of three test signals to generate both the video, sync and dot clock signals. These's signals are referenced on the schematic diagram as

Videotst, *Cmptst* and *Tstclkwr*. The main purpose of this test is to synthesize each portion of the video frame cycle and check the functionality of each circuit block progressively. The first section of the test program lines 8515-8526 checks the operation of the test signals and the routing/pipeline PAL function. This is performed by generating patterns of all possible video signal and verifying output's of the pipeline latch. The second test checks the clock synchronization PAL state machine. The test sequences from the rest state through several false video frame lock-on states, to true lock-on of a video frame, see lines 8700-8749. The state machine states are monitored throughout the first vertical interval and before frame acquire starts, at this point the signature accumulators are checked for reset values, see lines 8750-8779. Frame acquisition begins on the first video interval, the machine states and signatures are checked throughout the video and horizonal sync intervals, program lines 8780-8789. After several video intervals occur the ending vertical sync is initiated. At this point the state machine and signatures are checked until the signature acquisition stops. See lines 8780-8810. After this video frame completes a new video frame is attempted to test that the state control PAL and signature counters remain shutdown. At this time the video and sync signatures are checked for expected results, see lines 8840-8920. The step-by-step nature of this testing allows for better fault isolation for the repair technician. In addition

each step of this diagnostic program meets the requirements for error code trap operation.

Video Frame Acquisition Software

Specific Test Mode Message Parsing

Specific test mode message parsing occurs in the analysis program at lines 3800-4900, these programs parse the variable length data portion of the NITP command messages. The parsing procedure is based on the test number request found in the fixed portion of the NITP message. Message specific parsing is required because each video signal format has a different setup requirement for TTL and analog modes. The test sequence tables schedules the specific parsing routines required for a given test number request.

Video Acquisition Subroutine

A generic video acquisition subroutine is the by-product of the modular programming techniques used throughout this program. Once all hardware setups are completed the signature acquisition task has been reduced to a minor triggering subroutine. Therefore only a single acquisition subroutine is required for all video modes. This routine is at program lines 5000-5499. An assumption is made that subtest executed prior to this routine have configured all desired video control settings. This routines' function is to clears signature counters then triggers the frame acquisition cycle. After a sufficient frame acquisition time delay the software checks the frame acquire state machine for proper termination state.

From this point the software reads the first set of signatures for sync and video. The program reader also must notice that this subroutine performs two acquisition cycles on the same signal to check sync signature stability. Signature stability is considered a prerequisite to returning any video or sync signatures back to the test station. Signature Instability errors are reported back to the test host using the error message packet format. Secondary stability checks are made on the video signal and reported using an unstable video error message packet. If both stability tests pass then the sync and video signatures are returned to the test host in a signature response packet.

5. CONCLUSIONS AND FUTURE DEVELOPMENT

The primary goal of the video analysis system is to provide an automated alternative to operator-assisted video testing on the production floor. This testing strategy follows the goal of providing automated testing using a minimum amount of test operator or technical personnel intervention. The video testing alternatives considered was to purchase an over-the-counter video acquisition board or design a custom video analysis system. The major obstacle of any video analysis system is meeting the various video format requirements needed to capturing information about a computer generated video signal. Consumer available video acquisition boards investigated host only camera video output formats and do not interpret computer video monitor frequencies.

A fundamental hardware design goal of the video analysis system was to design a stand-alone video analysis instrument that could be commanded by different types of test hosts. The analysis system met the stand-alone requirement by providing its own hardware control intelligence by using an on-board microcontroller. The host-to-system minimal communication channel criteria was met using a RS-232 serial line interface. The capabilities of the BASIC-52 interpreter both in hardware support and in software structures allowed for easy software development. The execution speed of the microcontroller and minimal software overhead has kept the video acquisition cycle time from command to completion in a couple of seconds. The

main test time obstacle for video testing throughput should now be the test host program.

The future test role of this video analysis unit will allow this unit to communicate directly with the UUT and completely bypass the test host during video testing. This would require that the video testing control program be executed by either the UUT or the video analysis unit. The rewritten software would resemble a built-in self-test routine to the UUT being assisted by external analysis hardware. The major advantage would be reduced test time by obsoleting the test host responsibility as the middleman. The second advantage is that the UUT could execute assisted self-test with this video analysis system in a self-monitoring manufacturing burn-in. Future integration and development of this system will create evolutionary changes to the software to meet the goals of video testing requirements.

The analysis system developed in this thesis has provided a promising outlook in meeting all video testing goals. The systems' first manufacturing testset integration has been stalled by work force reduction due to change over from TI to Siemens. Also, longer than expected involvement in completing this document has tied up the hardware from manufacturing integration.

Major non-production testing milestones of this project have been met prior to integration into the manufacturing test environment. The original testset destination for the video

analysis system in the VME PLC system is still planned. In addition a new series of PC-compatible I/O modules also could benefit from this video testing capability.

Several design milestones were met early in the hardware development cycle. The first major milestone was completion of the functional blocks of the schematic to meet the various video format specifications. The first major challenge was the routing and conversion of various video signal formats into COMPOSITE TTL format expected by all downstream hardware.

The second major challenge was maintaining the intended functionality of the system while a continuous battle between PAL selection and propagation delay margins evolved the schematics. The first analysis system was envisioned to contain mainly discrete TTL components but worst case evaluation margins transformed the system into PAL-intensive video logic. The increased role the PAL logic played in this design created a hidden benefit of substituting PAL logic simulation for hardware debugging. This computer logic simulation allowed the seperate development of the project to be completed without requiring extensive troubleshoot effort. The final design completion of the six different PAL and functional simulations was a major pre-fabrication milestone.

The fabrication of the project hardware was the slowest portion of the development. This involved creative component board layout and planned wire-wrapping paths. Due to the high speed circuitry and high frequency clock the two wire-wrap

board layouts were optimized to make best use of the split hardware.

The first utility layer of diagnostic software was developed in parallel to the hardware fabrication. The diagnostic software was used to simulate the first video frame to be captured by the signature hardware. The built-in self-test capability provided exceptional feedback on the integrity of the wiring and PAL programs. The first repeatable lock-on and acquisition of the simulated video yielding the expected video signatures was a major hardware and software milestone. The complete synthesis of a software video frame is currently used in the self-test routines and will serve as a future debugging tool.

All software written from this point was targeted toward development of a video acquisition system capable of handling all expected video formats. The software development at this point is the easiest portion of this project due to the structured definition of these routines. This software was separated into command processor routine, test executive routine, video format hardware initialization routines and video frame acquisition routine.

The most important software milestone was the software development environment (VTB52). This allowed all video analysis hardware and software development to progress rapidly. The software allowed for text editor development of troubleshoot routines which could be downloaded to the BASIC-

52 CPU in seconds. The importance of this tool during hardware development is well understood when the designer considers the number of times the system is powered down and the program is lost due to hardware modifications.

The analysis system has been checked out with various TTL video signals formats such as an enhanced graphics adapter (EGA) and a TTL video graphic adapter (TTL VGA). Yet an early part availability problem of the high speed comparators has prevented the analog video operation checkout. The parts are now available, but work on this document has prevented further hardware checkout. However all analog DAC circuits were self-test verified using low speed comparators to prove the digital logic functionality. The video analysis system integration into the production floor test station will provide the real world video checkouts and needed experience with the system.

The overall performance of the BASIC-52 video analysis platform should allow for improved frame test times and a major increase in test coverage due to the increased number of video test patterns observed. The next major objective is to create two libraries of video patterns that will optimize both detection and isolation of video faults. The detection library will generate worst case video pattern situations for the video hardware. The isolation library will break down the detection pattern elements into their fundamental properties. The use of a dual library approach minimizes test time for a passing UUT by executing only the detection library. A failing

UUT will consume more test time due to the execution of the isolation pattern set to provide fault isolation for the technician. The test software will evolve more in the future by allowing more extensive software to allow the analysis board more stand-alone testing and fewer host communications.

LIST OF REFERENCES

LIST OF REFERENCES

1. INTEL, MCS BASIC-52 User's Manual
   Intel Literature Distribution, 1986
   Santa Clara, CA

2. Robert M. Bass, BASIC-52 Test System Documentation
   Texas Instruments, vol.88400, 1989
   Johnson City, Tn

3. Lattice, High Performance E^2-CMOS Array Logic
   Lattice Semiconductor Corporation, 1990
   Hillsboro, Oregon

4. Texas Instrument CRT Controller, Document Number 258950
   Texas Instrument Industrial Control Division
   Johnson City, Tn

5. PMI, Precision Analog Integrated Circuits Databook
   Precision Monolithics Inc., 1988
   Santa Clara, CA

6. Signetics, Linear Data Manual, Vol.2
   Signetics Corporation, 1987
   Sunnyvale, CA

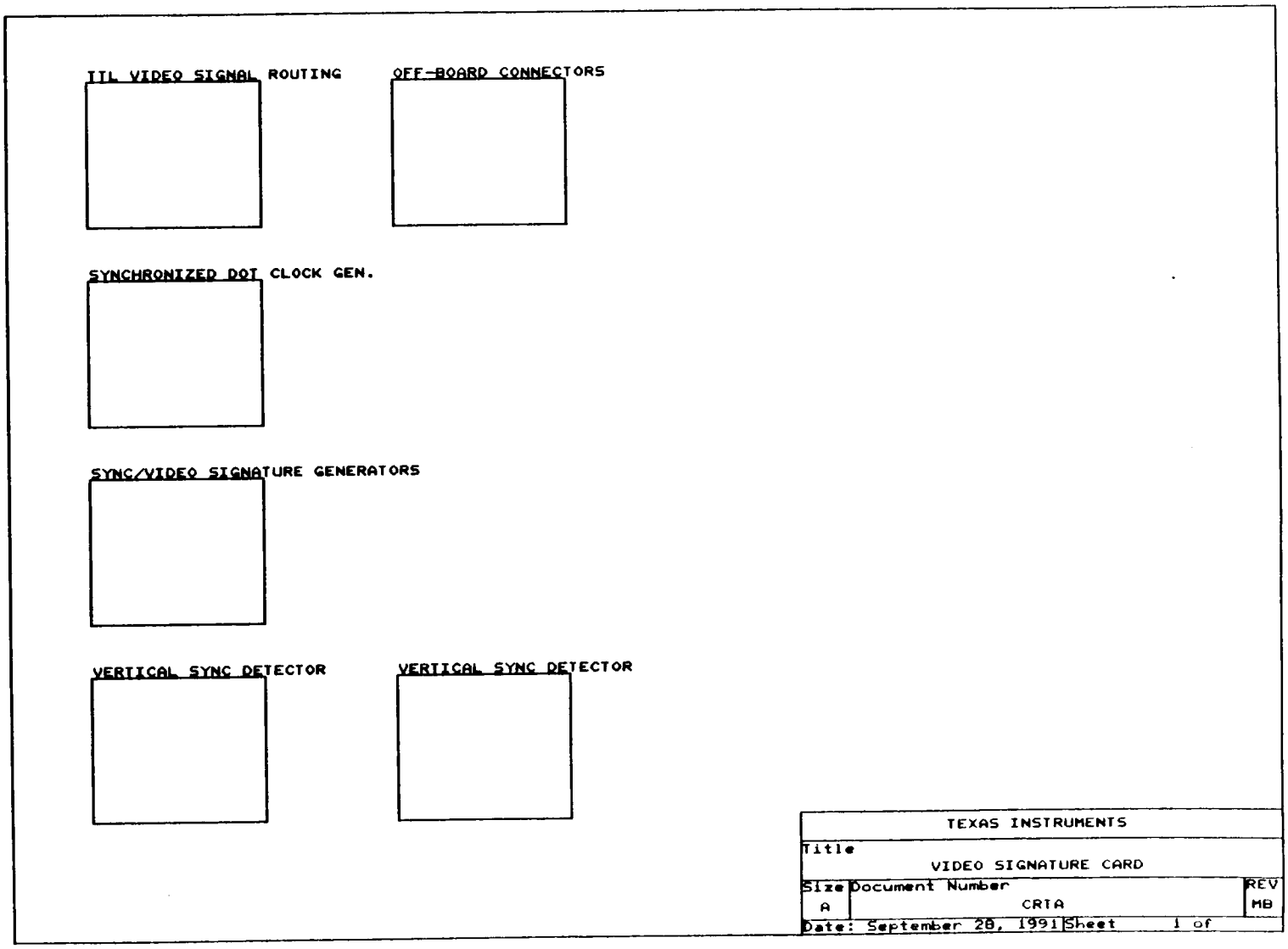
7. NEC, Multisync User Manual JC-1401P3A
   NEC Home Electronics (U.S.A.) Inc.
   Wood Dale, Illinois

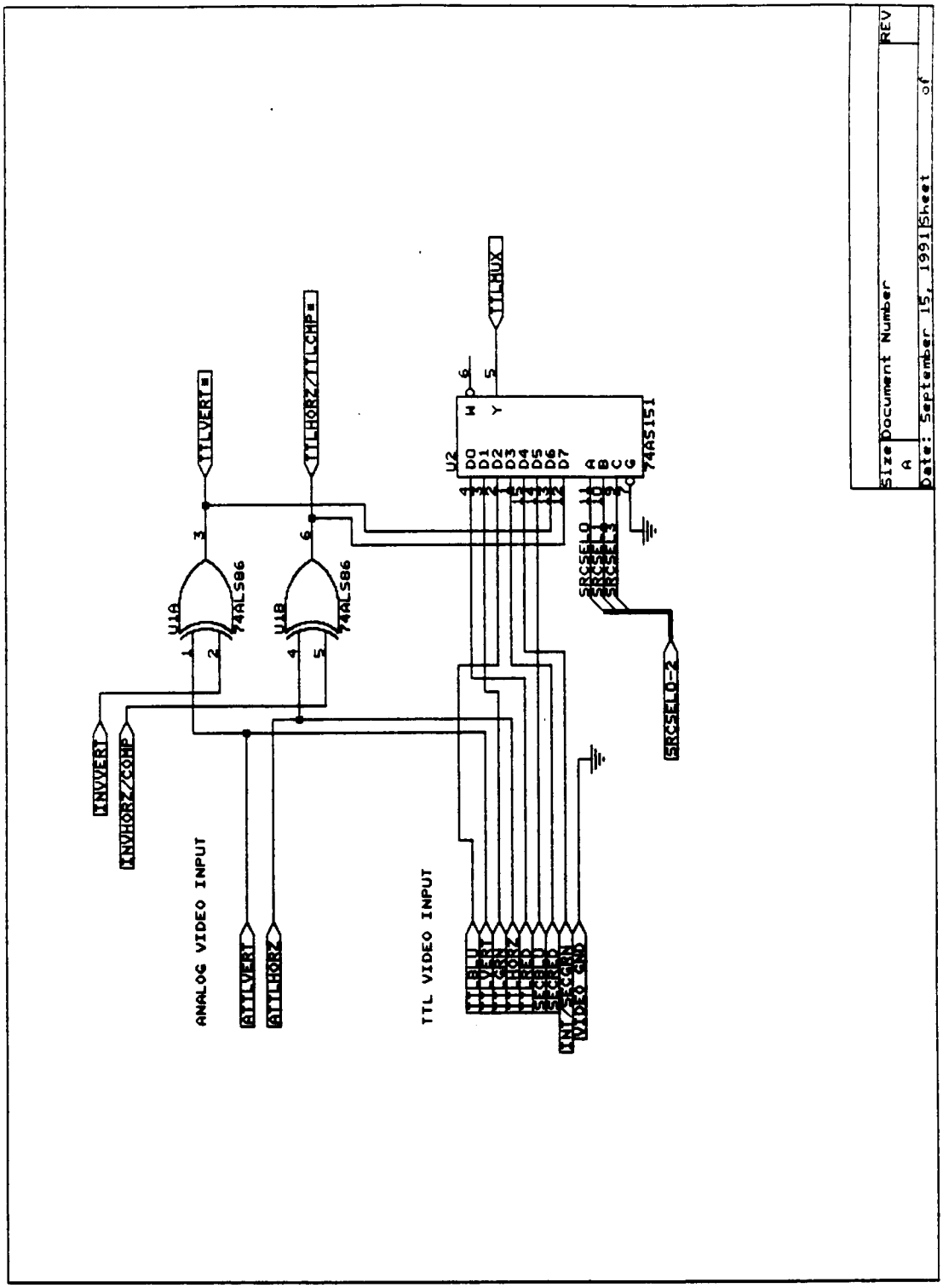
8. QuickBASIC 4.5
   Microsoft Corporation, 1990

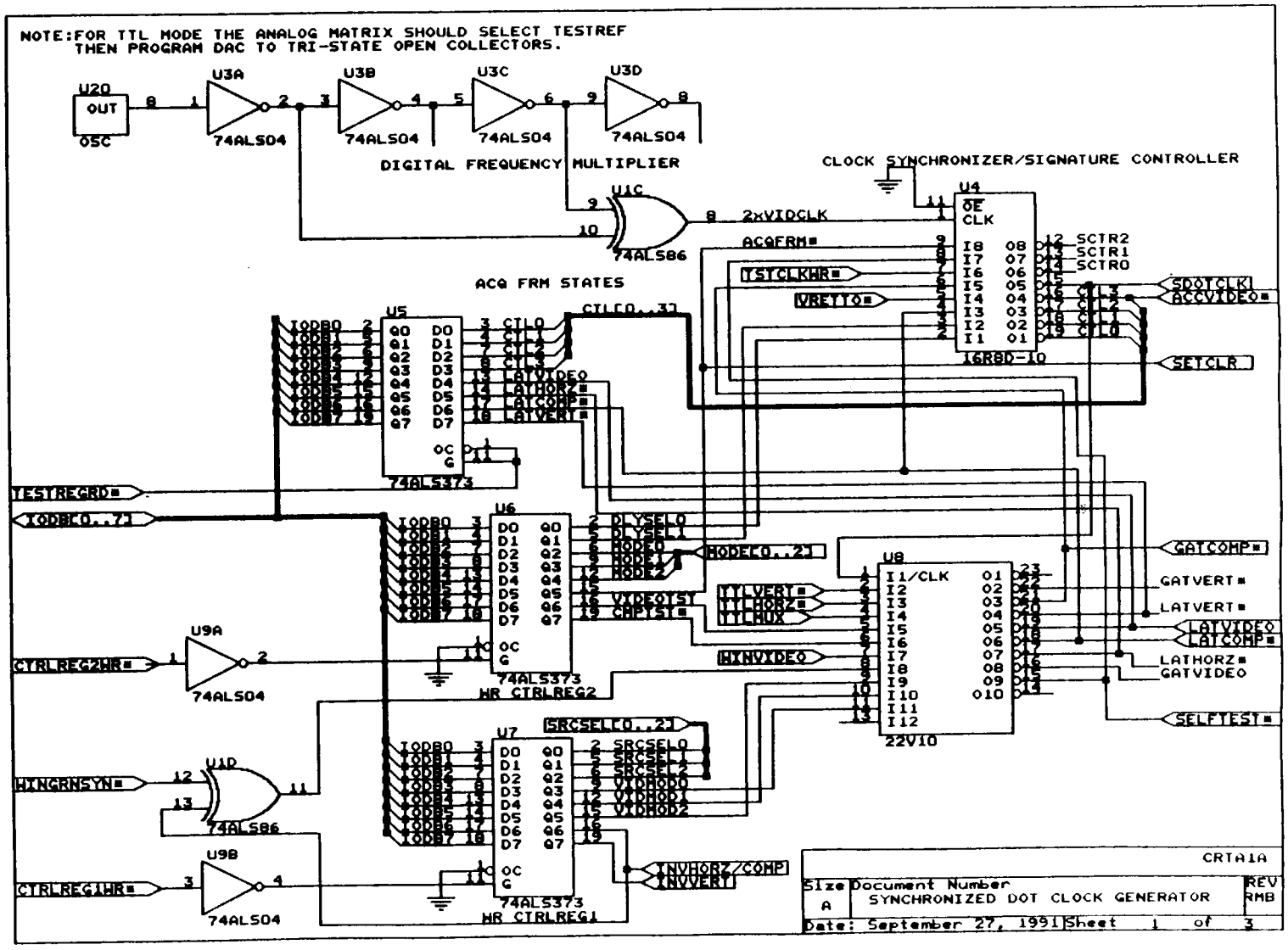
9. ADDS, Applied Digital Data System

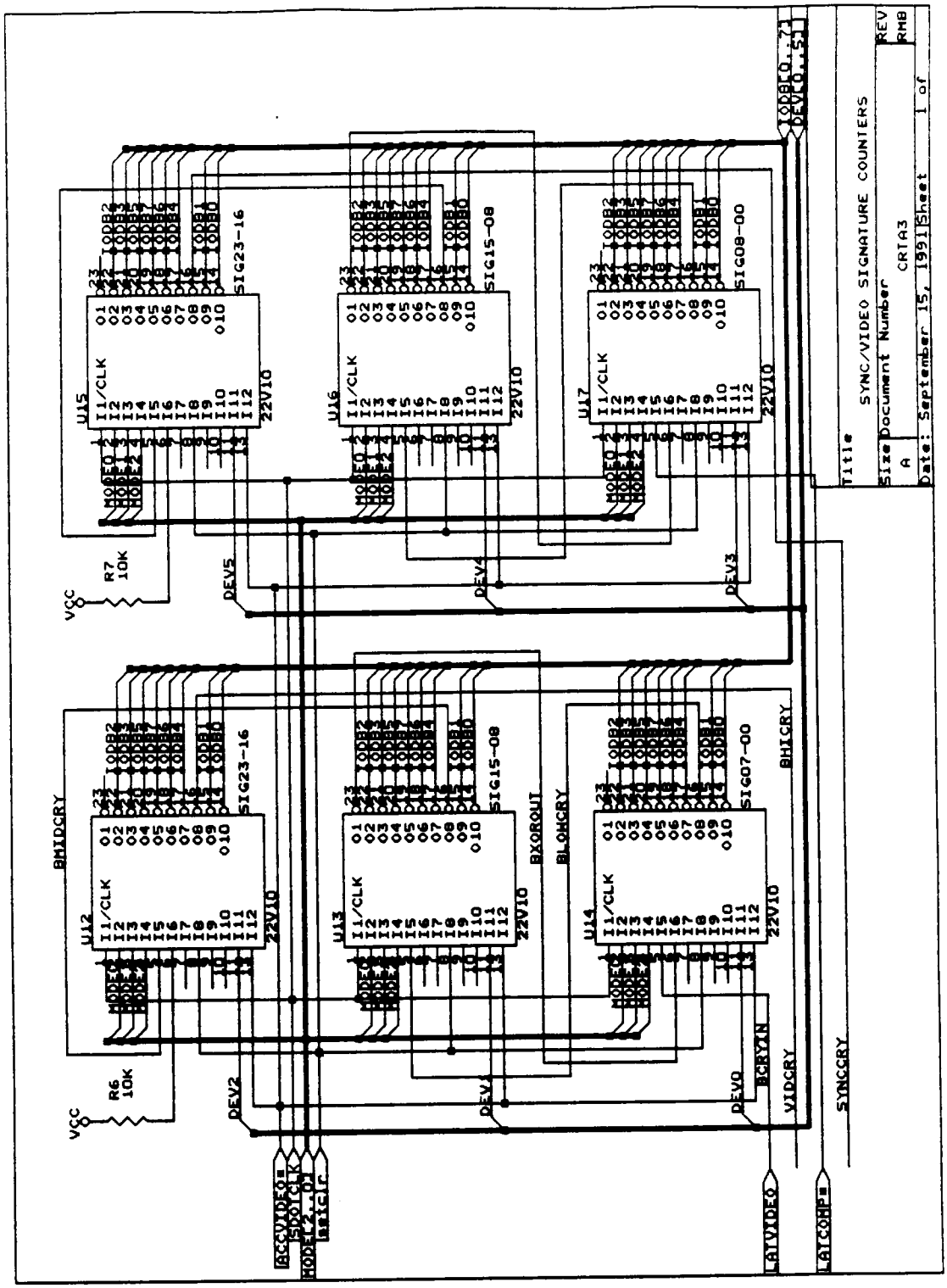
APPENDIX

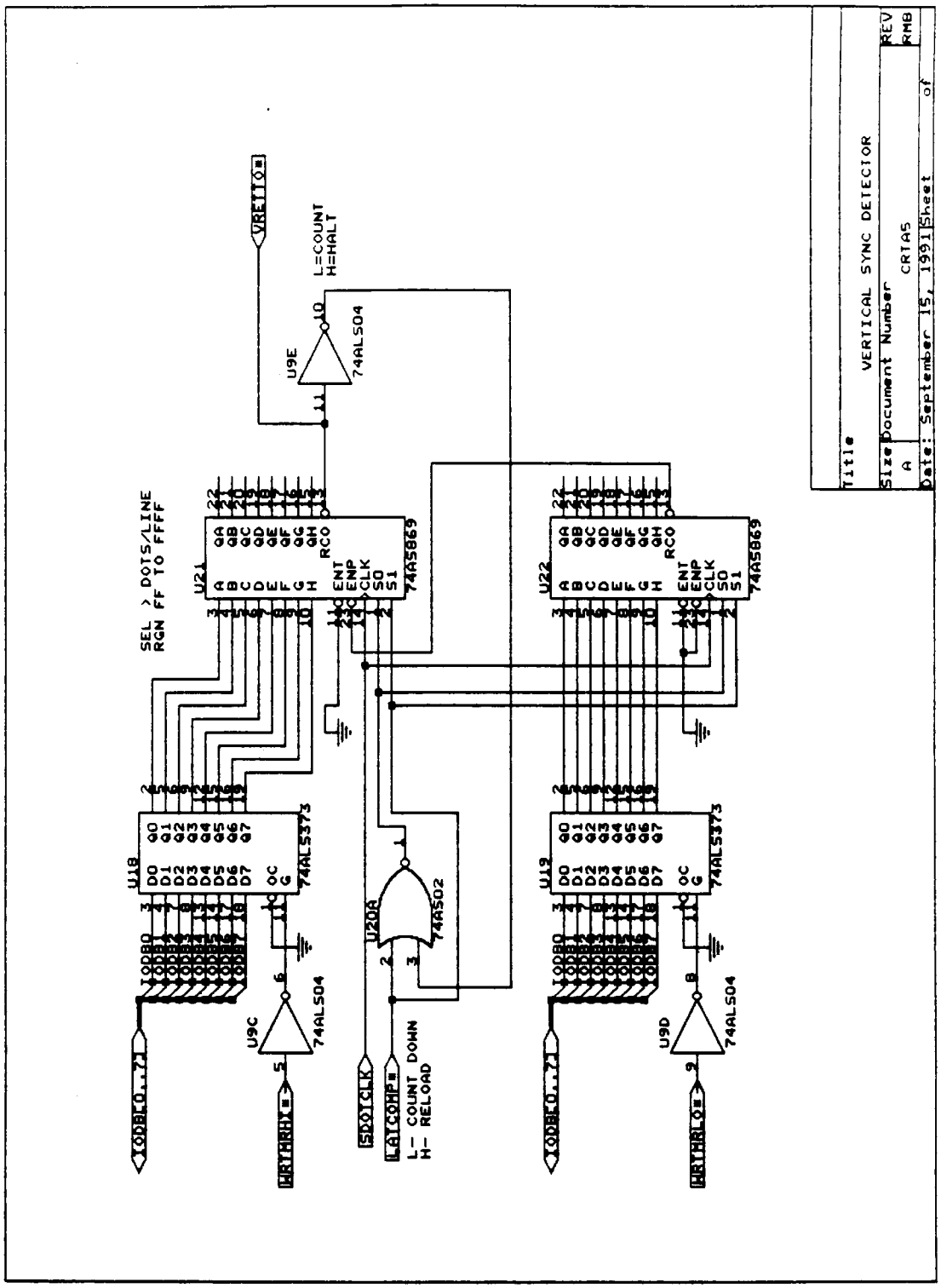
Analysis Board Schematic Diagrams


TIL VIDEO SIGNAL ROUTING
OFF-BOARD CONNECTORS
SYNCHRONIZED DOT CLOCK GEN.
SYNC/VIDEO SIGNATURE GENERATORS
VERTICAL SYNC DETECTOR
VERTICAL SYNC DETECTOR
TEXAS INSTRUMENTS
Title
VIDEO SIGNATURE CARD
Size
A
Document Number
CRTA
REV
MB
Date: September 28, 1991
Sheet
1 of

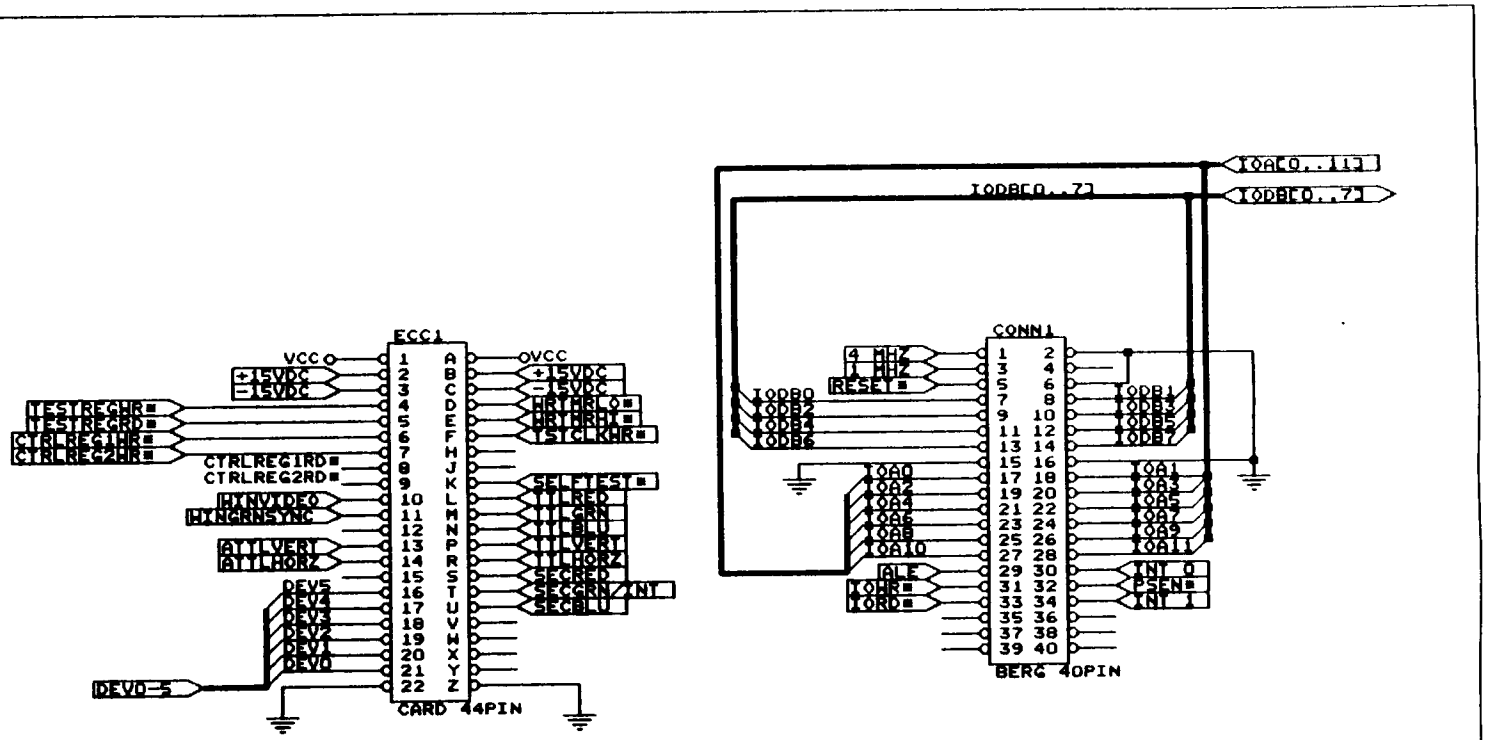

INVVERT
INVHORZ/COMP
ANALOG VIDEO INPUT
ATILVERT
ATILHORZ
TTL VIDEO INPUT
U10
74ALS86
U18
74ALS86
TTLVERT
TTLHORZ/TTLCMP
TTLMUX
U2
74AS151
SRCSELO-2
Size
A
Document Number
REV
Date: September 15, 1991
Sheet
of


NOTE:FOR TTL MODE THE ANALOG MATRIX SHOULD SELECT TESTREF
THEN PROGRAM DAC TO TRI-STATE OPEN COLLECTORS.
U20
OUT
OSC
U3A
U3B
U3C
U3D
74ALS04
DIGITAL FREQUENCY MULTIPLIER
U1C
74ALS86
2xVIDCLK
CLOCK SYNCHRONIZER/SIGNATURE CONTROLLER
U4
ACQFRM*
TSTCLKHR*
VRETTO*
SCTR2
SCTR1
SCTR0
SDOTCLK1
ACCVIDEO*
SETCLR
16R8D-10
ACQ FRM STATES
U5
CTLO..3]
LATVIDEO
LATHORZ*
LATCOMP*
LATVERT*
74ALS373
TESTREGRD*
IODBO..7]
U6
DLYSELO
DLYSEL1
MODEO
MODE1
MODE2
MODEO..2]
VIDEOTST
CMPTST*
TTLVERT*
TTLHORZ*
TTLAUX
HINVIDEO
74ALS373
HR CTRLREG2
U8
I1/CLK
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
O1
O2
O3
O4
O5
O6
O7
O8
O9
O10
22V10
GATCOMP*
GATVERT*
LATVERT*
LATVIDEO
LATCOMP*
LATHORZ*
GATVIDEO
SELFTEST*
U9A
CTRLREG2HR*
74ALS04
U1D
HINGRNSYN*
74ALS86
U7
SRCSELO..2]
SRCSELO
SRCSEL1
SRCSEL2
VIDMODO
VIDMOD1
VIDMOD2
74ALS373
HR CTRLREG1
U9B
CTRLREG1HR*
74ALS04
INVHORZ/COMP
INVVERT
CRTA1A
Size A
Document Number
SYNCHRONIZED DOT CLOCK GENERATOR
REV RMB
Date: September 27, 1991 Sheet 1 of 3

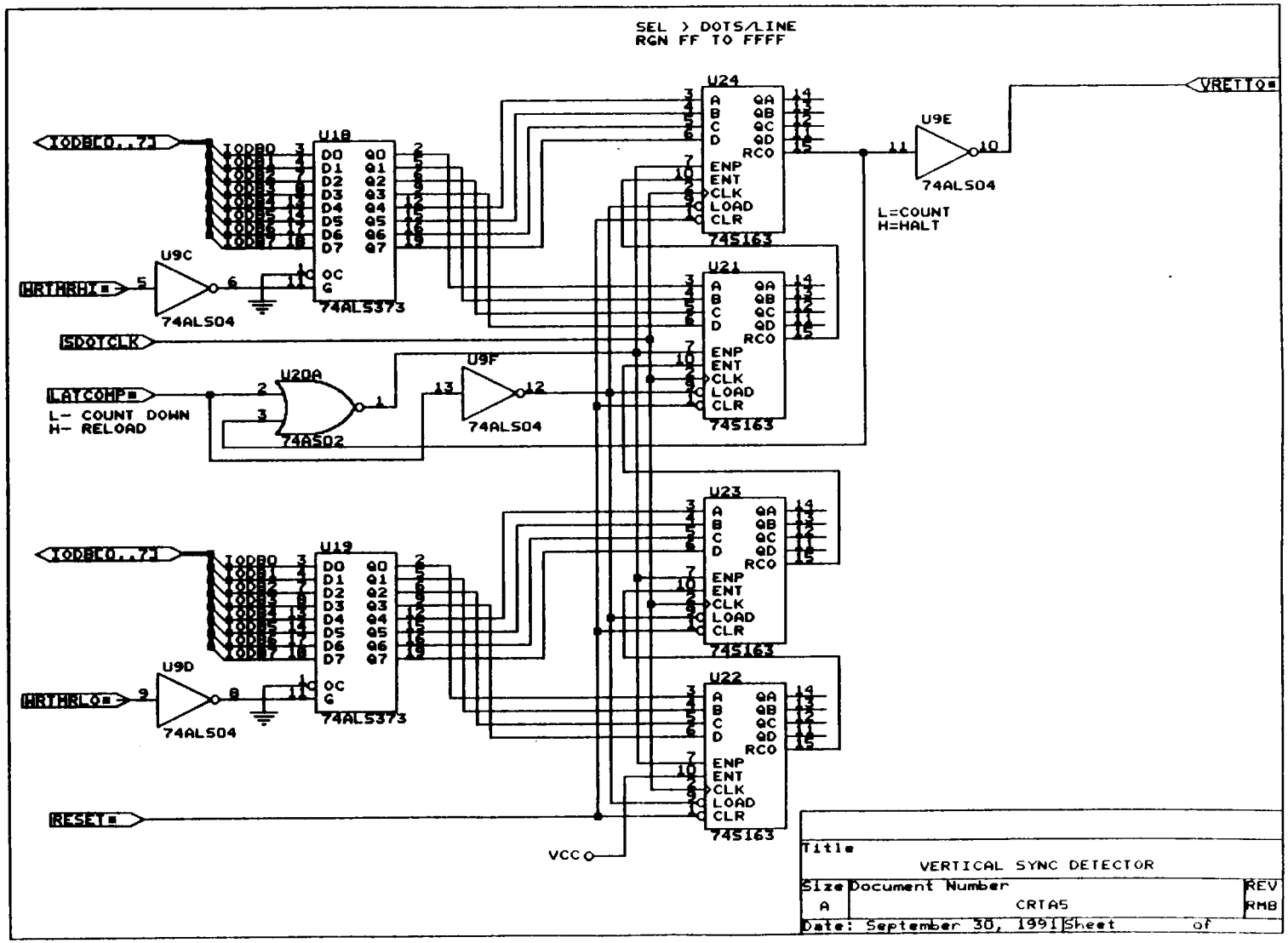

SYNC/VIDEO SIGNATURE COUNTERS
Size
A
Document Number
CRTA3
REV
RMB
Date: September 15, 1991
Sheet
1 of
U12
U13
U14
U15
U16
U17
22V10
SIG23-16
SIG15-08
SIG07-00
SIG08-00
R6 10K
R7 10K
VCC
BHIDCRY
BLOHCRY
BXOROUT
BHICRY
BCRYIN
VIDCRY
SYNCCRY
DEV0
DEV1
DEV2
DEV3
DEV4
DEV5
LATVIDEO
LATCOMP

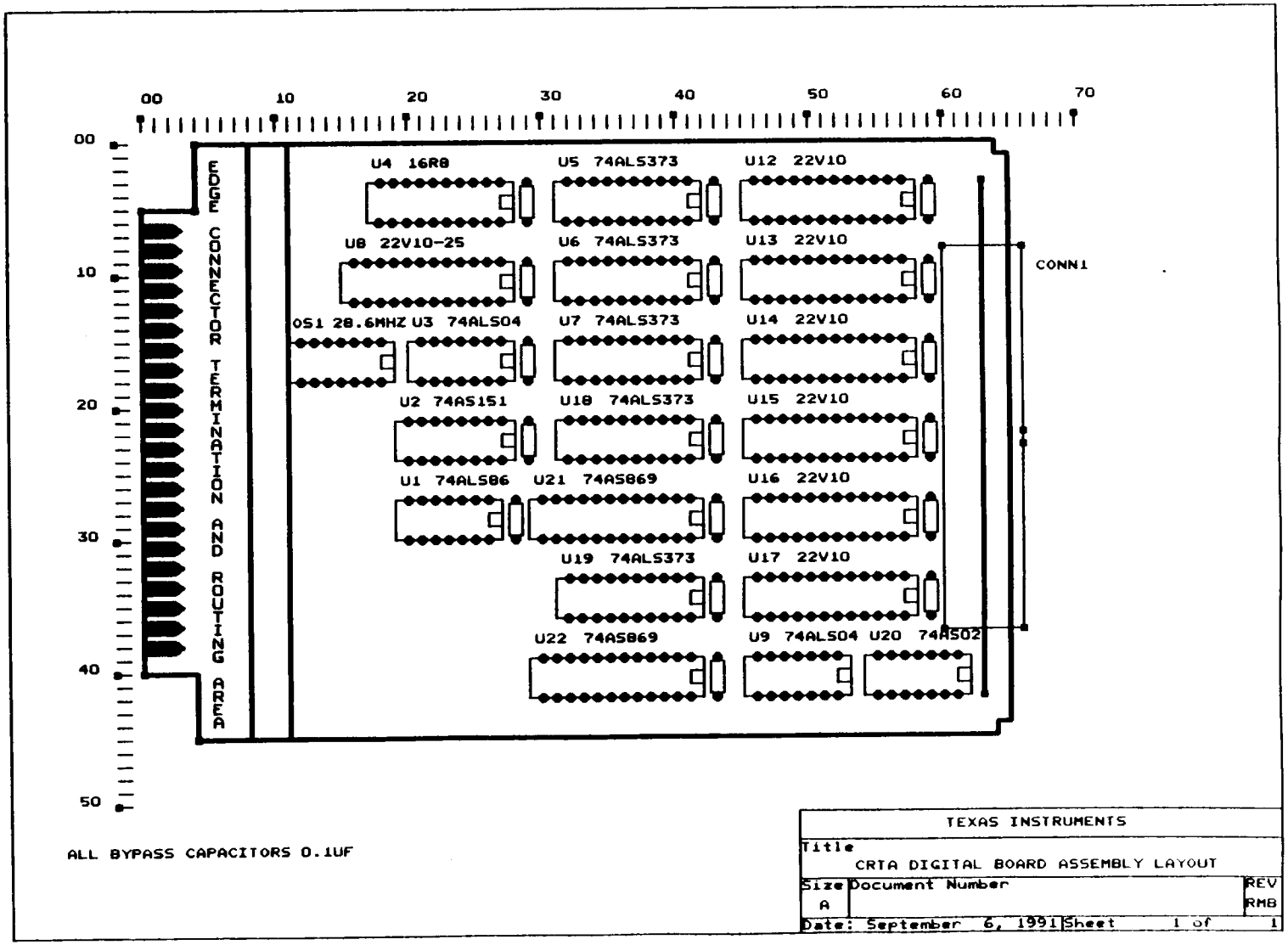


VERTICAL SYNC DETECTOR
Title
Size
A
Document Number
CRTA5
REV
RMB
Date: September 15, 1991
Sheet
of
VRETTO*
L=COUNT
H=HALT
U9E
74ALS04
SEL > DOTS/LINE
RGN FF TO FFFF
U21
74AS869
U22
74AS869
U18
74ALS373
U19
74ALS373
U20A
74AS02
U9C
74ALS04
U9D
74ALS04
IODBO..7
WRTMRHI*
SDOTCLK
LATCOMP*
L- COUNT DOWN
H- RELOAD
IODBO..7
WRTMRLO*



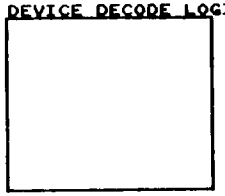
TEXAS INSTRUMENTS		
Title: OFF-BOARD CONNECTORS		
Size: A	Document Number: CRTA6	REV: RMB
Date: September 15, 1991	Sheet of	


SEL > DOTS/LINE
RGN FF TO FFFF
IODB[0..7]
U18
74ALS373
U9C
74ALS04
WRTMRHI=
SDOTCLK
LATCOMP=
L- COUNT DOWN
H- RELOAD
U20A
74AS02
U9F
74ALS04
U19
74ALS373
U9D
74ALS04
WRTMRLO=
RESET=
VCC
U24
U21
U23
U22
74S163
U9E
74ALS04
L=COUNT
H=HALT
VRETTO=
Title
VERTICAL SYNC DETECTOR
Size
A
Document Number
CRTA5
REV
RMB
Date: September 30, 1991
Sheet
of

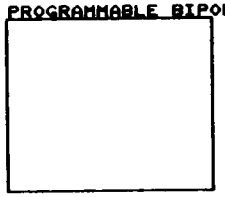

00
10
20
30
40
50
60
70
EDGE CONNECTOR TERMINATION AND ROUTING AREA
U4 16R8
U5 74ALS373
U12 22V10
U8 22V10-25
U6 74ALS373
U13 22V10
CONN1
OS1 28.6MHZ U3 74ALS04
U7 74ALS373
U14 22V10
U2 74AS151
U18 74ALS373
U15 22V10
U1 74ALS86
U21 74AS869
U16 22V10
U19 74ALS373
U17 22V10
U22 74AS869
U9 74ALS04 U20 74AS02
ALL BYPASS CAPACITORS 0.1UF
TEXAS INSTRUMENTS
Title
CRTA DIGITAL BOARD ASSEMBLY LAYOUT
Size A
Document Number
REV RMB
Date: September 6, 1991
Sheet 1 of 1

Decoder Board Schematic Diagrams

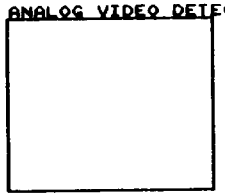
DEVICE DECODE LOGIC



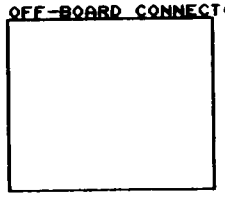
PROGRAMMABLE BIPOLAR ANALOG THRESHOLDS



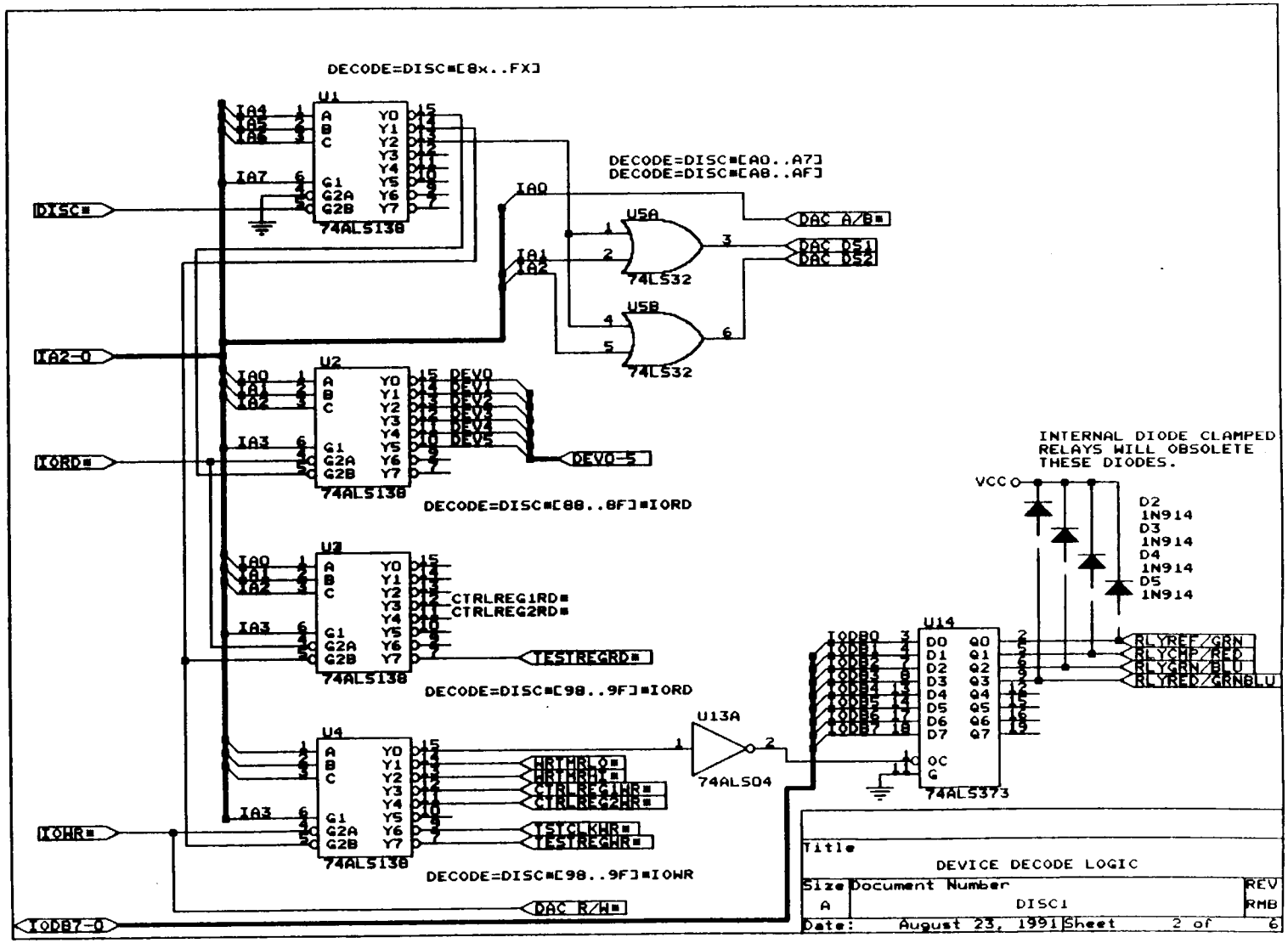
ANALOG VIDEO DETECTION

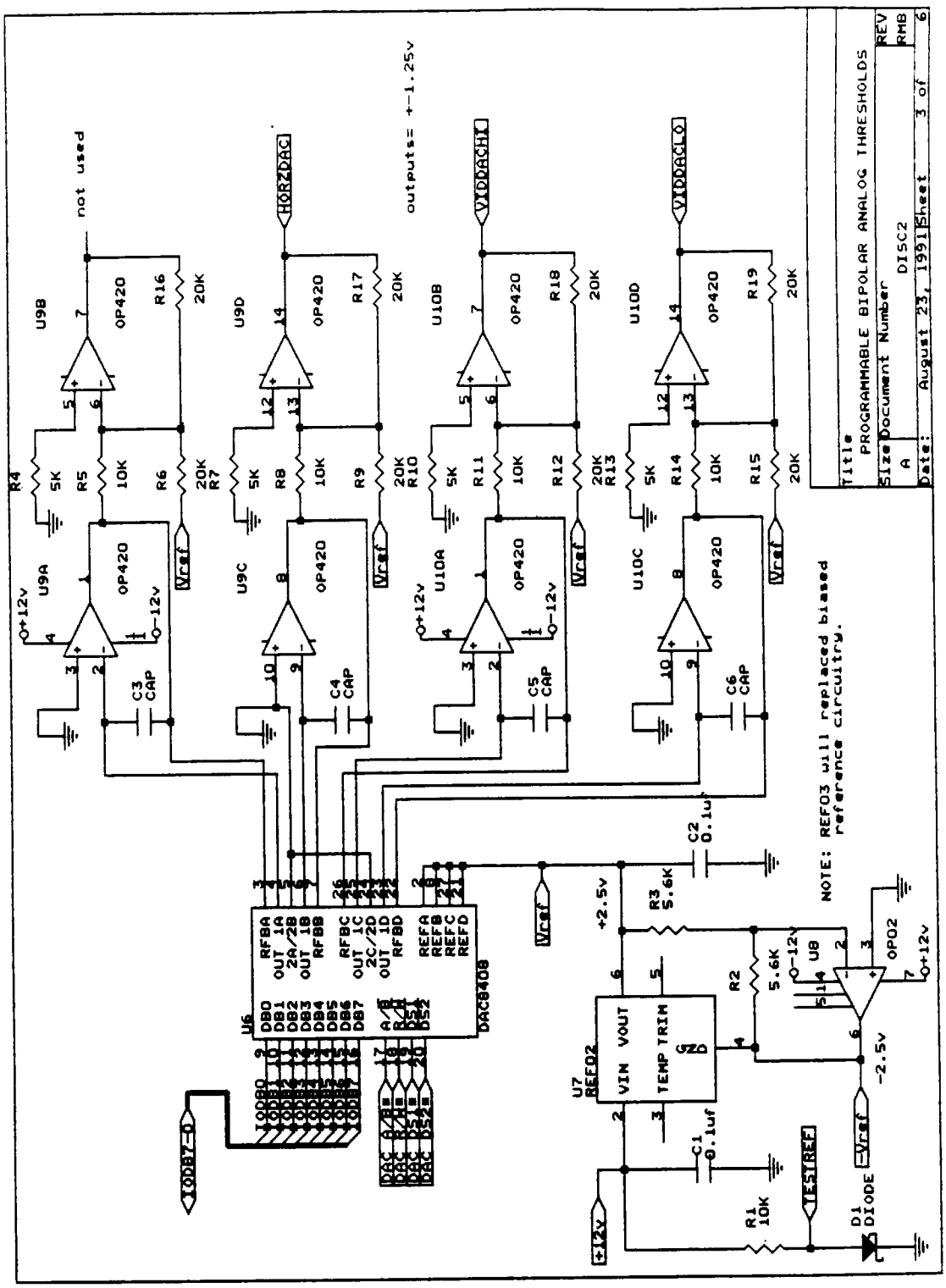


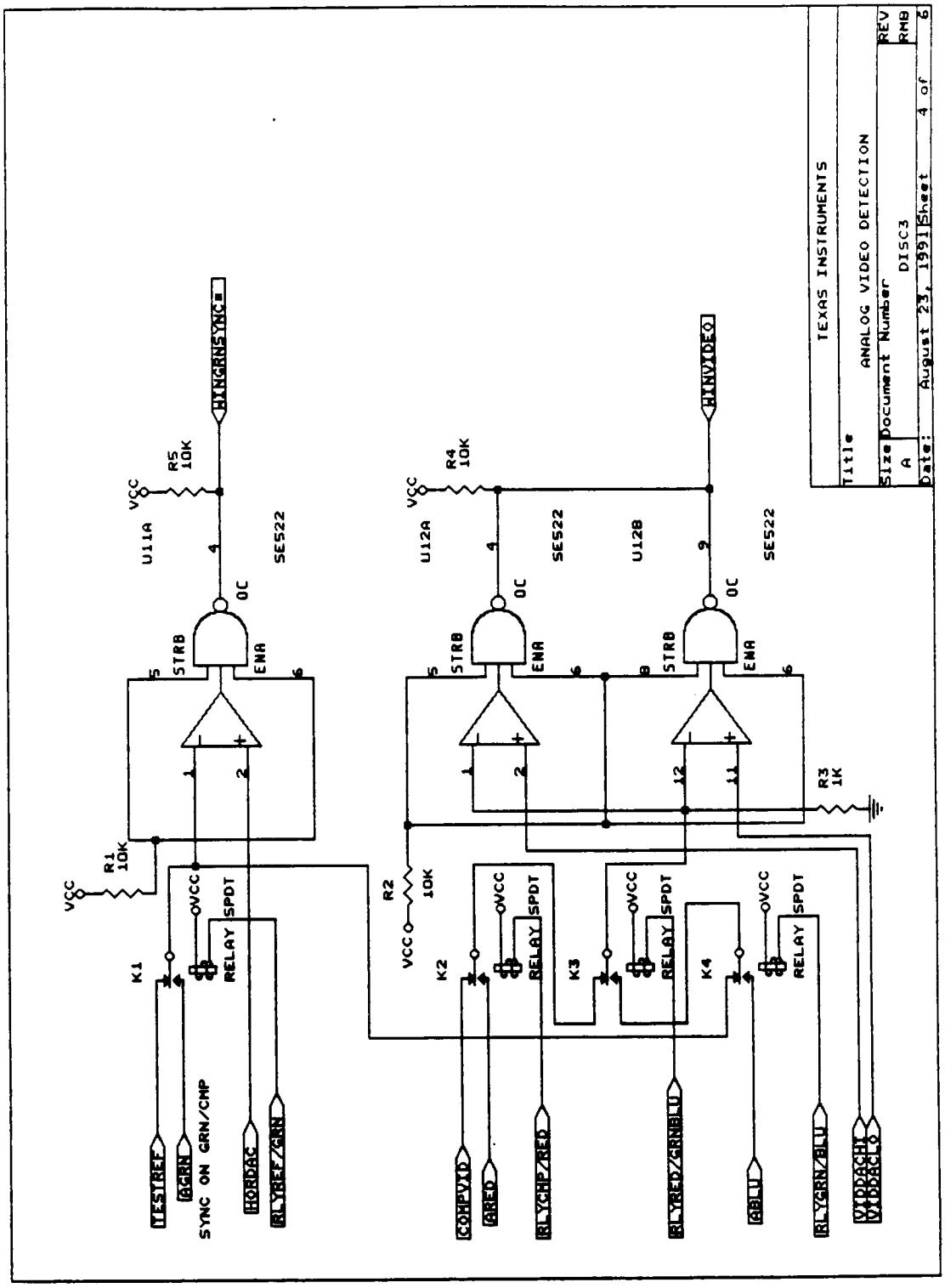
OFF-BOARD CONNECTORS

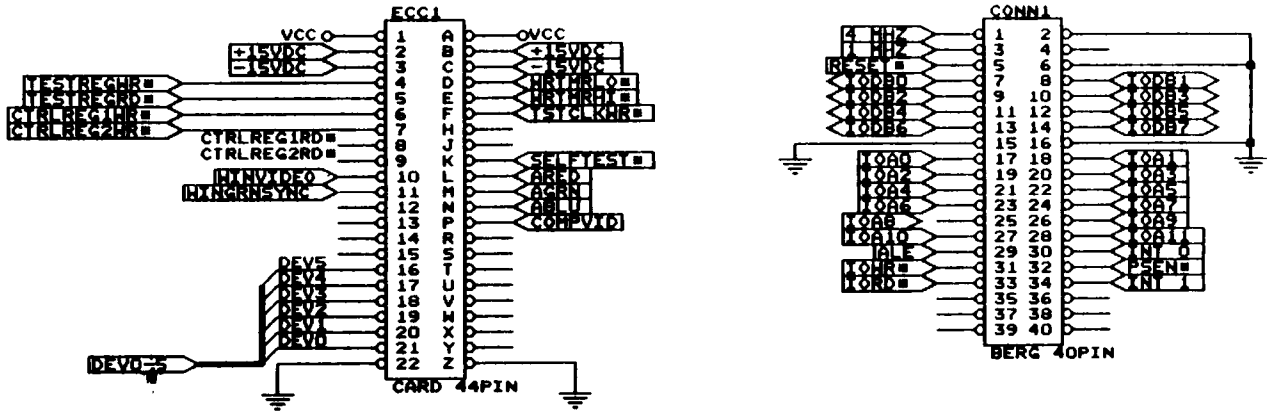


TEXAS INSTRUMENTS		
Title: DISCRETE DECODE BOARD		
Size: A	Document Number: CRTA	REV: RMB
Date: August 24, 1991	Sheet 1 of 6	

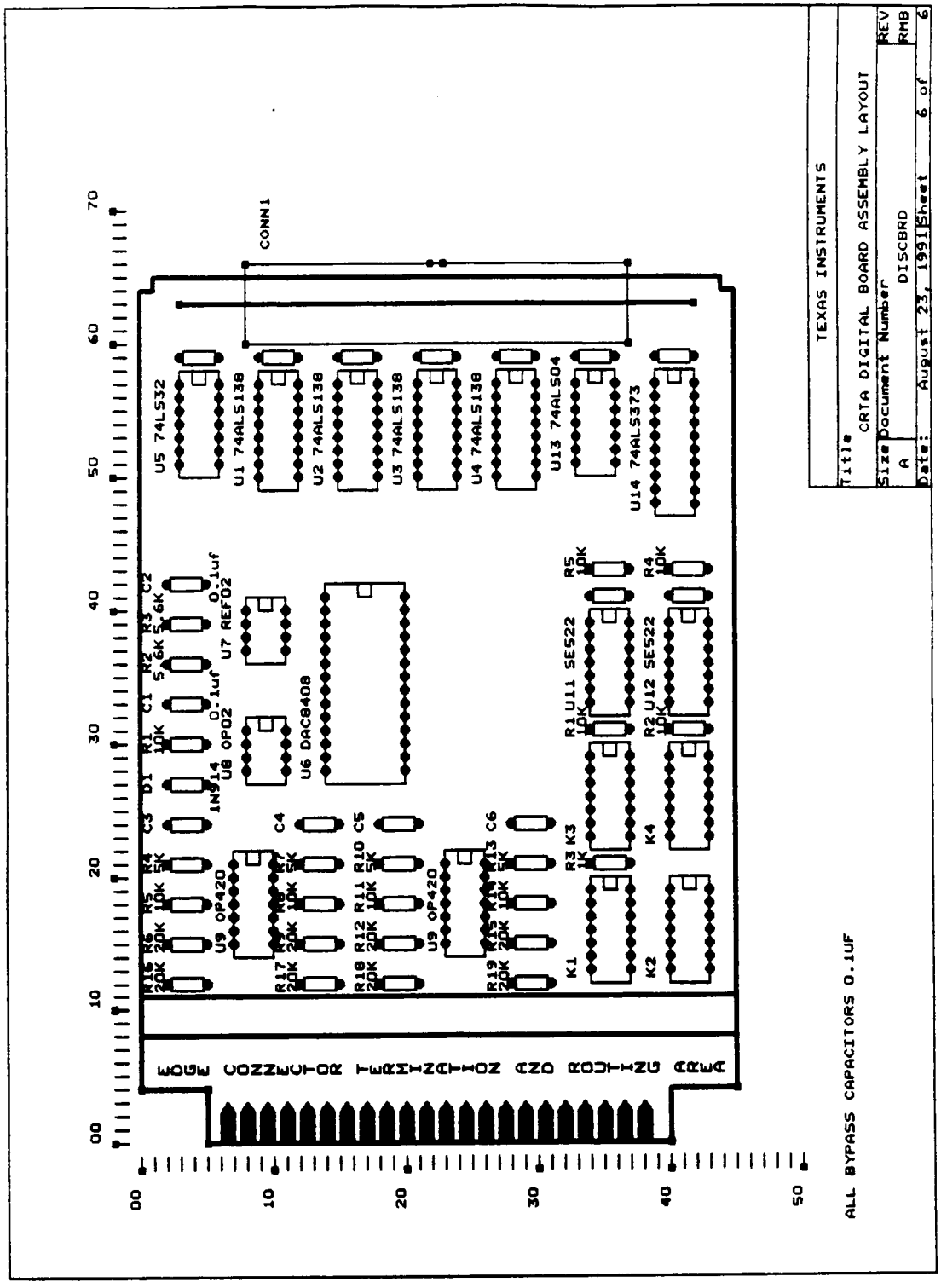

DECODE=DISC=[8x..FX]
U1
74ALS138
DISC=
IA7
IA4
IA5
IA6
DECODE=DISC=[A0..A7]
DECODE=DISC=[A8..AF]
IA0
IA1
IA2
U5A
U5B
74LS32
DAC A/B=
DAC DS1
DAC DS2
IA2-0
U2
DEV0
DEV1
DEV2
DEV3
DEV4
DEV5
DEV0-5
IORD=
IA3
DECODE=DISC=[80..8F]=IORD
U3
CTRLREG1RD=
CTRLREG2RD=
TESTREGRD=
DECODE=DISC=[90..9F]=IORD
U4
HRTHRL0=
HRTHRH1=
CTRLREG1HR=
CTRLREG2HR=
TSTCLKHR=
TESTREGHR=
IOWR=
DECODE=DISC=[90..9F]=IOWR
U13A
74ALS04
DAC R/W=
IODB7-0
INTERNAL DIODE CLAMPED
RELAYS WILL OBSOLETE
THESE DIODES.
VCC
D2
1N914
D3
1N914
D4
1N914
D5
1N914
U14
74ALS373
RLYREF/GRN
RLYGRB/RED
RLYGRN/BLU
RLYRED/GRNBLU
Title
DEVICE DECODE LOGIC
Size
A
Document Number
DISC1
REV
RMB
Date:
August 23, 1991
Sheet
2 of
6


not used
HORZDAC
VIDDACHI
VIDDACLO
outputs= +-1.25v
U9B
U9A
U9D
U9C
U10B
U10A
U10D
U10C
OP420
R16
R17
R18
R19
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
5K
10K
20K
+12v
-12v
Vref
C3 CAP
C4 CAP
C5 CAP
C6 CAP
U6
DAC8408
RFBA
OUT 1A
2A/2B
OUT 1B
RFBB
RFBC
OUT 1C
2C/2D
OUT 1D
RFBD
REFA
REFB
REFC
REFD
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
IODB7-0
U7
REF02
VIN
VOUT
TEMP TRIM
GND
C1 0.1uf
C2 0.1uf
R1 10K
R2 5.6K
R3 5.6K
+2.5v
-2.5v
U8
OP02
TESTREF
D1 DIODE
NOTE: REF03 will replaced biased reference circuitry.
Title
PROGRAMMABLE BIPOLAR ANALOG THRESHOLDS
Size
A
Document Number
DISC2
REV
RMB
Date: August 23, 1991
Sheet 3 of 6


TEXAS INSTRUMENTS
ANALOG VIDEO DETECTION
Title
Size A
Document Number DISC3
REV RMB
Date: August 23, 1991 Sheet 4 of 6
HINGRNSYNC
HINVIDEO
U11A
U12A
U12B
SE522
R1 10K
R2 10K
R3 1K
R4 10K
R5 10K
VCC
STRB
ENA
OC
K1
K2
K3
K4
RELAY
SPDT
TESTREF
AGRN
SYNC ON GRN/CMP
HORDAC
RLYREF/GRN
COMPVID
ARED
RLYCMP/RED
RLYRED/GRNBLU
ABLU
RLYGRN/BLU
VIDDACH
VIDDACL

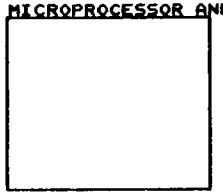


TEXAS INSTRUMENTS		
Title: OFF-BOARD CONNECTORS		
Size: A	Document Number: DISC4	REV: RMB
Date: August 23, 1991	Sheet 5 of 6	

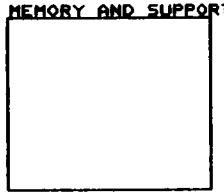

TEXAS INSTRUMENTS
Title
CRTA DIGITAL BOARD ASSEMBLY LAYOUT
Size Document Number REV
A DISCBRD RMB
Date: August 23, 1991 Sheet 6 of 6
CONN1
U1 74ALS138
U2 74ALS138
U3 74ALS138
U4 74ALS138
U5 74LS32
U6 DAC8408
U7 REF02
U8 OP02
U9 OP420
U9 OP420
U11 SE522
U12 SE522
U13 74ALS04
U14 74ALS373
K1
K2
K3
K4
ALL BYPASS CAPACITORS 0.1UF

BASIC-52 CPU Board Schematic Diagrams

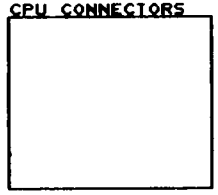
MICROPROCESSOR AND CONTROL



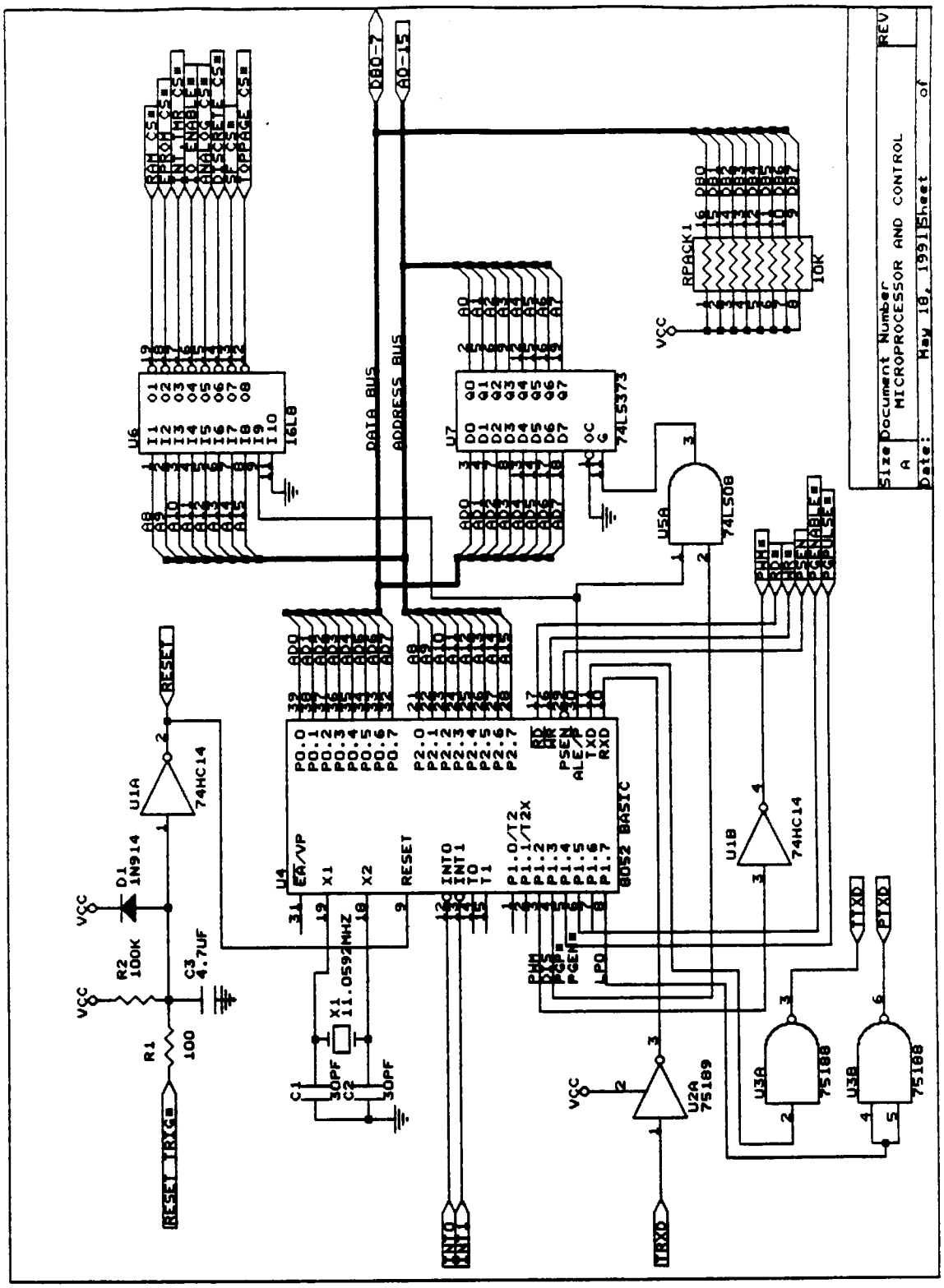
MEMORY AND SUPPORT DEVICES

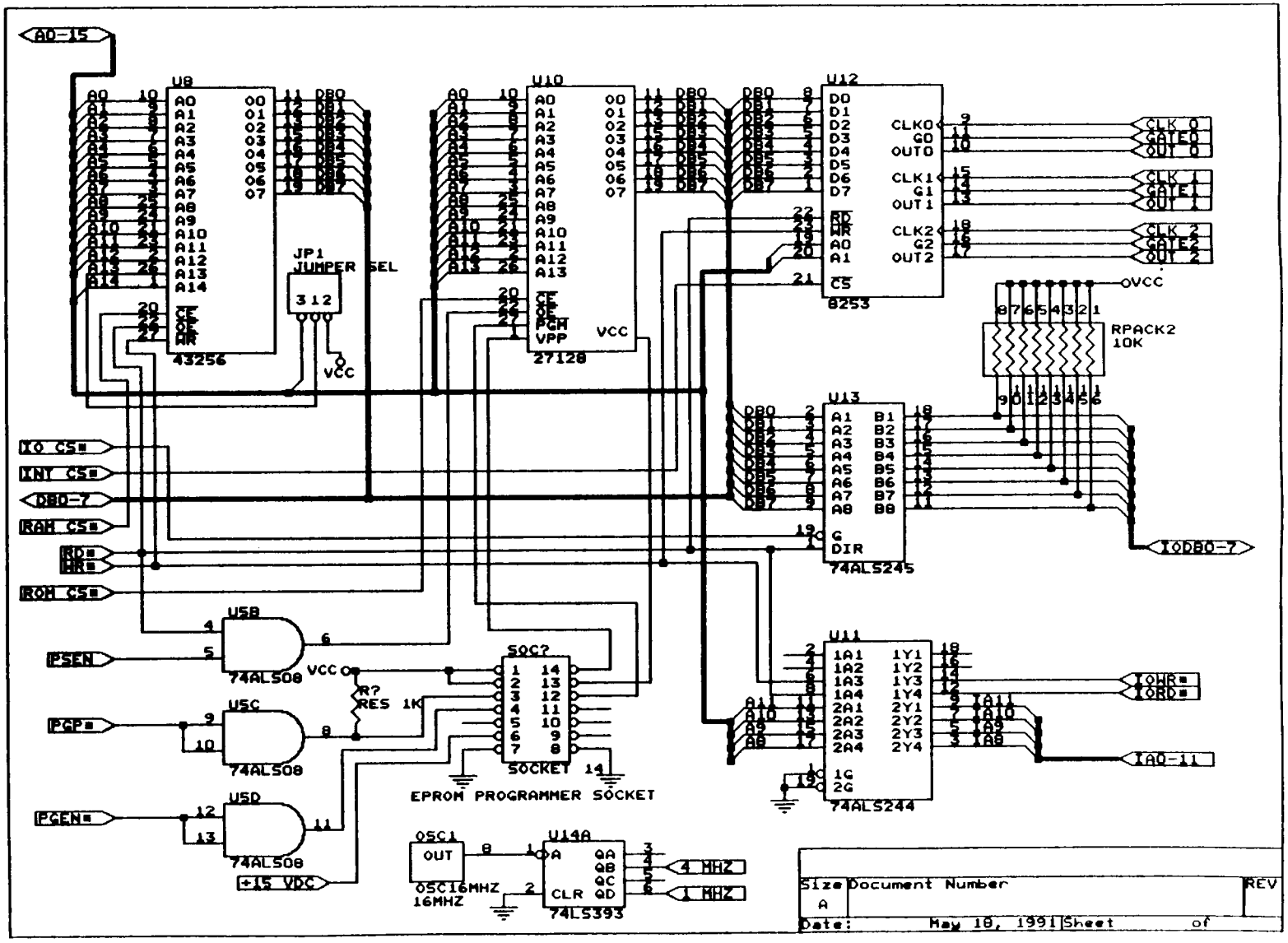


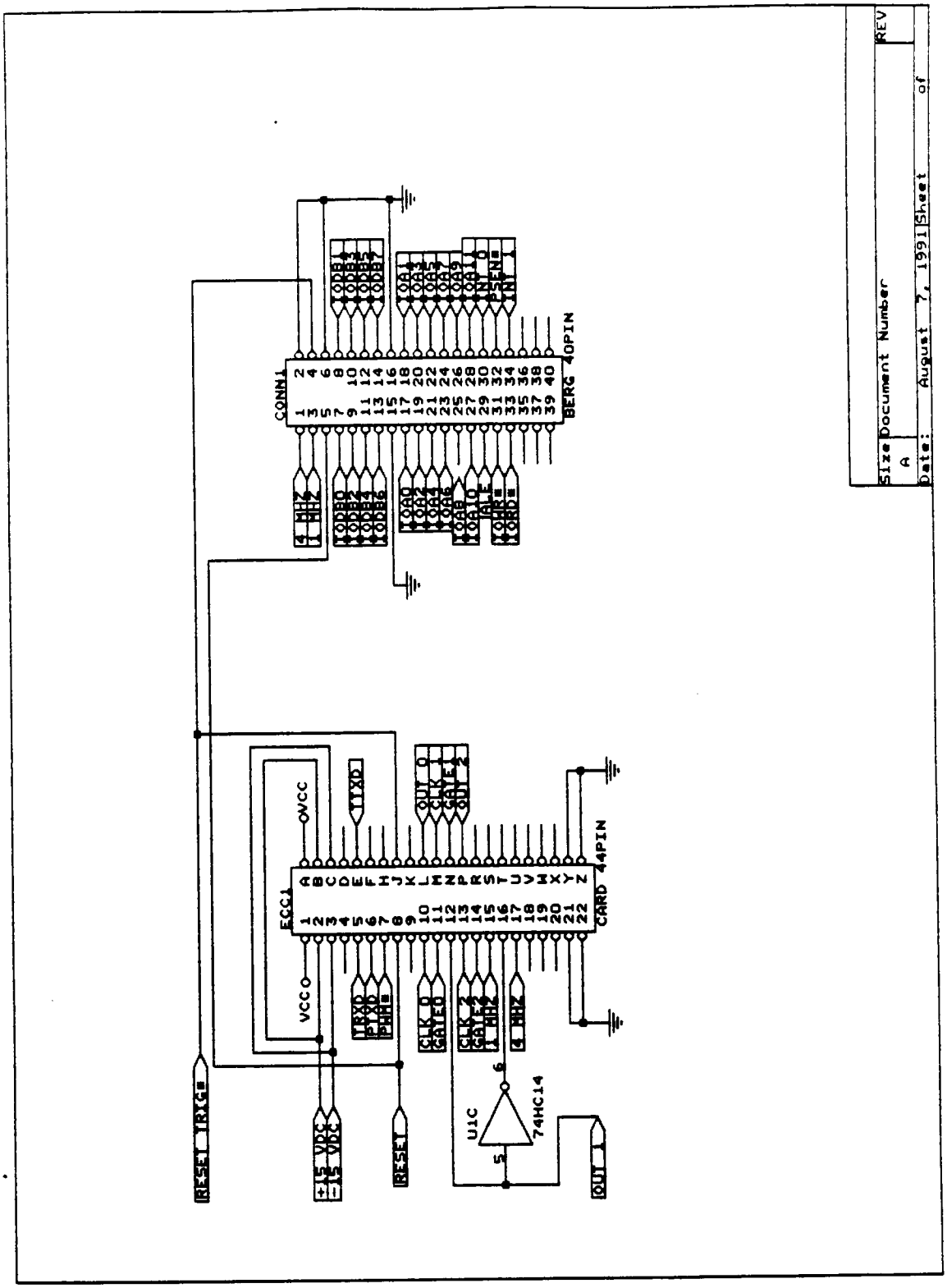
CPU CONNECTORS



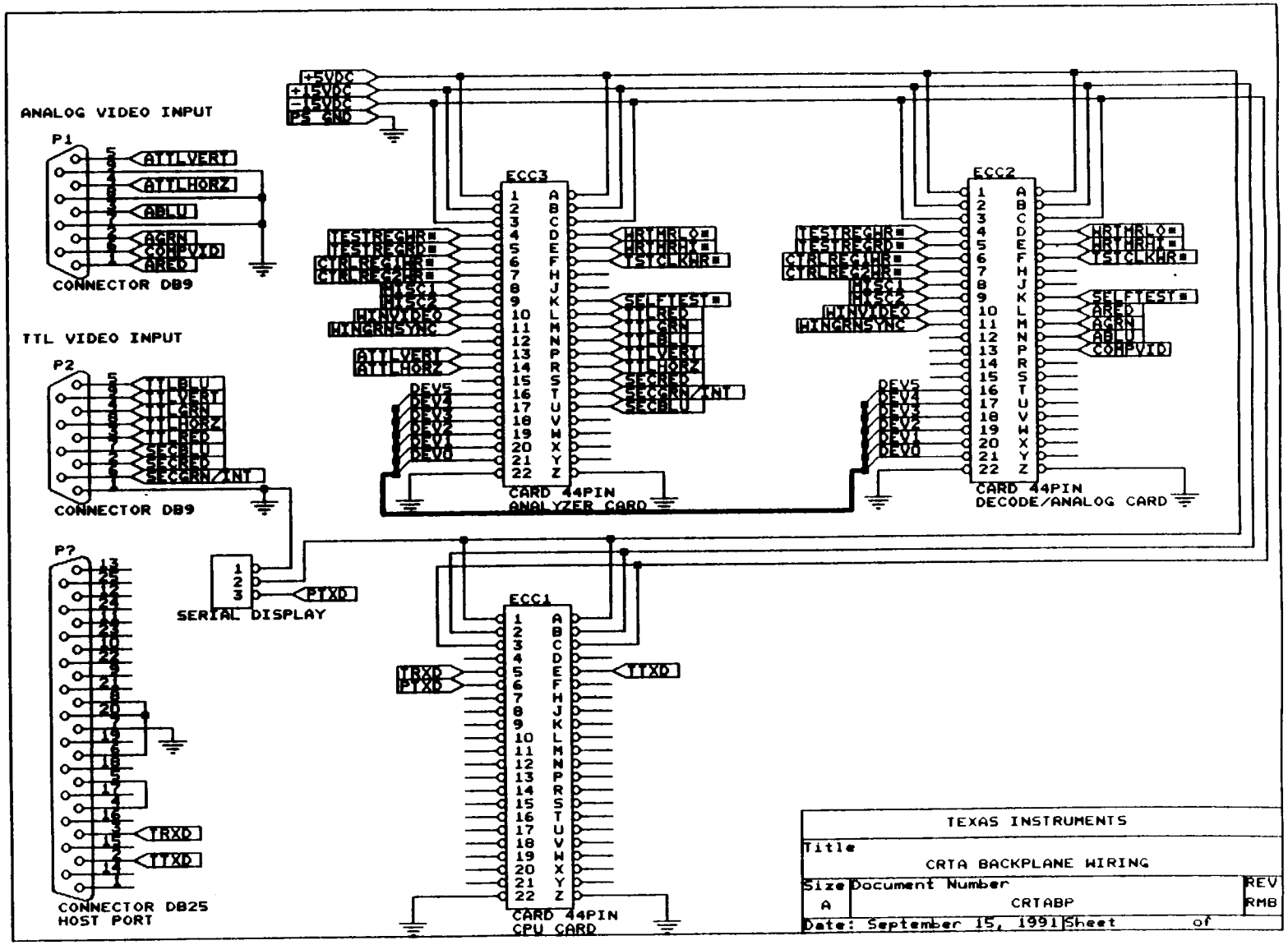
TEXAS INSTRUMENTS		
Title: BASIC-52 CPU BOARD		
Size: A	Document Number: CPU	REV: RMB
Date: October 24, 1991	Sheet 1 of 4	


Size
A
Document Number
MICROPROCESSOR AND CONTROL
REV
Date: May 18, 1991
Sheet of
DATA BUS
ADDRESS BUS
DB0-7
AD-15
U6
16L8
U7
74LS373
U5A
74LS08
U4
8052 BASIC
U1A
74HC14
U1B
74HC14
U2A
75189
U3A
75188
U3B
75188
RPACK1
10K
D1
1N914
R1
100
R2
100K
C3
4.7UF
X1
11.0592MHZ
C1
30PF
C2
30PF
VCC
RESET
RESET TRIG
INT0
INT1
TRXD
TTXD
PTXD


A0-15
U8
43256
JP1
JUMPER SEL
3 1 2
VCC
U10
27128
VPP
VCC
U12
8253
CLK 0
GATE 0
OUT 0
CLK 1
GATE 1
OUT 1
CLK 2
GATE 2
OUT 2
OVCC
RPACK2
10K
U13
74ALS245
IODB0-7
IO CS*
INT CS*
DB0-7
RAM CS*
RD*
WR*
ROM CS*
U5B
74ALS08
PSEN
U5C
74ALS08
PGP*
U5D
74ALS08
PGEN*
+15 VDC
R?
RES 1K
SOC?
SOCKET 14
EPROM PROGRAMMER SOCKET
U11
74ALS244
IOWR*
IORD*
IA0-11
OSC1
OUT
OSC16MHZ
16MHZ
U14A
74LS393
4 MHZ
1 MHZ
Size
A
Document Number
REV
Date: May 18, 1991
Sheet of


CONN1
BERG 40PIN
ECC1
CARD 44PIN
VCC
TIXD
OUT 0
CLK 0
GATE 1
OUT 2
TRXD
PTXD
PHHS
CLK 2
GATE 2
1 MHZ
4 MHZ
U1C
74HC14
RESET TRIG#
+15 VDC
-15 VDC
RESET
OUT 1
Size
A
Document Number
REV
Date: August 7, 1991
Sheet
of

ANALYSIS SYSTEM BACKPLANE


ANALOG VIDEO INPUT
P1
ATTLVERT
ATTLHORZ
ABLU
AGRN
COMPVID
ARED
CONNECTOR DB9
TTL VIDEO INPUT
P2
TTLBLU
TTLVERT
TTLGRN
TTLHORZ
TTLRED
SECBLU
SECRED
SECGRN/INT
CONNECTOR DB9
P?
TRXD
TTXD
CONNECTOR DB25
HOST PORT
SERIAL DISPLAY
PTXD
+5VDC
+15VDC
-15VDC
PS GND
ECC3
CARD 44PIN
ANALYZER CARD
ECC2
CARD 44PIN
DECODE/ANALOG CARD
ECC1
CARD 44PIN
CPU CARD
TEXAS INSTRUMENTS
Title
CRTA BACKPLANE WIRING
Size
A
Document Number
CRTABP
REV
RMB
Date: September 15, 1991
Sheet
of

BASIC-52 Video Analysis Program Listing

```
[
[        ****************************************************
[        ********        VIDTST01   8/16/91         *********
[        ********                                   *********
[        ****************************************************
[
[
[
[
[
[  AUTHOR/HWD DESIGNER: ROBERT M. BASS
[
[
[  CPU MEMORY MAP PORT ASSIGNMENTS[
[
[                   BASIC-52 DEFINED MEMORY MAP
[                  PROGRAM AREA        DATA AREA
[
[      0000-1FFF   INTERNAL ROM        EXTERNAL RAM
[                  INTERPRETER
[      2000-7FFF   EXTERNAL ROM        EXTERNAL RAM
[      8000-DFFF   EXTERNAL ROM        EXTERNAL ROM/RAM
[      E000-FFFF   USER IO/ROM         USER IO/ROM
[
[
[
[          ACTUAL BASIC-52 CPU IMPLIMENTATION OF MEMORY MAP
[                   DECODE PAL=ADDEC32
[
[      ADDRESS     PROGRAM AREA        DATA AREA
[                    "PSEN*"           "RD* WR*"
[      0000-1FFF   INTERNAL ROM        EXTERNAL RAM 8K
[                  INTERPRETER
[      2000-5FFF   EMPTY               EXTERNAL RAM 16K
[      6000-7FFF   EMPTY               EXTERNAL RAM 8K
[      8000-BFFF   EXTERNAL ROM 16K    EXTERNAL ROM 16K    DUAL MAPPED
[      C000-DFFF   EXTERNAL ROM  0K    EXTERNAL ROM  0K    DUAL MAPPED
[      E00X-E1FX   EMPTY               INTERVAL TIMER
[                                       X=0 LOAD COUNTER NO.0
[                                       X=1 LOAD COUNTER NO.1
[                                       X=2 LOAD COUNTER NO.2
[                                       X=3 MODE CONTROL WORD
[      E200-E2FF   EMPTY               ANALOG CARDS
[      E300-E3FF   EMPTY               ANALOG CARDS
[      E400-E4FF   EMPTY               DISCRETE CARDS
[      E500-E5FF                       DISCRETE CARDS OVERMAP
[      E600-E7FF   EMPTY               EMPTY
[      E800-EFFF   S.F.MEMORY AREA     S.F. MEMORY AREA
```

```
F000-FFFF   TOPPAGE              TOPPAGE MEMORY


        ADDRESS STRUCTURE FOR DISCRETE CARDS
   E400 +                        DISCRETE BASE ADDRESS
         000x-00Fx +             SUB-ADDRESS OF CARD
                    0000-0003    BYTE ADDRESS ON CARD

        ADDRESS STRUCTURE FOR ANALOG CARDS
   E400 +                        ANALOG BASE ADDRESS
         000x-00Fx +             SUB-ADDRESS OF CARD
                    0000-000F    BYTE ADDRESS ON CARD

        ADDRESS STRUCTURE FOR DAC/PORT CARD
   E400 +                        DISCRETE BASE ADDRESS
         008x-00Ax +             SUB-ADDRESS OF CARD
                    0000-000F    BYTE ADDRESS ON CARD


     SPECIFIC DISCRETE DAC/PORT ADDRESS MAP
    DECODE=DISC*[88..8F]*IORD
        E400+88  R/   SIGNATURE DEVICE 0
                       DATA= SYNC SIGNATURE LOW BYTE
        E400+89  R/   SIGNATURE DEVICE 1
                       DATA= SYNC SIGNATURE MID BYTE
        E400+8A  R/   SIGNATURE DEVICE 2
                       DATA= SYNC SIGNATURE HIGH BYTE
        E400+8B  R/   SIGNATURE DEVICE 3
                       DATA= VIDEO SIGNATURE LOW BYTE
        E400+8C  R/   SIGNATURE DEVICE 4
                       DATA= VIDEO SIGNATURE MID BYTE
        E400+8D  R/   SIGNATURE DEVICE 5
                       DATA= VIDEO SIGNATURE HIGH BYTE
        E400+8E  R/   UNUSED 6
        E400+8F  R/   UNUSED 7

    DECODE=DISC*[98..9F]*IORD
        E400+98  R/   UNUSED 0
        E400+99  R/   UNUSED 1
        E400+9A  R/   UNUSED 2
        E400+9B  R/   CTRLREG1RD
                       DB7..0=not defined
        E400+9C  R/   CTRLREG2RD
                       DB7..0=not defined
        E400+9D  R/   UNUSED 5
        E400+9E  R/   UNUSED 6
        E400+9F  R/   TESTREGRD
                       DB3..0=CTL3..0 ACQUIRE FRAME STATES
                       DB4=LATHORZ    LATCHED HORIZONAL SYNC PIPELINE
                       DB5=LATVIDEO   LATCHED VIDEO PIPELINE
                       DB6=LATCOMP    LATCHED COMPOSITE SYNC
                       DB7=LATVERT    LATCHED VERTICAL SYNC
```

```
[
[          DECODE=DISC*[98..9F]*IOWR
[              E400+98   /W  RLYSET RELAY CONTROL SET
[                            DB0=RLYREF/GRN
[                            DB1=RLYCMP/RED
[                            DB2=RLYGRN/BLU
[                            DB3=RLYRED/GRNBLU
[                            DB4-7=not defined
[              E400+99   /W  WRTMRLO
[                            DATA= VERTICAL SYNC TIMER LOW BYTE
[              E400+9A   /W  WRTMRHI
[                            DATA= VERTICAL SYNC TIMER HIGH BYTE
[              E400+9B   /W  CTRLREG1WR
[                            DB2..0=SRCSEL2..0  TTL VIDEO MULTIPLEXER CONTROL
[                            DB5..3=VIDMOD2..0  VIDEO MODE CONTROL
[                            DB6= INVERT HORIZONAL SYNC/COMPOSITE SYNC
[                            DB7= INVERT VERTICAL SYNC
[              E400+9C   /W  CTRLREG2WR
[                            DB1..0=DLYSEL1..0  TIMER SKEW CONTROL
[                            DB4..2=MODE2..0    SIGNATURE COUNTER OPERATING MODE
[                            DB5=ACQFRM*        ACQUIRE FRAME SIGNATURE
[                            DB6=CMPTST*        COMPOSITE SYNC TEST SIGNAL
[                            DB7=VIDEOTST       VIDEO TEST SIGNAL
[                            DB7..5=not used
[              E400+9D   /W  UNUSED 5
[              E400+9E   /W  TSTCLKWR
[                            DATA=is of no use
[              E400+9F   /W  TESTREQWR
[                            DB0=TEST COMPOSITE SYNC
[                            DB1=TEST VIDEO SIGNAL
[                            DB7..2=not used
[
[          DECODE=DISC*[A0..A7]*[IORD+IOWR]
[          DECODE=DISC*[A8..AF]*[IORD+IOWR]   OVERMAP
[             E400+A0+0 -/W                DAC CH.B & CH.D
[             E400+A0+0 R/-                DAC HOLD
[             E400+A0+1 -/W                DAC CH.A & CH.C
[             E400+A0+1 R/-                DAC HOLD
[             E400+A0+2 R/W                DAC CH.D VIDDACLO
[             E400+A0+3 R/W                DAC CH.C VIDDACHI
[             E400+A0+4 R/W                DAC CH.B SYNCDAC
[             E400+A0+5 R/W                DAC CH.A not used
[
[
[
[
[ VIDEO SIGNATURE MANAGER RESERVED VARIABLES
[
[ X6                                       [TEST NUMBER
[ X7                                       [SUBTEST NUMBER
[ X8                                       [STACK LEVEL FOR RETURN FAIL
[ X9                                       [LOOP COUNTER
```

```
[ E                                    [SUBTEST BASELINE ERROR CODE
[
[ HARDWARE ADDRESS SPACE RESERVED VARIABLES
[    ADDRESS SPACE IS SET BY CPU ADDRESS DECODE PAL
[ T = 57344                            [INTERVAL TIMER BASE ADDRESS
[ A = 57856                            [ANALOG BASE ADDRESS
[ D = 58368                            [DISCRETE BASE ADDRESS
[
[
[ UTILITIES RESERVED VARIABLES
[ Z  : Z1 : Z2                         [SEC DELAY,DELAY ROUTINE VARIABLE
[ O                                    [DISCRETE ON/OFF BOOLEAN
[ P1 : P2 : P3 : P4 : P5               [DISCRETE PORT VARIABLES 1-5
[ S1 : S2 : S3 : S4 : S5 :             [SCRATCH PAD LOCATIONS
[ S6 : S7                              [SCRATCH PAD LOCATIONS
[ B  : N                               [BIT NUMBER POINTER,NIBBLE
[


NEW
10 STRING 0,0
20 STRING 1536,250

   [INITIALIZE FIRST TIME VARIABLES FOR SELFTEST]
30 GOSUB 1700                          [REINITIALIZE ALL VARIABLES
32 XTAL=11.0592E6 : TIME=0 : CLOCK1    [INITIALIZE TIME FUNCTION
34 BAUD 1200                           [PRINTER PORT BAUD SETTING
35 MSGDLY=1.0
40 FOR C=2 TO 5: XBY(ADAC+C)=128: NEXT C   [CLEAR ALL DAC CHANNELS

   [POWER-UP SOFTWARE VERSION DISPLAY
50 PRINT # CHR(19),                    [SET SCROLL DISPLAY
          CHR(14):                     [SET INVISIBLE CURSOR
   PRINT # "VIDEO ANALYZER",!
52 Z=MSGDLY: GOSUB 14000
54 PRINT #: PRINT # "VER: VIDTST 11/01/91",!
56 Z=MSGDLY: GOSUB 14000

   [EVOKE POWER-UP SELFTEST]
60 GOSUB 8000                          [CALL SELFTEST HANDLER SET RESULTS

70 PRINT #: PRINT # "MENU-:0C00004000;",!
94 MSGDLY=0: MSGDLY2=0                 [SET TO FAST OPERATION
95 REG=0: ONERR 600                    [SETUP ERROR HANDLER

   [VIDTST MANAGER FIRST TIME UP DEFAULT FLAG SETTING
97 X=0 :                               ["CONTINUE ON FAIL"FLAG
   X1=0 :                              ["LOOP TEST" FLAG
   X5=0 :                              ["EXECUTE SINGLE TEST" FLAG
   X6=-1:                              ["TEST NUMBER"
   X2=0 :                              ["INACTIVE DIAGNOSTIC FLAG
   X4=0 :                              ["PAUSE ON FAIL" FLAG
```

```
   X9=0 :                                    [LOOP COUNTER
   MF=0 :                                    [MENU ENABLED FLAG
   TRAP=-1                                   [INITIALIZE ERROR CODE TRAP
98 $(3)=":0A000040;"                         [FIRST ARTIFICIAL RESPONSE
99 PRINT "READY"


  [VIDTST EXECUTIVE MAIN/RENTRY POINT]
100 GOSUB 1700                               [CALL NEW UUT INITIALIZATION
101 X8=DBY(17)                               [SET INITIAL RETURN STACK LEVEL
105 IF X1=1 THEN GOTO 250
110 IF MF=0 THEN GOTO 120
112   X6=0
115 GOTO 250
120 REM ""

  [VIDTST WAIT FOR TASK MESSAGE FROM HOST]
  [ RECEIVED MESSAGE FORMAT :LLBBBBCCSS.....;
  [ MY MESSAGE FORMAT :LL000040SS.....;
  [TOSS MESSAGES NOT DESTINATED FOR ME
  [ACKNOWLEDGE IMMEDIATELY MESSAGE RECEIVED
  [MY IMMEDIATE HOLD ACKNOWLEDGEMENT :LL000040FF;


  [NEW CHARACTER INPUT ROUTINE

   [WAITING NITP CMD
160 PRINT # "?",!                            [SEND DISPLAY CMD REQUEST
161 $(0)=":;": REG=10:                       [SET ERROR RETURN REGION POINTER
   INPUT "",$(0)                             [GET CURRENT COMMAND STRING
163 PRINT $(3):                              [SEND PREVIOUS RESPONSE STRING
   C=0: CMDEN=0:CMDST=0:
   PRINT # $(0),!                            [DISPLAY RECEIVED COMMAND
164 DO:C=C+1                                 [FIND START/END DELIMITERS IF EXISTS
165   IF ASC($(0),C)=ASC(;) THEN CMDEN=C
166   IF ASC($(0),C)=ASC(:) THEN CMDST=C
167 UNTIL (CMDEN<>0).OR.(C=255)
169 IF (CMDST<>0).AND.(CMDEN<>0).AND.CMDEN>CMDST THEN GOTO 171     [BAD DELIMITERS
170   $(3)= ":0EFF0040FFFF;":
   GOTO 160
171 IF CMDEN<>CMDST+1 THEN GOTO 174       [COMMAND REQUESTED LAST RESPONSE ONLY
172   PRINT #: PRINT # "MENU-:0C00004000;",:
   GOTO 160
174 $(1)="":
      ASC($(1),1)=ASC(S):                 [REPLACE PUNCTUATION CHARACTERS : ;
      ASC($(1),(CMDEN+1-CMDST))=ASC(Q)    [TO PREVENT EXTRA IGNORED MESSAGE
175 ASC($(1),(CMDEN+2-CMDST))=13          [ADD CR TO PREVENT EXTRA IGNORED FIXED
176 FOR C=2 TO (CMDEN-CMDST)
177   ASC($(1),C)=ASC($(0),C+CMDST-1):
   NEXT C

  [ERROR CHECK MESSAGE IS NITP HEX CHARACTERS
```

```
180 REG=0                                          [SET ERROR RETURN REGION POINTER
181 Z=MSGDLY2: GOSUB 14000
182  L=0
185  DO: L=L+1:BAD=1                               [CHECK VALID NITP CHARACTERS
188    IF ASC($(1),L)=ASC(S) THEN BAD=0
190    IF ASC($(1),L)=ASC(Q) THEN BAD=0
192    IF ASC($(1),L)>=ASC(0).AND.ASC($(1),L)<=ASC(9) THEN BAD=0
194    IF ASC($(1),L)>=ASC(A).AND.ASC($(1),L)<=ASC(F) THEN BAD=0
198  UNTIL ASC($(1),L)=ASC(Q).OR.BAD               [FIND LENGTH OF STRING
200  IF bad=0 THEN GOTO 210
201    $(3)= ":0EFF0040FFF0;":
     GOTO 100

   [CALL ASCII TO DECIMAL CONVERT THEN CHECK MESSAGE LENGTH
210  $(2)=$(1):PUSH 2,2,16: GOSUB 3000
211  POP ML
215  IF ML<>L THEN $(3)= ":0EFF0040FFF1;": GOTO 100

   [CHECK DESTINATION,COMMAND FIELD
220  $(2)="Scc000040ss"
225  FOR C= 4 TO 9 step 1
230    IF ASC($(1),c)<>asc($(2),c) then bad=1
235  NEXT C
240  IF bad=0 THEN GOTO 245
241    $(3)= ":0EFF0040FFF2;":
     GOTO 100

   [GET SUBCOMMAND SEQUENCE NUMBER
245  $(2)=$(1):PUSH 10,2,16: GOSUB 3000            [CONVERT ASCII TO HEX
246  POP X6

   [PROCESS BUTTON VARIABLES FOR TEST MODE OF OPERATION NORMAL/DIAG
250 IF (X6<>0).AND.X2 THEN 275                   [SKIP CONTROL FLAG RESTORE STARTUP
255 IF X6<>0 THEN 270                              [CHK NON-MENU STARTUP
   [DEBUG/DIAGNOSTIC SELECTED
260    X5=1:X2=1:X9=0:MF=1:GOSUB 1000              [SET DIAG FLAGS,CALL DIAG. MENU
265    GOTO 275                                    [AVOID FLAG RESET ENTRY POINT
270 X=0 :                                          ["CONTINUE ON FAIL"FLAG
    X1=0 :                                         ["LOOP TEST" FLAG
    X5=0 :                                         ["EXECUTE SINGLE TEST" FLAG
    X2=0 :                                         ["INACTIVE DIAGNOSTIC FLAG
    X4=0 :                                         ["PAUSE ON FAIL" FLAG
    X9=0 :                                         [LOOP COUNTER
    TRAP=-1                                        [INITIALIZE ERROR CODE TRAP
275 X9=X9+1:GOSUB 1700                             [INC.LOOP CT, SYSTEM,LOOP RENTRY
280 IF GET THEN 260                                [MONITOR KEYBOARD TO HALT LOOPING


   [SELECT TEST SEQUENCE BASED ON OPERATING MODE
   [SET X6 TO TEST NUMBER

   [TEST EXECUTIVE SEQUENCER BEGINS ]
```

```
  [CONFIGURATION SEQUENCE TERMINATOR IS "0"
300 RESTORE:
     Y2=0
305   DO : READ X7                                  [LOOK-UP TEST TABLE
307      IF X7=0 THEN Y2=Y2+1
310   UNTIL Y2=X6
311 ET=ET+TIME                                      [TOTAL RUNNING TIME
315 TIME=0: DBY(71)=0                               [RESET TIME CLOCK
320 READ X7
321 P.#: P.# USING(##),"T=",X6,"S=",X7,"L=",X9,!
322 P.# USING(##.#),"C=",TIME,!
323 P.# USING(0),!
324 %=MSGDLY2: GOSUB 14000
325 E=X7*1000: X8=DBY(17)-3                         [COMPUTE BASE ERROR CODE
                                                    [STACK LEVEL FOR FAIL RETURN
330 ON INT(X7/5) GOTO 340,345,350,355,360,365,370
332 STOP

    [SUBTEST 0-9 HANDLER
340 ON X7 GOSUB    1600,     [SUBROUTINE 0: END SEQUENCE
                   1400,     [SUBROUTINE 1: NEW UUT INITIALIZATION
                    500,     [SUBROUTINE 2: POWER-UP SUBROUTINE
                   8500,     [SUBROUTINE 3: DIGITAL VIDEO FRAME SELFTEST
                   7680      [SUBROUTINE 4: WRITE DAC VARIABLES TO DAC
342 GOTO 320

345 ON X7-5 GOSUB   4000,    [SUBROUTINE 5:TTL SEPERATE VARIABLE BUILD MENU
                    4200,    [SUBROUTINE 6:TTL COMPOSITE VARIABLE BUILD MENU
                    4400,    [SUBROUTINE 7:ANALOG SEPERATE VARIABLE BUILD MENU
                    4600,    [SUBROUTINE 8:ANALOG COMPOSITE VARIABLE BUILD MENU
                    4800     [SUBROUTINE 9:ANALOG GREEN SYNC VARIABLE BUILD MENU
347 GOTO 320

    [SUBTEST 10-19 HANDLER
350 ON X7-10 GOSUB   400,    [SUBROUTINE10: WRITE ANALOG DAC CHANNELS
                    4100,    [SUBROUTINE11: PARSE TTL SEPERATE SYNC CMD
                    4300,    [SUBROUTINE12: PARSE TTL COMPOSITE SYNC CMD
                    4500,    [SUBROUTINE13: PARSE ANALOG SEPERATE SYNC CMD
                    4700     [SUBROUTINE14: PARSE ANALOG COMPOSITE SYNC CMD
352 GOTO 320

355 ON X7-15 GOSUB  4900,    [SUBROUTINE15: PARSE ANALOG GREEN SYNC COMMAND
                    5000,    [SUBROUTINE16: ACQUIRE VIDEO SIGNATURE OF CHANNEL
                    4940,    [SUBROUTINE17: PARSE WRITE DAC COMMAND
                    5400,    [SUBROUTINE18: MENU SCREEN RESPONSE OF VARIABLES
                    4920     [SUBROUTINE19: OPERATOR INPUT MENU FOR DAC WRITE
357 GOTO 320

    [SUBTEST 20-24 HANDLER
360 ON X7-20 GOSUB  8100,    [SUBROUTINE20: ANALOG SELFTEST
                    1100,    [SUBROUTINE21: CHANGE OPTION LIST
                     700,    [SUBROUTINE22: EXIT MENU
```

```
                  400,     [SUBROUTINE23:
                  400      [SUBROUTINE24:
362 GOTO 320

    [SUBTEST 25-29 HANDLER
365 ON X7-20 GOSUB  400,     [SUBROUTINE25:
                    400,     [SUBROUTINE26:
                    400,     [SUBROUTINE27:
                    400,     [SUBROUTINE28:
                    400      [SUBROUTINE29:
366 GOTO 320

370 RESTORE :
   ON X7-30 GOTO 98          [SUBROUTINE30: EXIT TEST EXECUTIVE
371  STOP

400 RETURN                   [DUMMY FILL SUBROUTINE
  [ TEST SEQUENCER ENDS


449 DATA 0                    [**REQUIRED BY TEST EXECUTIVE PROGRAM**
                              [**DO NOT REMOVE **
                             [DIRECT MENU STARTUP TEST NUMBER=0

        [TEST SEQUENCE TABLE]
450 DATA 1,5,16,18,0          [TEST 1-MENU TTL SEPERATE SYNC
451 DATA 1,6,16,18,0          [TEST 2-MENU TTL COMPOSITE SYNC
452 DATA 1,7,16,18,0          [TEST 3-MENU ANALOG SEPERATE SYNC
456 DATA 1,8,16,18,0          [TEST 4-MENU ANALOG COMPOSITE SYNC
458 DATA 1,9,16,18,0          [TEST 5-MENU ANALOG GREEN SYNC
460 DATA 19,4,0               [TEST 6-MENU WRITE DAC OUTPUTS
462 DATA 16,18,0              [TEST 7-RE-ACQUIRE SIGNATURE
464 DATA 0                    [TEST 8-
466 DATA 0                    [TEST 9-
468 DATA 1,3,0                [TEST10-DIGITAL SELFTEST
470 DATA 1,20,0               [TEST11-ANALOG SELFTEST
472 DATA 0                    [TEST12-
474 DATA 21,0                 [TEST13-CHANGE OPTION LIST
476 DATA 0                    [TEST14-
478 DATA 0                    [TEST15-
480 DATA 0                    [TEST16-
482 DATA 0                    [TEST17-
484 DATA 0                    [TEST18-
485 DATA 22,0                 [TEST19-EXIT
486 DATA 0                    [TEST20-
487 DATA 1,11,16,0            [TEST21-PORT CMD TTL SEPERATE SYNC
488 DATA 1,12,16,0            [TEST22-PORT CMD TTL COMPOSITE SYNC
489 DATA 1,13,16,0            [TEST23-PORT CMD ANALOG SEPERATE SYNC
490 DATA 1,14,16,0            [TEST24-PORT CMD ANALOG COMPOSITE SYNC
491 DATA 1,15,16,0            [TEST25-PORT CMD ANALOG GREEN SYNC
492 DATA 1,17,4,16,0          [TEST26-WRITE DACS AND RE-ACQUIRE SIG
```

```
     [POWER-UP UUT
500 PRINT #: PRINT # "PU ",: RETURN

     [POWER DOWN UUT
550 RETURN

    [ON ERROR PROCESSING
600 IF REG<>10 THEN GOTO 620              [IF INPUT PROCESSING
605   IF XBY(257)<>40 THEN GOTO 620       [RECEIVED BAD ARGUMNT ERROR
606     DBY(17)=X8:                       [RESTORE STACK TO ABORT RETURN
        GOTO 160
620 REM
697 PRINT "BASIC-52 ERROR CODE=",XBY(257)
698 DBY(17)=X8                            [RESTORE STACK TO ABORT RETURN
699 GOTO 100


700 MF=0: X6=-1
720 RETURN

       [ TROUBLESHOOT MENU SUBROUTINE]
[THE FOLLOWING LINES WILL BE USED TO DISPLAY THE TROUBLESHOOT/DEBUG MENU
[THAT OCCURS WHEN THE DIAGNOSTIC FLAG BECOMES ACTIVE. THE BASIC FORM OF THIS
[SCREEN CAN BE CHANGED AS DESIRED FOR THE APPLICATION.

1000 P. "     *** MAIN TEST CONTROL MENU ***       "
1006 P. "TEST 1- TTL    SEPERATE  SYNC ANALYSIS    "
1008 P. "TEST 2- TTL    COMPOSITE SYNC ANALYSIS    "
1010 P. "TEST 3- ANALOG SEPERATE  SYNC ANALYSIS    "
1014 P. "TEST 4- ANALOG COMPOSITE SYNC ANALYSIS    "
1016 P. "TEST 5- ANALOG GREEN     SYNC ANALYSIS    "
1018 P. "TEST 6- INPUT & WRITE DAC OUTPUTS         "
1020 P. "TEST 7- RE-ACQUIRE ANY SIGNATURE          "
1022 P. "TEST 8-                                   "
1024 P. "TEST 9-                                   "
1026 P. "TEST10-DIGITAL SELFTEST                   "
1028 P. "TEST11-ANALOG SELFTEST                    "
1030 P. "TEST12-                                   "
1032 P. "TEST13-CHANGE OPTION LIST                 "
1034 P. "TEST14-                                   "
1036 P. "TEST15-                                   "
1037 P. "TEST16-                                   "
1039 P. "TEST19-EXIT MENU                          ":P.
1041 P. " ",CHR(7):Z1=GET                         [REQUIRED:TERMINAL BEEP
1042 INPUT "EXECUTE TEST NUMBER=?",X6             [REQUIRED:TEST INPUT
1043 IF X6<1.OR.X6>19 THEN GOTO 1000              [REJECT BAD TEST NUMBER
1049 RETURN                                       [REQUIRED:


          [SET/CLEAR OPTION FLAG SUBROUTINE/SUBTEST]
 [THE FOLLOWING DIAGNOSTIC FLAG SET PROVIDES ADEQUATE CONTROL IN MOST
 [APPLICATIONS HOWEVER ADDITION FLAGS CAN BE ADDED TO IMPROVE TEST
```

```
 [SYSTEM CONTROL.
 [NOTE: THESE FLAGS CAN BE SET/CLEARED IN USER ROUTINES TO PROVIDE SPECIAL
 [ ERROR CONTROL BUT THE FLAGS SHOULD BE RESTORED AFTER USE TO INSURE
 [ PREDICTABLE OPERATION.
1100  P. "P.O.F=",X4," C.F.=",X," L.F=",X1," S.T.E=",X5," P.P.=",X3,!
1101  P. " P.F.D=",X0," EC TRAP=",TRAP:P.
1150  P. "0-SET/CLEAR PAUSE ON FAIL FLAG"
1151  P. "1-SET/CLEAR CONTINUE ON FAIL FLAG"
1152  P. "2-SET/CLEAR LOOP TEST FLAG"
1153  P. "3-SET/CLEAR DEBUG/PRODUCTION* TEST EXECUTION"
1154  P. "4-SET/CLEAR PRINT ALL PARAMETERS FLAG"
1155  P. "5-SET/CLEAR PRINT PASS/FAIL DATA"
1156  P. "6-SET/CLEAR ERROR CODE TRAP"
1157  P. "7-TROUBLESHOOT MESSAGE DELAYS"
1158  P. "8-EXIT"
1160  INPUT"OPTION ?",S1
1161  IF S1>7 THEN 1195
1162  ON S1 GOSUB 1173,1165,1167,1169,1171,1175,1177,1180
1163  GOTO 1100
1165  IF X THEN X=0 ELSE X=1
1166  RETURN
1167  IF X1 THEN X1=0 ELSE X1=1
1168  RETURN
1169  IF X5 THEN X5=0 ELSE X5=1
1170  RETURN
1171  IF X3 THEN X3=0 ELSE X3=1
1172  RETURN
1173  IF X4 THEN X4=0 ELSE X4=1
1174  RETURN
1175  IF X0 THEN X0=0 ELSE X0=1
1176  RETURN
1177  P. "ERROR CODE TRAP  DISABLE=-1  ENABLED=error code"
1178  INPUT "TRAP CODE ?",TRAP
1179  RETURN
1180  P. "TROUBLESHOOT MESSAGE DELAY"
1181  IF MSGDLY=0 THEN GOTO 1182 ELSE GOTO 1185
1182    MSGDLY =5: MSGDLY2=1: RETURN
1185    MSGDLY =0: MSGDLY2=0: RETURN
1194  RETURN
1195  GOSUB 1000
1199  CLEARS:X9=0:GOTO 275


          [NEW UUT INITIALIZATION SUBTEST]
 [PROVIDES A UNIQUE SUBTEST! WHICH CAN BE USED AT ANY POINT IN THE
 [DEBUG/PRODUCTION TEST SEQUENCE TO PROVIDE TEST SYSTEM/UUT INITIALIZATION
1400  PRINT #: PRINT # "INIT ",!                [DISPLAY INITIALIZATION
1402  Z=MSGDLY2: GOSUB 14000                    [CALL DELAY
1410  GOSUB 1700
1499  RETURN
```

```
      [UUT TESTING EXIT SUBROUTINE/SUBTEST]
 [PROVIDES A UNIQUE SUBTEST! WHICH SHOULD BE USED ONLY AT THE END OF
 [TEST SEQUENCE TO PROVIDE PROPER EXIT TO TEST EXECUTIVE AND
 [PROPER MODULE PASS INDICATION.
1600  REM ""                                  [REQUIRED:SET/LOG RESULTS
1655  PRINT # "TEST PASS",!                   [PRINT TO OPERATOR DISPLAY
1402  Z=MSGDLY: GOSUB 14000                   [CALL DELAY
   [SECONDARY ENTRY POINT NON-PASSING
1660  GOSUB 550                               [CALL POWER DOWN
1665  CLEARS                        [REQUIRED:ADJ.C-STACK,A-STACK ABORT RETURN
1670  IF X1 THEN 275                [REQUIRED:LOOP TEST :TEST EXECUTIVE DIRECT
1699  GOTO 100                      [REQUIRED:RETURN TO TEST EXEC. ENTRY POINT


      [TEST STATION INITIALIZATION ROUTINE]
 [THIS ROUTINE HANDLES ALL RE/INITIALIZATION OF ALL SYSTEM VARIABLES
 [AND HARDWARE DEVICES THAT MUST SET BEFORE ENTERING THE TEST EXECUTIVE.
 [NOTE: THIS ROUTINE MUST BE CALL AFTER ANY "CLEAR" COMMAND IS EXECUTED
 [      BUT IS NOT REQUIRED AFTER "CLEARS" COMMAND.

1700 T = 57344:                            [REQUIRED:INTERVAL TIMER BASE ADDRESS
     A = 57856:                            [REQUIRED:ANALOG BASE ADDRESS
     DISC = 58368                          [REQUIRED:DISCRETE BASE ADDRESS

1702 ADAC=DISC+160:                        [BASE ADDRESS DAC
     ASIGX=DISC+136                        [BASE ADDRESS SIGNATURE COUNTER

1704 ARLY=DISC+152+0:                      [ADDRESS RELAY CONTROL SET
     ATMLO=DISC+152+1:                     [ADDRESS WR TIMER LO REGISTER
     ATMHI=DISC+152+2                      [ADDRESS WR TIMER HI REGISTER
1705 ARGC1=DISC+152+3:                     [ADDRESS CTRLREG1
     ARGC2=DISC+152+4:                     [ADDRESS CTRLREG1
     ATSCK=DISC+152+6                      [ADDRESS TSTCLKWR
1706 ATSR=DISC+152+7                       [ADDRESS TESTREGWR

1720 PRLY=0: XBY(ARLY)=PRLY                [INITIALIZE ALL PORTS & VARIABLES
1721 PTMLO=0: XBY(ATMLO)=PTMLO
1722 PTMHI=0: XBY(ATMHI)=PTMHI
1723 PRGC1=0: XBY(ARGC1)=PRGC1
1724 PRGC2=0: XBY(ARGC2)=PRGC2
1725 PTSR=0: XBY(ATSR)=PTSR
1799 RETURN

      [ UUT FAIL ENTRY POINT SUBROUTINE
 [PROVIDES THE PROPER METHOD OF HANDLING UUT FAILURE TO ALLOW FULL DIAGNOSTIC
 [IMPLEMENTATION OF PAUSE/CONTINUE ON FAIL, LOOP TEST or RENTRY INTO TEST
 [EXECUTIVE.

1800  POP J4,J3,J2,E1,J1                   [ACT,RECEIVED,ERRCODE,P/F
1801  IF TRAP=E1 THEN J1=1                 [EC TRAP MATCH THEN CHG SIGN
      [*** TEST CONDITION P/F FLAG ***
1802  IF J1 THEN GOTO 1810 ELSE GOTO 1804
```

```
      [*** PASS BLOCK ***
1804  REM
1809  RETURN

      [*** FAIL BLOCK ***
      [":1AFF0040ttsseeeeLLLLAAAAHHHH;"
1810  $(3)=":1AFF0040":
      P.# "FAIL",!                        [REQUIRED:DISPLAY FAIL

1811  PUSH X6,2,16: GOSUB 3100            [CONVERT TEST NUMBER TO ASCII
1812  ASC($(3),10)=ASC($(0),1):           [ASSEMBLE RESPONSE 1ST DIGIT
      ASC($(3),11)=ASC($(0),2)            [ASSEMBLE RESPONSE 2ND DIGIT

1815  PUSH X7,2,16: GOSUB 3100            [CONVERT TEST NUMBER TO ASCII
1816  ASC($(3),12)=ASC($(0),1):           [ASSEMBLE RESPONSE 1ST DIGIT
      ASC($(3),13)=ASC($(0),2)            [ASSEMBLE RESPONSE 2ND DIGIT

1820  PUSH E1,4,16: GOSUB 3100            [CONVERT ERROR CODE TO ASCII
1821  ASC($(3),14)=ASC($(0),1):           [ASSEMBLE RESPONSE 1ST DIGIT
      ASC($(3),15)=ASC($(0),2)            [ASSEMBLE RESPONSE 2ND DIGIT
1822  ASC($(3),16)=ASC($(0),3):           [ASSEMBLE RESPONSE 3ND DIGIT
      ASC($(3),17)=ASC($(0),4)            [ASSEMBLE RESPONSE 4ND DIGIT

1825  PUSH J2,4,16: GOSUB 3100            [CONVERT PARMETER 1 TO ASCII
1826  ASC($(3),18)=ASC($(0),1):           [ASSEMBLE RESPONSE 1ST DIGIT
      ASC($(3),19)=ASC($(0),2)            [ASSEMBLE RESPONSE 2ND DIGIT
1827  ASC($(3),20)=ASC($(0),3):           [ASSEMBLE RESPONSE 3ND DIGIT
      ASC($(3),21)=ASC($(0),4)            [ASSEMBLE RESPONSE 4ND DIGIT

1830  PUSH J3,4,16: GOSUB 3100            [CONVERT PARAMETER 2 TO ASCII
1831  ASC($(3),22)=ASC($(0),1):           [ASSEMBLE RESPONSE 1ST DIGIT
      ASC($(3),23)=ASC($(0),2)            [ASSEMBLE RESPONSE 2ND DIGIT
1832  ASC($(3),24)=ASC($(0),3):           [ASSEMBLE RESPONSE 3ND DIGIT
      ASC($(3),25)=ASC($(0),4)            [ASSEMBLE RESPONSE 4ND DIGIT

1835  PUSH J4,4,16: GOSUB 3100            [CONVERT PARAMETER 3 TO ASCII
1836  ASC($(3),26)=ASC($(0),1):           [ASSEMBLE RESPONSE 1ST DIGIT
      ASC($(3),27)=ASC($(0),2)            [ASSEMBLE RESPONSE 2ND DIGIT
1837  ASC($(3),28)=ASC($(0),3):           [ASSEMBLE RESPONSE 3ND DIGIT
      ASC($(3),29)=ASC($(0),4)            [ASSEMBLE RESPONSE 4ND DIGIT

1839  ASC($(3),30)=ASC(;)                 [ADD TERMINATION SEMICOLON

1840  Z=MSGDLY2: GOSUB 14000              [DELAY DISPLAY FAIL MESSAGE
1842  P.#: P.# "FAIL=",E1," ",!           [REQUIRED:DISPLAY FAIL
1843  Z=MSGDLY: GOSUB 14000               [CALL DELAY
1844  P.#: PH1.# USING(####),J2,J3,J4,!    [REQUIRED:DISPLAY FAIL
1845  PH1.# USING(0),!
1846  Z=MSGDLY: GOSUB 14000               [CALL DELAY
1847  IF X4=0 THEN 1860                   [REQUIRED:BRANCH IF P.O.F INACTIVE
1850    Z4=GET:PRINT # "POF ",!           [REQUIRED:CLEAR FALSE KEY STROKE
1852    DO:
```

```
        Z4=GET:
       UNTIL Z4<>0                    [WAIT FOR ANY KEY PRESS
1860  IF X THEN RETURN                [REQUIRED:"CONTINUE ON FAIL"FLAG CHECK
1880  GOTO 1660                              [REQUIRED:EXIT JUMP TO UUT PASS ROUTINE


      [ TEST SYSTEM FAIL OUTPUT ROUTINE]
[THIS SUBROUTINE/SUBTEST WAS DEVELOPED TO PROVIDE CONTINOUS TERMINAL OUTPUT OF
[PROGRAM DETECTED TEST SYSTEM FAILURE. THE VARIABLES PASSED CONTAIN NO
[PREDEFINED MEANING HOWEVER VARIBLE "J" SHOULD CORRESPOND TO EITHER AN ERROR
[CODE OR THE LINE NUMBER WHERE THE FAILURE IS DIAGNOSED. EXIT OF THIS ROUTINE
[REQUIRES <CTRL C>, RESET OR STATION POWER DOWN.
[ln     PUSH J,J1,J2,J3

1900    POP J5,J4,J3,J2,J1,J           [REQUIRED:PULL FAILURE DATA
1910    PRINT#: PRINT # USING(####),"SFAIL ",J," ",J1," ",J2,!
1911    PRINT # " ",J3," ",!           [REQUIRED:PRINT FAIL DATA
1912    PRINT USING(0),!
1915    FOR C=0 TO 999:NEXT C
1920    GOTO 1910                      [REQUIRED:REPEAT TERMINAL OUTPUT FAIL DATA


  [***** MAIN TEST PROGRAN BEGINS HERE *****]


  [***** USEFUL TEST SYSTEM UTILITIES BEGINS HERE *****]

  [DATE 8/16/91
  [ASCII TO DECIMAL CONVERT FOR HEX CHARACTER BASE 2,7,10,16
  [245  $(2)=$(1):PUSH 10,2,16: GOSUB 3000       [CONVERT ASCII TO HEX

3000 POP BASE,NC,ST      [POP NUMBER BASE,#OF CHAR TO CONVERT,STARTING CHARACTER
3005  RES=0
3010  FOR DPOS=0 TO NC-1 STEP 1                  [STEP THROUGH DIGIT POSITION
3015    A1=ASC($(2),ST+DPOS): IF A1>=ASC(A) THEN A1=A1-65+10 ELSE A1=A1-48
3025    RES=RES+A1*BASE**(NC-1-DPOS)
3030  NEXT DPOS
3035  PUSH RES                                   [DECIMAL CONVERTED NUMBER
3040 RETURN


  [NUMBER TO ASCII CONVERT FOR HEX CHARACTER BASE 2,7,10,16
  [245  PUSH RES,2,16: GOSUB 3100    [CONVERT NUMBER TO ASCII
  [WARNING: NUMBER CANNOT EXCEED NUMBER OF DIGIT CHECKED
3100 POP BASE,NC,RES     [POP NUMBER BASE,#OF DIGITS TO CONVERT,VALUE
3105  $(0)="" :$(5)="0123456789ABCDEF.."
3110  FOR DPOS=NC-1 TO 0 STEP -1                 [STEP THROUGH DIGIT POSITION
3115    A1=INT(RES/(BASE**DPOS)):RES=((RES/(BASE**DPOS))-A1)*BASE**DPOS
3120    ASC($(0),NC-DPOS)=ASC($(5),A1+1)
3130  NEXT DPOS
3140 RETURN
```

```
[********* PARSE VIDEO MESSAGE FORMATS ************

   [MENU GENERIC FIELD INPUT ROUTINE
   [ :LLBBBBCC TT ACC IHIV SRC                    ;
   [  ACC=02:SIGNATURE, 01:COUNT PIXEL
   [  IH=0:HORIZONAL SYNC, 1=INVERT HORIZONAL SYNC
   [  IV=0:VERTICAL SYNC, 1=INVERT VERTICAL SYNC
   [  SRC=00:TTL RED, 01:TTL GREEN, 02:TTL BLUE, 03:SEC RED, 04:SECGRN/INTENSITY
   [      05:SEC BLUE, 06:TTL/ANALOG VERTICAL SYNC, 07:TTL/ANALOG HORIZONAL SYNC
   [      10:ANALOG RED, 11:ANALOG GREEN, 12:ANALOG BLUE, 13:TEST REFERENCE
   [      14:COMPOSITE VIDEO
3800  PRINT:PRINT "ACQUISITION MODE"
3801  PRINT "  01=COUNT PIXEL           02=SIGNATURE"
3802  INPUT "ACC=",ACC
3804  PUSH ACC:GOSUB 7700                    [PROGRAM ACQUIAITION MODE SELECTION

3810  PRINT: PRINT "SOURCE SELECTION MODE"
3811  PRINT "  00=TTL RED               01=TTL GREEN"
3812  PRINT "  02=TTL BLUE              03=SEC RED"
3813  PRINT "  04=SECGRN/INTENSITY      05=SEC BLUE"
3814  PRINT "  06=TTL/ANALOG VERT. SYNC 07=TTL/ANALOG HORZ. SYNC"
3815  PRINT "  16=ANALOG RED            17=ANALOG GREEN"
3816  PRINT "  18=ANALOG BLUE           19=TEST REFERENCE"
3817  PRINT "  20=COMPOSITE VIDEO"
3818  INPUT "SRC=",SRC
3820  PUSH SRC:GOSUB 7700                    [PROGRAM VIDEO SOURCE SELECTION

3825  PRINT:PRINT "HORIZONAL SYNC INVERT"
3826  PRINT "  0=TRUE SYNC   1=INVERTED SYNC"
3827  INPUT "IH=",IH
3828  PUSH IH: GOSUB 7700                    [PROGRAM HORIZONAL SYNC INVERT

3830  PRINT:PRINT "VERTICAL SYNC INVERT"
3831  PRINT "  0=TRUE SYNC   1=INVERTED SYNC"
3832  INPUT "IV=",IV
3833  PUSH IV: GOSUB 7700                    [PROGRAM VERTICAL SYNC INVERT
3899  RETURN


   [GENERIC TTL MESSAGE PARSING ROUTINE
   [ :LLBBBBCC TT ACC IHIV SRC SKEW               ;
   [  ACC=02:SIGNATURE, 01:COUNT PIXEL
   [  IH=0:HORIZONAL SYNC, 1=INVERT HORIZONAL SYNC
   [  IV=0:VERTICAL SYNC, 1=INVERT VERTICAL SYNC
   [  SRC=00:TTL RED, 01:TTL GREEN, 02:TTL BLUE, 03:SEC RED, 04:SECGRN/INTENSITY
   [      05:SEC BLUE, 06:TTL/ANALOG VERTICAL SYNC, 07:TTL/ANALOG HORIZONAL SYNC
   [      10:ANALOG RED, 11:ANALOG GREEN, 12:ANALOG BLUE, 13:TEST REFERENCE
   [      14:COMPOSITE VIDEO
   [  SKEW=00, 01, 02, 03
3900  $(2)=$(1):PUSH 12,2,16: GOSUB 3000  [CONVERT ACQUISITION MODE ASCII TO HEX
3901  POP ACC:PUSH ACC:GOSUB 7800          [PROGRAM ACQUISITION MODE SELECTION
3910  $(2)=$(1):PUSH 16,2,16: GOSUB 3000  [CONVERT SRC ASCII TO HEX
```

```
3911  POP SRC:PUSH SRC:GOSUB 7700          [PROGRAM VIDEO SOURCE SELECTION
3915  $(2)=$(1):PUSH 14,1,16: GOSUB 3000   [CONVERT HORZ SYNC INV ASCII TO HEX
3916  POP IH:PUSH IH:GOSUB 7740            [PROGRAM HORIZONAL SYNC INVERT
3920  $(2)=$(1):PUSH 15,1,16: GOSUB 3000   [CONVERT VERT SYNC INV ASCII TO HEX
3921  POP IV:PUSH IV:GOSUB 7760            [PROGRAM VERTICAL SYNC INVERT
3920  $(2)=$(1):PUSH 18,2,16: GOSUB 3000   [CONVERT CLOCK SKEW ASCII TO HEX
3921  POP SKEW:PUSH SKEW:GOSUB 7780        [PROGRAM DOT CLOCK CLOCK SKEW
3999  RETURN


    [****** PARSING TTL MESSAGE SECTION **********
    [ :LLBBBBCC TT ACC IHIV SRC                    ;
    [  TT=21:TTL SEPERATE SYNC
    [  ACC=02:SIGNATURE, 01:COUNT PIXEL
    [  IH=0:HORIZONAL SYNC, 1=INVERT HORIZONAL SYNC
    [  IV=0:VERTICAL SYNC, 1=INVERT VERTICAL SYNC
    [  SRC=00:TTL RED, 01:TTL GREEN, 02:TTL BLUE, 03:SEC RED, 04:SECGRN/INTENSITY
    [      05:SEC BLUE, 06:TTL/ANALOG VERTICAL SYNC, 07:TTL/ANALOG HORIZONAL SYNC
    [      10:ANALOG RED, 11:ANALOG GREEN, 12:ANALOG BLUE, 13:TEST REFERENCE
    [      14:COMPOSITE VIDEO
    [  SKEW=00, 01, 02, 03

     [SUBROUTINE 5:TTL SEPERATE VARIABLE BUILD MENU
4000  PUSH 0: GOSUB 7720                    [SYNC MODE SELECTION=TTL SEPERATE SYNC
4005  GOSUB 3800                            [CALL GENERIC TTL MESSAGE INPUT ROUTINE
4099  RETURN

     [SUBROUTINE11: PARSE TTL SEPERATE SYNC CMD
     [  TT=21:TTL SEPERATE SYNC
     [
4100  PUSH 0: GOSUB 7720                   [SYNC MODE SELECTION=TTL SEPERATE SYNC
4105  GOSUB 3900                           [CALL GENERIC TTL MESSAGE PARSING ROUTINE
4199  RETURN


     [SUBROUTINE 6:TTL COMPOSITE VARIABLE BUILD MENU
4200  PUSH 1: GOSUB 7720                   [SYNC MODE SELECTION=TTL COMPOSITE SYNC
4205  GOSUB 3800                           [CALL GENERIC TTL MESSAGE INPUT ROUTINE
4299  RETURN

     [SUBROUTINE12: PARSE TTL COMPOSITE SYNC CMD
     [  TT=22:TTL COMPOSITE SYNC
     [
4300  PUSH 1: GOSUB 7720                   [SYNC MODE SELECTION=TTL COMPOSITE SYNC
4305  GOSUB 3900                           [CALL GENERIC TTL MESSAGE PARSING ROUTINE
4399  RETURN


    [****** PARSING ANALOG MESSAGE SECTION **********
    [ :LLBBBBCC TT ACC IH IV SRC VWINHI VWINLO SWIN;
    [  TT=24:ANALOG COMPOSITE SYNC
```

```
[  ACC=02:SIGNATURE, 01:COUNT PIXEL
[  IH=0:HORIZONAL SYNC, 1=INVERT HORIZONAL SYNC
[  IV=0:VERTICAL SYNC, 1=INVERT VERTICAL SYNC
[  SRC=00:TTL RED, 01:TTL GREEN, 02:TTL BLUE, 03:SEC RED, 04:SECGRN/INTENSITY
[     05:SEC BLUE, 06:TTL/ANALOG VERTICAL SYNC, 07:TTL/ANALOG HORIZONAL SYNC
[     10:ANALOG RED, 11:ANALOG GREEN, 12:ANALOG BLUE, 13:TEST REFERENCE
[     14:COMPOSITE VIDEO
[  SKEW=00, 01, 02, 03
[  VWINHI=00
[  VWINLO=00
[  SWIN=00


    [SUBROUTINE 7:ANALOG SEPERATE VARIABLE BUILD MENU
4400  PUSH 2: GOSUB 7720              [SYNC MODE SELECTION=ANALOG SEPERATE SYNC
4405  GOSUB 3800                      [CALL GENERIC TTL MESSAGE INPUT ROUTINE
4410  PUSH 258: GOSUB 7830            [WRITE VERTICAL INTERVAL COUNTER
4499  RETURN

    [SUBROUTINE13: PARSE ANALOG SEPERATE SYNC CMD
    [  TT=23:ANALOG SEPERATE SYNC
    [
4500  PUSH 2: GOSUB 7720              [SYNC MODE SELECTION=ANALOG SEPERATE SYNC
4505  GOSUB 3900                      [CALL GENERIC TTL MESSAGE PARSING ROUTINE
4510  PUSH 258: GOSUB 7830            [WRITE VERTICAL INTERVAL COUNTER
4599  RETURN


    [SUBROUTINE 8:ANALOG COMPOSITE VARIABLE BUILD MENU
4600  PUSH 3: GOSUB 7720              [SYNC MODE SELECTION=ANALOG COMPOSITE SYNC
4605  GOSUB 3800                      [CALL GENERIC TTL MESSAGE INPUT ROUTINE
4610  PUSH 258: GOSUB 7830            [WRITE VERTICAL INTERVAL COUNTER
4699  RETURN

    [SUBROUTINE14: PARSE ANALOG COMPOSITE SYNC CMD
    [  TT=24:ANALOG COMPOSITE SYNC
    [
4700  PUSH 3: GOSUB 7720              [SYNC MODE SELECTION=ANALOG COMPOSITE SYNC
4705  GOSUB 3900                      [CALL GENERIC TTL MESSAGE PARSING ROUTINE
4710  PUSH 258: GOSUB 7830            [WRITE VERTICAL INTERVAL COUNTER
4799  RETURN


    [SUBROUTINE 9:ANALOG GREEN SYNC VARIABLE BUILD MENU
4800  PUSH 4: GOSUB 7720              [SYNC MODE SELECTION=ANALOG GREEN SYNC
4805  GOSUB 3800                      [CALL GENERIC TTL MESSAGE INPUT ROUTINE
4810  PUSH 258: GOSUB 7830            [WRITE VERTICAL INTERVAL COUNTER
4899  RETURN

    [SUBROUTINE15: PARSE ANALOG GREEN SYNC COMMAND
    [  TT=25:ANALOG GREEN SYNC
    [
```

```
4900  PUSH 4: GOSUB 7720                [SYNC MODE SELECTION=ANALOG GREEN SYNC
4905  GOSUB 3900                        [CALL GENERIC TTL MESSAGE PARSING ROUTINE
4910  PUSH 258: GOSUB 7830              [WRITE VERTICAL INTERVAL COUNTER
4919  RETURN

   [MENU INPUT PROGRAM FOR DAC LEVELS
4920 PRINT "DAC VOLTAGE LEVEL MENU PROGRAM"
4925 INPUT "  DAC VIDEO HIGH CNT ?",VIDDACHI
4926 INPUT "  DAC VIDEO LOW CNT ?",VIDDACLO
4927 INPUT "  DAC VIDEO SYNC HIGH CNT ?",SYNCDAC
4939 RETURN


   [COMMAND PARSING PROGRAM FOR DAC VOLTAGE LEVELS
   [ :LLBBBBCC TT VIDDACHI VIDDACLO SYNCDAC ;
   [  TT=21:TTL SEPERATE SYNC
   [ VIDDACHI: VIDEO HI COMPARITOR DAC LEVEL
   [ VIDDACLO: VIDEO LO COMPARITOR DAC LEVEL
   [ SYNCDAC:   SYNC HI COMPARITOR DAC LEVEL
4940  $(2)=$(1):PUSH 12,2,16: GOSUB 3000  [CONVERT VIDEO DAC HI ASCII TO HEX
4941  POP VIDDACHI
4945  $(2)=$(1):PUSH 14,2,16: GOSUB 3000  [CONVERT VIDEO DAC LO ASCII TO HEX
4946  POP VIDDACLO
4950  $(2)=$(1):PUSH 16,2,16: GOSUB 3000  [CONVERT SYNC DAC HI ASCII TO HEX
4951  POP SYNCDAC
4959  RETURN


[************* ACQUIRE VIDEO SIGNATURE UTILITIES **************
     [SUBROUTINE16: ACQUIRE VIDEO SIGNATURE OF CHANNEL
5000   PUSH 0: GOSUB 7850                    [ACQUIRE FRAME=NO
5005   PUSH 0: GOSUB 7820                    [SET/CLEAR TEST VIDEO SIGNAL
5006   PUSH 0: GOSUB 7840                    [SET/CLEAR TEST COMPOSITE SYNC
5010   PUSH 0:GOSUB 9000                [CHK CONTROLLER STATE=RESET STATE
5015   PUSH 1: GOSUB 7850                    [ACQUIRE FRAME=YES

5020   Z=0.3: GOSUB 14000                    [WAIT FOR FRAME CAPTURE

5030   PUSH 7:GOSUB 9000                [CHK CONTROLLER STATE=END SIGNATURE HALT

5040   PUSH 0: GOSUB 7880                    [READ SYNC SIGNATURES
5041    POP V1H,V1M,V1L                      [SIG HIGH MEDIUM LOW
5042   PUSH 1: GOSUB 7880                    [READ VIDEO SIGNATURES
5043    POP S1H,S1M,S1L                      [SIG HIGH MEDIUM LOW
5044   P.#: PH0. #"SIG ",V1H,V1M,V1L,S1H,S1M,S1L," ",!


5100   PUSH 0: GOSUB 7850                    [ACQUIRE FRAME=NO
5110   PUSH 0: GOSUB 9000                    [CHK CONTROLLER STATE=RESET STATE
5115   PUSH 1: GOSUB 7850                    [ACQUIRE FRAME=YES

5120   Z=0.3: GOSUB 14000                    [WAIT FOR FRAME CAPTURE
```

```
5130   PUSH 7:GOSUB 9000                [CHK CONTROLLER STATE=END SIGNATURE HALT
5140   PUSH 0: GOSUB 7880                      [READ SYNC SIGNATURES
5141    POP V2H,V2M,V2L                        [SIG HIGH MEDIUM LOW
5142   PUSH 1: GOSUB 7880                      [READ VIDEO SIGNATURES
5143    POP S2H,S2M,S2L                        [SIG HIGH MEDIUM LOW
5144   P.#: PH0. # "SSIG",S2H,S2M,S2L,!
5145   Z=MSGDLY: GOSUB 14000                   [CALL DELAY
5146   P.#: PH0. # "VSIG",V2H,V2M,V2L,!
5147   Z=MSGDLY: GOSUB 14000                   [CALL DELAY

5150   C=0: IF (V1H<>V2H).OR.(V1M<>V2M).OR.(V1L<>V2L) THEN C=1
5151   IF C=1 THEN P. # "SYNC UNSTABLE ",!
5152   Z=MSGDLY: GOSUB 14000                   [CALL DELAY
5153   IF C=1 THEN PUSH 1 ELSE PUSH 0
5154    PUSH 0030, V1H.XOR.V2H, V1M.XOR.V2M, V1L.XOR.V2L

5160   C=0: IF (S1H<>S2H).OR.(S1M<>S2M).OR.(S1L<>S2L) THEN C=1
5161   IF C=1 THEN P. # "SIG UNSTABLE ",!
5162   Z=MSGDLY: GOSUB 14000                   [CALL DELAY
5163   IF C=1 THEN PUSH 1 ELSE PUSH 0
5164    PUSH 0030, S1H.XOR.S2H, S1M.XOR.S2M, S1L.XOR.S2L: GOSUB 1800

         [ ASSEMBLE RESPONSE STRING
         [ :18000040TTSSSSSSVVVVVV;
         [BUILD GOOD RESPONSE PACKET
5170   SH=S1H:SM=S1M:SL=S1L:VH=V1H:VM=V1M:VL=V1L: GOSUB 9300
5299   RETURN


         [MENU SCREEN FOR RESPONSE OF SIGNATURES
5400   PH0. "SYNC  SIGNATURE",SIH,SIM,SIL
5405   PH0. "VIDEO SIGNATURE",VIH,VIM,VIL
5499   RETURN


[**************************************************************************
[*** PROGRAM VIDEO CONTROL REGISTER 1 ROUTINES

    [ XXXX   PUSH X: GOSUB 7700          [PROGRAM TTL VIDEO SOURCE SELECTION
    [ XXXX   PUSH X: GOSUB 7720          [PROGRAM TTL SYNC SOURCE SELECTION
    [ XXXX   PUSH X: GOSUB 7740          [INVERT HORIZONAL/COMPOSITE SYNC
    [ XXXX   PUSH X: GOSUB 7760          [INVERT VERTICAL SYNC
    [ XXXX   PUSH X: GOSUB 7780          [PROGRAM DOT CLOCK SKEW DELAY
    [ XXXX   PUSH X: GOSUB 7800          [PROGRAM SIGNATURE MODE SELECTION
    [ XXXX   PUSH X: GOSUB 7820          [SET/CLEAR TEST VIDEO SIGNAL
    [ XXXX   PUSH X: GOSUB 7840          [SET/CLEAR TEST COMPOSITE SYNC
    [ XXXX   PUSH X: GOSUB 7850          [SET/CLEAR TEST VIDEO SIGNAL
    [ XXXX   GOSUB 7860                  [READ CURRENT CONTROL STATE
    [ XXXX   POP PARM
    [ XXXX   GOSUB 7870                  [READ CURRENT LATCHED VIDEO SIGNALS
    [ XXXX   POP PARM
```

```
    [WRITE DAC CHANNELS FROM DAC VARIABLES
7680  XBY(ADAC+2)= VIDDACLO:              [SET VIDEO DAC HI
      XBY(ADAC+3)= VIDDACHI               [SET VIDEO DAC LO
7682  XBY(ADAC+4)= SYNCDAC:               [SET SYNC  DAC HI
      RETURN

    [PROGRAM TTL VIDEO SOURCE SELECTION LINE
    [  SOURCE    SELECTION
    [     0       TTLRED
    [     1       TTLGRN
    [     2       TTLBLU
    [     3       SECRED
    [     4       SECGRN/INTENSITY
    [     5       SECBLU
    [     6       TTLVERT*
    [     7       TTL HORZ/TTLCMP*
    [    16       ANALOG RED
    [    17       ANALOG GREEN
    [    18       ANALOG BLUE
    [    19       TEST REFERENCE
    [    20       COMPOSITE VIDEO
    [ XXXX   PUSH X: GOSUB 7700          [PROGRAM TTL VIDEO SOURCE SELECTION
    [
7700 POP PARM
7701 IF PARM >=16 THEN 7710
7705   PRGC1=(PRGC1.AND.(255-7)).OR.PARM
7707   XBY(ARGC1)=PRGC1:                          [MULTIPLEXER PROGRAM
     GOTO 7719
7710   C5=0:
       IF PARM=16 THEN C5=8                     [RED ANALOG RELAY CONTROL
7711   IF PARM=17 THEN C5=4                     [GREEN ANALOG RELAY CONTROL
7712   IF PARM=18 THEN C5=0                     [BLUE ANALOG RELAY CONTROL
7713   IF PARM=19 THEN C5=5                     [TEST REFERENCE ANALOG RELAY CONTROL
7714   IF PARM=20 THEN C5=10                    [COMPOSITE ANALOG RELAY CONTROL
7718   PRLY=(PRLY.AND.(255-15)).OR.C5: XBY(ARLY)=PRLY     [RELAY PROGRAM
7719 RETURN


    [PROGRAM TTL SYNC SOURCE SELECTION
    [  SOURCE    SELECTION
    [     0       TTL SEPERATE SYNC
    [     1       TTL COMPOSITE SYNC
    [     2       ANALOG SEPERATE SYNC
    [     3       ANALOG COMPOSITE SYNC
    [     4       ANALOG GREEN SYNC
    [     5       ANALOG TEST MODE
    [     6       TTL TEST MODE
    [     7       not used
    [
    [ XXXX   PUSH X: GOSUB 7720          [PROGRAM TTL SYNC SOURCE SELECTION
    [
7720 POP PARM
```

```
7725   PRGC1=(PRGC1.AND.(255-56)).OR.(PARM*8)
7727   XBY(ARGC1)=PRGC1:
     RETURN

    [INVERT HORIZONAL/COMPOSITE SYNC
    [    0       NOT INVERTED
    [    1       INVERTED
    [
    [ xxxx   PUSH X: GOSUB 7740                    [INVERT HORIZONAL/COMPOSITE SYNC
    [
7740 POP PARM
7745   IF PARM THEN PRGC1=PRGC1.OR.64 ELSE PRGC1=PRGC1.AND.(255-64)
7747   XBY(ARGC1)=PRGC1:
     RETURN

    [INVERT VERTICAL SYNC
    [    0       NOT INVERTED
    [    1       INVERTED
    [
    [ xxxx   PUSH X: GOSUB 7760                    [INVERT VERTICAL SYNC
    [
7760 POP PARM
7765   IF PARM THEN PRGC1=PRGC1.OR.128 ELSE PRGC1=PRGC1.AND.(255-128)
7767   XBY(ARGC1)=PRGC1:
     RETURN

[*************************************************************************
[*** PROGRAM VIDEO CONTROL REGISTER 2 ROUTINES

    [PROGRAM DOT CLOCK SKEW DELAY
    [  SOURCE    SELECTION
    [    0       DOTCLK DELAY SELECTION 0
    [    1       DOTCLK DELAY SELECTION 1
    [    2       DOTCLK DELAY SELECTION 2
    [    3       DOTCLK DELAY SELECTION 3
    [
    [ xxxx   PUSH X: GOSUB 7780                    [PROGRAM DOT CLOCK SKEW DELAY
    [
7780 POP PARM
7785   PRGC2=(PRGC2.AND.(255-3)).OR.PARM
7787   XBY(ARGC2)=PRGC2:
     RETURN


    [PROGRAM SIGNATURE MODE SELECTION
    [  SOURCE    SELECTION
    [    0       ACTIVE HOLD CURRENT REGISTER VALUE
    [    1       ACTIVE ACCUMULATE PIXEL COUNT
    [    2       ACTIVE ACCUMULATE PIXEL SIGNATURE
    [    3       ACTIVE REGISTER SHIFTING OF VALUE
    [    4       FREEZE HOLD CURRENT VALUE
    [    5       FREEZE ACCUMULATE ACTIVE PIXEL COUNT
```

```
    [    6       FREEZE ACCUMULATE ACTIVE PIXEL SIGNATURE
    [    7       FREEZE REGISTER SHIFTING OF VALUE
    [
    [ xxxx   PUSH X: GOSUB 7800          [PROGRAM SIGNATURE MODE SELECTION
    [
7800 POP PARM
7805   PRGC2=(PRGC2.AND.(255-28)).OR.(PARM*4)
7807   XBY(ARGC2)=PRGC2:
     RETURN


    [VIDEO TEST SIGNAL SET/CLEAR
    [    0       CLEAR
    [    1       SET
    [
    [ xxxx   PUSH X: GOSUB 7820          [SET/CLEAR TEST VIDEO SIGNAL
    [
7820 POP PARM
7825   IF PARM THEN PRGC2=PRGC2.OR.64 ELSE PRGC2=PRGC2.AND.(255-64)
7827   XBY(ARGC2)=PRGC2:
     RETURN


    [WRITE VERTICAL INTERVAL COUNTERS
    [
    [ xxxx   PUSH X: GOSUB 7830          [WRITE VERTICAL INTERVAL COUNTER
    [
7830 POP PARM : PARM=65535-PARM:
       XBY(ATMLO)=PARM.AND.255                          [WRITE LOW COUNTER
7834   XBY(ATMHI)=(INT(PARM/256)).AND.255:              [WRITE HIGH COUNTER
     RETURN


    [COMPOSITE SYNC TEST SIGNAL SET/CLEAR
    [    0       CLEAR
    [    1       SET
    [
    [ xxxx   PUSH X: GOSUB 7840          [SET/CLEAR TEST COMPOSITE SYNC
    [
7840 POP PARM
7845   IF PARM THEN PRGC2=PRGC2.OR.128 ELSE PRGC2=PRGC2.AND.(255-128)
7847   XBY(ARGC2)=PRGC2:
     RETURN


    [ACQUIRE FRAME SIGNAL
    [    0       RESET
    [    1       ACQUIRE
    [
    [ xxxx   PUSH X: GOSUB 7850                [ACQUIRE VIDEO FRAME ENABLE
    [
7850 POP PARM
```

```
7855   IF PARM=0 THEN PRGC2=PRGC2.OR.32 ELSE PRGC2=PRGC2.AND.(255-32)
7857   XBY(ARGC2)=PRGC2:
     RETURN


[*****************************************************************************
[*** READBACK OF TEST REGISTER READ PORT

   [READ CURRENT CONTROL STATE INVERTED TO REFECT INTERNAL PAL STATE
   [XXXX GOSUB 7860                      [READ CURRENT CONTROL STATE
   [XXXX POP PARM
7860 PTSR=XBY(ATSR): PARM=(PTSR.AND.15).XOR.15: PUSH PARM
7865 RETURN

   [READ CURRENT LATCHED VIDEO SIGNALS
   [XXXX GOSUB 7870                      [READ CURRENT LATCHED VIDEO SIGNALS
   [XXXX POP PARM
7870 PTSR=XBY(ATSR): PARM=(PTSR.AND.240)/16: PUSH PARM
7875 RETURN

    [READ SIGNATURE'S OF VIDEO OR SYNC COUNTERS
    [    0       VIDEO
    [    1       SYNC
    [
    [ xxxx   PUSH X: GOSUB 7880                [READ VIDEO SIGNATURES
    [ xxxx   POP H,M,L                         [SIG HIGH MEDIUM LOW
    [
7880 POP PARM
7885   PUSH XBY(ASIGX+PARM*3+0)                [READ LOW SIGNATURE
7886   PUSH XBY(ASIGX+PARM*3+1)                [READ MEDIUM SIGNATURE
7887   PUSH XBY(ASIGX+PARM*3+2)                [READ HIGH SIGNATURE
7889 RETURN


[*****************************************************************************
[*** SPECIAL SELF TEST SUPPORT UTILITIES

       [WRITE TEST SYNC & PULSE RESULTS & RESTORE POLARITY
7900 POP CNT,SYNC
7905   IF SYNC=1 THEN XBY(ARGC2)=PRGC2.OR.128 ELSE XBY(ARGC2)=PRGC2.AND.127
                                              [WRITE SYNC LEVEL
7910   FOR C=1 TO CNT
7915     XBY(ATSCK)=0                         [PULSE TEST CLOCK CNT REPETITIONS
7920   NEXT C
7925   XBY(ARGC2)=PRGC2                       [RESTORE SYNC POLARITY
7930 RETURN


       [WRITE VIDEO & PULSE RESULTS & RESTORE POLARITY
7950 POP CNT,VIDDAT: K=0
7960   FOR C= 1 TO CNT
7965     VDB = VIDDAT.AND.2**K                [PICK VID DATA BIT
```

```
7970     IF VDB THEN XBY(ARGC2)=PRGC2.OR.64 ELSE XBY(ARGC2)=PRGC2.AND.191
7975     XBY(ATSCK)=0                          [PULSE TEST CLOCK CNT
7980     K=K+1: IF K=16 THEN K=0               [INCREMENT BIT COUNTER
7990   NEXT C
7991   XBY(ARGC2)=PRGC2                        [RESTORE ENTRY VIDEO LEVEL
7999 RETURN


       [POWER-UP SELF TEST ]
8000 PRINT #:PRINT # "POWER-UP SELFTEST  ",!
8005  FOR C= 32 TO 126
8006    FOR Z1=0 TO 60:NEXT Z1
8007    PRINT # CHR(C),!
8008  NEXT C
8099 RETURN

       [ANALOG VIDEO SELF TEST ]
8100 P.#: PRINT # "ANALOG TEST",!
8110 FOR Z1= 0 TO 1200: NEXT Z1
8499 RETURN

       [DIGITAL VIDEO FRAME SELFTEST}
8500 P.#: PRINT # CHR(19),"DIGITAL TEST VF ",!
8501 VERT=0: HORZ=0: VERHORZ=1
8504 DAT=1455                          [SET VIDEO DATA TEST PATTERN

   [PROGRAM NON-CRITICAL SETUPS
8505   PUSH 0: GOSUB 7740              [INVERT HORIZONAL/COMPOSITE SYNC= NON-INVERTED
8506   PUSH 0: GOSUB 7760              [INVERT VERTICAL SYNC= NON-INVERTED
8507   PUSH 0: GOSUB 7780              [PROGRAM DOT CLOCK SKEW DELAY= DLYSEL 0
8508   PUSH 0: GOSUB 7700              [PROGRAM TTL VIDEO SOURCE=RED DON'T CARE

   [PROGRAM ORDER CRITICAL SETUPS
8509   PUSH 6: GOSUB 7720              [PRG TTL SYNC SOURCE=TTL TEST/SELF-CLOCKED
8510   PUSH 1: GOSUB 7820              [SET/CLEAR TEST VIDEO SIGNAL
8511   PUSH 1: GOSUB 7840              [SET/CLEAR TEST COMPOSITE SYNC
8513   PUSH 1: GOSUB 7800              [PROGRAM SIGNATURE MODE=COUNT

   [TEST READBACK OF LATHORZ,LATVIDEO,LATCOMP,LATVERT
8515   FOR W=0 TO 1
8516     FOR Y=0 TO 1
8517       PUSH W: GOSUB 7820          [SET/CLEAR TEST VIDEO SIGNAL
8518       PUSH Y: GOSUB 7840          [SET/CLEAR TEST COMPOSITE SYNC
8519       XBY(ATSCK)=0                [PULSE TEST CLOCK CNT
8520       GOSUB 7870                  [READ CURRENT LATCHED VIDEO SIGNALS
8521        POP PARM
8522       PARM1=Y*8+Y*4+Y*2+W
8523       IF PARM<>PARM1 THEN PUSH 1 ELSE PUSH 0
8524        PUSH 8523,0,PARM1,PARM: GOSUB 1800
8525     NEXT Y
8526   NEXT W
```

```
[CHECK VIDEO FRAME SEQUENCING AND SIGNATURE GENERATOR SHIFTING

[CHECK SIGNATURE GENERATOR SHIFTING
8700   PUSH 1: GOSUB 7820                [SET/CLEAR TEST VIDEO SIGNAL
8702   PUSH 1: GOSUB 7840                [SET/CLEAR TEST COMPOSITE SYNC
8704   PUSH 2: GOSUB 7800                [PROGRAM SIGNATURE MODE=SIGNATURE MODE
8705   PUSH 10: GOSUB 7830               [WRITE VERTICAL INTERVAL COUNTER
8706   PUSH 0: GOSUB 7850                [ACQUIRE VIDEO FRAME RESET
8707   XBY(ATSCK)=0                      [PULSE TEST CLOCK CNT

8710   PUSH 8710,255,255,255,255,255,255: GOSUB 9100 [CHECK SYNC/VIDEO SIGNATURE

8714   PUSH 1: GOSUB 7850                [ACQUIRE VIDEO FRAME ENABLE
8715   XBY(ATSCK)=0                      [PULSE TEST CLOCK CNT

8719 PUSH 1:GOSUB 9000                   [CHK CONTROLLER STATE=SYNCHRONIZE

[**** END VIDEO PORTION OF PREVIOUS FRAME

8720 PUSH 1,20: GOSUB 7900               [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8722 PUSH 0,5: GOSUB 7900                [GENERATE HORIZONAL SYNC

8729 PUSH 2:GOSUB 9000                   [CHK CONTROLLER STATE=UNKNOWN SYNC START HORZ

8730 PUSH 1,20: GOSUB 7900               [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8732 PUSH 0,5: GOSUB 7900                [GENERATE HORIZONAL SYNC

8739 PUSH 4:GOSUB 9000                   [CHK CONTROLLER STATE=UNKNOWN SYNC START HORZ

8740 PUSH 1,20: GOSUB 7900               [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8742 PUSH 0,5: GOSUB 7900                [GENERATE HORIZONAL SYNC

8749 PUSH 6:GOSUB 9000                   [CHK CONTROLLER STATE=UNKNOWN SYNC START HOR/VER

[**** END VERTICAL SYNC OF LAST FRAME

8750 PUSH 8750,255,255,255,255,255,255: GOSUB 9100 [CHECK SYNC/VIDEO SIG RESET

8751 PUSH 1,20: GOSUB 7900               [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
                                         [VERTICAL RETRACE
8752 PUSH 0: GOSUB 7840                  [SET/CLEAR TEST COMPOSITE SYNC=VERTICAL
8754 PUSH 0,20: GOSUB 7900               [GENERATE VERTICAL SYNC CLOCKS
8756 PUSH 1,5: GOSUB 7900                [SUPERIMPOSE HORIZONAL SYNC IN VERTICAL

8759 PUSH 8:GOSUB 9000                   [CHK CONTROLLER STATE=1ST VERTICAL SYNC FOUND

8760 PUSH 0,20: GOSUB 7900               [GENERATE VERTICAL SYNC CLOCKS
8762 PUSH 1,5: GOSUB 7900                [SUPERIMPOSE HORIZONAL SYNC IN VERTICAL

8769 PUSH 8:GOSUB 9000                   [CHK CONTROLLER STATE=2ND VERTICAL SYNC FOUND

8770 PUSH 0,20: GOSUB 7900               [GENERATE VERTICAL SYNC CLOCKS
```

```
8772 PUSH 1,5: GOSUB 7900                    [SUPERIMPOSE HORIZONAL SYNC IN VERTICAL


8779 PUSH 8:GOSUB 9000                       [CHK CONTROLLER STATE=3RD VERTICAL SYNC FOUND

[START NEW VIDEO FRAME
8780 PUSH 0,20: GOSUB 7900                   [GENERATE VERTICAL SYNC CLOCKS

8781 PUSH 8781,255,255,255,57,183,52: GOSUB 9100 [CHK SYNC/VIDEO SIG END VERT

8782 PUSH 0: GOSUB 7820                      [SET/CLEAR TEST VIDEO SIGNAL
8783 PUSH 1: GOSUB 7840                      [SET/CLEAR TEST COMPOSITE SYNC=VIDEO
8784 PUSH 1,20: GOSUB 7900                   [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8786 PUSH 0,5: GOSUB 7900                    [GENERATE HORIZONAL SYNC

8787 PUSH 9:GOSUB 9000                       [CHK CONTROLLER STATE=1ST UNKNOWN SYNC/HORZ

8789 PUSH 1: GOSUB 7820                      [SET/CLEAR TEST VIDEO SIGNAL
8790 PUSH 1,20: GOSUB 7900                   [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8792 PUSH 0,5: GOSUB 7900                    [GENERATE HORIZONAL SYNC

8797 PUSH 11:GOSUB 9000                      [CHK CONTROLLER STATE=2ND UNKNOWN SYNC/HORZ

8799 PUSH 0: GOSUB 7820                      [SET/CLEAR TEST VIDEO SIGNAL
8800 PUSH 1,20: GOSUB 7900                   [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8802 PUSH 0,5: GOSUB 7900                    [GENERATE HORIZONAL SYNC

8804 PUSH 1: GOSUB 7820                      [SET/CLEAR TEST VIDEO SIGNAL
8809 PUSH 11:GOSUB 9000                      [CHK CONTROLLER STATE=3RD UNKNOWN SYNC/HORZ


[**** START VERTICAL SYNC OF CURRENT FRAME

8810 PUSH 1,20: GOSUB 7900                   [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
                                             [VERTICAL RETRACE

8811 PUSH 8811,161,197,128,35,52,145: GOSUB 9100   [CHK SYNC/VIDEO SIG END VID

8812 PUSH 0: GOSUB 7840                      [SET/CLEAR TEST COMPOSITE SYNC=VERTICAL
8814 PUSH 0,20: GOSUB 7900                   [GENERATE VERTICAL SYNC CLOCKS
8816 PUSH 1,5: GOSUB 7900                    [SUPERIMPOSE HORIZONAL SYNC IN VERTICAL

      [**** SIGNATURE STOP
8819 PUSH 7:GOSUB 9000                       [CHK CONTROLLER STATE=END SIGNATURE HALT

8820 PUSH 8820,53,100,50,217,44,98: GOSUB 9100    [CHK SYNC/VIDEO SIG END ACQ

8821 PUSH 0,20: GOSUB 7900                   [GENERATE VERTICAL SYNC CLOCKS
8822 PUSH 1,5: GOSUB 7900                    [SUPERIMPOSE HORIZONAL SYNC IN VERTICAL

8827 PUSH 7:GOSUB 9000                       [CHK CONTROLLER STATE=END SIGNATURE HALT
```

```
8829 PUSH 0: GOSUB 7820                    [SET/CLEAR TEST VIDEO SIGNAL
8830 PUSH 0,20: GOSUB 7900                 [GENERATE VERTICAL SYNC CLOCKS
8832 PUSH 1,5: GOSUB 7900                  [SUPERIMPOSE HORIZONAL SYNC IN VERTICAL

8837 PUSH 7:GOSUB 9000                     [CHK CONTROLLER STATE=END SIGNATURE HALT

[**** CHECK HALT MAINTAINED FOR NEW FRAME

8839 PUSH 1: GOSUB 7820                    [SET/CLEAR TEST VIDEO SIGNAL
8840 PUSH 0,20: GOSUB 7900                 [GENERATE VERTICAL SYNC CLOCKS
8842 PUSH 1: GOSUB 7840                    [SET/CLEAR TEST COMPOSITE SYNC=VIDEO

8843 PUSH 0: GOSUB 7820                    [SET/CLEAR TEST VIDEO SIGNAL
8844 PUSH 1,20: GOSUB 7900                 [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8846 PUSH 0,5: GOSUB 7900                  [GENERATE HORIZONAL SYNC

8849 PUSH 7:GOSUB 9000                     [CHK CONTROLLER STATE=END SIGNATURE HALT

8850 PUSH 1,20: GOSUB 7900                 [GENERATE VIDEO CLOCKS WITH SYNC GENERATOR
8852 PUSH 0,5: GOSUB 7900                  [GENERATE HORIZONAL SYNC

8859 PUSH 7:GOSUB 9000                     [CHK CONTROLLER STATE=END SIGNATURE HALT

8870 PUSH 8870,53,100,50,217,44,98: GOSUB 9100    [CHK SYNC/VIDEO SIG HOLD CHK

8878 XBY(ATSCK)=0                          [PULSE TEST CLOCK CNT
8880 PUSH 7:GOSUB 9000                     [CHK CONTROLLER STATE=END SIGNATURE HALT

8890 PUSH 8885,53,100,50,217,44,98: GOSUB 9100    [CHECK SYNC/VIDEO SIGNATURE
8891 GOSUB 9300                                   [BUILD GOOD SIGNATURE RESPONSE
8920   RETURN


       [READ & CHECK CONTROLLER STATE
9000  POP PAR2                                    [READ CURRENT CONTROL STATE
9005   GOSUB 7860
9006    POP PARM
9008   P.#: PH0.# "CS:",PARM,"=",PAR2," ",!
9009   Z=MSGDLY2: GOSUB 14000                      [CALL DELAY
9010   IF PARM<>PAR2 THEN PUSH 1 ELSE PUSH 0
9011        PUSH 9010,PAR2,PARM,0: GOSUB 1800
9019  RETURN

       [READ & CHECK SYNC/VIDEO SIGNATURES
9100   POP V1(6),V1(5),V1(4),V1(3),V1(2),V1(1),EC
9105    PUSH 0: GOSUB 7880                         [READ RESET SYNC SIGNATURES
9106     PUSH 1: GOSUB 7880                        [READ RESET VIDEO SIGNATURES
9110    POP V2(6),V2(5),V2(4),V2(3),V2(2),V2(1)
9111    VH=V2(3):VM=V2(2):VL=V2(1):SH=V2(6):SM=V2(5):SL=V2(4)
9112    P.#: PH0.# "SSIG",SH,SM,SL,!
9113    Z=MSGDLY: GOSUB 14000                      [CALL DELAY
9114    P.#: PH0.# "VSIG",VH,VM,VL,!
```

```
9115   Z=MSGDLY: GOSUB 14000                    [CALL DELAY
9120   C2=0
9121   FOR C=1 TO 3
9125     IF V1(C)<>V2(C) THEN C2=1
9129   NEXT C
9130   IF C2=1 THEN PUSH 1 ELSE PUSH 0
9131     PUSH EC+C,V1(3)*256+V2(3),V1(2)*256+V2(2),V1(1)*256+V2(1)
9132     GOSUB 1800
9140   C2=0
9141   FOR C=1 TO 3
9145     IF V1(C)<>V2(C) THEN C2=1
9149   NEXT C
9150   IF C2=1 THEN PUSH 1 ELSE PUSH 0
9151     PUSH EC+C,V1(3)*256+V2(3),V1(2)*256+V2(2),V1(1)*256+V2(1)
9152     GOSUB 1800
9199   RETURN


       [ ASSEMBLE SIGNATURE RESPONSE STRING
       [ :18000040TTSSSSSSVVVVVV;
9300   $(3)=":18000040"

9305   PUSH X6,2,16: GOSUB 3100                 [CONVERT TEST NUMBER TO ASCII
9306   ASC($(3),10)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),11)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9310   PUSH X7,2,16: GOSUB 3100                 [CONVERT TEST NUMBER TO ASCII
9311   ASC($(3),12)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),13)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9315   PUSH SH,2,16: GOSUB 3100                 [CONVERT SYNC SIG HIGH TO ASCII
9316   ASC($(3),14)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),15)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9320   PUSH SM,2,16: GOSUB 3100                 [CONVERT SYNC SIG MED TO ASCII
9321   ASC($(3),16)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),17)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9325   PUSH SL,2,16: GOSUB 3100                 [CONVERT SYNC SIG LOW TO ASCII
9326   ASC($(3),18)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),19)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9330   PUSH VH,2,16: GOSUB 3100                 [CONVERT VIDEO SIG HIGH TO ASCII
9331   ASC($(3),20)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),21)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9335   PUSH VM,2,16: GOSUB 3100                 [CONVERT VIDEO SIG MID TO ASCII
9336   ASC($(3),22)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
       ASC($(3),23)=ASC($(0),2)                 [ASSEMBLE RESPONSE 2ND DIGIT

9340   PUSH VL,2,16: GOSUB 3100                 [CONVERT VIDEO SIG LOW TO ASCII
9341   ASC($(3),24)=ASC($(0),1):                [ASSEMBLE RESPONSE 1ST DIGIT
```

```
      ASC($(3),25)=ASC($(0),2)                [ASSEMBLE RESPONSE 2ND DIGIT

9345  ASC($(3),26)=ASC(;)                     [ADD TERMINATION SEMICOLON
9349  RETURN

            [ PERIOD MEASUREMENT ROUTINE]
 [ PERIOD MEASUREMENT CALLING SETUP
 [ ln   PUSH 0:                          [SETUP:M=CH
 [ ln+1 GOSUB 10200
 [

10200 POP C                                   [PULL FREQ.PARAMETERS
10205 P7=(P7.AND.248)+C: XBY(D1+0)=P7         [SELECT MEASUREMENT CHANNEL
10210 PUSH 2**C,2,0,0 : GOSUB 15000           [SET DOWN TRIGGER RELAY
10220 XBY(T+3)= 112 :                         [70H COMMAND WORD:PROG. COUNT MODE
      XBY(T+1)= 255 :                         [LS BYTE
      XBY(T+1)= 255                           [MS BYTE
10230 PUSH 255,0,0,0 : GOSUB 15000            [CLEAR DOWN RELAYS
10260 FOR Z1 = 0 TO 666: NEXT Z1              [DELAY 1.5ms/ct
10265 XBY(T+3)= 40                            [LATCH CTR.1
10270 F1= XBY(T+0)                            [LS BYTE OF COUNTER READ
10280 F2=(65535-F1-(XBY(T+0)*256))/10:PUSH F2 [FFFF-LS-MS BYTE=CYCLES MEASURED
10290 RETURN



      [ CLOCK DIVIDER SETUP ROUTINE ]
 [ln   GOSUB 10400                        [SETUP FREQUENCY RANGE FOR COUNTER
10400 XBY(T+3)= 54                        [COMMAND WORD SQUAREWAVE GENERATOR
10430 XBY(T+2)= 100: XBY(T+2)=0           [LS,MS BYTE OF COUNTER WRITTEN
                                          [10KHZ OUTPUT FREQ.
10440 FOR Z1=0 TO 10 : NEXT Z1            [DELAY 15ms WORST CASE
10450 RETURN                              [CALLED FROM NEW UUT INIT.1410

         [CONTROL STACK POP OF 3 LEVELS THUS 1 GOSUB]
13200   DBY(17)=DBY(17)-3                 [ADJ.C-STACK 1 LEVEL TO ABORT RETURN
13210   RETURN


         [** DELAY ROUTINE 1sec TIMEBASE
         [xxxx       Z=1.0: GOSUB 14000
14000   Z1=TIME+Z-0.034                         [TIME DELAY ROUTINE
14010   IF TIME<Z1 THEN 14010 ELSE RETURN


65000 REM" LAST LINE OF DOWNLOAD"
65001 REM ""
```

BASIC-52 Software Development Environment Listing

```
DECLARE SUB KEYWAIT (N%, W$)
DECLARE SUB SELWDOW (wdnum%, wd%(), clrwd%, HOME%, lit%, title$)
DECLARE FUNCTION FORMAT$ (S$, cmdcol%, cmdlen%, EDIT%)
DECLARE FUNCTION CHARIN$ (S$, EDIT%)
DECLARE FUNCTION SHEXDEC% (S$, cmdcol%, fldlen%)
DECLARE FUNCTION SENDRCV$ (S$, stat%)
DECLARE FUNCTION PPSSDEF$ (PPSS$, FFLAG%)
DECLARE FUNCTION DIAGDEF$ (PPSS$, diag2$, FFLAG%)
DECLARE FUNCTION VECTOR$ (vecnum%, UUT%)
DECLARE FUNCTION STRGIN$ (FILENUM%, char%)
DECLARE FUNCTION GETSCN$ (toprow%, botrow%, lefcol%, rhtcol%, scnrow%, scncol%)
DECLARE SUB SETGOSB (KN!, B$)
DECLARE SUB SETFCTN (STK!, A$())
DECLARE SUB RTNFCTN (STK!, A$())
DECLARE FUNCTION CMDRSP$ (S$, t$, EDIT%)

DIM SHARED quitterm, col, row, QUIT, port AS INTEGER
DIM endoffile, TROW, TCOL  AS INTEGER
DIM wd%(5, 5)
CONST yes% = 1, no% = 0, attempt = 3
CONST wlabl% = 0, wtop% = 1, wbot% = 2, wfore% = 3, wback% = 4 'DISPLAY WINDOW PARAMETER LABELS
CONST wcmd% = 0, wterm% = 1, wmsg% = 2                         'WINDOW NAMES LABELS
'COMMON SHARED cr$, lf$, point$
COMMON SHARED POINT$

'****************
 fmax% = 80
 bmax% = 80
 DIM docstr(80) AS STRING
 DIM rechar(1) AS STRING
 DIM basstr(fmax%) AS STRING
'****************
 DIM errstr(80) AS STRING
 DIM tmpstr(80) AS STRING
 DIM echo(80) AS STRING
'****************

DIM A$(15, 9)
CLEAR , , 8000

rarrow$ = CHR$(0) + CHR$(77): larrow$ = CHR$(0) + CHR$(75)
uarrow$ = CHR$(0) + CHR$(72): darrow$ = CHR$(0) + CHR$(80)
ins$ = CHR$(0) + CHR$(82): del$ = CHR$(0) + CHR$(83)
col = 80: row = 43
SHOWERR = 1: portbuf = 1024: srecdly = 700
echost = 0
retry% = attempt
SEL$ = "": LSTSEL$ = ""
```

```
 '************* SET WINDOW DIMENSIONS ************************
cr$ = CHR$(13): lf$ = CHR$(10): crlf$ = cr$ + lf$
'SCREEN 0
PRINT " "

GOSUB RDDEFA                          'READ DEFAULT PROGRAM CONFIGURATION

OPEN cport$ FOR RANDOM AS #1 LEN = portbuf

IF SHOWERR THEN ON ERROR GOTO ERRORHDL
RESTART:
QUIT = 0: RESTART = 0
STK = 1
 '************* SET WINDOW ************************
SCREEN 0, , 0, 0

 '************* BEGIN MAIN PROGRAM ****************************
A$(0, 0) = "TERM": A$(0, 1) = "EDIT": A$(0, 2) = "EXTRACT": A$(0, 3) = "DOWNLD": A$(0, 4) =
"SETFILE": A$(0, 5) = "EXTEMP": A$(0, 6) = "": A$(0, 7) = "LIBRARY": A$(0, 8) = "CALLDOS": A$(0, 9)
= "QUIT"
CALL SETFCTN(STK, A$())



GOSUB CLRWDOW

MAIN:
DO
LOOP UNTIL INKEY$ = "Q" OR QUIT = 1
IF RESTART = 1 THEN GOTO RESTART
CLOSE #1
END


 '************* END MAIN PROGRAM *****************************

CLRWDOW:
  CLS
  WIDTH col, row
  TROW = wd%(wterm%, wtop%): TCOL = 1
  CALL SELWDOW(wmsg%, wd%(), yes%, yes%, yes%, " ")
  CALL SELWDOW(wterm%, wd%(), yes%, yes%, yes%, " ")
  RETURN



ERRORHDL:
   SELECT CASE ERR
     CASE 57                          'SERIAL COMM PORT ERROR OCCURRED
       RESUME
     CASE 69                          'SERIAL COMM buffer full
       PRINT CHR$(1);
```

```
      RESUME
    CASE 64                          'BAD FILENAME
      PRINT "BAD FILENAME ERROR"
      IF quitterm THEN RESUME abortld
      IF ERRL = DNLINE THEN RESUME DNINPUT
'***BAD PROCESSING FOR ILLEGAL FILENAMES
    CASE ELSE
      PRINT "UNRECOVERABLE ERROR ="; ERR
      PRINT "ERROR ON LINE NUMBER="; ERL

      ON ERROR GOTO 0
    END SELECT
  STOP



TERM:
  A$(0, 0) = "ECHO": A$(0, 1) = "": A$(0, 2) = "": A$(0, 3) = "": A$(0, 4) = "": A$(0, 5) = "":
A$(0, 6) = "": A$(0, 7) = "": A$(0, 8) = "": A$(0, 9) = "EXIT"
  CALL SETFCTN(STK, A$())
  CALL SELWDOW(wterm%, wd%(), no%, no%, yes%, "DUMB TERMINAL EMULATOR")
  CLS : TROW = 2: TCOL = 1
  LOCATE TROW, TCOL, 1

  quitterm = 0

  DO
    iky$ = INKEY$
    IF iky$ <> "" THEN
      IF echost THEN PRINT iky$;
      PRINT #1, iky$;
    END IF
    IF NOT EOF(1) THEN
      INTERM$ = INPUT$(LOC(1), #1)
      DO
        c = INSTR(1, INTERM$, lf$): IF c <> 0 THEN MID$(INTERM$, c, 1) = " "
      LOOP UNTIL c = 0
      PRINT INTERM$;
    END IF
  LOOP UNTIL quitterm
  quitterm = 0: TROW = CSRLIN: TCOL = POS(0)
  CALL SELWDOW(wmsg%, wd%(), no%, no%, yes%, "")
  CALL RTNFCTN(STK, A$())
  QUIT = 0: RETURN

QUIT:
  PRINT "EXIT SMH PROGRAM"
  INPUT "EXIT CONFIRM (Y/N)", B$
  IF B$ = "Y" OR B$ = "y" THEN QUIT = 1
  RETURN

EXITTERM:
```

```
   POINT$ = "": quitterm = 1: QUITFCTN = 1: EDIT% = no%: RETURN

echo:
   IF echost THEN echost = 0 ELSE echost = 1
   RETURN

READY:
   quitterm = 1: EDIT% = no%: RETURN

DOS:
   FOR c = 0 TO 9: A$(0, c) = "": NEXT
   CALL SETFCTN(STK, A$())
   CALL SELWDOW(wmsg%, wd%(), yes%, yes%, yes%, "DOS WINDOW")
   SHELL
   CLS 2: CALL RTNFCTN(STK, A$())
   RETURN


' basic52 documentation extract routine
EXTEMP:
   A$(0, 0) = "": A$(0, 1) = "": A$(0, 2) = "": A$(0, 3) = "": A$(0, 4) = "": A$(0, 5) = "": A$(0,
6) = "": A$(0, 7) = "": A$(0, 8) = "": A$(0, 9) = ""
   CALL SETFCTN(STK, A$())

 CALL SELWDOW(wterm%, wd%(), yes%, yes%, yes%, "BASIC52 EXTRACTION")
 SHELL "COPY " + sdir$ + "TEMP.PRG" + " " + sdir$ + SRCFILE$
 GOTO EXBYPASS

' basic52 documentation extract routine
EXTRACT:
   A$(0, 0) = "": A$(0, 1) = "": A$(0, 2) = "": A$(0, 3) = "": A$(0, 4) = "": A$(0, 5) = "": A$(0,
6) = "": A$(0, 7) = "": A$(0, 8) = "": A$(0, 9) = ""
   CALL SETFCTN(STK, A$())

 CALL SELWDOW(wterm%, wd%(), yes%, yes%, yes%, "BASIC52 EXTRACTION")
 SHELL "COPY " + sdir$ + PRGFILE$ + " " + sdir$ + SRCFILE$

EXBYPASS:
 fmax% = 80
 bmax% = 80
 basstr$ = ""

 OPEN SRCFILE$ FOR INPUT AS #3
 OPEN dnldfile$ FOR OUTPUT AS #2
 baspos% = 1
 newlreq% = 1

 DO
   endofcmd% = 0
   extend% = 0
   textmode% = 0
   posinline% = 1
```

```
   baspos% = 1

   DO
     IF newlreq% THEN GOTO readline ELSE GOTO rdchar

readline:
     LINE INPUT #3, docstr$
     PRINT docstr$
     newlreq% = 0: posinline% = 1
rdchar:
     rechar$ = MID$(docstr$, posinline%, 1): posinline% = posinline% + 1
     newlflg% = 0

     SELECT CASE rechar$
     CASE " ", CHR$(9)
       GOSUB space
     CASE "@"
       extend% = 1
     CASE "!"
       extend% = 0
     CASE ",", ":"
       GOSUB cont
     CASE "["
       GOSUB comment
     CASE CHR$(34)
       GOSUB text
     CASE ""
       GOSUB null
     CASE ELSE
       GOSUB char
     END SELECT

      IF posinline% <= 80 THEN GOTO jmp1
        newlreq% = 1: posinline% = 1: contspc% = 0: endofcmd% = 1
        IF extend THEN
          endofcmd% = 0
        ELSE
          baspos% = 1
        END IF
jmp1:

   LOOP WHILE endofcmd% = 0
   IF LEN(basstr$) > 1 THEN PRINT #2, basstr$
   basstr$ = ""
 LOOP WHILE EOF(3) = 0
 CLOSE #3, #2
 CALL RTNFCTN(STK, A$())
 RETURN


space:
```

```
    IF contspc% = 1 THEN
      contspc% = 0
      basstr$ = basstr$ + " ": baspos% = baspos% + 1
    END IF
    RETURN
cont:
    extend% = 1: contspc% = 1
    basstr$ = basstr$ + rechar$: baspos% = baspos% + 1
    RETURN
comment:
    newlreq% = 1
    IF extend% = 1 THEN RETURN
      baspos% = 1
      textmode% = 0: endofcmd% = 1
    RETURN
text:
    contspc% = 1: extend% = 0: endofcmd% = 0
    basstr$ = basstr$ + rechar$: baspos% = baspos% + 1
    textmode% = 1
    DO
      rechar$ = MID$(docstr$, posinline%, 1)
      IF rechar$ = CHR$(34) THEN
        GOSUB textend
      ELSEIF rechar$ = "" THEN
        GOSUB textnull
      ELSE
        GOSUB textchar
      END IF
    LOOP WHILE textmode% = 1
    RETURN
textend:
    textmode% = 0
    basstr$ = basstr$ + rechar$: baspos% = baspos% + 1
    posinline% = posinline% + 1
    RETURN
textnull:
    baspos% = 1
    posinline% = 1
    textmode% = 0
    endofcmd% = 1
    RETURN
textchar:
    basstr$ = basstr$ + rechar$: baspos% = baspos% + 1
    posinline% = posinline% + 1
    RETURN
null:
    newlreq% = 1
    posinline% = 1: contspc% = 0: endofcmd% = 1
    IF extend% THEN
      endofcmd% = 0
    ELSE
      baspos% = 1
```

```
    END IF
    RETURN
char:
    extend% = 0: contspc% = 1
    basstr$ = basstr$ + rechar$: baspos% = baspos% + 1
    RETURN


BLDCMD:
  A$(0, 0) = "SEND": A$(0, 1) = "": A$(0, 2) = "": A$(0, 3) = "": A$(0, 4) = "": A$(0, 5) = "":
A$(0, 6) = "": A$(0, 7) = "": A$(0, 8) = "": A$(0, 9) = "EXIT"
  CALL SETFCTN(STK, A$())
  quitterm = 0: QUIT = 0
  CALL RTNFCTN(STK, A$())
  RETURN

retry:
  RETURN
NOP:
  RETURN


SEND:
  QUIT = 0: quitterm = 1: EDIT% = no%
  RETURN

SENDCMD:
 RETURN


CLRBUF:
  CALL SELWDOW(wterm%, wd%(), no%, no%, yes%, "")
  LOCATE TROW, TCOL, 1
  DO:
    IF NOT EOF(1) THEN
      INTERM$ = INPUT$(LOC(1), #1)
      PRINT INTERM$;
    END IF
    FOR c = 0 TO 10000: NEXT
  LOOP UNTIL EOF(1)
  quitterm = 0: TROW = CSRLIN: TCOL = POS(0)
  RETURN


DOWNLOAD:
 A$(0, 0) = "": A$(0, 1) = "": A$(0, 2) = "": A$(0, 3) = "": A$(0, 4) = "": A$(0, 5) = "": A$(0, 6)
= "": A$(0, 7) = "": A$(0, 8) = "": A$(0, 9) = ""
 CALL SETFCTN(STK, A$())
 CALL SELWDOW(wmsg%, wd%(), yes%, yes%, yes%, "DOWNLOAD MESSAGE")
 PRINT "DOWNLOAD PROGRAM LINES WILL BE DISPLAYED"
 PRINT "BASIC52 BAD ECHO LINES WILL BE DISPLAYED"
 CALL SELWDOW(wterm%, wd%(), yes%, yes%, yes%, "DOWNLOAD MESSAGE")
```

```
REM basic52 download routine
REM ROBERT M. BASS
REM TEST ENGINEERING
PRINT "DOWNLOAD STARTED "
basstr$ = "": tmpstr$ = ""
infile$ = sdir$ + dnldfile$
outfile$ = sdir$ + errfile$
OPEN infile$ FOR INPUT AS #4  'data source
OPEN outfile$ FOR OUTPUT AS #2     'error file
FOR X = 1 TO 5
  PRINT #1, " ",
  FOR z1 = 0 TO 200: NEXT z1
NEXT X
PRINT #1,
FOR z1 = 0 TO 300: NEXT z1
PRINT #1, "RAM"
CLOSE #1

FOR z1 = 0 TO 300: NEXT z1
LINE INPUT #4, basstr$                'PRIMING READ

OPEN cport$ FOR RANDOM AS #1

DO
     tmpstr$ = INPUT$(LOC(1), #1)               'purge message buffer
     il% = LEN(basstr$): PRINT #1, basstr$
     tmpstr$ = "": LINE INPUT #4, tmpstr$      'DELAY/PREFETCH NEXT STRING

     abrt = 0: rl% = 0
     DO
      rl% = LOC(1)
      abrt = abrt + 1
      IF abrt > abortdly THEN
        PRINT "**** receive timeout abort ****"
        PRINT #2, "**** receive timeout abort ****"
        GOTO QUITDNLD
      END IF
     LOOP WHILE rl% < (il% + 3)

     echo$ = INPUT$(LOC(1), #1)                'STORE RETURNED MESSAGE
     IF MID$(echo$, il% + 5, 1) <> ">" AND MID$(echo$, il% + 3, 1) <> ">" THEN
       FOR abrt = 0 TO 300: NEXT abrt          'WAIT FOR MESSAGE RESIDUE
       echo$ = INPUT$(LOC(1), #1)              'UPDATE RETURNED MESSAGE
       PRINT "**** data echo failure ****"
       PRINT basstr$
       PRINT echo$
       PRINT #2, "**** data echo failure ****"
       PRINT #2, basstr$
       PRINT #2, echo$
     END IF
     PRINT echo$;
     basstr$ = "": basstr$ = tmpstr$
```

```
 LOOP WHILE NOT EOF(4)
 PRINT "TRANSFER COMPLETED NORMALLY"
QUITDNLD:
 CLOSE #4, #2
 CALL RTNFCTN(STK, A$())
 RETURN




DNLDSREC:
DNINPUT:
   INPUT "Filename to download= <filename>.no extension ", A$
DNFIXED:
   QUIT = 0

   srecfile$ = A$ + ".run"
   errfile$ = A$ + ".err"


DNLINE:
   OPEN srecfile$ FOR INPUT AS #4          'data source
   OPEN errout$ FOR OUTPUT AS #2          'error file
   PRINT "DOWNLOAD STARTED "
   basstr$ = "": tmpstr$ = "": echo$ = ""

   LINE INPUT #4, basstr$                  'PRIMING READ
   basstr$ = basstr$ + CHR$(13)            'ADD CR
   ON COM(port) GOSUB BLDECHO              'SET RECEIVE INTERRRUPT SERVICE

 DO
      QUIT = 0: echo$ = "": msg$ = "": fault% = 0
      endoffile = EOF(4)

      IF endoffile THEN GOTO skipread
       tmpstr$ = "": LINE INPUT #4, tmpstr$      'READ NEXT LINE
       tmpstr$ = tmpstr$ + CHR$(13)              'ADD CR

skipread:
      QUIT = 0
      COM(port) ON                          'ACTIVATE RECEIVE INTERRUPT ROUTINE
      il% = LEN(basstr$): PRINT #1, basstr$, 'SEND S-RECORD

      abrt = 0
      DO
       abrt = abrt + 1
       IF abrt > 10000 THEN
          PRINT "**** receive timeout abort ****"
          PRINT #2, "**** receive timeout abort ****"
          QUIT = 1
          fault% = fault% + 1
```

```
        PRINT #2, basstr$, echo$
        PRINT basstr$, echo$
      END IF
    LOOP WHILE QUIT = 0

' DELAY FOR UUT PROCESSING TIME
    FOR X = 0 TO srecdly: NEXT X

' CHECK FOR TIMEOUT FAILURE
'     IF fault% > 0 AND fault% < 3 THEN GOTO skipread 'attempt retry of last srecord
    IF QUIT = 1 THEN GOTO QUITLD            'TIMEOUT ABORT ENCOUNTERED

    START = INSTR(1, echo$, "S")            'CHECK FOR START OF S-RECORD ECHO
    SEND = INSTR(1, echo$, cr$)       'CHECK FOR END OF S-RECORD ECHO
    IF START = 0 OR SEND = 0 THEN GOTO MISSING

    msg$ = MID$(echo$, START, SEND - START + 1)  'BUILD ECHO STRING
MISSING:

    IF LEFT$(msg$, il% - 0) <> LEFT$(basstr$, il% - 0) THEN          'ECHO FAILURE
      PRINT "**** data echo failure ****"
      PRINT basstr$
      PRINT msg$
      fault% = fault% + 1
      PRINT #2, "**** data echo failure ****"
      PRINT #2, basstr$
      PRINT #2, msg$
    ELSE
      PRINT basstr$;                         'DISPLAY ECHO
      QUIT = 0
    END IF
'     IF fault% > 0 AND fault% < 3 THEN GOTO skipread 'attempt retry of last srecord
    basstr$ = "": basstr$ = tmpstr$
QUITLD:
  LOOP WHILE QUIT = 0 AND NOT endoffile

  COM(port) OFF
  PRINT echo$
  PRINT "**** TRANSFER COMPLETED ****"
  CLOSE #4, #2
abortld:
  RETURN



BLDECHO:
    COM(port) STOP
    echo$ = echo$ + INPUT$(LOC(1), #1)
    IF INSTR(1, echo$, cr$) <> 0 THEN
      echo$ = LEFT$(echo$, INSTR(1, echo$, cr$))
      QUIT = 2
'      COM(port) STOP
```

```
      ELSE
        COM(port) ON
      END IF
      RETURN


LIBRARY:
   PRINT "NOT CURRENTLY SUPPORTED"
   RETURN


EDIT:
   CALL SELWDOW(wterm%, wd%(), yes%, yes%, yes%, "DFEDIT SHELL")
   SHELL "ued " + sdir$ + PRGFILE$
   RESTART = 1: QUIT = 1
   RETURN


 ' EXAMPLE COMFILE$="C:\QUICK40\PRG\TERM.DAT"
 ' EXAMPLE  CPORT="COM1:9600,N,8,1,BIN,OP,RS,CD,CS,DS"
 ' EXAMPLE  SDIR$="C:\BACKUP\TCOMAU\"

RDDEFA:
  cnfgfile$ = ENVIRON$("CNFGFILE")  'get powerup configuration filename
  IF cnfgfile$ = "" THEN cnfgfile$ = "VTB52.CFG"
  newcnfg$ = cnfgfile$

RDCNFG:
  CLS 2: X = 1
  PRINT "LOADING FROM CONFIGURATION FILE LOADING= "; newcnfg$
  OPEN newcnfg$ FOR INPUT AS #2   'open powerup configuration file

  DO:
   INPUT #2, LABL$
   PRINT "  Reading setup= "; LABL$
   SELECT CASE LABL$
    CASE "COMPORT"
      INPUT #2, cport$
      port = VAL(MID$(cport$, 4, 1))
    CASE "B52BASE"
      INPUT #2, sdir$
    CASE "PRGFILE"
      INPUT #2, PRGFILE$
    CASE "SRCFILE"
      INPUT #2, SRCFILE$
    CASE "DNLDFILE"
      INPUT #2, dnldfile$
    CASE "ERRFILE"
      INPUT #2, errfile$
    CASE "SRECDLY"
      INPUT #2, srecdly
    CASE "ABORTDLY"
```

```
      INPUT #2, abortdly
    CASE "PORTBUF"
      INPUT #2, portbuf
    CASE "SHOWERR"
      INPUT #2, SHOWERR
    CASE "TERMOUT"
      INPUT #2, comfile$
    CASE "SCNSIZE"
      INPUT #2, row, col
    CASE "MSGFIELD"
      INPUT #2, wd%(wmsg%, wlabl%), wd%(wmsg%, wtop%), wd%(wmsg%, wbot%)
    CASE "MSGCOLOR"
      INPUT #2, wd%(wmsg%, wfore%), wd%(wmsg%, wback%)
    CASE "TERMFIELD"
      INPUT #2, wd%(wterm%, wlabl%), wd%(wterm%, wtop%), wd%(wterm%, wbot%)
    CASE "TERMCOLOR"
      INPUT #2, wd%(wterm%, wfore%), wd%(wterm%, wback%)
    CASE ""          'COMMENT FIELD IGNORE THIS LINE
    CASE ELSE
      PRINT "INVALID ENTRY IN CSMH.CFG AT LINE NUMBER=;X"
    END SELECT

    X = X + 1        'INCREMENT LINE COUNTER
  LOOP UNTIL EOF(2)

  CLOSE #2
  CLS 2
  RETURN
```

```
SETFILE:
   A$(0, 0) = "": A$(0, 1) = "": A$(0, 2) = "": A$(0, 3) = "": A$(0, 4) = "": A$(0, 5) = "": A$(0,
6) = "": A$(0, 7) = "": A$(0, 8) = "": A$(0, 9) = "EXIT"
   CALL SETFCTN(STK, A$())
   CALL SELWDOW(wterm%, wd%(), yes%, yes%, yes%, "USER PROGRAM FILESPEC DEFINITIONS")
   EDIT% = 0
   DO
     LOCATE 2, 1
     PRINT ""
     PRINT "DEVELOPMENT DIRECTORY   ="; : sdir$ = CHARIN$(sdir$, EDIT%): PRINT
     PRINT "PROGRAM SOURCE FILESPEC ="; : PRGFILE$ = CHARIN$(PRGFILE$, EDIT%): PRINT
     PRINT "EXTRACTION  FILESPEC    ="; : SRCFILE$ = CHARIN$(SRCFILE$, EDIT%): PRINT
     PRINT "DWNLD SOURCE FILESPEC   ="; : dnldfile$ = CHARIN$(dnldfile$, EDIT%): PRINT
     PRINT "ERROR REPORT FILESPEC   ="; : errfile$ = CHARIN$(errfile$, EDIT%): PRINT
     EDIT% = 1
   LOOP UNTIL QUITFCTN
   quitterm = 0: QUIT = 0: QUITFCTN = 0
```

```
   CALL RTNFCTN(STK, A$())
   RETURN

SUB RTNFCTN (STK, A$())
   COLOR 15, 0
   STK = STK - 1
   IF STK = 0 THEN STOP
   FOR c = 0 TO 9
     CALL SETGOSB(c + 1, A$(STK, c))
     KEY c + 1, A$(STK, c)
     IF A$(STK, c) = "" THEN KEY(c + 1) OFF ELSE KEY(c + 1) ON
   NEXT
   KEY ON
END SUB

SUB SETFCTN (STK, A$())
   COLOR 15, 0
   STK = STK + 1
   IF STK = 10 THEN STOP
   FOR c = 0 TO 9
     CALL SETGOSB(c + 1, A$(0, c))
     A$(STK, c) = A$(0, c)
     KEY c + 1, A$(STK, c)
     IF A$(STK, c) = "" THEN KEY(c + 1) OFF ELSE KEY(c + 1) ON
   NEXT
   KEY ON
END SUB

SUB SETGOSB (KN, B$)

SELECT CASE B$
CASE "TERM"
  ON KEY(KN) GOSUB TERM
CASE "ECHO"
  ON KEY(KN) GOSUB echo
CASE "EDIT"
  ON KEY(KN) GOSUB EDIT
CASE "EXTRACT"
  ON KEY(KN) GOSUB EXTRACT
CASE "DOWNLD"
  ON KEY(KN) GOSUB DOWNLOAD
CASE "EXIT"
  ON KEY(KN) GOSUB EXITTERM
CASE "QUIT"
  ON KEY(KN) GOSUB QUIT
CASE "CALLDOS"
  ON KEY(KN) GOSUB DOS
CASE "LIBRARY"
  ON KEY(KN) GOSUB LIBRARY
CASE "EXTEMP"
  ON KEY(KN) GOSUB EXTEMP
CASE "RDCNFG"
```

```
  ON KEY(KN) GOSUB RDCNFG
CASE "SETFILE"
  ON KEY(KN) GOSUB SETFILE
CASE ""
CASE ELSE
  STOP                          'SELECTED SUBROUTINE NOT IN THIS MODULE
END SELECT

END SUB
```

CUPL PAL Listings

SRCSEL4 PAL Program Listing

SRCSEL4.PLD 8690 08-24-91 12:55pm

```
               PARTNO    SRCSEL;
               NAME      SRCSEL;
               DATE      4/28/91;
               REV       ;
               DESIGNER  ROBERT M. BASS;
               COMPANY   TEXAS INSTRUMENTS;
               ASSEMBLY  ;
               LOCATION  TEST ENGINEERING;

/*************************************************************************/
/*                                                                       */
/*   TTL SIGNATURE SOURCE SELECT & SIGNAL PIPELINE PAL                   */
/*                                                                       */
/*************************************************************************/
/*  Allowable Target Device Types: ADV. u DEV. (AMD) - 22V10             */
/*************************************************************************/
/*   NOTE: All video & sync signals will retain the same polarity */
/*         NON-INVERTING entering & exiting the pal to maintain   */
/*         compatiability with existing hardware.                 */
/*   NOTE: All sync signals entering this pal are considered to be*/
/*         avtive low thus sync polarity can be stated to be      */
/*         negative. any other sync polarities must be inverted   */
/*         externally before entering this pal.                   */
/*   NOTE: SELFTEST will consists of two parts                     */
/*          1.ANALOG THRESHOLD TEST MODE which will allow the     */
/*            the dac thresholds to be tested against the analog  */
/*            test ref by reading back the levels detected and    */
/*            free-run clocked in to LATVIDEO,LATVERT,LATHORZ reg.*/
/*          2.TTL VIDEO SELFTEST where ttl test signals will      */
/*            generate VIDEOTST,HORZTST,VERTTST,TSTCLK. This will */
/*            be used to test the pal, the clock synchronizer,    */
/*            the frame synchronizer and also the signature       */
/*            analyzers.                                          */
/*                                                                */
/*    ANALOG & TTL SHOULD ENCOUNTER SYMMETRIC PROPAGATION DELAYS  */
/*    ANALOG DELAYS THROUGH WINDOW DETECTION CIRCUIT TO WINVIDEO  */
/*    SHOULD BE APPROXIMATELY THE SAME DELAY AS TTL SIGNALS       */
/*    THROUGH MUX LOGIC.                                          */
/*************************************************************************/
/*          BASIC TRUTH TABLE FOR EXTERNAL MUX OPERATIONS        */
/*   TTL VIDEO SOURCE SELECTION                                  */
/*      SRCSEL                                                   */
/*        0       RED                                            */
/*        1       GREEN                                          */
/*        2       BLUE                                           */
/*        3       SECONDARY RED                                  */
/*        4       SECONDARY GREEN/INTENSITY                      */
/*        5       SECONDARY BLUE                                 */
```

```
/*        6       HORIZONAL SYNC/COMPOSITE SYNC                    */
/*        7       VERTICAL SYNC                                    */
/*                                                                 */
/*                                                                 */
/*   ANALOG VIDEO SOURCE SELECTION                                 */
/*      SRCSEL                                                     */
/*        0       RED                                              */
/*        1       GREEN                                            */
/*        2       BLUE                                             */
/*        3                                                        */
/*        4                                                        */
/*        5                                                        */
/*        6       HORIZONAL SYNC/COMPOSITE SYNC                    */
/*        7       VERTICAL SYNC                                    */
/*                                                                 */
/*          BASIC TRUTH TABLE FOR PAL MODE OPERATIONS              */
/*                                                                 */
/*                                                                 */
/*   TTL VIDEO MODES                                               */
/*      VIDMODE                                                    */
/*        0       TTL SEPERATE SYNC       TTLSEP                   */
/*        1       TTL COMPOSITE SYNC      TTLCMP                   */
/*        2       ANALOG SEPERATE SYNC    ASEPSYN                  */
/*        3       ANALOG COMPOSITE SYNC   ACMPSYN                  */
/*        4       ANALOG SYNC ON GREEN    AGRNSYN                  */
/*        5       ANALOG TEST MODE        ANATST                   */
/*        6       TTL TEST MODE           TTLTST                   */
/*        7                                                        */
/*                                                                 */
/*                                                                 */
/*******************************************************************/

/** Inputs **/

PIN 1        = SDOTCLK    ;     /* SYNCHRONIZED DOT CLOCK */
PIN 2        = !TTLVERT   ;     /* TTL VERTICAL SYNCHRONIZATION */
PIN 3        = !TTLHORZ   ;     /* TTL HORIZONAL SYNCHRONIZATION */
PIN 4        = !TTLMUX    ;     /* TTL RED VIDEO */
PIN 5        = !VIDEOTST  ;     /* TTL VIDEO TEST SIGNAL */
PIN 6        = !CMPTST    ;     /* TTL COMPOSITE TEST SIGNAL */
PIN 7        = !WINVIDEO  ;     /* ANALOG WINDOW VIDEO */
PIN 8        = !WINGRNSYN ;     /* ANALOG WINDOW GREEN SYNC */
PIN [9,10,11]= [VIDMOD2..0] ;   /* SOURCE SELECTION CONTROL LINES*/
PIN 13       = inul       ;

/** REGISTERED OUTPUT  **/
PIN 17       =!LATHORZ    ;     /* DOT CLOCK LATCHED HORIZONAL SYNC SIGNAL */
PIN 18       =!LATCOMP    ;     /* DOT CLOCK LATCHED GEN.COMPOSITE SYNC SIGNAL */
PIN 19       =!LATVIDEO   ;     /* DOT CLOCK LATCHED VIDEO PIXEL */
PIN 20       =!LATVERT    ;     /* DOT CLOCK LATCHED VERTICAL SYNC SIGNAL */

/** LOGIC INPUT/OUTPUT **/
```

```
PIN 15       =!SELFTEST  ;       /* SELFTEST DECODED OUTPUT */
PIN 16       =!GATVIDEO  ;       /* TTL WINDOW VIDEO SIGNAL */
PIN 21       =!GATCOMP   ;       /* TTL WINDOW GEN COMPOSITE SYNC */
PIN 22       =!GATVERT   ;       /* TTL WINDOW VERTICAL SYNC */

PIN 14       = ionu1     ;
PIN 23       = ionu2     ;
PIN 24       = VCC;

/* THESE SELECT SIGNALS ARE NOT REQUIRED INTERNAL TO THIS PAL */
/* FIELD SRCSEL=[SRCSEL2..0]; */
/* SELRED  =( !SRCSEL2 & !SRCSEL1 & !SRCSEL0); */
/* SELGRN  =( !SRCSEL2 & !SRCSEL1 &  SRCSEL0); */
/* SELBLU  =( !SRCSEL2 &  SRCSEL1 & !SRCSEL0); */
/* SELSRED =( !SRCSEL2 &  SRCSEL1 &  SRCSEL0); */
/* SELSGRN =(  SRCSEL2 & !SRCSEL1 & !SRCSEL0); */
/* SELSBLU =(  SRCSEL2 & !SRCSEL1 &  SRCSEL0); */
/* SELHORZ =(  SRCSEL2 &  SRCSEL1 & !SRCSEL0); */
/* SELVERT =(  SRCSEL2 &  SRCSEL1 &  SRCSEL0); */

FIELD VIDMOD=[VIDMOD2..0];

$DEFINE ON  'b'1
$DEFINE OFF 'b'0

TTLSEP  =( !VIDMOD2 & !VIDMOD1 & !VIDMOD0);
TTLCMP  =( !VIDMOD2 & !VIDMOD1 &  VIDMOD0);
ASEPSYN =( !VIDMOD2 &  VIDMOD1 & !VIDMOD0);
ACMPSYN =( !VIDMOD2 &  VIDMOD1 &  VIDMOD0);
AGRNSYN =(  VIDMOD2 & !VIDMOD1 & !VIDMOD0);
ANATST  =(  VIDMOD2 & !VIDMOD1 &  VIDMOD0);
TTLTST  =(  VIDMOD2 &  VIDMOD1 & !VIDMOD0);
NUvm7   =(  VIDMOD2 &  VIDMOD1 &  VIDMOD0);

/**  Logic Equations  **/

/** STATE DIAGRAM GENERATED EQUATIONS NO DIRECT MINIMIZATION ATTEMPTED **/

VID = (TTLSEP # TTLCMP) & TTLMUX #              /*GATE MULTIPLEXED TTL VIDEO*/
      (ASEPSYN # ACMPSYN # AGRNSYN # ANATST) & WINVIDEO # /*GATE ANALOG VIDEO*/
      TTLTST & VIDEOTST;                /*GATE TTLTST VIDEO*/

       /* THIS EQU. SENDS THE HORIZ SYNC AVAILABLE FOR THE SELECTED MODE*/
       /* WHICH MAY BE SEPERATE OR COMPOSITE */
HORZ = (TTLSEP # TTLCMP # ASEPSYN # ACMPSYN) & TTLHORZ # /*GATE TTL HORZ SYNC*/
       (AGRNSYN # ANATST) & WINGRNSYN #   /*GATE SYNC ON GREEN*/
       TTLTST & CMPTST;                  /*GATE TEST COMPOSITE SYNC */

       /* THIS EQU. ALWAYS SENDS COMPOSITE SYNC IF AVAILABLE OR GENERATES
       /* COMPOSITE SYNC FOR SEPERATE MODES */
       /* THIS EQU.IS USED TO GEN. SYNCHRONIZED DOTCLK & SIG START/STOP    */
       /* EXTERNAL COMPOSITE GEN.TRUTH TABLE IS Exclusive-Nor OF HORZ. AND */
```

```
      /* VERT SYNC INPUTS */
COMP = (TTLCMP # ACMPSYN) & TTLHORZ #              /*GATE TTL COMPOSITE SYNC*/
      (TTLSEP # ASEPSYN) & TTLVERT & !TTLHORZ # /* 1ST TERM TO GEN.COMPOSTIE*/
      (TTLSEP # ASEPSYN) & !TTLVERT & TTLHORZ # /* 2ST TERM TO GEN.COMPOSTIE*/
      (AGRNSYN # ANATST) & WINGRNSYN #    /*GATE COMPOSITE SYNC ON GREEN*/
      TTLTST & CMPTST;                    /*GATE TEST COMPOSITE SYNC */

   /* THIS EQUATION SENDS VERTICAL SYNC IF AVAILABLE OTHERWISE COMPOSITE */
   /* SYNC IS ROUTED TO THIS PIN.*/
VERT = TTLCMP & TTLVERT #                  /* GATE TTL COMPOSITE SYNC */
      (TTLSEP # ASEPSYN) &  TTLVERT #      /* GATE SYNC SEPERATE   */
      ACMPSYN & TTLHORZ #                  /* GATE ANALOG COMPOSITE SYNC */
      (AGRNSYN # ANATST) & WINGRNSYN #  /* GATE SYNC ON GREEN SYNC */
      TTLTST & CMPTST;           /* GATE TTL TEST COMPOSITE SYNC */

/** NOTE: USING REDUNDANT EQUATIONS IN LAT*.D & GAT* WILL PROVIDE **/
/**       IMPROVED SETUP TIME FOR LATCHING REGISTER.                **/

GATVIDEO.OE= ON;              /** DRIVE OUTPUT FOR ALL MODES **/
GATVIDEO   = VID;             /** GATED VIDEO SIGNAL LOGIC PASS THROUGH **/

LATVIDEO.AR=OFF;              /** ASYNCHRONOUS RESET DISABLE **/
LATVIDEO.SP=OFF;              /** SYNCHRONOUS PRESET DISABLE **/
LATVIDEO.OE= ON;              /** DRIVE OUTPUT FOR ALL MODES **/
LATVIDEO.D = VID;             /** DOTCLK LATCHED VIDEO SIGNAL**/

GATVERT.OE= ON;               /** DRIVE OUTPUT FOR ALL MODES **/
GATVERT   = VERT;             /** GATED VERTICAL SYNC LOGIC PASS THROUGH **/
LATVERT.AR=OFF;               /** ASYNCHRONOUS RESET DISABLE **/
LATVERT.SP=OFF;               /** SYNCHRONOUS PRESET DISABLE **/
LATVERT.OE= ON;               /** DRIVE OUTPUT FOR ALL MODES **/
LATVERT.D = VERT;             /** DOTCLK LATCHED VERTICAL SYNC **/

LATHORZ.AR=OFF;               /** ASYNCHRONOUS RESET DISABLE **/
LATHORZ.SP=OFF;               /** SYNCHRONOUS PRESET DISABLE **/
LATHORZ.OE= ON;               /** DRIVE OUTPUT FOR ALL MODES **/
LATHORZ.D = HORZ;             /** DOTCLK LATCHED HORIZONAL SYNC SIGNAL**/

GATCOMP.OE= ON;               /** DRIVE OUTPUT FOR ALL MODES **/
GATCOMP   = COMP;             /** GATED COMPOSITE SYNC LOGIC PASS THROUGH **/
LATCOMP.AR=OFF;               /** ASYNCHRONOUS RESET DISABLE **/
LATCOMP.SP=OFF;               /** SYNCHRONOUS PRESET DISABLE **/
LATCOMP.OE= ON;               /** DRIVE OUTPUT FOR ALL MODES **/
LATCOMP.D = COMP;             /** DOTCLK LATCHED COMPOSITE SYNC**/

SELFTEST.OE=ON;
SELFTEST = TTLTST # ANATST;   /**SELFTEST:ENABLE TSTCLK AND TEST VIDEO **/
```

SRCSEL4 PAL Simulation Output

SRCSEL4.SO

CSIM: CUPL Simulation Program
Version 3.00 Serial# MD-300-5446
Copyright (C) 1983,1988 Logical Devices, Inc.
CREATED Sun Sep 22 18:18:03 1991

LISTING FOR SIMULATION FILE: SRCSEL4.si

```
  1:
  2:          PARTNO    SRCSEL;
  3:          NAME      SRCSEL;
  4:          DATE      4/28/91;
  5:          REV       ;
  6:          DESIGNER  ROBERT M. BASS;
  7:          COMPANY   TEXAS INSTRUMENTS;
  8:          ASSEMBLY  ;
  9:          LOCATION  TEST ENGINEERING;
 10:
 11: /*************************************************************************/
 12: /*                                                                       */
 13: /*************************************************************************/
 14: /*  Allowable Target Device Types: ADV. u DEV. (AMD) - 22V10              */
 15: /*************************************************************************/
 16:
 17: BASE: HEX;
 18:
 19:
 20: ORDER:
 21:    %7,SDOTCLK,%2,!TTLMUX,%1,!TTLHORZ,%1,!TTLVERT,%3,!VIDEOTST,%1,!CMPTST,%1,
 22:    %1,%1,!WINVIDEO,%1,!WINGRNSYN,%2,%3,%1,VIDMOD,%3,!GATVIDEO,%1,
 23:    !GATCOMP,%1,!GATVERT,%2,!SELFTEST,%2,!LATVIDEO,%1,!LATCOMP,%1,!LATVERT,
 24:    %1,!LATHORZ;
 25:
```

```
================================================================================
                              Simulation Results
================================================================================
 NOTE:SIMULATION IS BASED ON EXTERNAL PIN CONNECTION
     LEVEL NOT INTERNAL TRUE ACTIVE LEVELS

 S  T T T   V C   W W     V    G G G  S  L L L L
 D  T T T   I M   I I     I    A A A  E  A A A A
 O  L L L   D P   N N     D    T T T  L  T T T T
 T  M H V   E T   V G     M    V C V  F  V C V H
 C  U O E   O S   I R     O    I O E  T  I O E O
 L  X R R   T T   D N     D    D M R  E  D M R R
 K    Z T   S     E S          E P T  S  E P T Z
           T     O Y          O     T  O
                   N

 EXTERNAL COMPOSITE GEN.TRUTH TABLE IS Exclusive-Nor OF HORZ. AND
 VERT SYNC INPUTS
  MODE TTLSEP,MUX VIDEO,SEPERATE SYNC, MUST GEN COMPOSITE SYNC
0001:        0  0 0 0   0 0   0 0      000   L H L  H  N N N N
0002:        C  0 0 0   0 0   0 0      000   L H L  H  L H L L
0003:        C  0 0 1   0 0   0 0      000   L L H  H  L L H L
0004:        C  0 1 0   0 0   0 0      000   L L L  H  L L L H
0005:        C  1 0 0   0 0   0 0      000   H H L  H  H H L L
0006:        C  1 1 1   0 0   0 0      000   H H H  H  H H H H
0007:        C  0 0 0   1 1   1 1      000   L H L  H  L H L L
  MODE TTLCMP,MUX VIDEO,COMPOSITE SYNC
0008:        0  0 0 0   0 0   0 0      001   L L L  H  L H L L
0009:        C  0 0 0   0 0   0 0      001   L L L  H  L L L L
0010:        C  0 0 1   0 0   0 0      001   L L H  H  L L H L
0011:        C  0 1 0   0 0   0 0      001   L H L  H  L H L H
0012:        C  1 0 0   0 0   0 0      001   H L L  H  H L L L
0013:        C  1 1 1   0 0   0 0      001   H H H  H  H H H H
0014:        C  0 0 0   1 1   1 1      001   L L L  H  L L L L
  MODE ASEPSYN,WINVIDEO VIDEO,SEPERATE SYNC,MUST GENERATE COMPOSITE SYNC
0015:        0  0 0 0   0 0   0 0      010   L H L  H  L L L L
0016:        C  0 0 0   0 0   0 0      010   L H L  H  L H L L
0017:        C  0 0 1   0 0   0 0      010   L L H  H  L L H L
0018:        C  0 1 0   0 0   0 0      010   L L L  H  L L L H
0019:        C  0 0 0   0 0   1 0      010   H H L  H  H H L L
0020:        C  0 1 1   0 0   1 0      010   H H H  H  H H H H
0021:        C  1 0 0   1 1   0 1      010   L H L  H  L H L L
  MODE ACMPSYN,WINVIDEO VIDEO,COMPOSITE SYNC
0022:        0  0 0 0   0 0   0 0      011   L L L  H  L H L L
0023:        C  0 0 0   0 0   0 0      011   L L L  H  L L L L
0024:        C  0 0 1   0 0   0 0      011   L L L  H  L L L L
0025:        C  0 1 0   0 0   0 0      011   L H H  H  L H H H
0026:        C  0 0 0   0 0   1 0      011   H L L  H  H L L L
0027:        C  0 1 1   0 0   1 0      011   H H H  H  H H H H
0028:        C  1 0 0   1 1   0 1      011   L L L  H  L L L L
  MODE AGRNSYN,WINVIDEO VIDEO,WINGRNSYNC
```

```
0029:       0  0 0 0   0 0   0 0     100   L L L  H  L L L L
0030:       C  0 0 0   0 0   0 0     100   L L L  H  L L L L
0031:       C  0 0 0   0 0   0 1     100   L H H  H  L H H H
0032:       C  0 0 0   0 0   1 0     100   H L L  H  H L L L
0033:       C  0 0 0   0 0   1 1     100   H H H  H  H H H H
0034:       C  0 0 0   0 0   1 1     100   H H H  H  H H H H
0035:       C  1 1 1   1 1   0 0     100   L L L  H  L L L L
  MODE ANATST,WINVIDEO VIDEO,WINGRNSYNC
0036:       0  0 0 0   0 0   0 0     101   L L L  L  L L L L
0037:       C  0 0 0   0 0   0 1     101   L H H  L  L H H H
0038:       C  0 0 0   0 0   1 0     101   H L L  L  H L L L
0039:       C  0 0 0   0 0   1 1     101   H H H  L  H H H H
0040:       C  1 1 1   1 1   0 0     101   L L L  L  L L L L
  MODE TTLTST,VIDEOTST,CMPTST
0041:       0  0 0 0   0 0   0 0     110   L L L  L  L L L L
0042:       C  0 0 0   0 1   0 0     110   L H H  L  L H H H
0043:       C  0 0 0   1 0   0 0     110   H L L  L  H L L L
0044:       C  0 0 0   1 1   0 0     110   H H H  L  H H H H
0045:       C  1 1 1   0 0   1 1     110   L L L  L  L L L L
```

SYNC5 PAL Program Listing

SYNC5.PLD 14816 09-22-91 06:38pm

```
               PARTNO    SYNC5;
               NAME      SYNC5;
               DATE      9/22/91;
               REV       ;
               DESIGNER  ROBERT M. BASS;
               COMPANY   TEXAS INSTRUMENTS;
               ASSEMBLY  ;
               LOCATION  TEST ENGINEERING;

/*********************************************************************/
/*                                                                   */
/*   VIDEO CLOCK SYNCHRONIZER/SIGNATURE CONTROL ANALYZER    */
/*                                                                   */
/*********************************************************************/
/*  Allowable Target Device Types: ADV. u DEV. (AMD) - G16V8        */
/*********************************************************************/

/**  Inputs  **/

/** NOTE: VERTSYNC INPUT IS NOT CURRENTLY BEING USED THUS PIN IS AVAILABLE **/

PIN 1        = CLK        ;      /* 4X CLOCK LINE    */
PIN [3..2]   = [OF1..0]   ;      /* CLOCK OFFSET/SKEW SELECT */
PIN 4        = LATHORZ    ;     /* LATCHED HORZ/COMPOSITE SYNC LINE */
PIN 5        = !VTMREXP   ;      /* VERTICAL TIMER EXPIRED TIMEOUT */
PIN 6        = !SYNC      ;      /* HORIZONAL/COMPOSITE SYNC SIGNAL    */
PIN 7        = !CSWR      ;      /* uP WRITE GENERATED CLOCK */
PIN 8        = !SELFTEST  ;      /* ENABLES uP GENERATED DOTCLK */
PIN 9        = !ACQUIRE   ;      /* ACQUIRE FRAME*/
PIN 10       = GND        ;
PIN 11       = OUTENA     ;

/**  OUTPUT  **/

PIN [14..12] = ![SCTR2..0];      /*STATE COUNTER  */
PIN 15       = !SDOTCLK;         /*SYNC DOT CLOCK */
PIN [16..19] = ![CTL3..0];       /*SIGNATURE START/STOP CONTROL STATES */
PIN 20       = VCC;


/* CLOCK SYNCHRONIZER/GENERATOR */

FIELD SSEQ=[SCTR2,SCTR1,SCTR0];
FIELD SK=[OF1,OF0];

$DEFINE ON-  'b'1;
$DEFINE OFF- 'b'0;
```

```
$define S0 'b'000
$define S1 'b'001
$define S2 'b'010
$define S3 'b'011
$define S4 'b'100
$define S5 'b'101
$define S6 'b'110
$define S7 'b'111


/** Logic Equations **/

/** STATE DIAGRAM GENERATED EQUATIONS NO DIRECT MINIMIZATION ATTEMPTED **/

SEQUENCE SSEQ {
present S0    IF !SYNC    NEXT S1;
              IF  SYNC    NEXT S4;
              IF SK:[2] & !SELFTEST OUT SDOTCLK.D;
              IF SK:[3] & !SELFTEST OUT SDOTCLK.D;


present S1    IF !SYNC    NEXT S2;
              IF  SYNC    NEXT S4;
              IF SK:[1] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[2] & !SELFTEST  OUT SDOTCLK.D;

present S2    IF !SYNC    NEXT S3;
              IF  SYNC    NEXT S4;
              IF SK:[0] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[1] & !SELFTEST  OUT SDOTCLK.D;


present S3    IF !SYNC    NEXT S0;
              IF  SYNC    NEXT S4;
              IF SK:[0] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[3] & !SELFTEST  OUT SDOTCLK.D;

present S4    IF !SYNC    NEXT S1;
              IF  SYNC    NEXT S5;
              IF SK:[2] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[3] & !SELFTEST  OUT SDOTCLK.D;

present S5    IF !SYNC    NEXT S2;
              IF  SYNC    NEXT S6;
              IF SK:[1] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[2] & !SELFTEST  OUT SDOTCLK.D;

present S6    IF !SYNC    NEXT S3;
              IF  SYNC    NEXT S7;
              IF SK:[0] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[1] & !SELFTEST  OUT SDOTCLK.D;
```

```
present S7    IF !SYNC    NEXT S0;
              IF  SYNC    NEXT S4;
              IF SK:[0] & !SELFTEST  OUT SDOTCLK.D;
              IF SK:[3] & !SELFTEST  OUT SDOTCLK.D;
}

APPEND SDOTCLK.D = SELFTEST & CSWR;

APPEND SCTR2.D = SELFTEST;
APPEND SCTR1.D = SELFTEST;
APPEND SCTR0.D = SELFTEST;
/** ADDITIONS MUST BE MADE TO EXTEND SDOTCLK OR CYCLE OUTPUT TWO CLK PERIODS **/
/** NXTCYC.D   = SDOTCLK & !NXTCYC & !ACQUIRE;              **/



/* SIGNATURE START/STOP CONTROL STATE DEFINITION                        */
/*                                                                      */
/*   STATE DIAGRAM SYNC LEVELS ARE EXTERNAL CHIP ACTIVE LEVELS          */
/* CURRENT STATE     NEXT STATE       NEXT STATE CRITERIA      */
/* ______________ ____________________ ____________________________ */
/* RESET             SYNCHRONIZE STATE       ACQ & HIGH SYNC            */
/*                   RESET STATE             ALL OTHER CONDITIONS       */
/*                                                                      */
/*   ----_-----_--                                                      */
/*         ^                                                            */
/* SYNCHRONIZE_     ANY SYNC #1___________ ACQ & FALLING SYNC____________*/
/*                   SYNCHRONIZE             ACQ & HIGH SYNC___________    */
/*                                                                      */
/*   ----_-----_--                                                      */
/*         ^   ^                                                        */
/* ANY SYNC #1       1ST HORIZONAL INVL____ ACQ & !TMREXPIRE & RISING SYNC  */
/*                   SYNCHRONIZE___________ ACQ & TMREXPIRE & RISING SYNC   */
/*                   ANY SYNC #1             ACQ & LOW SYNC             */
/*                                                                      */
/*   ----_-----_--                                                      */
/*         ^   ^ ^                                                      */
/* 1ST HORIZONAL INV ANY SYNC #2___________ ACQ & FALLING SYNC___________ */
/*                   1ST HORIZONAL INV       ACQ & HIGH SYNC______________ */
/*                                                                      */
/*   ----_-----_-----_--                                                */
/*         ^   ^ ^    ^                                                 */
/* ANY SYNC #2_____ 2ND HORIZONAL INVL____ ACQ & !TMREXPIRE & RISING SYNC_ */
/* ______________ SYNCHRONIZE___________ ACQ &  TMREXPIRE & RISING SYNC_ */
/* ______________ ANY SYNC #2___________ ACQ & LOW SYNC_______________ */
/*                                                                      */
/*   ----_-----_-----_--                                                */
/*         ^   ^ ^    ^ ^                                               */
/* 2ND HORIZONAL INV ANY SYNC #3___________ ACQ & FALLING SYNC___________ */
/* ______________ 2ND HORIZONAL INV_____ ACQ & HIGH SYNC______________ */
/*                                                                      */
/*   ----_-----_-----_-----____--_                                      */
```

```
/*          ^   ^ ^    ^ ^    ^                                          */
/* ANY SYNC #3_____ Nth VERTICAL INVL_____ ACQ &  TMREXPIRE & RISING SYNC_ */
/* _______________ 2ND HORIZONAL INV_____ ACQ & !TMREXPIRE & RISING SYNC_ */
/* _______________ ANY SYNC #3___________ ACQ & LOW SYNC_______________ */
/*                                                                       */
/*   ----_-----_-----_-----____--_                                       */
/*         ^   ^ ^    ^ ^    ^    ^                                     */
/* Nth VERTICAL INVL ANY SYNC #4___________ ACQ & FALLING SYNC_____________ */
/* _______________ Nth VERTICAL INVL_____ ACQ & HIGH SYNC_______________ */
/*                                                                       */
/*        OUTPUT:START SIGNATURE CAPTURE ON THIS STATE                   */
/*                                                                       */
/*   ----_-----_-----_-----____--____--_.....__-----_-----_              */
/*         ^   ^ ^    ^ ^    ^    ^                    ^                 */
/* ANY SYNC #4_____ Nth HORIZONAL INVL____ ACQ & !TMREXPIRE & RISING SYNC_ */
/* _______________ Nth VERTICAL INVL_____ ACQ &  TMREXPIRE & RISING SYNC_ */
/* _______________ ANY SYNC #4___________ ACQ & LOW SYNC_______________ */
/*                                                                       */
/*   ----_-----_-----_-----____--____--_.....__-----_-----_              */
/*         ^   ^ ^    ^ ^    ^    ^                     ^ ^              */
/* Nth HORIZONAL INV ANY SYNC #5___________ ACQ & FALLING SYNC_____________ */
/* _______________ NTH HORIZONAL SYNC____ ACQ & HIGH SYNC_______________ */
/*                                                                       */
/*                                                                       */
/*   ----_-----_-----_-----____--____--_.....__-----_-----____--____ */
/*         ^   ^ ^    ^ ^    ^    ^                      ^ ^    ^       */
/* ANY SYNC #5_____ STOP SIGNATURE_______ ACQ &  TMREXPIRES & RISING SYNC */
/* _______________ Nth HORIZONAL INVL____ ACQ & !TMREXPIRES & RISING SYNC */
/* _______________ ANY SYNC #5___________ ACQ & LOW SYNC_______________ */
/*                                                                       */
/*   ----_-----_-----_-----____--____--_.....__-----_-----____--____ */
/*         ^   ^ ^    ^ ^    ^    ^                      ^ ^  ^    ^    */
/* STOP SIGNATURE___ STOP SIGNATURE_______ ACQ_____________________________ */
/* _______________ RESET STATE___________ !ACQ_____________________________ */
/*                                                                       */
/*        STOP SIGNATURE CAPTURE ON THIS STATE                           */
/*                                                                       */


/* NOTE: STATE WERE GROUPED TO DEFINE ACQUIRE SIGNATURE REGION */
/* HOWEVER IF EXCLUSION OF SYNC INFORMATION IS NECESSARY THEN */
/* REGROUPING STATES OR EXTERNAL DECODING THEN OPEN COLLECTOR WIRED-OR MAY */
/* BE NECESSARY. */
FIELD CSEQ=[CTL3,CTL2,CTL1,CTL0];

$define REST        'b'0000  /* RESET STATE WAITING FOR ACQUIRE & HIGH SYNC */
$define SYNCHRZ     'b'0001  /* SYNCHRONIZE STATE TO NON SYNC INTERVAL */
$define ANYSYNC1    'b'0010  /*     UNKNOWN SYNC STARTED */
$define HORIZ1      'b'0011  /* 1st HORIZONAL SYNC FOUND */
$define ANYSYNC2    'b'0100  /*     UNKNOWN SYNC FOUND */
$define HORIZ2      'b'0101  /* 2nd or Nth HORIZONAL SYNC FOUND */
$define ANYSYNC3    'b'0110  /* Nth UNKNOWN SYNC FOUND */
```

```
$define STOPSIG     'b'0111  /* END OF FRAME VERTICAL SYNC FOUND --STOP SIG.--*/

 /* SIGNATURE CAN BE GATED DIRECTLY FROM CTL3 FOR START OF FRAME TO EOF */
$define VERTnth     'b'1000  /* 1st or Nth VERTICAL SYNC FOUND --START SIG.--*/
$define ANYSYNC4    'b'1001  /* Nth UNKNOWN SYNC FOUND */
$define HORIZnth    'b'1010  /* 1ST or Nth HORIZONAL SYNC AFTER VERTICAL INVL*/
$define ANYSYNC5    'b'1011  /* Nth UNKNOWN SYNC FOUND */

$define nus12       'b'1100  /* unused state */
$define nus13       'b'1101  /* unused state */
$define nus14       'b'1110  /* unused state */
$define nus15       'b'1111  /* unused state */


/**  Logic Equations  **/

/** STATE DIAGRAM GENERATED EQUATIONS NO DIRECT MINIMIZATION ATTEMPTED **/

/*   STATE DIAGRAM & EQN. SYNC LEVELS ARE INTERNAL & EXTERNAL ACTIVE LEVELS */
/*   ALL OTHER TERMS ARE INTERNAL ACTIVE LEVELS                          */

SEQUENCE CSEQ {
/* SIGNATURE START/STOP CONTROL STATE DEFINITION                        */
/*                                                                      */
/* CURRENT STATE     NEXT STATE       NEXT STATE CRITERIA      */
/* ______________ ____________________ ____________________________ */
/* RESET             SYNCHRONIZE STATE       ACQ & HIGH SYNC                */
/*                   RESET STATE             ALL OTHER CONDITIONS           */
/*                                                                      */
present REST      IF !ACQUIRE # !LATHORZ NEXT REST;
                  IF  ACQUIRE &  LATHORZ NEXT SYNCHRZ;

/*   ----_-----_--                                                      */
/*          ^                                                           */
/* SYNCHRONIZE__     ANY SYNC #1___________ ACQ & FALLING SYNC____________*/
/*                   SYNCHRONIZE             ACQ & HIGH SYNC___________    */
/*                                                                      */
present SYNCHRZ   IF  ACQUIRE & !LATHORZ NEXT ANYSYNC1;
                  IF  ACQUIRE &  LATHORZ NEXT SYNCHRZ;
                  IF  !ACQUIRE NEXT REST;

/*   ----_-----_--                                                      */
/*          ^   ^                                                       */
/* ANY SYNC #1       1ST HORIZONAL INVL____ ACQ & !TMREXPIRE & RISING SYNC  */
/*                   SYNCHRONIZE___________ ACQ & TMREXPIRE & RISING SYNC   */
/*                   ANY SYNC #1             ACQ & LOW SYNC                 */
/*                                                                      */
present ANYSYNC1    IF   ACQUIRE & !LATHORZ NEXT ANYSYNC1;
                    IF   ACQUIRE & !VTMREXP & LATHORZ NEXT HORIZ1;
                    IF   ACQUIRE &  VTMREXP & LATHORZ NEXT SYNCHRZ;
                    IF   !ACQUIRE NEXT REST;
```

```
/*   ----_-----_--                                                    */
/*         ^   ^^                                                     */
/* 1ST HORIZONAL INV ANY SYNC #2_____ ACQ & FALLING SYNC_____ */
/*                   1ST HORIZONAL INV       ACQ & HIGH SYNC_____ */
/*                                                                    */
present HORIZ1       IF  ACQUIRE & LATHORZ NEXT HORIZ1;
                     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC2;
                     IF  !ACQUIRE NEXT REST;

/*   ----_-----_-----_--                                              */
/*         ^   ^^    ^                                                */
/* ANY SYNC #2_____ 2ND HORIZONAL INVL____ ACQ & !TMREXPIRE & RISING SYNC_ */
/* _____ SYNCHRONIZE_____ ACQ &  TMREXPIRE & RISING SYNC_ */
/* _____ ANY SYNC #2_____ ACQ & LOW SYNC_____ */
/*                                                                    */
present ANYSYNC2     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC2;
                     IF  ACQUIRE & !VTMREXP & LATHORZ NEXT HORIZ2;
                     IF  ACQUIRE &  VTMREXP & LATHORZ NEXT SYNCHRZ;
                     IF  !ACQUIRE NEXT REST;

/*   ----_-----_-----_--                                              */
/*         ^   ^^    ^^                                               */
/* 2ND HORIZONAL INV ANY SYNC #3_____ ACQ & FALLING SYNC_____ */
/* _____ 2ND HORIZONAL INV_____ ACQ & HIGH SYNC_____ */
/*                                                                    */
present HORIZ2       IF  ACQUIRE & LATHORZ NEXT HORIZ2;
                     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC3;
                     IF  !ACQUIRE NEXT REST;

/*   ----_-----_-----_-----____--_                                    */
/*         ^   ^^    ^^    ^                                          */
/* ANY SYNC #3_____ Nth VERTICAL INVL_____ ACQ &  TMREXPIRE & RISING SYNC_ */
/* _____ 2ND HORIZONAL INV_____ ACQ & !TMREXPIRE & RISING SYNC_ */
/* _____ ANY SYNC #3_____ ACQ & LOW SYNC_____ */
/*                                                                    */
present ANYSYNC3     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC3;
                     IF  ACQUIRE &  VTMREXP & LATHORZ NEXT VERTnth;
                     IF  ACQUIRE & !VTMREXP & LATHORZ NEXT HORIZ2;
                     IF  !ACQUIRE NEXT REST;

/*   ----_-----_-----_-----____--_                                    */
/*         ^   ^^    ^^    ^   ^                                     */
/* Nth VERTICAL INVL ANY SYNC #4_____ ACQ & FALLING SYNC_____ */
/* _____ Nth VERTICAL INVL_____ ACQ & HIGH SYNC_____ */
/*                                                                    */
/*        OUTPUT:START SIGNATURE CAPTURE ON THIS STATE                */
/*                                                                    */
present VERTnth      IF  ACQUIRE & LATHORZ NEXT VERTnth;
                     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC4;
                     IF  !ACQUIRE NEXT REST;
```

```
/*   ----_-----_-----_-----____--____--_.....__-----_-----_                 */
/*          ^   ^ ^   ^ ^   ^   ^                      ^                    */
/* ANY SYNC #4_____ Nth HORIZONAL INVL____ ACQ & !TMREXPIRE & RISING SYNC_ */
/* _______________ Nth VERTICAL INVL_____ ACQ &  TMREXPIRE & RISING SYNC_ */
/* _______________ ANY SYNC #4___________ ACQ & LOW SYNC_______________ */
/*                                                                          */
present ANYSYNC4     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC4;
                      IF  ACQUIRE & !VTMREXP &  LATHORZ NEXT HORIZnth;
                      IF  ACQUIRE &  VTMREXP &  LATHORZ NEXT VERTnth;
                      IF  !ACQUIRE NEXT REST;

/*   ----_-----_-----_-----____--____--_.....__-----_-----_                 */
/*          ^   ^ ^   ^ ^   ^   ^                      ^ ^                  */
/* Nth HORIZONAL INV ANY SYNC #5___________ ACQ & FALLING SYNC___________ */
/* _______________ NTH HORIZONAL SYNC____ ACQ & HIGH SYNC_______________ */
/*                                                                          */
/*                                                                          */
present HORIZnth     IF  ACQUIRE & LATHORZ NEXT HORIZnth;
                      IF  ACQUIRE & !LATHORZ NEXT ANYSYNC5;
                      IF  !ACQUIRE NEXT REST;


/*   ----_-----_-----_-----____--____--_.....__-----_-----____--____ */
/*          ^   ^ ^   ^ ^   ^   ^                      ^ ^   ^            */
/* ANY SYNC #5_____ STOP SIGNATURE________ ACQ &  TMREXPIRES & RISING SYNC */
/* _______________ Nth HORIZONAL INVL____ ACQ & !TMREXPIRES & RISING SYNC */
/* _______________ ANY SYNC #5___________ ACQ & LOW SYNC_______________ */
/*                                                                          */
present ANYSYNC5     IF  ACQUIRE & !LATHORZ NEXT ANYSYNC5;
                      IF  ACQUIRE &  VTMREXP &  LATHORZ NEXT STOPSIG;
                      IF  ACQUIRE & !VTMREXP &  LATHORZ NEXT HORIZnth;
                      IF  !ACQUIRE NEXT REST;

/*   ----_-----_-----_-----____--____--_.....__-----_-----____--____ */
/*          ^   ^ ^   ^ ^   ^   ^                      ^ ^   ^   ^       */
/* STOP SIGNATURE__ STOP SIGNATURE________ ACQ_______________________ */
/* _______________ RESET STATE___________ !ACQ_______________________ */
/*                                                                          */
/*        STOP SIGNATURE CAPTURE ON THIS STATE                              */
/*                                                                          */
present STOPSIG      IF  ACQUIRE NEXT STOPSIG;
                      IF  !ACQUIRE NEXT REST;


present nus12        NEXT REST;
present nus13        NEXT REST;
present nus14        NEXT REST;
present nus15        NEXT REST;

}
```

```
SYNC5 PAL Simulation Output

SYNC5.SO

CSIM: CUPL Simulation Program
Version 3.00 Serial# MD-300-5446
Copyright (C)  1983,1988 Logical Devices, Inc.
CREATED Thu Oct 31 00:20:15 1991

LISTING FOR SIMULATION FILE: SYNC5.si

  1:
  2:          PARTNO    SYNC5;
  3:          NAME      SYNC5;
  4:          DATE      9/22/91   ;
  5:          REV       ;
  6:          DESIGNER  ROBERT M. BASS;
  7:          COMPANY   TEXAS INSTRUMENTS;
  8:          ASSEMBLY  ;
  9:          LOCATION  TEST ENGINEERING;
 10:
 11: /*************************************************************************/
 12: /*                                                                       */
 13: /*************************************************************************/
 14: /*  Allowable Target Device Types: ADV. u DEV. (AMD) - G16V8              */
 15: /*************************************************************************/
 16:
 17: BASE: HEX;
 18:
 19: /* ALL INPUT'S AND OUTPUT'S ARE ACTIVE HIGH LOGIC FOR THIS SIMULATION */
 20:
 21: ORDER:
 22:       %12,CLK,%2,SYNC, %1,SELFTEST,%2,SK, %2,CSWR,%1,OUTENA,%1,ACQUIRE,%1,
 23:       VTMREXP,%1,LATHORZ,%9,SSEQ, %4, SDOTCLK,%2,%1,%2,CSEQ;
 24:
```

```
=============================================================================
                              Simulation Results
=============================================================================
  SIMULATION OF CLOCK SYNCHRONIZER SECTION OF PAL

     C  S S  00  W O A V L        CCC    S     CCCC
     L  Y E  FF  R U C T A        TTT    D     TTTT
     K  N L  10  C T Q M T        RRR    O     LLLL
        C F      S E U R H        210    T     3210
          T        N I E O              C
          E        A R X R              L
          S          E P Z              K
     ---------------------------------------------------
 POWER UP CONDITION
0001:            C  0 1  00  0 0 0 0 0        NNN    N     NNNN
0002:            C  0 1  00  0 0 0 0 0        NNN    N     NNNN
0003:            0  0 0  00  0 0 0 0 0        HHH    N     NNNN
0004:            C  0 0  00  0 0 0 0 0        LLL    H     NNNN
 SIMULATION WILL NOT USE INVERTED OUTPUT CONDITION
 CHECK STRAIGHT COUNTING 0 TO 3 FOR SYNC =0
0005:            0  0 0  00  0 0 0 0 0        LLL    H     NNNN
0006:            C  0 0  00  0 0 0 0 0        LLH    L     NNNN
0007:            C  0 0  00  0 0 0 0 0        LHL    L     NNNN
0008:            C  0 0  00  0 0 0 0 0        LHH    H     NNNN
0009:            C  0 0  00  0 0 0 0 0        LLL    H     NNNN
0010:            C  0 0  00  0 0 0 0 0        LLH    L     NNNN
0011:            C  0 0  00  0 0 0 0 0        LHL    L     NNNN
0012:            C  0 0  00  0 0 0 0 0        LHH    H     NNNN
0013:            C  0 0  00  0 0 0 0 0        LLL    H     NNNN
0014:            C  0 0  00  0 0 0 0 0        LLH    L     NNNN
0015:            C  0 0  00  0 0 0 0 0        LHL    L     NNNN
0016:            C  0 0  00  0 0 0 0 0        LHH    H     NNNN
 CHECK STRAIGHT COUNTING 0 TO 4,5,6,7 FOR SYNC =1
0017:            C  0 0  00  0 0 0 0 0        LLL    H     NNNN
0018:            C  1 0  00  0 0 0 0 0        HLL    L     NNNN
0019:            C  1 0  00  0 0 0 0 0        HLH    L     NNNN
0020:            C  1 0  00  0 0 0 0 0        HHL    L     NNNN
0021:            C  1 0  00  0 0 0 0 0        HHH    H     NNNN
0022:            C  1 0  00  0 0 0 0 0        HLL    H     NNNN
0023:            C  1 0  00  0 0 0 0 0        HLH    L     NNNN
0024:            C  1 0  00  0 0 0 0 0        HHL    L     NNNN
0025:            C  1 0  00  0 0 0 0 0        HHH    H     NNNN
0026:            C  1 0  00  0 0 0 0 0        HLL    H     NNNN
0027:            C  1 0  00  0 0 0 0 0        HLH    L     NNNN
0028:            C  1 0  00  0 0 0 0 0        HHL    L     NNNN
0029:            C  1 0  00  0 0 0 0 0        HHH    H     NNNN
 CHECK SYNC ACT.TRANSITION COUNTING 0 TO 4, 1 TO 4, 2 TO 4, 3 TO 4
0030:            C  0 0  00  0 0 0 0 0        LLL    H     NNNN
0031:            C  1 0  00  0 0 0 0 0        HLL    L     NNNN
0032:            C  0 0  00  0 0 0 0 0        LLH    L     NNNN
0033:            C  1 0  00  0 0 0 0 0        HLL    L     NNNN
```

```
0034:            C 0 0  00  0 0 0 0 0        LLH    L     NNNN
0035:            C 0 0  00  0 0 0 0 0        LHL    L     NNNN
0036:            C 1 0  00  0 0 0 0 0        HLL    H     NNNN
0037:            C 0 0  00  0 0 0 0 0        LLH    L     NNNN
0038:            C 0 0  00  0 0 0 0 0        LHL    L     NNNN
0039:            C 0 0  00  0 0 0 0 0        LHH    H     NNNN
0040:            C 1 0  00  0 0 0 0 0        HLL    H     NNNN
 CHECK SYNC INACT.TRANSITION COUNTING 5 TO 2, 1 TO 4, 2 TO 4, 3 TO 4
0041:            C 1 0  00  0 0 0 0 0        HLH    L     NNNN
0042:            C 0 0  00  0 0 0 0 0        LHL    L     NNNN
0043:            C 1 0  00  0 0 0 0 0        HLL    H     NNNN
0044:            C 1 0  00  0 0 0 0 0        HLH    L     NNNN
0045:            C 1 0  00  0 0 0 0 0        HHL    L     NNNN
0046:            C 0 0  00  0 0 0 0 0        LHH    H     NNNN
0047:            C 1 0  00  0 0 0 0 0        HLL    H     NNNN
0048:            C 1 0  00  0 0 0 0 0        HLH    L     NNNN
0049:            C 1 0  00  0 0 0 0 0        HHL    L     NNNN
0050:            C 1 0  00  0 0 0 0 0        HHH    H     NNNN
0051:            C 0 0  00  0 0 0 0 0        LLL    H     NNNN
   SIMULATION OF SIGNATURE CLOCK START/STOP CONTROL

     C  S S  00  W O A V L        CCC   S     CCCC
     L  Y E  FF  R U C T A        TTT   D     TTTT
     K  N L  10  C T Q M T        RRR   O     LLLL
        C F      S D U R H        210   T     3210
          T        R I E O              C
          E        V R X R              L
          S          E P Z              K
     ----------------------------------------------------
 POWER UP CONDITION
0052:            C 0 0  00  0 0 0 0 0        NNN    N     NNNN
0053:            C 0 0  00  0 0 0 0 0        NNN    N     LLLL
0054:            C 0 0  00  0 0 0 0 0        NNN    N     LLLL
 STATE 1 SETUP SYNCHRONIZE
0055:            C 0 0  00  0 0 1 0 0        NNN    N     LLLL
0056:            C 0 0  00  0 0 0 0 1        NNN    N     LLLL
0057:            C 0 0  00  0 0 1 0 1        NNN    N     LLLH
0058:            C 0 0  00  0 0 1 0 1        NNN    N     LLLH
 STATE 2 ANY SYNC #1
0059:            C 0 0  00  0 0 1 0 1        NNN    N     LLLH
0060:            C 0 0  00  0 0 1 0 0        NNN    N     LLHL
0061:            C 0 0  00  0 0 1 0 0        NNN    N     LLHL
 STATE 2 ANY SYNC #1 NOT HORIZONAL INTERVAL ABORT
0062:            C 0 0  00  0 0 1 1 0        NNN    N     LLHL
0063:            C 0 0  00  0 0 1 1 0        NNN    N     LLHL
0064:            C 0 0  00  0 0 1 1 1        NNN    N     LLLH
0065:            C 0 0  00  0 0 1 1 1        NNN    N     LLLH
 REESTABLISH STATE 2 ANY SYNC #1
0066:            C 0 0  00  0 0 1 0 1        NNN    N     LLLH
0067:            C 0 0  00  0 0 1 0 0        NNN    N     LLHL
0068:            C 0 0  00  0 0 1 0 0        NNN    N     LLHL
 STATE 3 1ST HORIZONAL SYNC
```

```
0069:          C 0 0  00  0 0 1 0 0      NNN   N    LLHL
0070:          C 0 0  00  0 0 1 0 1      NNN   N    LLHH
0071:          C 0 0  00  0 0 1 0 1      NNN   N    LLHH
 STATE 4 ANY SYNC #2
0072:          C 0 0  00  0 0 1 0 1      NNN   N    LLHH
0073:          C 0 0  00  0 0 1 0 0      NNN   N    LHLL
0074:          C 0 0  00  0 0 1 0 0      NNN   N    LHLL
 STATE 4 ANY SYNC #2 NOT HORIZONAL INTERVAL ABORT
0075:          C 0 0  00  0 0 1 1 0      NNN   N    LHLL
0076:          C 0 0  00  0 0 1 1 0      NNN   N    LHLL
0077:          C 0 0  00  0 0 1 1 1      NNN   N    LLLH
0078:          C 0 0  00  0 0 1 1 1      NNN   N    LLLH
 REESTABLISH STATE 2 ANY SYNC #1
0079:          C 0 0  00  0 0 1 0 1      NNN   N    LLLH
0080:          C 0 0  00  0 0 1 0 0      NNN   N    LLHL
0081:          C 0 0  00  0 0 1 0 0      NNN   N    LLHL
 STATE 3 1ST HORIZONAL SYNC
0082:          C 0 0  00  0 0 1 0 0      NNN   N    LLHL
0083:          C 0 0  00  0 0 1 0 1      NNN   N    LLHH
0084:          C 0 0  00  0 0 1 0 1      NNN   N    LLHH
 STATE 4 ANY SYNC #2
0085:          C 0 0  00  0 0 1 0 1      NNN   N    LLHH
0086:          C 0 0  00  0 0 1 0 0      NNN   N    LHLL
0087:          C 0 0  00  0 0 1 0 0      NNN   N    LHLL
 STATE 5 2ND HORIZONAL SYNC
0088:          C 0 0  00  0 0 1 0 0      NNN   N    LHLL
0089:          C 0 0  00  0 0 1 0 1      NNN   N    LHLH
0090:          C 0 0  00  0 0 1 0 1      NNN   N    LHLH
 STATE 6 ANY SYNC #3
0091:          C 0 0  00  0 0 1 0 1      NNN   N    LHLH
0092:          C 0 0  00  0 0 1 0 0      NNN   N    LHHL
0093:          C 0 0  00  0 0 1 0 0      NNN   N    LHHL
 STATE 5 Nth HORIZONAL SYNC
0094:          C 0 0  00  0 0 1 0 0      NNN   N    LHHL
0095:          C 0 0  00  0 0 1 0 1      NNN   N    LHLH
0096:          C 0 0  00  0 0 1 0 1      NNN   N    LHLH
 STATE 6 NTH HORIZONAL ANY SYNC #3
0097:          C 0 0  00  0 0 1 0 1      NNN   N    LHLH
0098:          C 0 0  00  0 0 1 0 0      NNN   N    LHHL
0099:          C 0 0  00  0 0 1 0 0      NNN   N    LHHL
 STATE 8 1ST VERTICAL SYNC=START OF SIGNATURE GOTO STATE 8
0100:          C 0 0  00  0 0 1 0 0      NNN   N    LHHL
0101:          C 0 0  00  0 0 1 1 0      NNN   N    LHHL
0102:          C 0 0  00  0 0 1 1 1      NNN   N    HLLL
 STATE 9 ANY SYNC #4
0103:          C 0 0  00  0 0 1 1 1      NNN   N    HLLL
0104:          C 0 0  00  0 0 1 0 0      NNN   N    HLLH
0105:          C 0 0  00  0 0 1 0 0      NNN   N    HLLH
 STATE 8 Nth VERTICAL SYNC
0106:          C 0 0  00  0 0 1 0 0      NNN   N    HLLH
0107:          C 0 0  00  0 0 1 1 0      NNN   N    HLLH
0108:          C 0 0  00  0 0 1 1 1      NNN   N    HLLL
```

```
 STATE 9 Nth ANY SYNC #4
0109:            C 0 0  00  0 0 1 1 1        NNN    N     HLLL
0110:            C 0 0  00  0 0 1 0 0        NNN    N     HLLH
0111:            C 0 0  00  0 0 1 0 0        NNN    N     HLLH
 STATE 10 1ST FRAME HORIZONAL SYNC
0112:            C 0 0  00  0 0 1 0 0        NNN    N     HLLH
0113:            C 0 0  00  0 0 1 0 0        NNN    N     HLLH
0114:            C 0 0  00  0 0 1 0 1        NNN    N     HLHL
 STATE 9 ANY SYNC #5 RETRY HORIZONAL
0115:            C 0 0  00  0 0 1 0 1        NNN    N     HLHL
0116:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
0117:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
 STATE 10 Nth FRAME HORIZONAL SYNC
0118:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
0119:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
0120:            C 0 0  00  0 0 1 0 1        NNN    N     HLHL
 STATE 11 ANY SYNC #5
0121:            C 0 0  00  0 0 1 0 1        NNN    N     HLHL
0122:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
0123:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
 STATE 12 END OF FRAME VERTICAL SYNC
0124:            C 0 0  00  0 0 1 0 0        NNN    N     HLHH
0125:            C 0 0  00  0 0 1 1 0        NNN    N     HLHH
0126:            C 0 0  00  0 0 1 1 1        NNN    N     LHHH
0127:            C 0 0  00  0 0 1 0 1        NNN    N     LHHH
0128:            C 0 0  00  0 0 1 0 0        NNN    N     LHHH
 STATE 7 EXIT STOP SIGNATURE
0129:            C 0 0  00  0 0 1 0 0        NNN    N     LHHH
0130:            C 0 0  00  0 0 0 0 0        NNN    N     LLLL
   SIMULATION OF SELFTEST CLOCK SYNCHRONIZER SECTION OF PAL

     C  S S  00  W O A V L        CCC    S     CCCC
     L  Y E  FF  R U C T A        TTT    D     TTTT
     K  N L  10  C T Q N T        RRR    O     LLLL
        C F      S D U R H        210    T     3210
          T        R I E O              C
          E        V R X R              L
          S          E P Z              K
     ------------------------------------------------------
 POWER UP CONDITION
0131:            C 0 1  00  0 0 1 0 0        NNN    N     NNNN
0132:            C 0 1  00  0 0 1 0 0        NNN    N     NNNN
0133:            0 0 1  00  0 0 0 0 0        NNN    N     NNNN
0134:            C 0 1  00  0 0 0 0 0        NNN    L     NNNN
 SIMULATION WILL NOT USE INVERTED OUTPUT CONDITION
 CHECK STRAIGHT COUNTING 0 TO 3 FOR SYNC =0
0135:            C 0 1  00  0 0 0 0 0        NNN    L     NNNN
0136:            C 0 1  00  0 0 0 0 0        NNN    L     NNNN
0137:            C 0 1  00  1 0 0 0 0        NNN    H     NNNN
0138:            C 0 1  00  1 0 0 0 0        NNN    H     NNNN
0139:            C 0 1  00  0 0 0 0 0        NNN    L     NNNN
0140:            C 0 1  00  0 0 0 0 0        NNN    L     NNNN
```

| | | |

CTRSIG3L PAL Program Listing

CTRSIG3L.PLD 6088 10-25-91 10:32am

```
               PARTNO    CTRSIG3L;
               NAME      CTRSIG3L;
               DATE      8/24/91 ;
               REV       ;
               DESIGNER  ROBERT M. BASS;
               COMPANY   TEXAS INSTRUMENTS;
               ASSEMBLY  ;
               LOCATION  TEST ENGINEERING;

/*************************************************************************/
/*                                                                       */
/*                                                                       */
/*************************************************************************/
/*  Allowable Target Device Types: ADV. u DEV. (AMD) - 22V10             */
/*************************************************************************/

/**  Inputs  **/

PIN 1        = CLK       ;      /*                */
PIN 2        = M0        ;      /*                */
PIN 3        = M1        ;      /*                */
PIN 4        = M2        ;      /*                */
PIN 5        = BCCIN     ;      /*                */
PIN 6        = XORIN     ;      /*                */
PIN 7        = nc4       ;      /*                */
PIN 8        = SETCLR    ;      /*                */
PIN 9        = nc2       ;      /*                */
PIN 10       = nc1       ;      /*                */
PIN 11       = !DEVRD    ;      /*                */
PIN 12       = GND       ;      /*                */
PIN 13       = !GATE     ;      /*                */

/**  OUTPUT  **/

PIN 14       = DB0       ;      /*          */
PIN 15       = DB1       ;      /*          */
PIN 16       = BCCOUT    ;      /*          */
PIN 17       = DB4       ;      /*          */
PIN 18       = DB6       ;      /*          */
PIN 19       = DB7       ;      /*          */
PIN 20       = DB5       ;      /*          */
PIN 21       = DB3       ;      /*          */
PIN 22       = DB2       ;      /*          */
PIN 23       = XOROUT    ;      /*          */
PIN 24       = VCC       ;

$DEFINE ON  'b'1
$DEFINE OFF 'b'0
```

```
$DEFINE BCC0 (!DB0)
$DEFINE BCC1 (DB0)
$DEFINE BCC2 (DB1 & DB0)
$DEFINE BCC3 (DB2 & DB1 & DB0)
$DEFINE BCC4 (DB3 & DB2 & DB1 & DB0)
$DEFINE BCC5 (DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC6 (DB5 & DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC7 (DB6 & DB5 & DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC8 (DB7 & DB6 & DB5 & DB4 & DB3 & DB2 & DB1 & DB0)

/**  MODES CURRENTLY UNRELATED **/
$DEFINE HLD    (!M2 & !M1 & !M0) /** HOLD REGISTER RESULT   **/
$DEFINE CTR    (!M2 & !M1 &  M0) /** ACCUMULATE COUNT        **/

/**  MODES RELATED TO LOGICAL SHIFTING **/
$DEFINE SIG    (!M2 &  M1 & !M0) /** ACCUMULATE SIGNATURE   **/
$DEFINE SHF    (!M2 &  M1 &  M0) /** SHIFT PIPELINE **/

/**  MODES RELATED TO FREEZE LAST SETTING **/
$DEFINE NOTUSED0 (M2 & !M1 & !M0) /**  **/
$DEFINE NOTUSED1 (M2 & !M1 &  M0) /**  **/
$DEFINE NOTUSED2 (M2 &  M1 & !M0) /**  **/
$DEFINE NOTUSED3 (M2 &  M1 &  M0) /**  **/


/**  Logic Equations  **/
LSF = (SIG # SHF);                          /** MODES THAT LOGICAL SHFT **/


/**  IC POSITION DEVELOPMENT CODE FOR PRODUCING EACH PAL **/
LOW  =  ON;                           /** LOWER  REGISTER B0..B7 **/


 [DB0..7].AR = CTR & SETCLR;                /** SET COUNTER TO 00     **/
 [DB0..7].SP = SIG & SETCLR;                /** SET COUNTER TO FF     **/

/** GENERAL FORMULA USED FOR CRC16 GENERATION **/
/** NOTE:  CRC16  G(X)= X^16 + X^12 + X^5 + 1      **/
/**               G(X)= !(X^16 + X^12 + X^5)       **/
/** LOW- REPRESENTS LEAST SIG. DEVICE IN COUNTER CHAIN **/

XOROUT.OE = ON;                                    /** OUTPUT ALWAYS ACTIVE  **/
XOROUT=  !DB4 & !XORIN   &  BCCIN & LOW            /** X^5 XOR DATA XOR XORIN**/
     #  !DB4 &  XORIN   & !BCCIN & LOW
     #   DB4 & !XORIN   & !BCCIN & LOW
     #   DB4 &  XORIN   &  BCCIN & LOW;



[DB0..DB7].OE = DEVRD;                        /** uP READ COUNTER REGISTER **/

 DB0.D =  BCC0   &  CTR &  BCCIN  &  GATE /** ON/OFF/ON TOGGLE TERM **/
```

```
      #  DB0                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB0    &  HLD                        /** HOLD FREEZE           **/
      #  BCCIN &  SHF           &  GATE       /** SERIAL SHIFT INPUT     **/
      # !XOROUT & SIG &  LOW    &  GATE;      /**SIG TERM LOW  DEVICE   **/


DB1.D = !DB1 &  BCC1 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
      #  DB1 & !BCC1 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
      #  DB1                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB1    &  HLD                        /** HOLD FREEZE           **/
      #  DB0    &  SHF          &  GATE       /** SERIAL SHIFT INPUT     **/
      #  DB0    &  SIG          &  GATE;      /** SIGNATURE SHIFT        **/


DB2.D = !DB2 &  BCC2 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
      #  DB2 & !BCC2 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
      #  DB2                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB2    &  HLD                        /** HOLD FREEZE           **/
      #  DB1    &  SHF          &  GATE       /** SERIAL SHIFT INPUT     **/
      #  DB1    &  SIG          &  GATE;      /** SIGNATURE SHIFT        **/


DB3.D = !DB3 &  BCC3 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
      #  DB3 & !BCC3 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
      #  DB3                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB3    &  HLD                        /** HOLD FREEZE           **/
      #  DB2    &  SHF          &  GATE       /** SERIAL SHIFT INPUT     **/
      #  DB2    &  SIG          &  GATE;      /** SIGNATURE SHIFT        **/


DB4.D = !DB4 &  BCC4 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
      #  DB4 & !BCC4 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
      #  DB4                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB4    &  HLD                        /** HOLD FREEZE           **/
      #  DB3    &  SHF          &  GATE       /** SERIAL SHIFT INPUT     **/
      #  DB3    &  SIG          &  GATE;      /** SIGNATURE SHIFT        **/


DB5.D = !DB5 &  BCC5 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
      #  DB5 & !BCC5 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
      #  DB5                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB5    &  HLD                        /** HOLD FREEZE           **/
      #  DB4    &  SHF          &  GATE       /** SERIAL SHIFT INPUT     **/
      #  DB4    &  SIG          &  GATE;      /** SIGNATURE SHIFT        **/


DB6.D = !DB6 &  BCC6 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
      #  DB6 & !BCC6 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
      #  DB6                    & !GATE       /** ANYMODE FREEZE        **/
      #  DB6    &  HLD                        /** HOLD FREEZE           **/
      #  DB5    &  SHF          &  GATE       /** SERIAL SHIFT INPUT     **/
      #  DB5    &  SIG          &  GATE;      /** SIGNATURE SHIFT        **/
```

```
DB7.D = !DB7 &  BCC7 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB7 & !BCC7 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB7                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB7    &  HLD                         /** HOLD FREEZE            **/
      #  DB6    &  SHF          &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB6    &  SIG          &  GATE;       /** SIGNATURE SHIFT        **/


BCCOUT    =  BCC8 & CTR          /** COUNTER LOOK AHEAD CARRY**/
          #  DB7 & SHF                  /** SERIAL SHIFT OUTPUT     **/
          #  DB7 & SIG;                 /** SIGNATURE SHIFT OUTPUT  **/
BCCOUT.OE = ON;                           /** OUTPUT ALWAYS ACTIVE    **/
```

CTRSIG3L PAL Simulation Output

CTRSIG3L.SO

CSIM: CUPL Simulation Program
Version 3.00 Serial# MD-300-5446
Copyright (C) 1983,1988 Logical Devices, Inc.
CREATED Sun Sep 22 18:05:55 1991

LISTING FOR SIMULATION FILE: CTRSIG3L.si

```
  1:
  2:          PARTNO    CTRSIG3L;
  3:          NAME      CTRSIG3L;
  4:               DATE      4/18/88   ;
  5:          REV       ;
  6:          DESIGNER  ROBERT M. BASS;
  7:          COMPANY   TEXAS INSTRUMENTS;
  8:          ASSEMBLY  ;
  9:          LOCATION  TEST ENGINEERING;
 10:
 11: /*************************************************************************/
 12: /*                                                                       */
 13: /*************************************************************************/
 14: /*  Allowable Target Device Types: ADV. u DEV. (AMD) - 22V10              */
 15: /*************************************************************************/
 16:
 17: BASE: HEX;
 18: FIELD MODE=[M2..M0];
 19: FIELD DBUS=[DB7..0];
 20:
 21: ORDER:
 22:    %7,CLK,%2,MODE,%2,GATE,%2,BCCIN,%2,XORIN,%2,DEVRD,%3,
 23:    !SETCLR,%4,DBUS,%2,BCCOUT,%2,XOROUT;
 24:
 25:
```

```
================================================================================
                                Simulation Results
================================================================================

 C  MMM  G  B  X  D   S    DDDDDDDD  B  X
 L  210  A  C  O  E   E    BBBBBBBB  C  O
 K       T  C  R  V   T    76543210  C  R
         E  I  I  R   C              O  O
            N  N  D   L              U  U
                      R              T  T
                      *
  POWER-UP PRESET FLIP=FLOP SET TO FF
0001:       0  000  0  1  X  1   1    LLLLLLLL  L  N
  CHECK TRI-STATE OUTPUT CAPABILITY
0002:       0  000  0  1  X  0   1    ZZZZZZZZ  L  N
0003:       0  000  0  1  X  1   1    LLLLLLLL  L  N
  CHECK MODE SYNCHRONOUS PRESET SIGNATURE =FF CAPABILITY
0004:       0  010  1  1  X  1   1    LLLLLLLL  L  N
0005:       C  010  1  1  X  1   0    HHHHHHHH  H  N
0006:       C  010  1  1  X  1   0    HHHHHHHH  H  N
  CHECK MODE ASYNCHRONOUS RESET COUNTER=00 CAPABILITY
0007:       0  001  0  1  X  1   1    HHHHHHHH  H  N
0008:       0  001  0  1  X  1   1    HHHHHHHH  H  N
0009:       C  001  0  1  X  1   0    LLLLLLLL  L  N
0010:       C  001  0  1  X  1   0    LLLLLLLL  L  N
0011:       C  001  0  1  X  1   1    LLLLLLLL  L  N
0012:       C  001  1  1  X  1   1    LLLLLLLH  L  N
  CHECK SYNCHRONOUS SHIFT IN LOGIC ONE CAPABILITY
0013:       0  011  1  1  X  1   1    LLLLLLLH  L  N
0014:       C  011  1  1  X  1   1    LLLLLLHH  L  N
0015:       C  011  1  1  X  1   1    LLLLLHHH  L  N
0016:       C  011  1  1  X  1   1    LLLLHHHH  L  N
0017:       C  011  1  1  X  1   1    LLLHHHHH  L  N
0018:       C  011  1  1  X  1   1    LLHHHHHH  L  N
0019:       C  011  1  1  X  1   1    LHHHHHHH  L  N
0020:       C  011  1  1  X  1   1    HHHHHHHH  H  N
  CHECK SYNCHRONOUS SHIFT IN LOGIC ZERO CAPABILITY
0021:       C  011  1  0  X  1   1    HHHHHHHL  H  N
0022:       C  011  1  0  X  1   1    HHHHHHLL  H  N
0023:       C  011  1  0  X  1   1    HHHHHLLL  H  N
0024:       C  011  1  0  X  1   1    HHHHLLLL  H  N
0025:       C  011  1  0  X  1   1    HHHLLLLL  H  N
0026:       C  011  1  0  X  1   1    HHLLLLLL  H  N
0027:       C  011  1  0  X  1   1    HLLLLLLL  H  N
0028:       C  011  1  0  X  1   1    LLLLLLLL  L  N
0029:       C  011  1  0  X  1   1    LLLLLLLL  L  N
  CHECK SYNCHRONOUS SHIFT IN ALTERNATE ONE CAPABILITY
0030:       C  011  1  1  X  1   1    LLLLLLLH  L  N
0031:       C  011  1  0  X  1   1    LLLLLLHL  L  N
0032:       C  011  1  1  X  1   1    LLLLLHLH  L  N
0033:       C  011  1  0  X  1   1    LLLLHLHL  L  N
```

```
0034:       C  011  1  1  X  1   1    LLLHLHLH  L  N
0035:       C  011  1  0  X  1   1    LLHLHLHL  L  N
0036:       C  011  1  1  X  1   1    LHLHLHLH  L  N
0037:       C  011  1  0  X  1   1    HLHLHLHL  H  N
0038:       C  011  1  1  X  1   1    LHLHLHLH  L  N
  CHECK MODE SYNCHRONOUS PRESET SIGNATURE=FF CAPABILITY
0039:       C  010  1  1  X  1   0    HHHHHHHH  H  N
  CHECK SHIFT GATE CONTROL CAPABILITY
0040:       C  011  0  0  X  1   1    HHHHHHHH  H  N
0041:       C  011  1  0  X  1   1    HHHHHHHL  H  N
0042:       C  011  0  0  X  1   1    HHHHHHHL  H  N
0043:       C  011  1  0  X  1   1    HHHHHHLL  H  N
0044:       C  011  0  0  X  1   1    HHHHHHLL  H  N
0045:       C  011  1  0  X  1   1    HHHHHLLL  H  N
0046:       C  011  0  0  X  1   1    HHHHHLLL  H  N
0047:       C  011  1  0  X  1   1    HHHHLLLL  H  N
0048:       C  011  0  0  X  1   1    HHHHLLLL  H  N
0049:       C  011  1  0  X  1   1    HHHLLLLL  H  N
0050:       C  011  0  0  X  1   1    HHHLLLLL  H  N
0051:       C  011  1  0  X  1   1    HHLLLLLL  H  N
0052:       C  011  0  0  X  1   1    HHLLLLLL  H  N
0053:       C  011  1  0  X  1   1    HLLLLLLL  H  N
0054:       C  011  0  0  X  1   1    HLLLLLLL  H  N
0055:       C  011  1  0  X  1   1    LLLLLLLL  L  N

 C  MMM  G  B  X  D  S    DDDDDDDD  B  X
 L  210  A  C  O  E  E    BBBBBBBB  C  O
 K       T  C  R  V  T    76543210  C  R
         E  I  I  R  C              O  O
            N  N  D  L              U  U
                     R              T  T
                     *
  CHECK COUNTER CONTROL CAPABILITY 00-07
0056:       C  001  0  1  X  1   1    LLLLLLLL  L  N
0057:       C  001  1  1  X  1   1    LLLLLLLH  L  N
0058:       C  001  1  1  X  1   1    LLLLLLHL  L  N
0059:       C  001  1  1  X  1   1    LLLLLLHH  L  N
0060:       C  001  1  1  X  1   1    LLLLLHLL  L  N
0061:       C  001  1  1  X  1   1    LLLLLHLH  L  N
0062:       C  001  1  1  X  1   1    LLLLLHHL  L  N
0063:       C  001  1  1  X  1   1    LLLLLHHH  L  N
  CHECK COUNTER CONTROL CAPABILITY 08-10
0064:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0065:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0066:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0067:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0068:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0069:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0070:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0071:       C  001  1  1  X  1   1    LLLLHHHH  L  N
0072:       C  001  1  1  X  1   1    LLLHLLLL  L  N
  CHECK COUNTER CONTROL CAPABILITY 11-20
```

```
0073:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0074:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0075:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0076:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0077:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0078:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0079:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0080:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0081:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0082:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0083:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0084:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0085:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0086:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0087:       C  001  1  1  X  1   1    LLLHHHHH  L  N
0088:       C  001  1  1  X  1   1    LLHLLLLL  L  N
  CHECK COUNTER CONTROL CAPABILITY 21-30
0089:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0090:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0091:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0092:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0093:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0094:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0095:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0096:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0097:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0098:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0099:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0100:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0101:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0102:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0103:       C  001  1  1  X  1   1    LLHLHHHH  L  N
0104:       C  001  1  1  X  1   1    LLHHLLLL  L  N
  CHECK COUNTER CONTROL CAPABILITY 31-40
0105:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0106:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0107:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0108:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0109:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0110:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0111:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0112:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0113:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0114:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0115:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0116:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0117:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0118:       C  001  1  1  X  1   1    NNNNNNNN  L  N
0119:       C  001  1  1  X  1   1    LLHHHHHH  L  N
0120:       C  001  1  1  X  1   1    LHLLLLLL  L  N
  CHECK COUNTER CONTROL CAPABILITY 41-80
0121:       C  001  1  1  X  1   1    NNNNNNNN  L  N
```

```
0122:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0123:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0124:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0125:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0126:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0127:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0128:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0129:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0130:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0131:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0132:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0133:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0134:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0135:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0136:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0137:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0138:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0139:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0140:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0141:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0142:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0143:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0144:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0145:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0146:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0147:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0148:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0149:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0150:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0151:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0152:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0153:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0154:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0155:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0156:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0157:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0158:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0159:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0160:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0161:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0162:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0163:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0164:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0165:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0166:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0167:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0168:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0169:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0170:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0171:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0172:      C  001  1  1  X  1  1    NNNNNNNN  L  N
0173:      C  001  1  1  X  1  1    NNNNNNNN  L  N
```

```
0174:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0175:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0176:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0177:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0178:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0179:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0180:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0181:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0182:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0183:       C  001  1  1  X  1  1    LHHHHHHH  L  N
0184:       C  001  1  1  X  1  1    HLLLLLLL  L  N
0185:       C  001  1  1  X  1  1    HLLLLLLH  L  N
  CHECK COUNTER CONTROL CAPABILITY 81-00
0186:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0187:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0188:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0189:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0190:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0191:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0192:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0193:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0194:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0195:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0196:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0197:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0198:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0199:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0200:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0201:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0202:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0203:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0204:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0205:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0206:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0207:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0208:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0209:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0210:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0211:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0212:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0213:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0214:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0215:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0216:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0217:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0218:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0219:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0220:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0221:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0222:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0223:       C  001  1  1  X  1  1    NNNNNNNN  L  N
0224:       C  001  1  1  X  1  1    NNNNNNNN  L  N
```

```
0225:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0226:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0227:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0228:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0229:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0230:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0231:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0232:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0233:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0234:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0235:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0236:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0237:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0238:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0239:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0240:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0241:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0242:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0243:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0244:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0245:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0246:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0247:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0248:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0249:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0250:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0251:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0252:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0253:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0254:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0255:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0256:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0257:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0258:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0259:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0260:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0261:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0262:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0263:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0264:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0265:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0266:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0267:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0268:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0269:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0270:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0271:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0272:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0273:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0274:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0275:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
0276:     C  001  1  1  X  1  1   NNNNNNNNN  L  N
```

```
0277:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0278:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0279:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0280:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0281:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0282:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0283:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0284:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0285:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0286:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0287:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0288:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0289:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0290:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0291:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0292:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0293:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0294:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0295:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0296:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0297:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0298:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0299:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0300:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0301:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0302:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0303:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0304:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0305:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0306:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0307:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0308:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0309:        C  001  1  1  X  1   1    NNNNNNNN  L  N
0310:        C  001  1  1  X  1   1    HHHHHHHL  L  N
0311:        C  001  1  1  X  1   1    HHHHHHHH  H  N
0312:        C  001  1  1  X  1   1    LLLLLLLL  L  N
  CHECK HOLD MODE RETAINS VALUE
0313:        C  000  1  1  1  1   0    LLLLLLLL  L  N
0314:        C  000  1  1  1  1   0    LLLLLLLL  L  N
0315:        C  000  1  1  1  1   0    LLLLLLLL  L  N

 C  MMM  G  B  X  D   S    DDDDDDDD  B  X
 L  210  A  C  O  E   E    BBBBBBBB  C  O
 K       T  C  R  V   T    76543210  C  R
         E  I  I  R   C              O  O
            N  N  D   L              U  U
                      R              T  T
                      *
  CHECK SIGNATURE SHIFT MODE --LOW EQUATION TEST--
  XOROUT TRUTH TABLE LOW DEVICE TEST BCCIN,XORIN,DB4=0
0316:        0  010  1  0  0  1   1    LLLLLLLL  L  L
0317:        0  010  1  0  1  1   1    LLLLLLLL  L  H
```

```
0318:       0  010  1  1  0  1  1    LLLLLLLL  L  H
0319:       0  010  1  1  1  1  1    LLLLLLLL  L  L
  SET DB4 BY SHIFTING IN ONE
0320:       C  011  1  1  0  1  1    LLLLLLLH  L  N
0321:       C  011  1  0  0  1  1    LLLLLLHL  L  N
0322:       C  011  1  0  0  1  1    LLLLLHLL  L  N
0323:       C  011  1  0  0  1  1    LLLLHLLL  L  N
0324:       C  011  1  0  0  1  1    LLLHLLLL  L  N
  XOROUT TRUTH TABLE LOW DEVICE TEST BCCIN,XORIN,DB4=1
0325:       0  010  1  0  0  1  1    LLLHLLLL  L  H
0326:       0  010  1  0  1  1  1    LLLHLLLL  L  L
0327:       0  010  1  1  0  1  1    LLLHLLLL  L  L
0328:       0  010  1  1  1  1  1    LLLHLLLL  L  H
  SHIFT IN PATTERN OF 55 TO TEST SIGNATURE GENERATION
0329:       0  010  1  0  0  1  1    LLLHLLLL  L  H
0330:       C  010  1  0  0  1  1    LLHLLLLL  L  L
0331:       C  010  1  1  0  1  1    LHLLLLLL  L  H
0332:       C  010  1  0  0  1  1    HLLLLLLH  H  L
0333:       C  010  1  1  0  1  1    LLLLLLHL  L  H
0334:       C  010  1  0  0  1  1    LLLLLHLH  L  L
0335:       C  010  1  1  0  1  1    LLLLHLHL  L  H
0336:       C  010  1  0  0  1  1    LLLHLHLH  L  H
0337:       C  010  1  1  0  1  1    LLHLHLHH  L  H
  SHIFT IN PATTERN OF AA TO TEST SIGNATURE GENERATION
0338:       C  010  1  1  0  1  1    LHLHLHHL  L  L
0339:       C  010  1  0  0  1  1    HLHLHHLL  H  L
0340:       C  010  1  1  0  1  1    LHLHHLLL  L  L
0341:       C  010  1  0  0  1  1    HLHHLLLL  H  H
0342:       C  010  1  1  0  1  1    LHHLLLLH  L  H
0343:       C  010  1  0  0  1  1    HHLLLLHH  H  L
0344:       C  010  1  1  0  1  1    HLLLLHHL  H  H
0345:       C  010  1  0  0  1  1    LLLLHHLH  L  L
  SHIFT IN PATTERN OF FF TO TEST SIGNATURE GENERATION
0346:       C  010  1  1  0  1  1    LLLHHLHL  L  L
0347:       C  010  1  1  0  1  1    LLHHLHLH  L  L
0348:       C  010  1  1  0  1  1    LHHLHLHH  L  H
0349:       C  010  1  1  0  1  1    HHLHLHHL  H  L
0350:       C  010  1  1  0  1  1    HLHLHHLH  H  H
0351:       C  010  1  1  0  1  1    LHLHHLHL  L  L
0352:       C  010  1  1  0  1  1    HLHHLHLH  H  L
0353:       C  010  1  1  0  1  1    LHHLHLHH  L  H
  SHIFT IN PATTERN OF 00 TO TEST SIGNATURE GENERATION
0354:       C  010  1  0  0  1  1    HHLHLHHH  H  H
0355:       C  010  1  0  0  1  1    HLHLHHHL  H  L
0356:       C  010  1  0  0  1  1    LHLHHHLH  L  H
0357:       C  010  1  0  0  1  1    HLHHHLHL  H  H
0358:       C  010  1  0  0  1  1    LHHHLHLL  L  H
0359:       C  010  1  0  0  1  1    HHHLHLLL  H  L
0360:       C  010  1  0  0  1  1    HHLHLLLH  H  H
0361:       C  010  1  0  0  1  1    HLHLLLHL  H  L
```

CTRSIG3M PAL Program Listing

CTRSIG3M.PLD 6141 10-25-91 10:33am

```
PARTNO    CTRSIG3M;
NAME      CTRSIG3M;
DATE      8/24/91 ;
REV       ;
DESIGNER  ROBERT M. BASS;
COMPANY   TEXAS INSTRUMENTS;
ASSEMBLY  ;
LOCATION  TEST ENGINEERING;

/*************************************************************************/
/*                                                                       */
/*                                                                       */
/*************************************************************************/
/*  Allowable Target Device Types: ADV. u DEV. (AMD) - 22V10             */
/*************************************************************************/

/**  Inputs  **/

PIN 1       = CLK       ;     /*                */
PIN 2       = M0        ;     /*                */
PIN 3       = M1        ;     /*                */
PIN 4       = M2        ;     /*                */
PIN 5       = BCCIN     ;     /*                */
PIN 6       = XORIN     ;     /*                */
PIN 7       = nc4       ;     /*                */
PIN 8       = SETCLR    ;     /*                */
PIN 9       = nc2       ;     /*                */
PIN 10      = nc1       ;     /*                */
PIN 11      = !DEVRD    ;     /*                */
PIN 12      = GND       ;     /*                */
PIN 13      = !GATE     ;     /*                */

/**  OUTPUT  **/

PIN 14      = DB0       ;     /*          */
PIN 15      = DB1       ;     /*          */
PIN 16      = BCCOUT    ;     /*          */
PIN 17      = DB4       ;     /*          */
PIN 18      = DB6       ;     /*          */
PIN 19      = DB7       ;     /*          */
PIN 20      = DB5       ;     /*          */
PIN 21      = DB3       ;     /*          */
PIN 22      = DB2       ;     /*          */
PIN 23      = XOROUT    ;     /*          */
PIN 24      = VCC       ;

$DEFINE ON  'b'1
```

```
$DEFINE OFF 'b'0
$DEFINE BCC0 (!DB0)
$DEFINE BCC1 (DB0)
$DEFINE BCC2 (DB1 & DB0)
$DEFINE BCC3 (DB2 & DB1 & DB0)
$DEFINE BCC4 (DB3 & DB2 & DB1 & DB0)
$DEFINE BCC5 (DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC6 (DB5 & DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC7 (DB6 & DB5 & DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC8 (DB7 & DB6 & DB5 & DB4 & DB3 & DB2 & DB1 & DB0)

/**  MODES CURRENTLY UNRELATED **/
$DEFINE HLD    (!M2 & !M1 & !M0) /** HOLD REGISTER RESULT   **/
$DEFINE CTR    (!M2 & !M1 &  M0) /** ACCUMULATE COUNT        **/

/**  MODES RELATED TO LOGICAL SHIFTING **/
$DEFINE SIG    (!M2 &  M1 & !M0) /** ACCUMULATE SIGNATURE   **/
$DEFINE SHF    (!M2 &  M1 &  M0) /** SHIFT PIPELINE **/

/**  MODES RELATED TO FREEZE LAST SETTING **/
$DEFINE NOTUSED0 (M2 & !M1 & !M0) /**  **/
$DEFINE NOTUSED1 (M2 & !M1 &  M0) /**  **/
$DEFINE NOTUSED2 (M2 &  M1 & !M0) /**  **/
$DEFINE NOTUSED3 (M2 &  M1 &  M0) /**  **/


/**  Logic Equations  **/
LSF = (SIG # SHF);                          /** MODES THAT LOGICAL SHFT **/


/**  IC POSITION DEVELOPMENT CODE FOR PRODUCING EACH PAL **/
MID  =  ON;                           /** MIDDLE REGISTER B8..B15**/


 [DB0..7].AR = CTR & SETCLR;                /** SET COUNTER TO 00     **/
 [DB0..7].SP = SIG & SETCLR;                /** SET COUNTER TO FF     **/

/** GENERAL FORMULA USED FOR CRC16 GENERATION **/
/** NOTE:  CRC16  G(X)= X^16 + X^12 + X^5 + 1      **/
/**               G(X)= !(X^16 + X^12 + X^5)       **/
/** MID- REPRESENTS MIDDLE DEVICE IN COUNTER CHAIN **/

XOROUT.OE = ON;                                  /** OUTPUT ALWAYS ACTIVE  **/
XOROUT=  !DB7 & !DB3 &   XORIN &  MID       /** X^12 X^16 CRC16 SET XORIN=0**/
     #  !DB7 &  DB3 & !XORIN &  MID
     #   DB7 & !DB3 & !XORIN &  MID
     #   DB7 &  DB3 &   XORIN &  MID;
/** NOTE MID DEVICE XORIN WILL BE GROUNDED FOR CRC16 **/


[DB0..DB7].OE = DEVRD;                       /** uP READ COUNTER REGISTER **/
```

```
DB0.D =  BCC0   &  CTR &  BCCIN  &  GATE /** ON/OFF/ON TOGGLE TERM **/
      #  DB0                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB0    &  HLD                          /** HOLD FREEZE           **/
      #  BCCIN &  SHF            &  GATE        /** SERIAL SHIFT INPUT    **/
      #  BCCIN &  SIG &  MID     &  GATE;       /**SIG TERM MID  DEVICE   **/


DB1.D = !DB1 &  BCC1 &  CTR                     /** OFF TO ON  TOGGLE TERM **/
      #  DB1 & !BCC1 &  CTR                     /** ON  TO OFF TOGGLE TERM **/
      #  DB1                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB1    &  HLD                          /** HOLD FREEZE           **/
      #  DB0    &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
      #  DB0    &  SIG           &  GATE;       /** SIGNATURE SHIFT       **/


DB2.D = !DB2 &  BCC2 &  CTR                     /** OFF TO ON  TOGGLE TERM **/
      #  DB2 & !BCC2 &  CTR                     /** ON  TO OFF TOGGLE TERM **/
      #  DB2                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB2    &  HLD                          /** HOLD FREEZE           **/
      #  DB1    &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
      #  DB1    &  SIG           &  GATE;       /** SIGNATURE SHIFT       **/


DB3.D = !DB3 &  BCC3 &  CTR                     /** OFF TO ON  TOGGLE TERM **/
      #  DB3 & !BCC3 &  CTR                     /** ON  TO OFF TOGGLE TERM **/
      #  DB3                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB3    &  HLD                          /** HOLD FREEZE           **/
      #  DB2    &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
      #  DB2    &  SIG           &  GATE;       /** SIGNATURE SHIFT       **/


DB4.D = !DB4 &  BCC4 &  CTR                     /** OFF TO ON  TOGGLE TERM **/
      #  DB4 & !BCC4 &  CTR                     /** ON  TO OFF TOGGLE TERM **/
      #  DB4                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB4    &  HLD                          /** HOLD FREEZE           **/
      #  DB3    &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
      #  DB3    &  SIG           &  GATE;       /** SIGNATURE SHIFT       **/


DB5.D = !DB5 &  BCC5 &  CTR                     /** OFF TO ON  TOGGLE TERM **/
      #  DB5 & !BCC5 &  CTR                     /** ON  TO OFF TOGGLE TERM **/
      #  DB5                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB5    &  HLD                          /** HOLD FREEZE           **/
      #  DB4    &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
      #  DB4    &  SIG           &  GATE;       /** SIGNATURE SHIFT       **/


DB6.D = !DB6 &  BCC6 &  CTR                     /** OFF TO ON  TOGGLE TERM **/
      #  DB6 & !BCC6 &  CTR                     /** ON  TO OFF TOGGLE TERM **/
      #  DB6                     & !GATE        /** ANYMODE FREEZE        **/
      #  DB6    &  HLD                          /** HOLD FREEZE           **/
      #  DB5    &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
```

```
     #  DB5    &  SIG          &  GATE;       /** SIGNATURE SHIFT       **/


DB7.D = !DB7 &  BCC7 &  CTR                   /** OFF TO ON  TOGGLE TERM **/
     #  DB7 & !BCC7 &  CTR                   /** ON  TO OFF TOGGLE TERM **/
     #  DB7                 & !GATE          /** ANYMODE FREEZE         **/
     #  DB7    &  HLD                        /** HOLD FREEZE            **/
     #  DB6    &  SHF       &  GATE          /** SERIAL SHIFT INPUT     **/
     #  DB6    &  SIG       &  GATE;         /** SIGNATURE SHIFT        **/


BCCOUT    =  BCC8 & CTR          /** COUNTER LOOK AHEAD CARRY**/
          #  DB7 & SHF                  /** SERIAL SHIFT OUTPUT     **/
          #  DB7 & SIG;                 /** SIGNATURE SHIFT OUTPUT  **/
BCCOUT.OE = ON;                           /** OUTPUT ALWAYS ACTIVE    **/
```

CTRSIG3H PAL Program Listing

CTRSIG3H.PLD 5949 10-25-91 10:32am

```
               PARTNO    CTRSIG3H;
               NAME      CTRSIG3H;
               DATE      8/24/91 ;
               REV       ;
               DESIGNER  ROBERT M. BASS;
               COMPANY   TEXAS INSTRUMENTS;
               ASSEMBLY  ;
               LOCATION  TEST ENGINEERING;

/*************************************************************************/
/*                                                                       */
/*                                                                       */
/*************************************************************************/
/*  Allowable Target Device Types: ADV. u DEV. (AMD) - 22V10             */
/*************************************************************************/

/**  Inputs  **/

PIN 1        = CLK       ;      /*                */
PIN 2        = M0        ;      /*                */
PIN 3        = M1        ;      /*                */
PIN 4        = M2        ;      /*                */
PIN 5        = BCCIN     ;      /*                */
PIN 6        = XORIN     ;      /*                */
PIN 7        = nc4       ;      /*                */
PIN 8        = SETCLR    ;      /*                */
PIN 9        = nc2       ;      /*                */
PIN 10       = nc1       ;      /*                */
PIN 11       = !DEVRD    ;      /*                */
PIN 12       = GND       ;      /*                */
PIN 13       = !GATE     ;      /*                */

/**  OUTPUT  **/

PIN 14       = DB0       ;      /*          */
PIN 15       = DB1       ;      /*          */
PIN 16       = BCCOUT    ;      /*          */
PIN 17       = DB4       ;      /*          */
PIN 18       = DB6       ;      /*          */
PIN 19       = DB7       ;      /*          */
PIN 20       = DB5       ;      /*          */
PIN 21       = DB3       ;      /*          */
PIN 22       = DB2       ;      /*          */
PIN 23       = XOROUT    ;      /*          */
PIN 24       = VCC       ;

$DEFINE ON  'b'1
```

```
$DEFINE OFF 'b'0
$DEFINE BCC0 (!DB0)
$DEFINE BCC1 (DB0)
$DEFINE BCC2 (DB1 & DB0)
$DEFINE BCC3 (DB2 & DB1 & DB0)
$DEFINE BCC4 (DB3 & DB2 & DB1 & DB0)
$DEFINE BCC5 (DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC6 (DB5 & DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC7 (DB6 & DB5 & DB4 & DB3 & DB2 & DB1 & DB0)
$DEFINE BCC8 (DB7 & DB6 & DB5 & DB4 & DB3 & DB2 & DB1 & DB0)

/** MODES CURRENTLY UNRELATED **/
$DEFINE HLD    (!M2 & !M1 & !M0) /** HOLD REGISTER RESULT   **/
$DEFINE CTR    (!M2 & !M1 &  M0) /** ACCUMULATE COUNT        **/

/** MODES RELATED TO LOGICAL SHIFTING **/
$DEFINE SIG    (!M2 &  M1 & !M0) /** ACCUMULATE SIGNATURE   **/
$DEFINE SHF    (!M2 &  M1 &  M0) /** SHIFT PIPELINE **/

/** MODES RELATED TO FREEZE LAST SETTING **/
$DEFINE NOTUSED0 (M2 & !M1 & !M0) /**  **/
$DEFINE NOTUSED1 (M2 & !M1 &  M0) /**  **/
$DEFINE NOTUSED2 (M2 &  M1 & !M0) /**  **/
$DEFINE NOTUSED3 (M2 &  M1 &  M0) /**  **/


/** Logic Equations **/
LSF = (SIG # SHF);                          /** MODES THAT LOGICAL SHFT **/


/** IC POSITION DEVELOPMENT CODE FOR PRODUCING EACH PAL **/
HIGH =  ON;                          /** HIGH REGISTER B16..B19 **/


 [DB0..7].AR = CTR & SETCLR;                /** SET COUNTER TO 00     **/
 [DB0..7].SP = SIG & SETCLR;                /** SET COUNTER TO FF     **/

/** GENERAL FORMULA USED FOR CRC16 GENERATION **/
/** NOTE:  CRC16  G(X)= X^16 + X^12 + X^5 + 1      **/
/**              G(X)= !(X^16 + X^12 + X^5)      **/
/** HIGH-  REPRESENTS MOST SIG. DEVICE IN COUNTER CHAIN **/

XOROUT.OE = ON;                                  /** OUTPUT ALWAYS ACTIVE  **/
XOROUT =  HIGH;                                  /** NO FUNCTION ASSIGNED  **/

[DB0..DB7].OE = DEVRD;                        /** uP READ COUNTER REGISTER **/

 DB0.D =  BCC0  &  CTR &  BCCIN  &  GATE /** ON/OFF/ON TOGGLE TERM **/
       #  DB0                    & !GATE        /** ANYMODE FREEZE        **/
       #  DB0   &  HLD                         /** HOLD FREEZE           **/
       #  BCCIN &  SHF           &  GATE        /** SERIAL SHIFT INPUT    **/
       #  BCCIN &  SIG &  HIGH   &  GATE;       /**SIG TERM HIGH DEVICE   **/
```

```
DB1.D = !DB1 &  BCC1 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB1 & !BCC1 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB1                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB1   &  HLD                         /** HOLD FREEZE            **/
      #  DB0   &  SHF           &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB0   &  SIG           &  GATE;       /** SIGNATURE SHIFT        **/

DB2.D = !DB2 &  BCC2 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB2 & !BCC2 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB2                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB2   &  HLD                         /** HOLD FREEZE            **/
      #  DB1   &  SHF           &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB1   &  SIG           &  GATE;       /** SIGNATURE SHIFT        **/

DB3.D = !DB3 &  BCC3 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB3 & !BCC3 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB3                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB3   &  HLD                         /** HOLD FREEZE            **/
      #  DB2   &  SHF           &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB2   &  SIG           &  GATE;       /** SIGNATURE SHIFT        **/

DB4.D = !DB4 &  BCC4 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB4 & !BCC4 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB4                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB4   &  HLD                         /** HOLD FREEZE            **/
      #  DB3   &  SHF           &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB3   &  SIG           &  GATE;       /** SIGNATURE SHIFT        **/

DB5.D = !DB5 &  BCC5 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB5 & !BCC5 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB5                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB5   &  HLD                         /** HOLD FREEZE            **/
      #  DB4   &  SHF           &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB4   &  SIG           &  GATE;       /** SIGNATURE SHIFT        **/

DB6.D = !DB6 &  BCC6 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB6 & !BCC6 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB6                    & !GATE        /** ANYMODE FREEZE         **/
      #  DB6   &  HLD                         /** HOLD FREEZE            **/
      #  DB5   &  SHF           &  GATE        /** SERIAL SHIFT INPUT     **/
      #  DB5   &  SIG           &  GATE;       /** SIGNATURE SHIFT        **/

DB7.D = !DB7 &  BCC7 &  CTR                    /** OFF TO ON  TOGGLE TERM **/
      #  DB7 & !BCC7 &  CTR                    /** ON  TO OFF TOGGLE TERM **/
      #  DB7                    & !GATE        /** ANYMODE FREEZE         **/
```

```
     # DB7   & HLD                          /** HOLD FREEZE          **/
     # DB6   & SHF          & GATE          /** SERIAL SHIFT INPUT   **/
     # DB6   & SIG          & GATE;         /** SIGNATURE SHIFT      **/


BCCOUT    = BCC8 & CTR        /** COUNTER LOOK AHEAD CARRY**/
          # DB7 & SHF                  /** SERIAL SHIFT OUTPUT     **/
          # DB7 & SIG;                 /** SIGNATURE SHIFT OUTPUT  **/
BCCOUT.OE = ON;                         /** OUTPUT ALWAYS ACTIVE    **/
```

CAD32-24 PAL Program Listing

CAD32-24.PLD 2036 04-18-88 03:28pm

```
               PARTNO    ;
               NAME      16L8 ;
               DATE      4/18/88 ;
               REV       ;
               DESIGNER  ROBERT M. BASS;
               COMPANY   TEXAS INSTRUMENTS;
               ASSEMBLY  ;
               LOCATION  TEST ENGINEERING;

/*************************************************************************/
/*                                                                       */
/*       CADEC MODIFIED:WOW MICROCONTROLLER ADDRESS DECODER              */
/*          ENLARGED EPROM MAP FROM 16Kx8 TO 24Kx8                       */
/*            EPROM NOW MAPPS UP TO HARDWARE SPACE                       */
/*                                                                       */
/*************************************************************************/
/*  Allowable Target Device Types: ADV. u DEV. (AMD) - 16L8               */
/*************************************************************************/

/**  Inputs  **/

PIN 1        = A8         ;      /* 8052 ADDRESS A15 */
PIN 2        = A9         ;      /* 8052 ADDRESS A14 */
PIN 3        = A10        ;      /* 8052 ADDRESS A13 */
PIN 4        = A11        ;      /* 8052 ADDRESS A12 */
PIN 5        = A12        ;      /* 8052 ADDRESS A11 */
PIN 6        = A13        ;      /* 8052 ADDRESS A10 */
PIN 7        = A14        ;      /* 8052 ADDRESS A9  */
PIN 8        = A15        ;      /* 8052 ADDRESS A8  */
PIN 9        = NC1        ;
PIN 11       = NC2        ;

/**  OUTPUT  **/

PIN 12       = !TOPAGE    ;      /* F000-FFFF */
PIN 13       = !SF        ;      /* E800-EFFF */
PIN 14       = !DISCRETE  ;      /* E400-E5FF */
PIN 15       = !ANALOG    ;      /* E200-E3FF */
PIN 16       = !IOBUSEN   ;      /* E200-FFFF */
PIN 17       = !INTERVAL  ;      /* E000-E1FF */
PIN 18       = !EPROM     ;     /* 8000-DFFF ENLARGED */
PIN 19       = !RAM       ;     /* 0000-7FFF */

/**  Logic Equations  **/


 RAM      = !A15 ;
```

```
EPROM    = A15 & !A14  #  A15 & A14 & !A13 ;

INTERVAL = A15 & A14 & A13 & !A12 & !A11 & !A10 & !A9 ;

ANALOG   = A15 & A14 & A13 & !A12 & !A11 & !A10 & A9  ;

IOBUSEN  = A15 & A14 & A13 &  A12 #
           A15 & A14 & A13 & !A12 & !A11 & !A10 & A9  #
           A15 & A14 & A13 & !A12 & !A11 & A10 #
           A15 & A14 & A13 & !A12 & A11;

DISCRETE = A15 & A14 & A13 & !A12 & !A11 & A10 & !A9  ;

SF       = A15 & A14 & A13 & !A12 & A11 ;

TOPAGE   = A15 & A14 & A13 & A12;
```

CAD32-24 PAL Simulation Output

CAD32-24.SO

```
  1:
  2:          PARTNO    ;
  3:          NAME      16L8  ;
  4:              DATE      4/18/88   ;
  5:          REV       ;
  6:          DESIGNER  ROBERT M. BASS;
  7:          COMPANY   TEXAS INSTRUMENTS;
  8:          ASSEMBLY  ;
  9:          LOCATION  TEST ENGINEERING;
 10:
 11:/*************************************************************************/
 12:/*                                                                       */
 13:/*        CADEC MODIFIED: WOW MICROCOMPUTER ADDRESS DECODER              */
 14:/*              ENLARGED EPROM MAP FROM 16Kx8 TO 24Kx8                   */
 15:/*              EPROM NOW MAPS UP TO HARDWARE SPACE                      */
 16:/*************************************************************************/
 17:/*  Allowable Target Device Types: ADV. u DEV. (AMD) - 16L8              */
 18:/*************************************************************************/
 19:
 20:ORDER:
 21:   %12,A15, %2 ,A14, %2 ,A13, %2 ,A12, %2 ,A11, %2 ,
 22:   A10, %2 ,A9 , %2 ,A8, %4 ,!RAM, %2 ,!EPROM, %2 ,
 23:   !INTERVAL, %2 ,!IOBUSEN, %2 ,!ANALOG, %2 ,!DISCRETE, %2 ,
 24:   !SF,%2 ,!TOPAGE;
 25:
 26:
```

==========

Simulation Results

==========

```
     A  A  A  A  A  A  A  A    R  E  I  I  A  D  S  T
     1  1  1  1  1  1           A  P  N  O  N  I  F  O
     5  4  3  2  1  0  9  8    M  R  T  B  A  S     P
                                 O  E  U  L  C     A
                                 M  R  S  O  R     G
                                    V  E  G  E     E
                                    A  N     T
     ------------------------------L--------E-------

0001:          0  0  0  0  0  0  0  X    L  H  H  H  H  H  H  H
0002:          0  0  0  1  0  0  0  X    L  H  H  H  H  H  H  H
0003:          0  0  1  0  0  0  0  X    L  H  H  H  H  H  H  H
0004:          0  1  1  1  1  1  1  X    L  H  H  H  H  H  H  H

0005:          1  0  0  0  0  0  0  X    H  L  H  H  H  H  H  H
```

0006:	1	0	0	1	0	0	0	X	H	L	H	H	H	H	H	H
0007:	1	0	1	0	0	0	0	X	H	L	H	H	H	H	H	H
0008:	1	0	1	1	1	1	1	X	H	L	H	H	H	H	H	H
0009:	1	1	0	0	0	0	0	X	H	L	H	H	H	H	H	H
0010:	1	1	0	1	1	1	1	X	H	L	H	H	H	H	H	H
0011:	1	1	1	0	0	0	0	X	H	H	L	H	H	H	H	H
0012:	1	1	1	0	0	0	1	X	H	H	H	L	L	H	H	H
0013:	1	1	1	0	0	1	0	X	H	H	H	L	H	L	H	H
0014:	1	1	1	0	0	1	1	X	H	H	H	L	H	H	H	H
0015:	1	1	1	0	1	0	0	X	H	H	H	L	H	H	L	H
0016:	1	1	1	0	1	0	1	X	H	H	H	L	H	H	L	H
0017:	1	1	1	0	1	1	0	X	H	H	H	L	H	H	L	H
0018:	1	1	1	0	1	1	1	X	H	H	H	L	H	H	L	H
0019:	1	1	1	1	0	0	0	X	H	H	H	L	H	H	H	L
0020:	1	1	1	1	1	0	0	X	H	H	H	L	H	H	H	L
0021:	1	1	1	1	1	1	1	X	H	H	H	L	H	H	H	L

VITA

Robert Michael Bass was born in Knoxville, Tennessee on May 5, 1957. He attended elementary school in the Knoxville School District and then graduated from Central High School in June, 1975. He entered Walter State Community College in September, 1975 in the Manufacturing Technology Program. On completion of this program he transferred to Knoxville State Technical Institute and received an Associate of Science in Electronic Technology in June, 1979. Upon graduation, he accepted a position at Texas Instruments for two years as an Electronic Technician. In June, 1981 he took a leave of absence to enter the University of Tennessee, Knoxville and later received the Bachelor of Science in Electrical Engineering in June, 1984. He resumed previous his employment at Texas Instruments in the product engineering department and later moved to the test engineering group in 1986. In September, 1987 he reentered the University of Tennessee, Knoxville and in June, 1991 received a Master of Science degree in Electrical Engineering.

Michael is currently employed as a test engineer with Siemens Industrial Automation Inc. and was able to apply his experience toward completion of this project.