Accession No. N81-23889

NASA Contractor Report 164286

FEASIBILITY STUDY OF A MICROPROCESSOR-BASED OCULOMETER SYSTEM

Murali R. Varanasi

OLD DOMINION UNIVERSITY
Norfolk, Virginia 23508

Grant NSG-1379
January 1981

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

DEPARTMENT OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA

FEASIBILITY STUDY OF A MICROPROCESSOR-BASED OCULOMETER SYSTEM

By

Murali R. Varanasi, Principal Investigator

Final Report
For the period January 1, 1977 - December 31, 1980

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Research Grant NSG 1379
Patrick A. Gainer, Technical Monitor
Flight Dynamics and Control Division

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508

January 1981

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES - CONCLUDED

LIST OF FIGURES

FEASIBILITY STUDY OF A MICROPROCESSOR-BASED OCULOMETER SYSTEM

By

Murali R. Varanasi*

BACKGROUND

Several vision movement recording instruments are in existence today. A survey of all these instruments is provided by Young and Sheena (ref. 1). These include Honeywell's remote oculometer (ref. 2), Department of Transportation's remote oculometer, EG and G/Human Engineering Laboratory facility (ref. 3), University of Alberta's remote oculometer (ref. 4), the Whittaker Corporation eye view monitor (1973), and a TV pupilometer system developed by Gulf and Western Applied sciences laboratory.

The first of these, Honeywell's remote oculometer, is primarily used by the National Aeronautics and Space Administration/Langley Research Center (NASA/LaRC) for conducting studies in flight management. The instrument is configured around a minicomputer as a signal processor, collects information using a TV camera, and has provisions for headtracking. Its design and construction are aimed at using it in a laboratory environment and have not utilized the space and weight savings offered by the large-scale integrated circuit technology.

Old Dominion University has undertaken a feasibility study of a microprocessor-based oculometer system. The primary emphasis in the study centered upon real-time processing of oculometer data in the most efficient manner and bringing about a system design that was portable in size and flexible in use. A secondary design consideration was to eliminate redundancy in data so that processing speed could be maximized and storage requirements minimized. The results of this investigation are reported here, and recommendations for a future system are included.

---

*Formerly Associate Professor, Department of Electrical Engineering, Old Dominion University, Norfolk, Virginia 23508, currently employed by Department of Computer Science and Engineering, University of South Florida, Tampa, Florida 33620.

SCOPE

Introduction

The research undertaken in the grant was aimed at defining strategies to design a future flight-worthy oculometer system. Specifically, the investigation was directed at an appropriate architectural design of the signal processor, improved optics, and reduction of size, weight and power of the system. A strategy of design is given in Figure 1 in a flow chart form as an aid to understanding. This was also presented to the flight management researchers at NASA/LaRC in August 1977. Subsequent to the presentation, several meetings with various researchers were held to define the features for a future system. Based on the suggestions of all the researchers, a list of essential features for the oculometer was integrated into the research and development effort as goals to pursue in the research. For completeness sake, these are listed below.

Goals

For this research the following aspects were considered highly desirable:

1. Improved optical subsystem,
2. Systematic design of the interface electronics,
3. Investigation of architectural variations for efficient processing of data,
4. Study of possible hardware-software tradeoffs,
5. Choice of control and processing elements that reflect state of the art, and
6. Elimination of redundant data.

Certain implicit features considered were:

a. Higher resolution,

b. Reduction of computational complexity,

c. Elimination of computational bottlenecks, and

d. Incorporation of testability into the system.

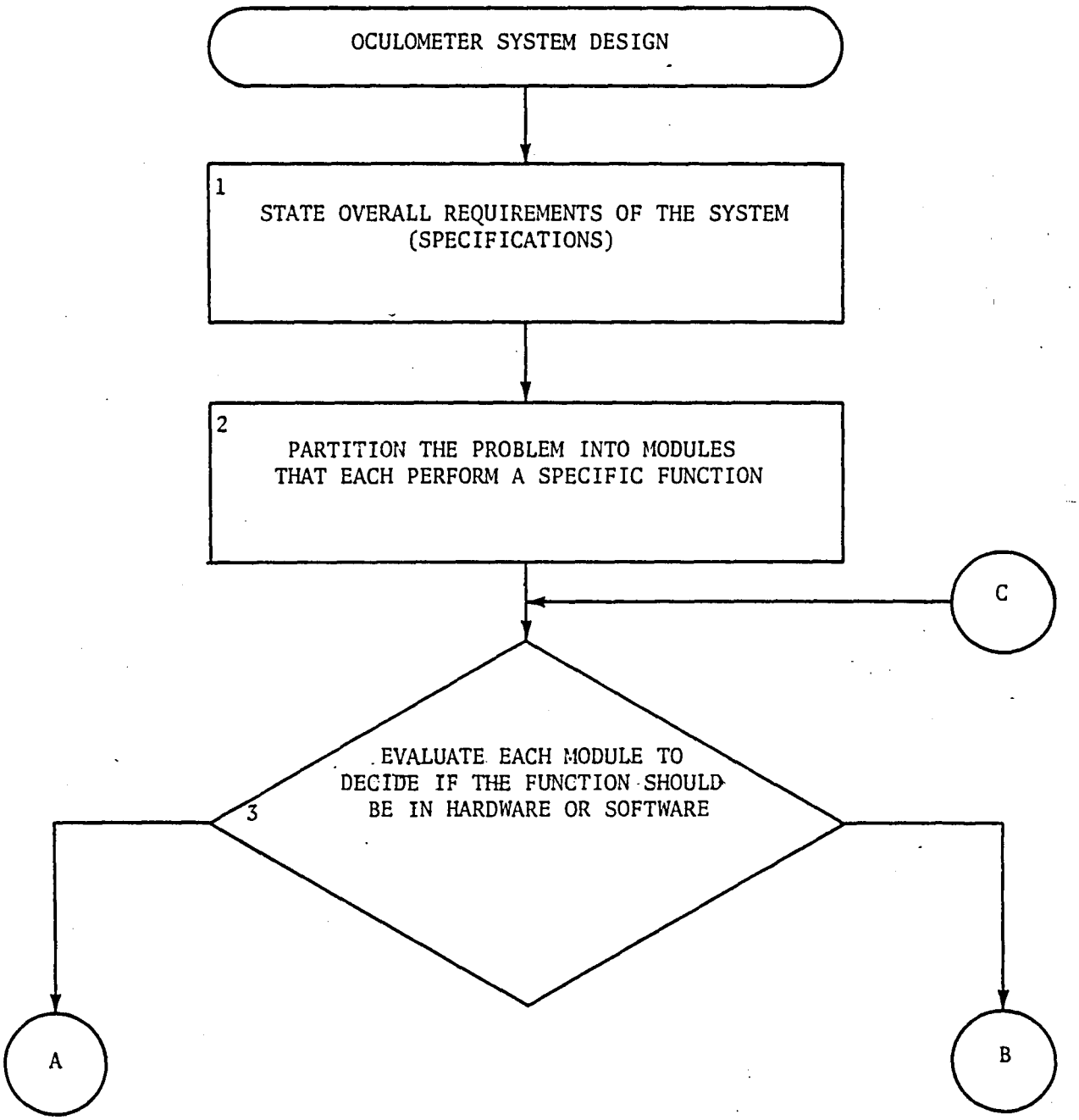


Figure 1. Flow chart of design strategy.

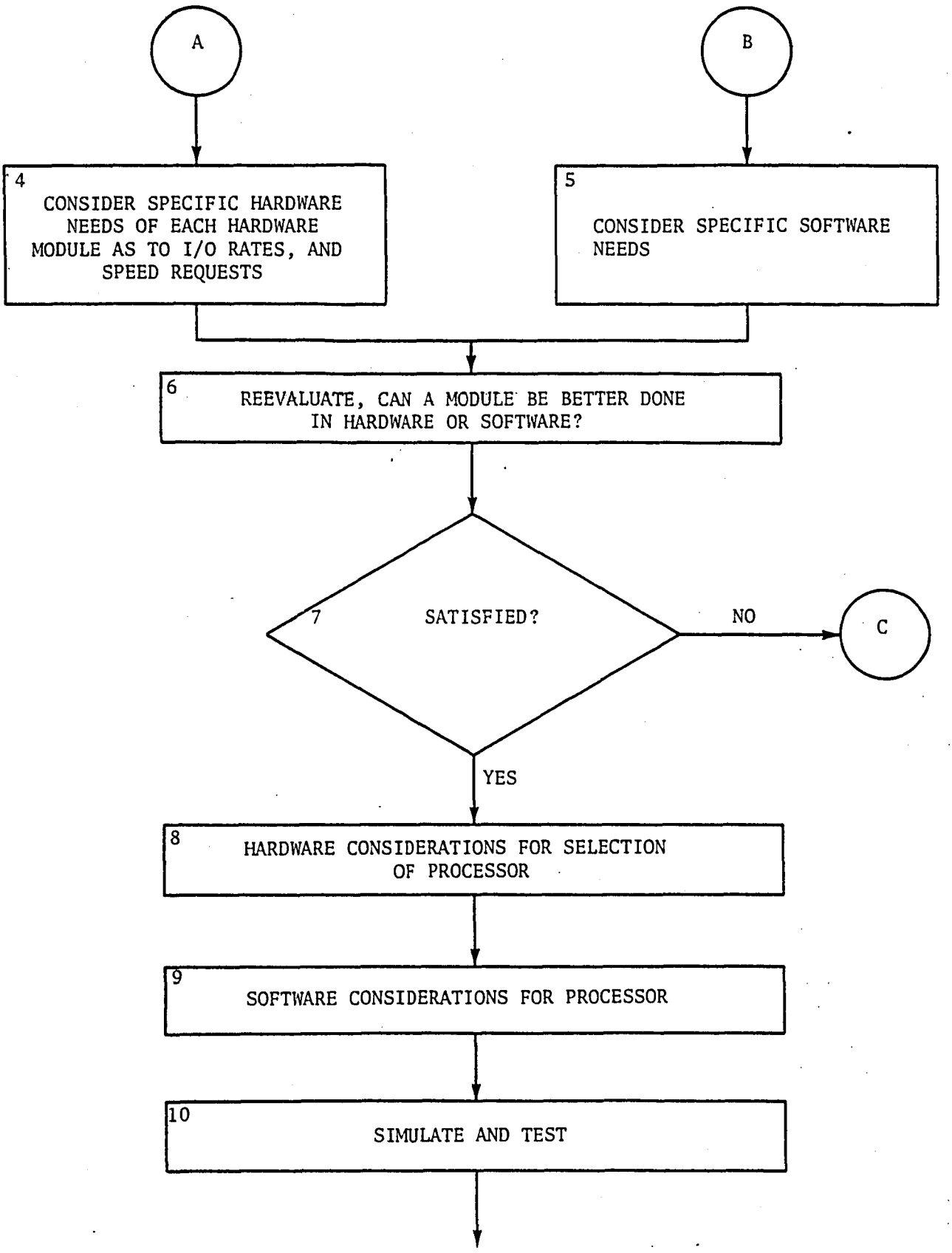


FIGURE 1. (Continued).

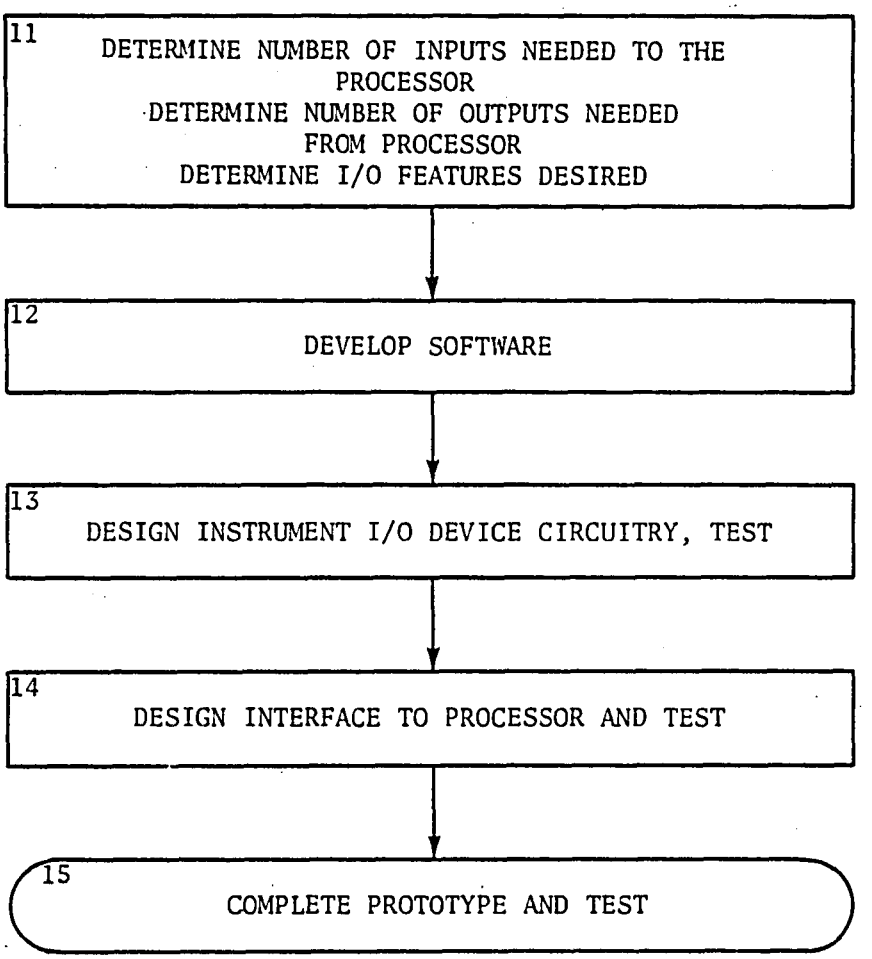


Figure 1. (Concluded).

Throughout this investigation it was assumed that autofocusing, head-tracking, and mirror search systems were external to the signal processor. It was further assumed that the oculometer system will be used in many operating environments with different instrument panel configurations, and therefore the computational aspects included calculation of fixation point with respect to a two-dimensional reference plane. Consequently, it was necessary to augment the basic program with additional software features to match the exact planar configuration from one experiment to another. Therefore, throughout the effort considerable attention was devoted to designing hardware and software subsystems with reasonable flexibility and modularity.

System Configuration

The important subsystems of the oculometer are shown in Figure 2 and include the electro-optical subsystem, the synchronization subsystem, the high-speed algorithmic processor, the digital interface and the software subsystems coordinated by an INTEL 8086 microprocessor. Some of the important components, e.g. the high-speed algorithmic processor, the simulator for developing microroutines for the high-speed algorithmic processor, and the Karhunen-Loève Transform technique for data compression are briefly discussed within this report. Complete discussions of the simulator and Karhunen-Loève transfer are presented separately in Appendixes A and B, respectively. Specific subsystems along with the design considerations are discussed in the next section, "System Design," and recommendations for future research are reported under "Summary and Recommendations."

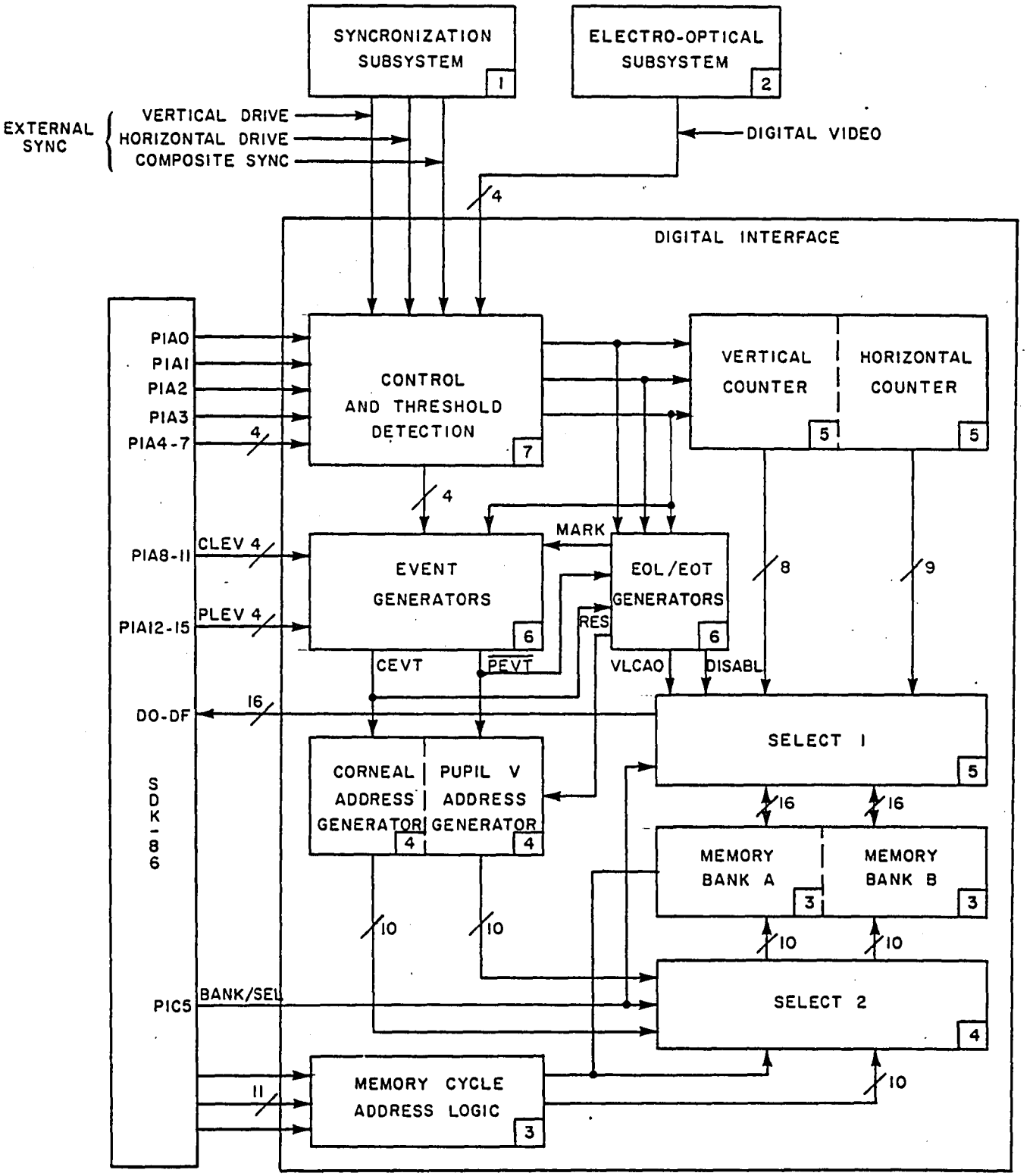


Figure 2. Digital interface block diagram.

SYSTEM DESIGN

Introduction

The primary emphasis in the design was placed on efficient processing of pupil and corneal data in the most expedient manner so that the system can be used in real time. In anticipation of the system's being evaluated for functional completeness, minimal effort was made to eliminate data for reasons of statistical significance. The signal processor functions using data generated in a direct manner except for threshold detection of corneal and pupil events. To minimize the impact of design modifications during the development cycle and to ease the field maintenance of the completed prototype, the system was partitioned into functional hardware and software modules. To facilitate this, the hardware design was carried out utilizing state-of-the-art components resulting in minimization of system complexity. As each subsystem was completed, it was extensively tested for its correct operation as well as for its compatibility with other subsystems. Based on the experience gained during prototype development and evaluation, techniques for significant performance enhancements are summarized as recommendations for further refinement of the system.

The method of sensing eye movement is in principle identical to that used in the Honeywell MARK III oculometer system. The relative displacement of the center of the corneal reflection from the pupil center is assumed to be unchanged as a result of lateral head movements and changes with eye rotation only. The measurement is based on the principle that the displacement of corneal reflection from the center of the pupil is a function of the angular direction of the eye (and is independent of the position of the eye). An EL-12B lamp (with Wratten 87A filter) is used as an infrared light source and a DAGE 650 silicon diode television camera with a telephoto lens is used for sensing the pupil and corneal reflections. The intensity of the light is chosen to provide a safe radiation level at all times.

The remainder of this section is devoted to a description of the hardware and software elements that comprise the oculometer developed during the research effort. Descriptions of the functional subsystems comprising the oculometer are included and serve as a basis for understanding

hardware-software interdependencies of the system. For clarity, this section is divided into subsections as follows:

System Processor (SPB)—an overview of the SDK-86 system processor;

Synchronization Subsystem (SS)—the circuitry comprising the clock and EIA RS-160 synchronization generators;

Electro-optical Subsystem (EOS)—the features and alignment procedures for the camera and A/D signal conditioning;

Digital Interface (DI) Subsystem—dual banks of high-speed memory, direct memory access (DMA) controller, and event detection;

High-Speed Arithmetic Processor (HSAP)—special purpose hardware specifically designed for high-speed computation of transcendental functions.

Software—the software necessary to collect a field of data and generate gaze vector information.

Information provided in this report requires a knowledge of TTL and LSI integrated circuits as well as INTEL's microprocessor programming language, PL/M-86. This information may be found in the following publications:

The TTL Data Book for Design Engineers, Texas Instruments Incorporated, Dallas, TX;

SDK-86 (MCS-86) System Design Kit User's Guide, INTEL Corporation, Santa Clara, CA;

PL/M-86 Programming Manual, INTEL Corporation, Santa Clara, CA; and

SDK-86 (MCS-86) System Design Kit Monitor Listings, INTEL Corporation, Santa Clara, CA.

System Processor

The system processor board (SPB) is an INTEL SDK-86 system development kit. The SPB is a complete microcomputer system featuring an 8086 microprocessor, 48 lines of parallel I/O, and a serial communications channel. Specifications of the SPB are included in Table 1. Descriptions of the SDK-86 may be found in SDK-86 (MCS-86) System Design Kit User's Guide and SDK-86 (MCS-86) System Design Kit Monitor Listings; therefore, only details

Table 1. SDK-86 specifications.

Processor:	8086
Clock Frequency:	5 MHz
RAM:	4K bytes 2142
ROM:	4K bytes 2616 with sockets for additional 4K bytes
Memory Address Space:	0-FFFFF_H
I/O Address Space:	0-FFFF_H
Serial I/O:	1 channel, RS-232 or current loop, 110-4800 baud
Parallel I/O:	48 programmable I/O
Interrupts:	Not used
Power Requirements:	5 V at 3.5 amp -12 V at 0.3 amp

necessary for understanding processor interaction with other system components are included here.

Serial I/O channel 1 includes an 8251A programmable USART as well as several MSI components forming a baud-rate generator. The USART may be programmed to support several word formats, parity options, and external clock rates. Two jumper matrices on the SDK-86 allow selection of baud-rates from 110 to 4800 baud and either EIA RS-232C or current-loop protocols.

Control and status information is handled by the parallel I/O subsystem through 3 16-bit ports designated PORT$A, PORT$B, and PORT$C. The 16-bit control port (PORT$A) format is shown in Figure 3. Bit 0, the debug flag, determines the source of the external control and synchronization signals to the digital interface. When the debug flag is set, external synchronization signals must be provided under software control by bits 1 to 3 of the control word. With appropriate debug routines and a logic state analyzer, hardware within the DI may be checked to the chip level. When the debug flag is reset, bits 1 to 3 are nonfunctional and external synchronization signals must be provided. Bits 4 to 7 provide amplitude information to the DI when in debug mode. When debug is reset, these bits control a gain stage in the E/O subsystem. Bits 8 to 15 set the levels applied to the pupil and corneal comparators.

Status information is returned to the system processor through a 16-bit port designated "PORT$B." In the current version of the oculometer, only one status flag is used to monitor the vertical synchronization pulse. Data is latched into the status port latches on the positive transition of a pulse applied to bits 2 and 10 of PORT$C. One additional control signal, the bank select flag, is located at bit 5 of PORT$C.

Fifteen status bits of PORT$B and three control bits of PORT$C have been allocated for expanding the capabilities of future versions. Parallel I/O is implemented with two 8255A programmable peripheral interfaces. These LSI chips must be initialized during system startup by sending a command $0A6A6_H$ to the control port located at address $0FFFE_H$; I/O port allocations are given in Table 2.

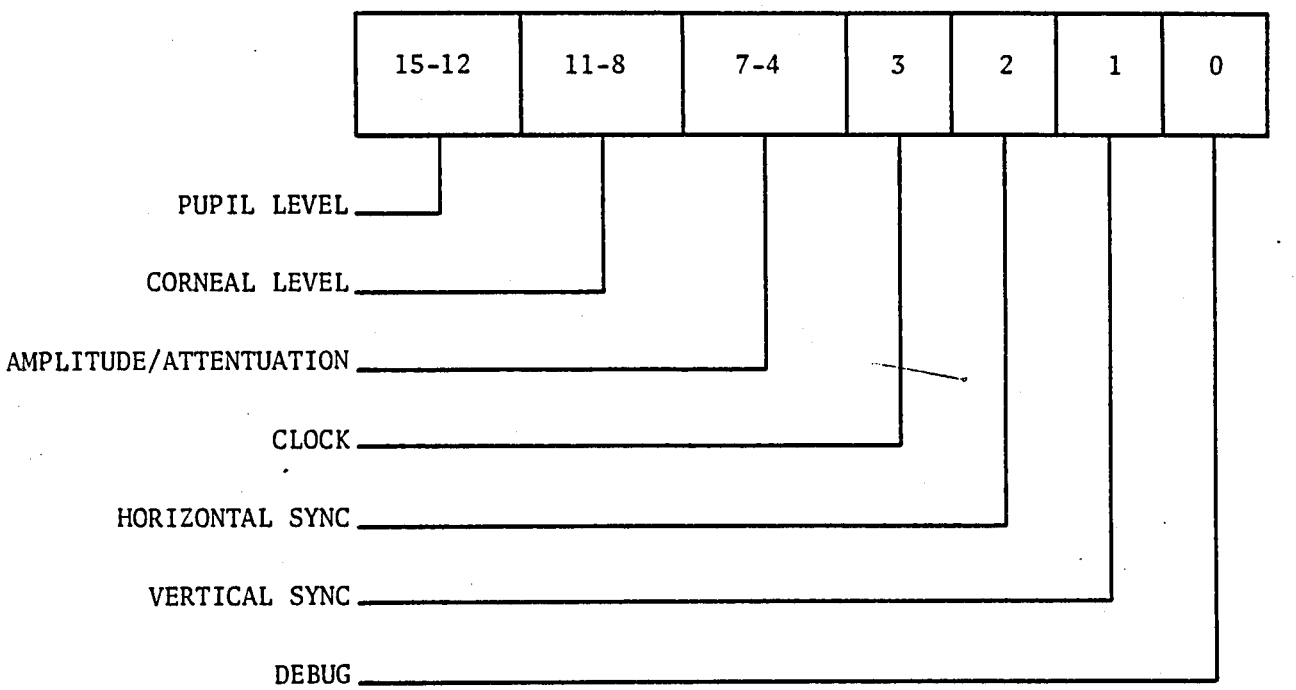


Figure 3. PORT$A format.

Table 2. I/O port allocations.

PORT ADDRESS*	PORT FUNCTION
0000 to FFE7	Not used
FFE8, FEEA	On board keyboard and display (not used at this time)
FFE9, FFEB, FFED, FFEF	Reserved
FFF0	Read/write serial data
FFF1	Reserved
FFF2	Read/serial status/write serial command
FFF3 to FFF7	Reserved
FFF8	Read/write LO(PORT$A)
FFF9	Read/write HI(PORT$A)
FFFA	Read/write LO(PORT$B)
FFFB	Read/write HI(PORT$B)
FFFC	Read/write LO(PORT$C)
FFFD	Read/write HI(PORT$C)
FFFE	Write LO(CTL$PORT)
FFFF	Write HI(CTL$PORT)

*All addresses in hexidecimal representation.

Synchronization Subsystem

The function of the clock and synchronizing circuit is to provide a single phase, 10-MHz system clock and also to provide the necessary synchronizing signals for control of the video drive circuitry. Video timing is also provided to the 8086-based microcomputer.

A 20-MHz square wave is generated by a crystal-controlled oscillator built around 3 inverting gates with positive feedback. The signal is divided by an offset modulo ten counter to produce 10-MHz and 2-MHz signals. The counter is composed of a 74S169 synchronous counter and a 74S00 two-input NAND gate used for decoding. The 10-MHz signal is inverted to provide a system clock, and the 2-MHz signal drives a 3262B TV synchronizing generator, which provides horizontal and vertical drive signals, composite synchronization, and composite blanking signals. The horizontal and vertical signals drive 4N25 opto-isolators which isolate digital and analog grounds to prevent noise pickup. The isolated signals are connected by 75123 line drivers to 75 Ω coaxial cable which is in turn connected to the camera system. Unisolated horizontal and vertical drive signals are also sent to the system processor for timing purposes.

The circuit requires +5 and -12 V power supplies. It is recommended that this circuit be constructed close to other digital circuitry in any future system to minimize noise generation; that is, all digital circuitry other than the microcomputer should be constructed on the same printed circuit board. (See Figure C1, Appendix C, for a diagram of the synchronization subsystem).

Electro-optical Subsystem

The function of the electro-optical subsystem (E/O) is to monitor the test subject and provide a digital representation of the scene. Figure 4 contains a functional block diagram of the subsystem. As illustrated by the figure, considerable signal conditioning is accomplished before the composite video signal is converted to digital form. For detailed schematics please refer to Figure C2.

Since the incoming video signal varies greatly from subject to subject, it is necessary to be able to change bias level and gain. This is the function of the first two stages. The bias level may be adjusted via

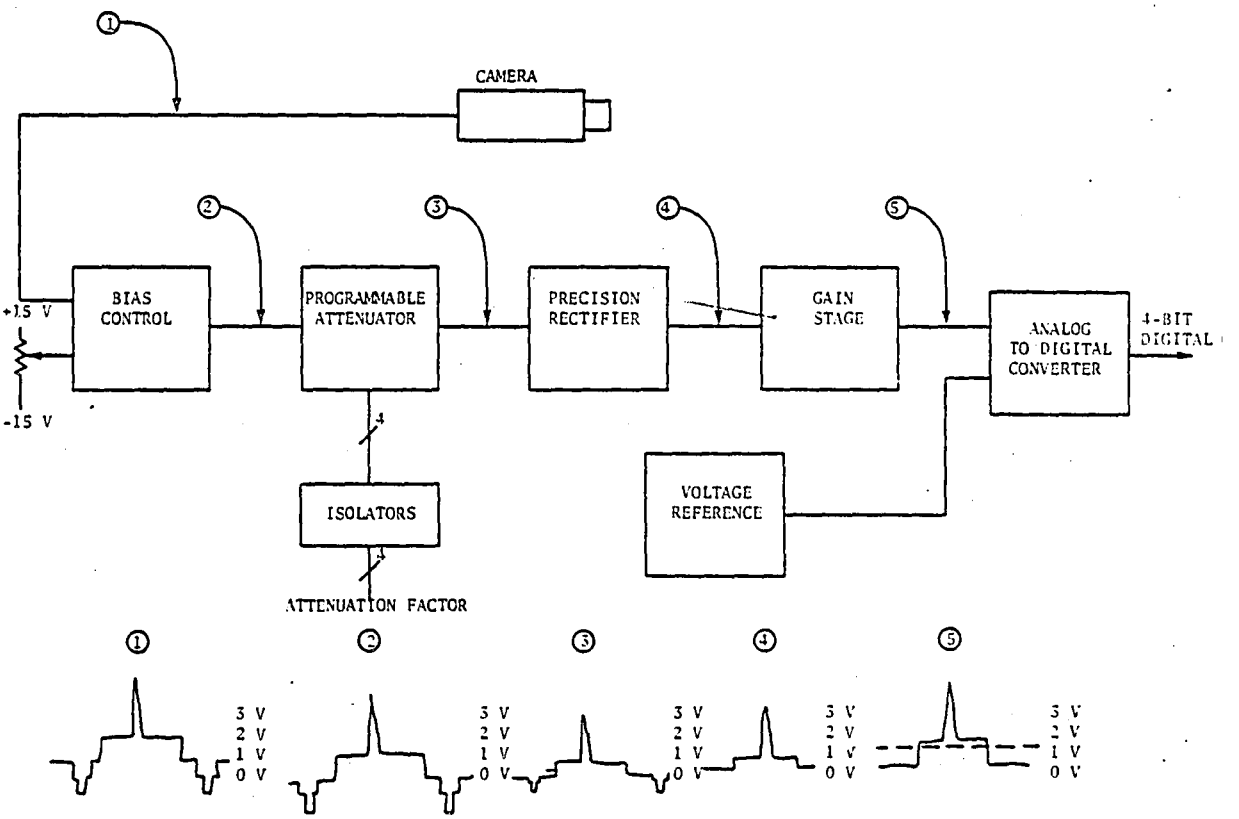


Figure 4. Electro-optical subsystem.

potentiometer used in conjunction with a summing circuit composed of a Harris HA2-2515 operational amplifier (A1).. For future designs this circuit may be changed to allow computer control using a low-resolution digital-to-analog converter isolated by opto-isolators. The next stage functions as a computer-controlled gain stage composed of a four-bit multiplying digital-to-analog converter. The multiplying digital-to-analog converter is implemented by a summing amplifier (A2) with binary weights. The video signal is connected to the four inputs of the summer by four analog switches which are controlled by the computer through four opto-isolators. The isolators help to prevent noise induction into the analog signal by the digital circuitry. This circuit may be replaced by an integrated version if one of sufficient bandwidth is available.

Once the bias and gain are set, the signal is rectified by a high-speed rectifier (A3 and A4) to obtain the negative part. This is necessary for the analog-to-digital converter, and it also functions to eliminate the synchronizing pulses. The analog-to-digital converter (TRW-TDC 1021J) produces a four-bit result and is clocked at 10 MHz by the system clock. The reference voltage consists of a zener diode-potentiometer circuit buffered by a unity gain amplifier (A5). Ground isolation is provided by the analog-to-digital converter.

This circuit should be constructed with careful attention to isolation of digital signals and grounds from analog signals and grounds. This board should be contained by an aluminum box at analog ground to prevent noise pickup. Connections should be made using feedthroughs and coaxial cable.

The camera used with the system is a Dage Model 650; however, any camera with similar characteristics may be used. For more information consult Model 60, 65, and 650 MKII Series Cameras; manual No. 970265-02 available from Dage MTI, Inc.

Digital Interface Subsystem

The hardware within the oculometer extracts contour information from an illuminated scene and places the boundary points into memory for later examination by the system processor. Within the digital interface subsystem,

amplitude information from the analog subsystem is compared against computer-generated thresholds producing a ternary representation of the scene. The signal may be thought of as residing in one of the three mutually exclusive states illustrated in Table 3. Note that other states could be defined (i.e., corneal signal but no pupil), but in this application only the states in Table 3 are considered valid. The pupil and corneal thresholds are generated by the system processor and no error checking is performed by the digital interface; therefore, software checks must be implemented to insure that the corneal threshold is always greater than or equal to the pupil threshold. Each state transition generates an event as tabulated in Table 4.

From Table 4 one should observe that corneal events have priority over pupil events. If during a single clock cycle the state transitions corresponding to lines 3 or 7 occur, only the corneal event will be recorded. Experience has shown that this event is most rare and poses no problem to the processing algorithm. At each detected state transition the x and y coordinate of the illuminated pixel is stored in memory.

A functional block diagram of the digital interface is contained in Figure 2. The corresponding schematic in Appendix C is indicated in the lower left corner of each block. Using timing signals from the synchrozation subsystem and control information from the system processor, the digital interface places coordinate data for each corneal and pupil event into one of two banks of high-speed memory.

Memory banks A and B (refer to Fig. C3) consist of two 1K × 16-bit banks of 80-nsec memory organized such that, while data is being placed in one bank, the contents of the other bank are accessible to the system processor. At any given time only the bank of the memory selected by the $A/\overline{B}$ line is in the address space of the system processor. This organization eliminates cycle stealing associated with many DMA controllers at the expense of marginally increased memory requirements. INTEL 2148 1K × 4-bit memory chips were used in this implementation because of their speed and low standby power dissipation. Each system clock cycle is divided into two subcycles by the memory cycle address logic. As illustrated in Figure 5, the low active chip enable is active during the

Table 3. State table for illuminated eye.

STATE	CONDITIONS
0	No signal: Signal below both pupil and corneal threshold
1	Pupil signal: Signal above pupil threshold but below corneal threshold
2	Corneal Signal: Signal above pupil threshold and above corneal threshold

Table 4. Summary of events and actions taken.

LINE	PRESENT STATE	NEXT STATE	EVENT	COMMENTS
1	0	0	NULL	No action taken
2	0	1	PEVT	Pupil event logged
3	0	2	CEVT	Only corneal event logged No pupil event logged
4	1	0	PEVT	Pupil event logged
5	1	1	NULL	No action taken
6	1	2	CEVT	Corneal event logged
7	2	0	CEVT	Only corneal event logged No pupil event logged
8	2	1	CEVT	Corneal event logged
9	2	2	NULL	No action taken

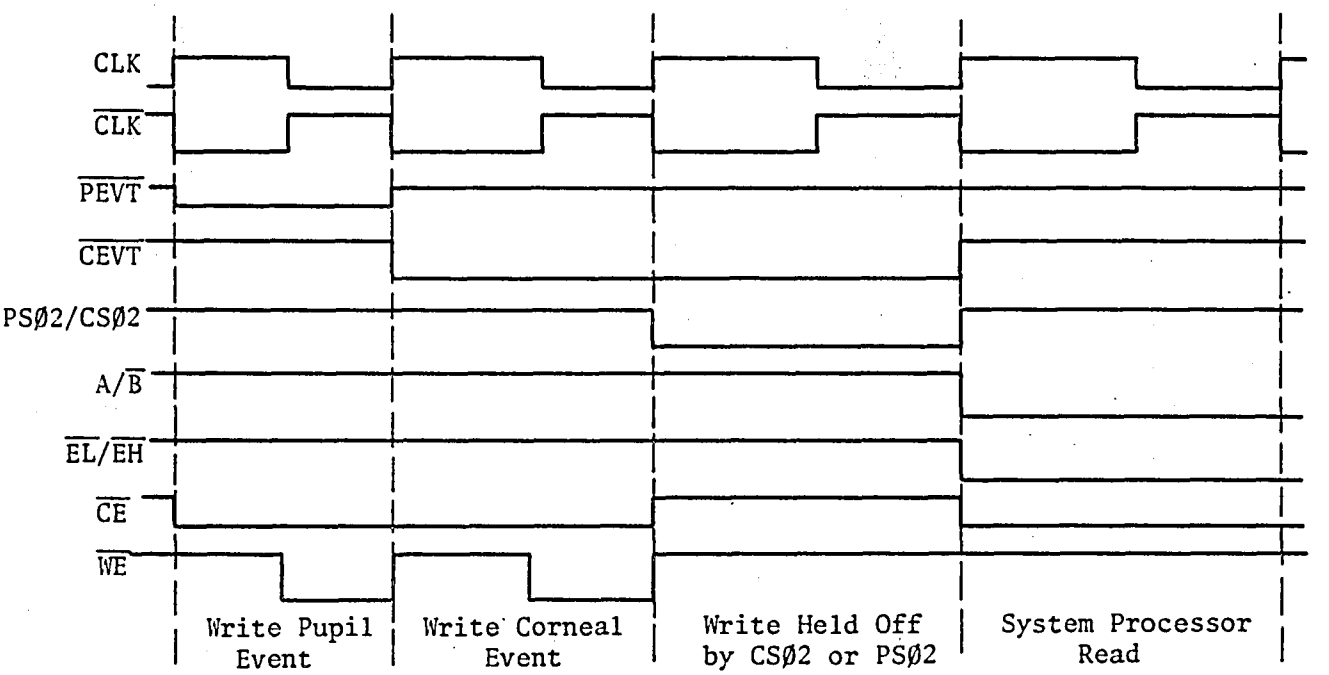


Figure 5. Memory cycle timing diagram.

entire write and read cycle. The write enable signal is active during the second subcycle but may be held off by PSØ2 or CSØ2.

The pupil and cornea address vectors are generated by the Select 2 and DMA address generator circuitry. The pupil and corneal tables are organized as shown in Figure 6. The cornea address generator (CAG) is implemented with 74LS161 binary counters. Before the start of each video frame the counter is preset by the end of file signal (EOF) to OFFFH. The counter is then incremented during each clock cycle that CEVT is high. The pupil address generator (PAG), although similar to the CAG, consists of 74LS169 counters configured to count down each clock cycle the PEVT is active. The PAG is preset to 0 by RES.

The address vector into the memory banks must be selected from one of three sources. This is accomplished with two banks of two to one data selectors in the Select 2 module. The bank labeled Select 2A selects either the output of PAG or CAG based on corneal event signal CEVT. Note that the normal output of this bank is the pupil address vector. Inputs to the Select 2B module are 10-bits of the system processor address bus and the output of the Select 2A module. Two sets of address vectors, controlled by the $A/\overline{B}$ signal, are generated as outputs.

The x and y coordinate information is generated by the horizontal and vertical counter modules respectively. Both modules are composed of 74LS161 synchronous counters. The clock input to the horizontal counter is the 10-MHz system clock. The counter is reset by the horizontal blanking signal from the synchronization subsystem and increments from 0 to 535 (requiring a 10-bit representation) between successive resets. The vertical counter is incremented by the horizontal blanking signal and reset to 0 by the vertical blanking signal. The counter increments from 0 to 254 or 255, depending on the field being processed, and thus requires an 8-bit resolution. The outputs are denoted "HCNT" and "VCNT."

Entries within the pupil and corneal tables are organized as shown in Figure 7. On each line that a state transition is detected, a sequence of horizontal counts followed by a vertical count is entered into the appropriate table. When the vertical blanking pulse is detected, an end of table signal (-1) is placed in memory.

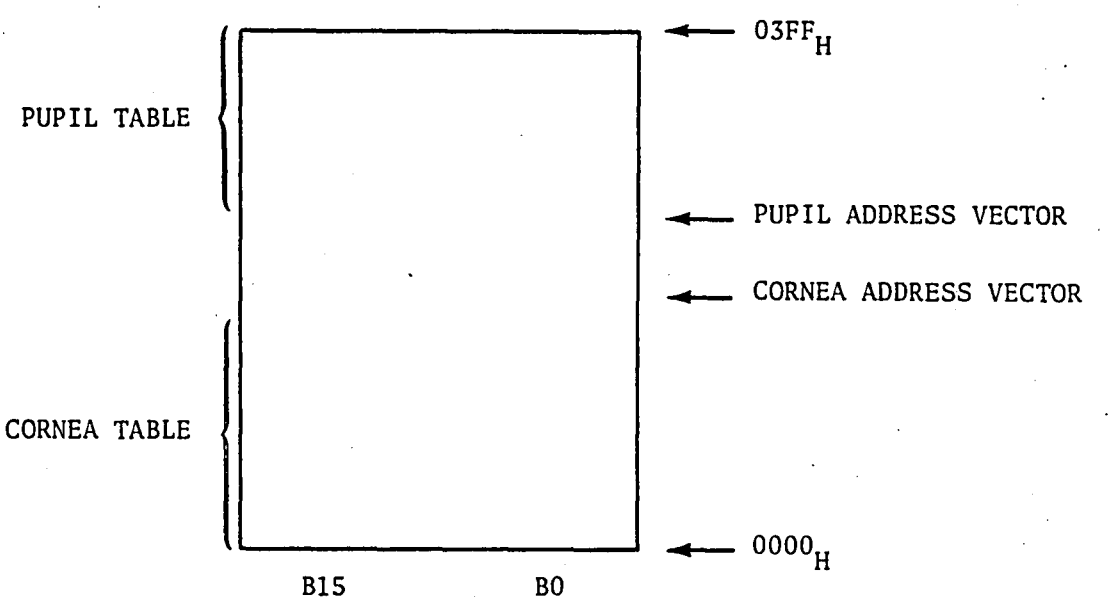


Figure 6. Organization of pupil and corneal tables.

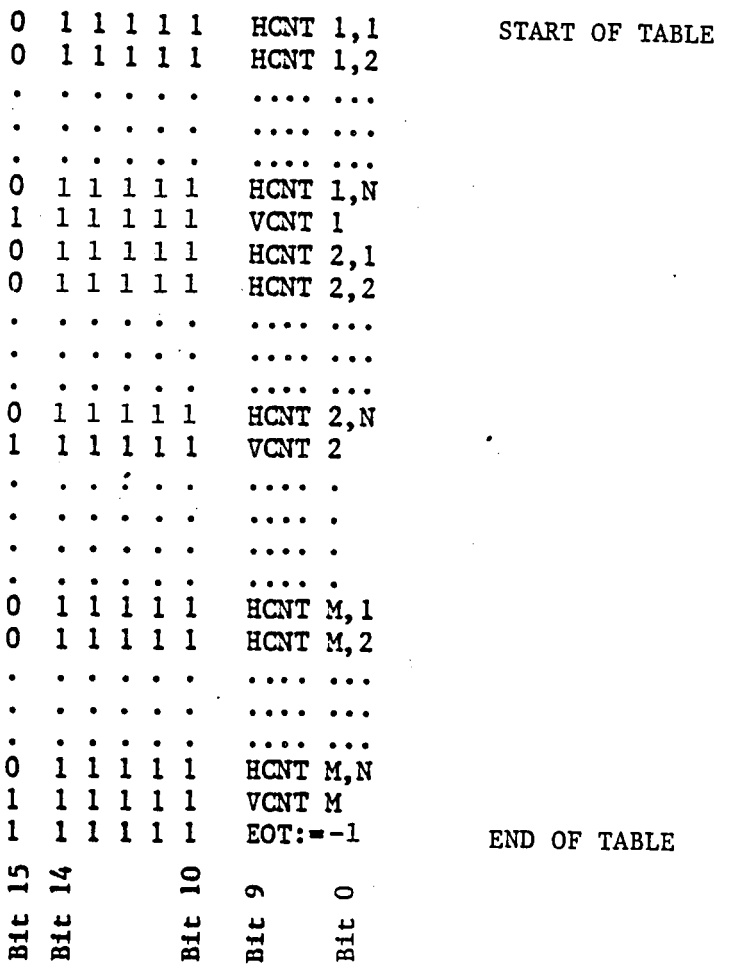


Figure 7. Organization of entries within pupil and corneal tables.

The logic necessary to route the data to or from memory is denoted the Select 1 module and is further partitioned into four submodules. The Select 1A and Select 1B modules are composed of 2 to 1 data selectors and are functionally identical, but have outputs connected to memory banks A and B, respectively. The inputs to these modules are HCNT and VCNT. The outputs of these modules are controlled by the Select 1 control module and are summarized in Table 5; RP1, RP2, and RP3 are resistive pull-ups to generate the EOT marker. The Select 1C module routes data from either memory bank A or B to the processor data bus.

The outputs of the Select 1C module are enabled whenever the address decoder detects a valid memory address. The pupil and corneal event signals (PEVT and CEVT) are produced by a digital edge detector within the event generator module. The circuit is composed of a simple 16-state machine with 2 inputs ($\overline{PS}$ and $\overline{CS}$) and 2 outputs (PEVT and CEVT). Table 6 denotes the state transition where PS and CS are the outputs of the pupil and corneal comparators.

Most interface timing is produced by the EOL/EOT generator. The key components are the 74LS161 binary counter and a 74154 4-to-16 line decoder comprising the control sequencer. The sequencer normally resides in state 11; that is, the pin corresponding to output 11 of the 74154 is low and the counter is disabled. The sequencer remains in this state until the counter is either cleared or preset by the horizontal and vertical pulsers. Each horizontal and vertical blanking pulse sets its respective pulser, and, when the decoder settles into its initial state, the pulser is reset. The flag register contains a pupil event flag, corneal event flag, and an end of file flag. Each flag is set when the corresponding event is detected and is reset by the control sequencer. The sequence of states is summarized in Table 7.

The debug control and threshold detection module determines the source of external control and data signals for the digital interface and compares the input digital video against computer-generated thresholds. The debug control submodules consist of a set of 74LS257 data selectors configured such that, for normal operation (Debug = 0) control, synchronization and video signals are routed from the synchronization and analog subsystems. As previously discussed (under "System Processor")

Table 5. Select 1 module function table.

INPUTS				OUTPUTS		
A/B	EOT	VLOAD	EH/L	Select 1B	Select 1A	Select 1C
0	0	0	0	TRISTATE	TRISTATE	BANK A
0	0	0	1	TRISTATE	TRISTATE	TRISTATE
0	0	1	0	TRISTATE	TRISTATE	BANK A
0	0	1	1	TRISTATE	TRISTATE	TRISTATE
0	1	0	0	TRISTATE	VCNT	BANK A
0	1	0	1	TRISTATE	VCNT	TRISTATE
0	1	1	0	TRISTATE	HCNT	BANK A
0	1	1	1	TRISTATE	HCNT	TRISTATE
1	0	0	0	TRISTATE	TRISTATE	BANK B
1	0	0	1	TRISTATE	TRISTATE	TRISTATE
1	0	1	0	TRISTATE	TRISTATE	BANK B
1	0	1	1	TRISTATE	TRISTATE	TRISTATE
1	1	0	0	VCNT	TRISTATE	BANK B
1	1	0	1	VCNT	TRISTATE	TRISTATE
1	1	1	0	HCNT	TRISTATE	BANK B
1	1	1	1	HCNT	TRISTATE	TRISTATE

Table 6. State transition table.

PRESENT STATE				NEXT STATE				OUTPUTS
Q4	Q3	Q2	Q1	Q4	Q3	Q2	Q1	$CEVT = Q3 \oplus Q4$
A	B	C	D	B	CS	D	PS	$\overline{PEVT} = CEVT \oplus (Q1 + \overline{Q2})$

Table 7. State sequence.

STATE	COMMENTS
0	Initial state of horizontal sequence, reset HORP, generate corneal event if CEVTFLG is set
1	
2	Generate vertical load signal
3	Reset CEVTFLG, generate vertical load signal, hold off second (CSØ2) write into memory
4	General pupil event if PEVTFLG is set
5	
6	Generate VLOAD
7	Generate VLOAD, reset PEVTFLG, hold off second write of pupil event (PSØ2)
8	Generate $\overline{\text{RES}}$
9	
10	
11	Idle state, reset to 0 by $\overline{\text{HORP}}$, preset to E by $\overline{\text{VERTP}}$ reset EOF flag
12	
13	
14	Initial state of vertical sequence, reset $\overline{\text{VERTP}}$, set EOF
15	Go to state 0

these signals may be software generated when debug is true. A debug routine is provided in two 2716 EPROMS. To use this program the system software EPROMS (addresses $FE000_H$-$FEFFF_H$) must be replaced with the debug EPROM.

The threshold detection submodule compares the video signal against computer-generated thresholds and generates two low active open collector outputs ($\overline{PS}$ and $\overline{CS}$) whenever the video signal is greater than the threshold. The outputs are connected to the edge detector previously discussed. The outputs of the 7485 comparators are useful as test points during setup. Representative waveforms expected at test points TP1 to TP4 are shown in Figure 8.

Architecture and Implementation of the High-Speed Arithmetic Processor

The high-speed arithmetic processor (HSAP) is designed to maximize use of the fast processing capabilities made available by its architectural composition. High-speed Schottky logic is used throughout, and particular emphasis is placed on parallel operation where possible. All data interconnection buses within the unit are one-to-one and unidirectional; therefore, delays due to data transfer are minimized. Organization of data flow between subunits is accomplished by extensive use of data selectors under horizontal microprogram control.

As with many digital systems, the HSAP can be architecturally partitioned into a control unit and an arithmetic unit. The arithmetic unit is composed of four subunits, each capable of performing one or more elementary arithmetic operations controlled by bits contained in microprogram memory. The four subunits are (1) a register file, (2) two accumulators, (3) an arithmetic and logic unit and (4) a multiplier (See figs. C8-C19).

The register file serves two functions in the HSAP. First, it is used as an input-output buffer so that all data flow between the associated microcomputer and the HSAP is done through the register file. Second, partial results of current computations are stored in the register file; that is, it acts as a small scratch pad memory for the HSAP. The register file is organized as four addressable 16-bit words, and is accessible by the associated microcomputer, the HSAP accumulators, and the

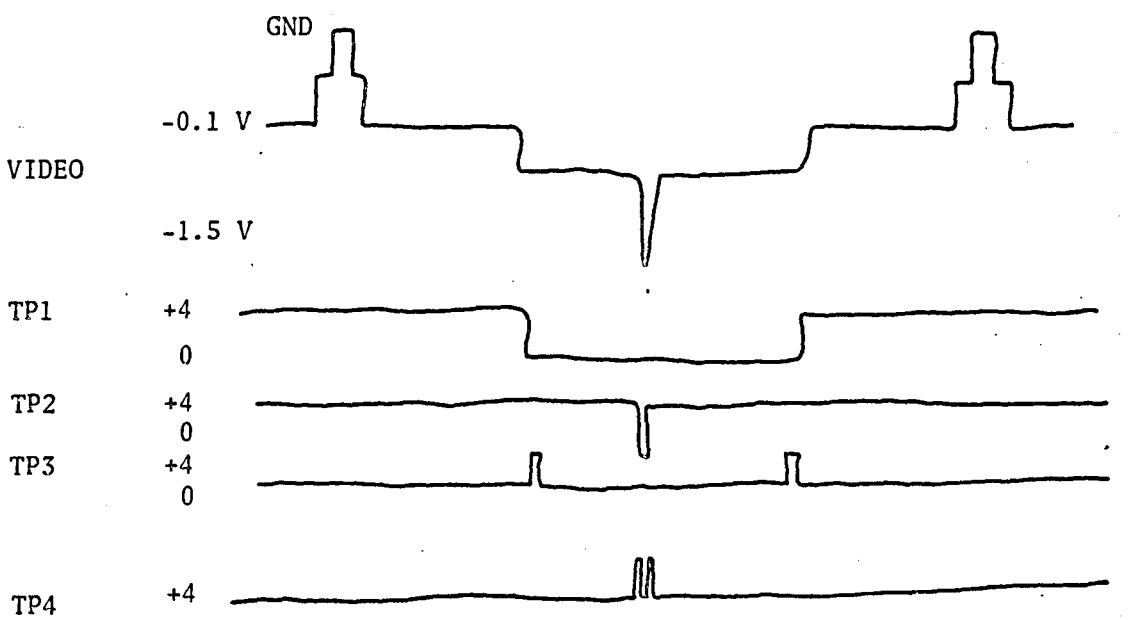


Figure 8. Representative waveforms expected at test points.

emit field of the microprogram. Data may be written to and read from two different registers simultaneously, decreasing the data transfer time between the HSAP and its accompanying system. Control of addressing and read-write functions is accomplished using a two-to-one data selector to choose between the microprogram and the microcomputer. At the start of each algorithm, the associated microcomputer is able to load the register file locations with up to four operands. During computations, the microprogram has exclusive control allowing transfer to and from the accumulators and insertion of constants from the emit field. Once the algorithm has terminated, the HSAP goes into a wait or halt state with the results of the previous algorithm located in the register file again readily accessible by the microcomputer. The register file is realized using 74S153 four-to-one data selectors and 74LS670 four by four register files. One problem with using the 74LS670's is that the read and write enables are level triggered. Because of this, the timing of the write enable pulse is critical since input data must remain stable during the entire length of the pulse. This problem is solved by logically NANDing the write enable pulse with the complement of the system clock and its complement. A better solution would be pin compatible register files with edge triggered write enables.

The accumulator registers, A and B, function both as accumulators and shift registers. They are organized as 16 bits and are capable of performing left and right 2's complement shifts. Access to the accumulators is provided to the register file, the multiplier, the arithmetic logic unit and each other using four-to-one data selectors. The accumulators are realized using 74S194 universal shift registers and 74S153 four-to-one data selectors.

The arithmetic logic unit, ALU, performs two's complement additions, subtractions, and five other logic functions selected by a function code. Additions and subtractions are performed using look-ahead carry to minimize propagation delays. Accumulator registers A and B serve as operands to the ALU, and results of an operation by the ALU are made available to each accumulator. The ALU is realized using 74S381 arithmetic logic units and a 74S182 look-ahead carry generator.

The remaining subunit is a single-chip 16 × 16 bit multiplier capable of producing a 2's complement 32-bit product in 100 nsec. The multiplier is a 64 pin chip and is manufactured by TRW, Inc. The use of a high-speed monolithic multiplier reduces multiplication to an elementary operation. Accumulators A and B serve as multiplier and multiplicand, but there are edge triggered registers internal to the chip. Because of the internal registers, there is a one clock pulse delay to data transfer between the accumulators and the multiplier. The most significant part of the product is directly available to accumulator A. The least significant part is multiplexed onto a bidirectional bus which serves as the input and output bus to accumulator B. Accumulator B is isolated from this bus during an output transfer by tristate buffers. When using fractional notation, only the most significant part of the product is retained, and typically rounding is performed based on the least significant part. This function is available and is under microprogram control. The multiplier is of the MPY-16HJ series, and the buffers are 8T97 tristate buffers.

The control unit is a loop-free sequencer using a counter which provides the address space for the microprogram memory. The control is organized as a 12-bit binary up counter providing up to 4096 states, although only 512 are used in the prototype. Algorithm selection is accomplished by loading the desired address into the counter as an output port. A wait state is produced by disabling the counter via a control bit in the microcode. The microprogram is stored horizontally in high-speed, programmable, read-only memories, and all microinstructions are 48 bits wide including a 16-bit emit field. The control unit is implemented using 74S169 synchronous counters and Fairchild 93448 high-speed proms. All synchronous elements in the HSAP are clocked by a single phase 10-MHz clock.

The prototype HSAP has been realized on two boards, one totally devoted to arithmetic and logic operations and the other devoted to control and interface functions. Microcode interconnections are made along edge connectors (see Figs. C17-18). This organization provides a great degree of flexibility in that the controller board may be completely redesigned to suit a given problem. For example, if a chain calculation

is desired, the present controller board may be used with the routine preprogrammed into the on board PROMS. A looping or jumping program would require a new control board. In any case the arithmetic board remains unchanged.

Several classes of algorithms have been developed for evaluating elementary functions. Volder (ref. 5) proposed the CORDIC method and an accompanying architecture for computing trigonometric functions using additions and shifts as elementary operations. Walther (ref. 6) generalized the CORDIC algorithm to include multiplication, division, and hyperbolic functions using the same basic architecture; deLugish (ref. 7) and Chen (ref. 8) have proposed other types of algorithms and architectures for computing elementary functions, again using additions and shifts as elementary operations.

The use of polynomial approximations is a well-known method for evaluating elementary functions, but their previous use in high-speed applications has suffered due to the number of multiplications involved. This problem no longer exists with the availability of fast LSI multipliers. Using the HSAP, a fifth order polynomial can be evaluated on the order of several microseconds. A big advantage gained is the ability to evaluate any function capable of being approximated by a ratio of polynomials. The problem now becomes finding the best polynomial for some given error cirterion. Truncation of Taylor series expansions is an obvious solution, but it may not be the best solution in terms of minimal order and approximation error. Production of optimum approximations and error curve leveling are discussed by Hastings (ref. 9).

Functions are usually approximated over a finite range of the input variable. In polynomial approximations the range is typically -1 to 1. This fits well within the fractional arithmetic of the HSAP. Operands outside this range must be suitably scaled to fall within the range. This is best accomplished using the decision-making capabilities of the accompanying microcomputer. A list of scalings for the elementary functions is presented by Walther (ref. 6).

The logistics of evaluating a polynomial must be considered when using fractional arithmetic. Polynomial coefficients and the results

of evaluation may exceed the range of fractional arithmetic. Scaling by powers of two appears to be the easiest method of solving this problem, since the only operations needed are shifting. Consider a polynomial of the form:

$$a_{n-1} X^{n-1} + a_{n-2} X^{n-2} + \ldots + a_1 X + a_0$$

In terms of computation, the least number of multiplications is required if the polynomial is represented in a continued product form:

$$\left\{\left(\ldots\left[\left(a_{n-1} X + a_{n-2}\right)X + a_{n-1}\right] X + \ldots + a_1\right)X + a_0\right\}$$

If a scaling by two is required, it may be accomplished by:

$$2^{(\frac{1}{2})}\left\{\left(\ldots\left[\left(a_{n-1} X + a_{n-2}\right)X + a_{n-3}\right] X + \ldots + a_1\right)X + a_0\right\}$$

$$= 2\left\{\left(\ldots\left[\left(a_{n-1} X + a_{n-2}\right)X + a_{n-3}\right] X + \ldots + a_1\right)\frac{X}{2} + \frac{a_0}{2}\right\}$$

$$= 2\left\{\left(\ldots\left[\left(\frac{a_{n-1}}{2} X + \frac{a_{n-2}}{2}\right)X + \frac{a_{n-3}}{2}\right] X + \ldots + \frac{a_1}{2}\right)X + \frac{a_0}{2}\right\}$$

If the scaling is needed, the constants may be stored in the emit field preshifted. If it is necessary to also scale the operand, then a shift will be required after each multiplication.

In summary, in order to implement a function on the HSAP, the function must first be approximated by a polynomial or ratio of polynomials. The approximation must be economized to reduce its order and level its error curve. Finally, the approximation must be scaled to fit the fractional arithmetic of the HSAP. The microroutines may then be developed from the polynomials and implemented in the PROM's on the control board.

Software

The oculometer software was developed subject to the organization of its supporting hardware. The function of the routines comprised in the software is to accept data in the form of pupil and cornea contour coordinates

and produce data representing the angles of deviation between the gaze vector and the optical axis. Like the hardware, the software has been modularized to accentuate flexibility in upgrading and maintenance. The program is organized as a collection of procedures, all written in PL/M-86 or 8086 Assembler, and sequenced by a main program (see Fig. 9). Current subroutines called by the main program are:

(1) Hardware initialization,

(2) Switch banks, and

(3) Center and verify.

Two routines, CALIBRATE and ANGULAR DEVIATION are currently under development. All operator interfaces, such as input/output operations and command interpretation and execution, are performed by the main program. A short overview of each routine follows; for further detail see the software listings being reported separately.

Hardware initialization. - The title "hardware initialization" is self-descriptive. All hardware parameters controlled by the microcomputer are set to initial conditions by this routine. Currently, the gain in the analog signal conditioning stage and the pupil and corneal comparator levels are set to values which will produce valid data. This action is performed via output instructions to PORT$A. Additions of computer control to other hardware functions may be accomplished by expanding the PORT I/O space and adding the appropriate output instructions in this routine.

Switch banks. - The digital interface is essentially a double-buffered memory organized as two banks. While one bank is being accessed by the microcomputer, the other bank is under control of special purpose DMA hardware. Each time through the loop the banks are switched by an appropriate output instruction to PORT$C. This implies that, in order to process every frame of data, the total time through the main program loop must not exceed the field period of 16.6 msec.

Center and verify. - The routine CENTER computes the average values of the pupil and corneal coordinates and finds their relative displacements: XREL and YREL Initially the pupil and corneal signals are represented by the coordinates (addresses in the video field) of their

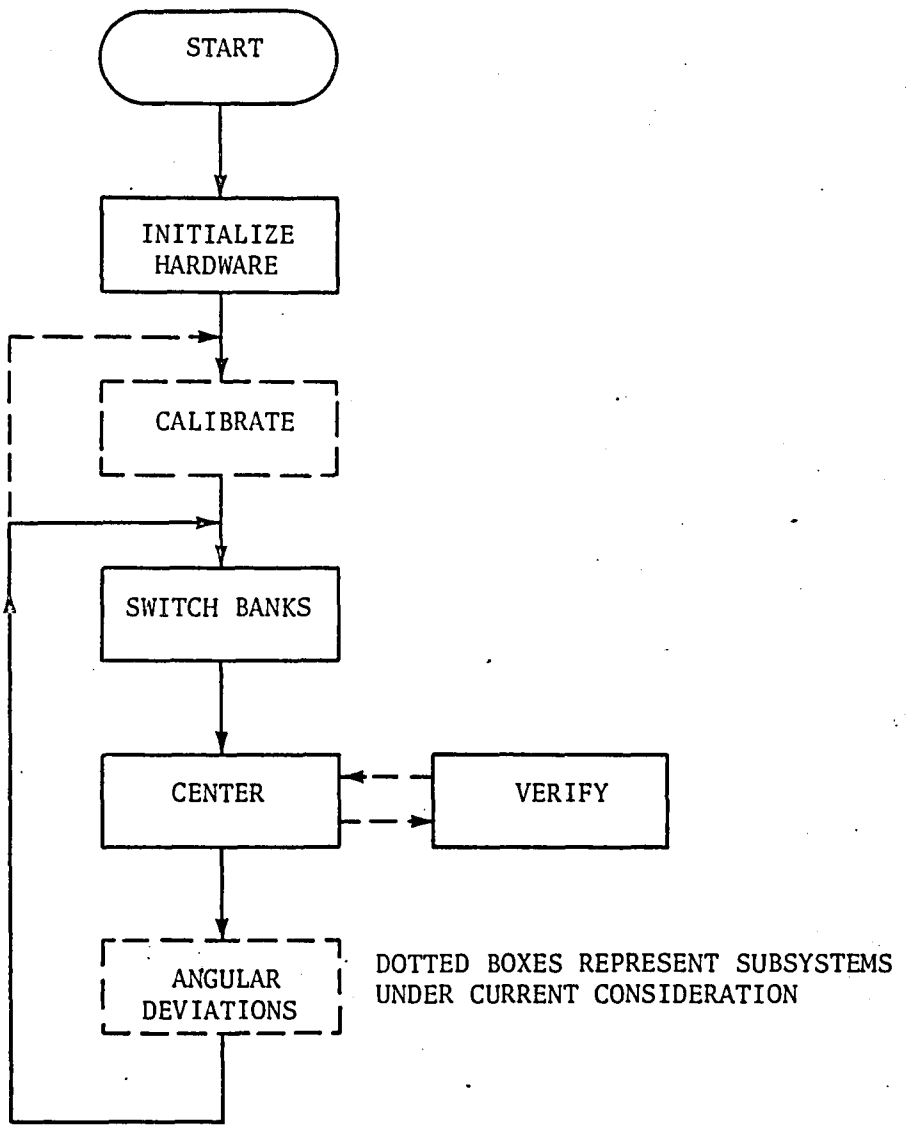


Figure 9. Main program flow chart.

edges. CENTER computes the center coordinates by averaging the X coordinates and Y coordinates for both the pupil and corneal tables:

$$XAVG = \frac{\sum_{i=0}^{n-1} x_i}{n} \quad \text{and} \quad YAVG = \frac{\sum_{i=0}^{n-1} y_i}{n}$$

Once the center coordinates are found, the displacements are calculated by:

XREL = XAVG cornea - XAVG pupil

YREL = YAVG cornea - YAVG pupil.

In order to insure valid pupil data, a window is placed about the current center and is used in the next field. Any data outside the window is rejected as nonpupil.

VERIFY is an offline routine which may be used as an occasional check as to how accurately the "center" routine is working. The routine is offline because of its complexity and, hence, the amount of time it requires. VERIFY assumes that the pupil signal is essentially a circle and uses geometrical calculations to compute its center. The results obtained by this routine are compared to that of CENTER to give some measure of CENTER's performance.

CALIBRATE and ANGULAR DEVIATION are being devised to enable the system to fulfill its intended purpose: to determine the lookpoint of a subject under test. The basis of these routines is the assumption that the angles between the gaze vector and the optical axis are linearly dependent in the relative displacements of the pupil and corneal centers. Calibration becomes a linear regression using a least squares approximation where the input parameters are the XREL and YREL values of known angles. During operation, the output angles are produced by evaluating the linear equations generated by the regression. Preliminary experiments have shown a strong linear relationship. However, if higher order approximations are required, they may be implemented using least squares polynominal fit in place of the linear fit. This means an increase in processing time, but since the calibration is performed only once at the beginning stages, this is not detrimental to the real-time performance of the system.

SUMMARY AND RECOMMENDATIONS

Several aspects of special purpose hardware and software pertaining to the feasibility study of a microprocessor-based oculometer have been discussed. All completed phases of the research have been discussed in the previous sections. In summarizing the research, it seems appropriate to make recommendations for future research for an orderly transition to make the bread board model into an operational model.

The experience gained with the prototype suggests that relatively minor refinements in the allocation of hardware/software functions coupled with recent advances in VLSI and LSI technology could yield significant improvements in system performance. Figure 10 gives one such approach to partitioning tasks into a hierarchial structure where the calculation of gaze vector information can be viewed as a composite function suitable for a realization by a pseudo-pipeline architecture. The data in its most coarse form starts at the bottom of the figure and is refined at each stage until, at the topmost level, the operator is provided with an indication of lookpoint. As shown, information in the form of operator-generated commands also flows in the reverse direction. It is believed that the illumination and headtracker functions are best implemented as semi-autonomous subsystems where global parameters are passed to and from the operator.

At the lowest level of the hierarchy is data collection. Currently this function is implemented with almost all MSI components and no effort is made to preprocess or eliminate any data; therefore, the burden of all data processing falls on the system processor and limits servicing of multiple E/O heads. An effective alternative strategy is to use a single-chip microcomputer to preprocess the input data stream. Used in conjunction with a high-speed monolithic FIFO (first-in, first-out shift register), this approach yields a significant increase in thoughput while reducing component count. The processing responsibility at this level is to place the input data stream in a structure which aids processing by subsequent stages while providing rough estimates of the signal statistics, pupil diameter, and other measures. In the next stage of the pipeline, data within a tracking window is smoothed and confirmed as valid pupil and corneal information. The output of this

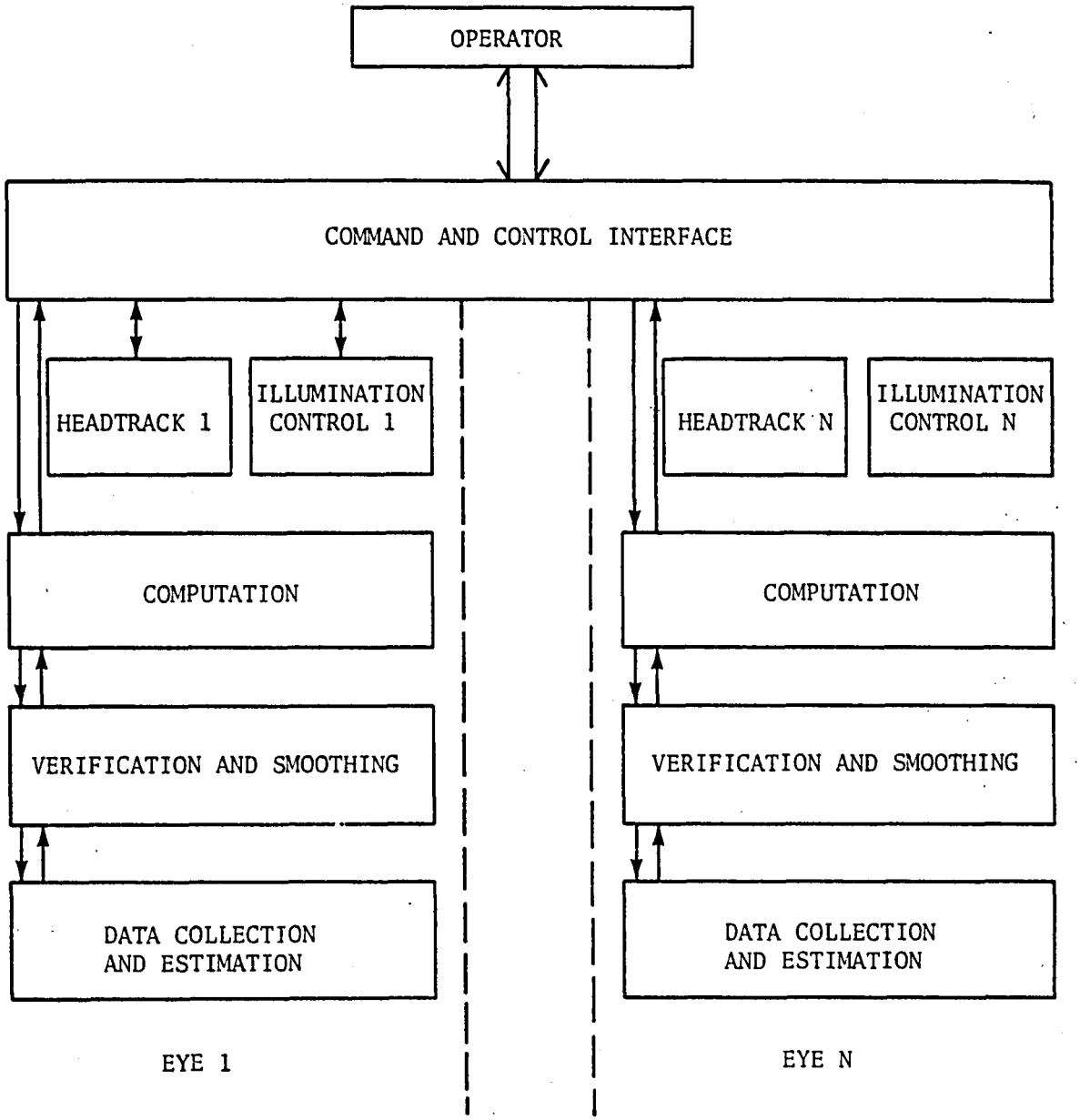


Figure 10. Model of computation and control.

stage includes pupil diameter, the confirmed centers of pupil and corneal reflections, and the relative displacement of the centers.

Lookpoint is calculated in the next stage. Additional functions include accumulation of intermediate statistics regarding scan patterns instrument dwell times, and other physiological responses. These indications of performance are passed to the operator and data logging equipment through the command and control interface.

The operator must perceive the system to be friendly. The command and control interface creates this impression with the generation of positive cues and immediate system recognition and acknowledgment of operator actions. This module controls the overall program flow and should provide the user with pertinent information from setup and calibration through all operational phases in which the system is likely to be used. It is believed that incorporation of these recommendations will enhance the system into a viable instrument for its projected use in flight management research. Complete and detailed discussion of this approach will be provided in a separate communication.

The constraint to process data in near real time, coupled with the relatively low information bandwidth of the current generation of microprocessors, imposes severe restrictions on the categories of signal-processing algorithms that may be utilized in an operational oculometer; however, the new generation of microprocessors to be introduced in the next two years offer performance gains of 100 to 300 percent (ref. 10). If modular design techniques are adhered to, exploitation of these new technologies can be accomplished with minimal impact on other system elements; but as the complexity and sophistication of these components grow, increasing demands will be placed on the hardware designer. To meet these demands, development of special purpose hardware must be limited to those functions (such as data collection) which cannot be accomplished with commercially available equipment. It is recommended that future oculometer designs be standardized around board level components from a single vendor with a common bus and development language.

ACKNOWLEDGMENTS

In any research investigation spanning several years, it is impractical to properly acknowledge every contribution. Nevertheless, the author would like to acknowledge the guidance provided by flight management branch researchers P.A. Gainer, M.A. Wise, M. Waller, A. Meintel, and M. Kurbjun. The author also acknowledges the invaluable contributions provided by the following research assistants during the investigation: L. Johnston, H. Tran, D. Livingston, F.W. Harrison, S. Charlambous, L. Ray, Y. Chong, M. Arunachalam, and W. Dalton. Finally, it is with utmost admiration that the author acknowledges the administrative support provided by Hope M. Howard during critical phases of the project.

REFERENCES

1. Young, L.R.; and Sheena, D.: Survey of Eye Movement Recording Methods. Behavior Research Methods and Instrumentation, Vol. 7, No. 5, 1975, pp. 397-429.

2. Merchant, J.; Morrissette, R.; and Porterfield, J.L.: Remote Measurement of Eye Direction Allowing Subject Motion over One Cubic Foot of Space. Engineering, Vol. 21, No. 4, July 1974, pp. 309-317.

3. Monty, R.A.: An Advanced Eye Movement Measuring and Recording System Featuring Unobtrusive Monitoring and Automatic Data Processing. American Psychologist, Vol. 30, 1975, pp. 331-335.

4. Petruk, M.W.; and Hunka, S.: Oculometer Laboratory Systems. The Canada Council Research Project 571-890, Final Report, Feb. 1974.

5. Volder, J.E.: The CORDIC Trigonometic Computing Technique. IRE Transactions on Electronic Computers, Sept. 1959, pp. 330-334.

6. Walther, J.S.: A Unified Algorithm for Elementary Functions. Spring Joint Computer Conference Proceedings, 1971, pp. 379-385.

7. DeLugish, B.G. A Class of Algorithms for Automatic Evaluation of Certain Elementary Functions in a Binary Computer. Report No. 399, Department of Computer Science, Univ. of Illinois, June 1, 1970.

8. Chen, T.C.: Automatic Computation of Exponentials, Logorithms, Ratios, and Square Roots. J. Res. Development, July 1972, pp. 380-388.

9. Hastings, C., Jr.: Approximations for Digital Computer. Princeton Univ. Press (Princeton, NJ), 1955.

10. Microsystem 80 Advance Information. Intel Corporation, 1980, pp. 2-7.

APPENDIX A

ALGORITHMIC PROCESSOR SIMULATOR

Introduction

The algorithmic processor simulator was developed to test the programs to be programmed into the PROMS of the algorithmic processor.

Each hardware-associated instruction set has a hexidecimal representation, used only by the simulator. The desired program must be preassembled manually and then loaded into memory before the simulator can run.

The simulator operates on the same rules that govern the algorithmic processor, except that it allows one concurrent instruction that the processor does not: that is, information can be read into and out of the same register file location at the same time. This is an error condition which the simulator does not detect.

The simulator accepts each instruction and adjusts the code to represent the memory location where the routine is stored. It then calls this routine with a variable jump. The simulator checks the sign of each instruction to see if it is concurrent with the next instruction. In order to handle concurrency of the machine, the simulator stores the results of each instruction in temporary locations in memory, exchanging these locations with the A and B registers and the register files after the multiplication is performed. Multiplication of the A and B registers is performed after each machine cycle. The register representations are locations in the memory that hold the results of an operation.

Each instruction or set of concurrent instructions is disassembled after each cycle by adjusting the original instruction with a mask, a series of shifts, and by adding a constant. This obtains the pointer to the ASCII representation table of each instruction. This instruction is then displayed along with the contents of the registers and the register files. This allows the user to check for errors in his routine.

Assembling the Code

Each instruction has a hexidecimal representation that the simulator uses to execute the desired routine. The hexidecimal word or constant that is to be loaded into a register file must follow the instruction byte, with the low-order byte first and the high-order byte second. Concurrency is implemented by adding 80H to all but the last instruction in the concurrent set. This sets the sign bit of the instruction, which is checked by a mask for concurrency with the next instruction. The last instruction to the simulator must be a halt. The assembler code is listed in Table A1 which follows.

Table A1. Assembler code.

INSTRUCTION	HEXIDECIMAL REPRESENTATION	FUNCTION
R0≠NNH	28	Load register file Ø with a word constant.
R1=NNH	29	Load register file 1 with a word constant.
R2=NNH	2A	Load register file 2 with a word constant.
R3=NNH	2B	Load register file 3 with a word constant.
RA=RØ	ØØ	Load the contents of register file Ø into register A.
RA=R1	Ø1	Load the contents of register file 1 into register A.
RA=R2	Ø2	Load the contents of register file 2 into register A.
RA=R3	Ø3	Load the contents of register file 3 into register A.
RB=RØ	1Ø	Load the contents of register file Ø into register B.
RB=R1	11	Load the contents of register file 1 into register B.
RB=R2	12	Load the contents of register file 2 into register B.
RB=R3	13	Load the contents of register file 3 into register B.
RA=RB	Ø4	Load the contents of register file B into register A.
RB=RA	14	Load the contents of register A into register B.
RA=U	Ø5	Load the high-order byte of the multiplication into register A.
RB=L	15	Load the low-order byte of the multiplication into register B.
RA=SRRA	Ø6	Shift the contents of register A 1 bit to the right.
RA=SLRA	Ø7	Shift the contents of register A 1 bit to the left.
RB=SRRB	16	Shift the contents of register B 1 bit to the right.
RB=SLRB	17	Shift the contents of register B 1 bit to the left.
RA=Ø	Ø8	Reset all bits of register A (clear register A).

Table A1. (Concluded)

INSTRUCTION	HEXIDECIMAL REPRESENTATION	FUNCTION
RB=Ø	18	Reset all bits of register B (clear register B).
RA=RB-RA	Ø9	Subtract register A from register B and load into register A.
RA=RA-RB	ØA	Substract register B from register A.
RA=RA+RB	ØB	Add register B to register A.
RB=RB-RA	19	Subtract register A from register B.
RB=RA-RB	1A	Subtract register B from register A and load into register B.
RB=RA+RB	1B	Add register A to register B.
RA=RA XR RB	ØC	Exclusive OR register A with register B and load into register A.
RA=RA OR RB	ØD	OR register A with register B and load into register A.
RA=RA AN RB	ØE	AND register A with register B and load into register A.
RB=RA XR RB	1C	Exclusive OR register A with register B and load into register B.
RB=RA OR RB	1D	OR register A with register B and load into register B.
RB=RA AN RB	1E	AND register A with register B and load into register B.
RA=1	ØF	Set all bits of register A.
RB=1	1F	Set all bits of register B.

APPENDIX B

DIMENSIONALITY REDUCTION OF THE KARHUNEN-LOÈVE TRANSFORM

By

Salomi T. Charalambous

Abstract

It is generally agreed that, when the minimum L_2 norm is used as the performance measure in data compression applications, the Karhunen-Loève Transform (KLT) is the optimum compressor. In spite of its optimality, however, it has not been possible to derive a fast implementation comparable to other orthogonal transforms. It is the purpose of this research to demonstrate that preceding the transform by a zero-error predictor yields a viable solution to the implementation of the Karhunen-Loève Transform. This will require reduction of the covariance matrix computation time, the eigenvector computation time, and the transformation time.

Introduction

Among their wide spectrum of applications, orthogonal transforms offer a theoretical basis for representing data in data compression applications. Since most often such signal-processing applications are realized in a Euclidean vector space, the minimum L_2 norm (minimum mean square error) has been accepted as a satisfactory performance measure. For this performance measure, the Karhunen-Loève Transform (KLT) has been shown (refs. 1-3) to be the optimum data-reduction algorithm for processes belonging to a given distribution class with the same second order statistics. The performance of the KLT is followed by the Fourier Transform (FT) and the Hadamard Transform, respectively. In terms of ease of implementation, the order is reversed (refs. 1-3). As compared to other transforms, for a given mean square error, the KLT requires the minimum number of basis functions to represent a signal. Consequently, for an equal number of basis functions, the KLT yields the best representation of the original process; but, unlike other transforms, no fast implementation has yet been determined.

Assuming that M basis vectors are necessary to represent an N-dimensional data sequence, then MN multiplications and additions are required to transform the data. In addition, the basis functions of the transform are the eigenvectors of the covariance of the input process. This implies either prior knowledge of the covariance, or a need to compute the covariance and its corresponding eigenvectors.

In this study we propose a strategy which assures a reduction of the dimensionality difficulties of the KLT with minimal effect on its performance. Our strategy is to precede the KLT with a predictor which introduces no error. The predictor reduces the dimension of the data vector from N to K, where $K \leq N$, thus reducing the number of transform operations from MN to MK, a reduction factor of M(N-K). Also, for an N-dimensional sequence, $\{\zeta(n)\}$, the dimension of the covariance matrix Σ_ζ is $N \times N$. The presence of the predictor reduces this dimension to $K \times K$, consequently reducing both the covariance computation time and its eigenvector computation time.

In the next section ("Signal Statistics") we discuss the desired statistical properties of the input process, and under "Karhunen-Loève Transform" the properties of the KLT are described. "Proposed Solution" describes further the proposed strategy to reduce the dimensionality difficulties of the KLT, and the section titled "Verification" presents the results obtained. In the final section of the test, conclusions are presented.

Signal Statistics

When designing a system or an algorithm, the engineer must know something about the input signal and its statistical properties. For this reason, some effort is spent here to determine some of the statistical properties which the input to the proposed system is assumed to possess. Also, we restrict our attention to the discrete case only, since the continuous case is an extension of the discrete.

Assume that a discrete sample function can be represented by a finite sequence, $\{\zeta(n)\}$. This sequence consists of second-order, stationary, zero-mean, random variables ζ_i's, such that

$$E\{\zeta_i\} = 0 \tag{B1}$$

$$E\{\zeta_i^2\} = \sigma^2{}_i \tag{B2}$$

$$E\{\zeta_i \zeta_j\} = \sigma_{ij}^2 = \sigma_{ji}^2 \tag{B3}$$

where $E\{.\}$ is the expected value. The zero-mean assumption, expressed by equation (B1), is made for simplicity of mathematics. This assumption also leads to equation (B2), the variance of the variables. Equation (B3) expresses the property of second-order stationarity, which means that the correlation function is invariant to time translation.

With these properties in mind, the nth sample function Z_n is defined by

$$Z_n^T = \{\zeta_1, \zeta_2, \ \ldots \ , \zeta_N\} \tag{B4}$$

The ensemble of this random process can be expressed by the column vector Ξ as

$$\Xi^T = \{Z_1, Z_2, \ \ldots \ Z_L\} \tag{B5}$$

where L is the number of discrete sample functions.

This discussion involves the Karhunen-Loève Transform (KLT), whose basis vectors are the eigenvectors of the covariance matrix of the random process. The covariance is defined by

$$\begin{aligned} \sum_Z &= E\left\{[Z - E\{Z\}][Z - E\{Z\}]^T\right\} \\ &= E\left\{Z\ Z^T\right\} = E\left\{Z\right\}\ E\left\{Z^T\right\} \\ &= E\left\{Z\ Z^T\right\} \end{aligned} \tag{B6}$$

for the zero-mean case. Equation (B6) can be expanded into

$$\sum_Z = E\left\{\begin{bmatrix}\zeta_1\\ \zeta_2\\ \vdots\\ \zeta_N\end{bmatrix}[\zeta_1,\zeta_2,\ \ldots,\ \zeta_N]\right\}$$

$$= E\begin{bmatrix}\zeta_1\zeta_1,\ \zeta_1\zeta_2\ \ldots,\ \zeta_1\zeta_N\\ \zeta_2\zeta_1\\ \vdots\\ \zeta_N\zeta_1 \qquad\qquad \zeta_N\zeta_N\end{bmatrix}$$

$$\sum_Z = \begin{bmatrix}E\{\zeta_1\zeta_1\},\ E\{\zeta_1\zeta_2\}\ \ldots\ E\{\zeta_1\zeta_N\}\\ E\{\zeta_2\zeta_1\}\\ \vdots\\ E\{\zeta_N\zeta_1\} \qquad\qquad E\{\zeta_N\zeta_N\}\end{bmatrix} \tag{B7}$$

It is necessary to assume that the process is second-order stationary since the basis functions of the KLT are the eigenvectors of the covariance matrix. Otherwise, the basis functions will change, and the period of stationarity must be known so that new basis functions can be determined for that period.

Karhunen-Loève Transform

The Karhunen-Loève Transform is a transformation which completely preserves the information of the original process. It uses an optimal set of orthonormal functions derived from the covariance matrix of the random process (refs. 4-12). The optimality results because, compared to other orthonormal transforms, a minimum number of basis vectors is needed to represent the signal within a given mean square error. Figure B1 displays a representation of the forward and inverse KLT of the data vector $Z = \{\zeta(n)\}$.

The sequence $\{\zeta(n)\}$ can be represented by the inner product between the transform coefficient vector A and the basis vectors of the tranform. This relationship is expressed by

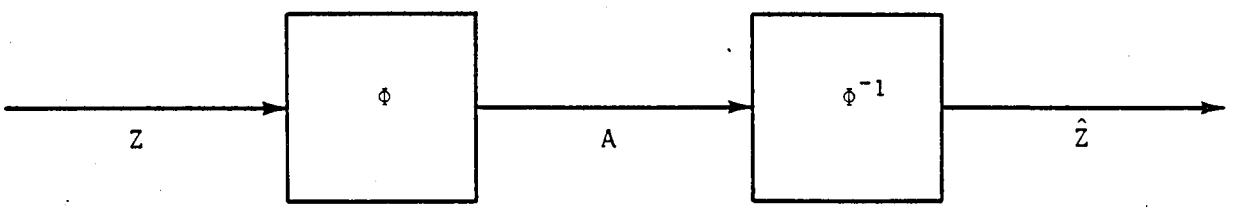


Figure B1. Forward and inverse transformation of vector Z.

$$\zeta_n = < A | \phi_n > \tag{B8}$$

where

$$A^T = [\alpha_1, \alpha_2, \ldots, \alpha_M] \tag{B9}$$

and

$$\phi_n^T = [\phi_{1n}, \phi_{2n}, \ldots, \phi_{Mn}] \tag{B10}$$

In the Euclidean vector space, equation (B8) results in

$$\hat{\zeta}_n = \sum_{i=1}^{M} \alpha_i \phi_{in} \cdot n = 1,2, \ldots, N \tag{B11}$$

The transform coefficients α_i's are computed from the inner product between the input sequence $\{\zeta(n)\}$ and the basis functions. Therefore, the ith coefficient is

$$\alpha_i = < \phi_i | Z > \tag{B12}$$

where Z is defined by equation (4) and ϕ_i by equation (B10). Further, it can be shown (ref. 7) that these transform coefficients are completely uncorrelated such that

$$E\{\alpha_i \alpha_j\} = \lambda_i \delta_{ij}, \; i,j = 1,2, \ldots, M \tag{B13}$$

and the λ_i's are the eigenvalues corresponding to the eigenvectors.

The basis vectors form an orthonormal set since they arise from a symmetric covariance matrix. The set is formed by considering only those eigenvectors (of the covariance matrix) with corresponding largest eigenvalues arranged in monotonically descending order. Therefore, although the dimension of the eigenvectors is N, only M eigenvectors are used to approximate the signal, thus reducing the data by (N-M) components.

The number of eigenvectors used is determined by the minimum mean square error. It is shown (ref. 7) that if M eigenvectors with corresponding

largest eigenvalues are used to approximate the signal by equation (B11), the minimum mean square error between $\{\zeta(n)\}$ and $\{\hat{\zeta}(n)\}$ is

$$\epsilon_{min} = \sum_{i=M+1}^{N} \lambda_i \tag{B14}$$

where the λ_i's represent the remaining (N-M) eigenvalues.

Therefore, when the minimum mean square error is used as the performance measure for data compression techniques, the KLT is optimum. For a mean square error, it maximizes data compression by generating a minimal set of completely uncorrelated transform coefficients $\{\alpha_i\}$. However, its optimality is not entirely ideal. Precise calculation of the transformation matrix presumes prior knowledge of the covariance matrix. Calculation of the matrix is normally a long and complex process. Furthermore, equation (B11) requires MN operations and MN is normally a large number.

Proposed Solution

It has been shown (refs. 6, 13) that, if a transformation matrix consists of a large number of redundancy, it may be possible to factor the matrix into Kronecker products of sparce matrices. When such factorization is established, a fast implementation of that transform is possible. Since the KLT matrix is not predefined but must be determined from the input process, such fatorization is generally not easily derived. Consequently, alternative approaches for fast implementations have been studied.

The discussion of the previous section leads to the conclusion that the greatest limitation of the KLT is its dimensionality: i.e. the large number of computations required. Since the dimensionality arises from the large dimension of the data vector and consequently the basis vectors, one approach to reduce the dimensionality difficulties (of the transform) is to reduce the dimension of the data before applying the KLT. This is the approach taken by this study.

Before proceeding, it should be noted that the solution must satisfy certain objectives: it must introduce no additional error to that

introduced by the KLT; it must be simple to implement; and it must be able to transform a second-order stationary process. A simple redundancy reduction technique such as a predictor or an interpolator can realize these objectives. However, an additional requirement is that the redundancy reduction must be real time. Since the interpolator is not a real-time process, it leaves the predictor as the most appropriate. A description of the proposed solution is shown in Figure B2.

A predictor is a system which can predict the value of each new data sample based on the past history of the data (ref. 4). Several orders of polynomial predictors are possible, the zero-order being the simplest (see Fig. B3). It predicts that each new data value will be the same as the preceding within a $\pm T_o$ tolerance aperture. This implies that the data can be approximated by a horizontal line (see Fig. B3). It has been shown (ref. 4) that for most applications the zero-order predictor is adequate; thus, further discussion will concentrate on it only. If the predictor introduces no error, the tolerance aperture must be zero so that the predicted value exactly matches the actual value, or that sample is not considered redundant. Therefore, the zero-order predictor satisfies all the criteria stated for the system.

As defined earlier (see "Signal Statistics"), the data vector is of the form

$$Z_i^T = [\zeta_1, \zeta_2 \ \ldots, \ \zeta_N]$$

where Z_i is the input to the predictor (see Fig. B2). The predictor reduces the data dimension from N to K so that its output vector is of the form

$$Z_i^T = [\zeta_1^*, \zeta_2^* \ , \ldots, \ \zeta_K^*] \tag{B16}$$

and the ijth component σ_{ij}^2 of the covariance matrix is

$$\sigma_{ij}^2 = E\{\zeta_i \zeta_j\}$$

$$= \frac{1}{L} \sum_{n=1}^{L} \zeta_i^n \, \zeta_j^n, \ i,j = 1, \ \ldots \ K \tag{B17}$$

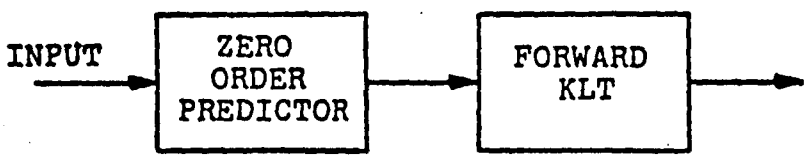


Figure B2. Proposed algorithm.

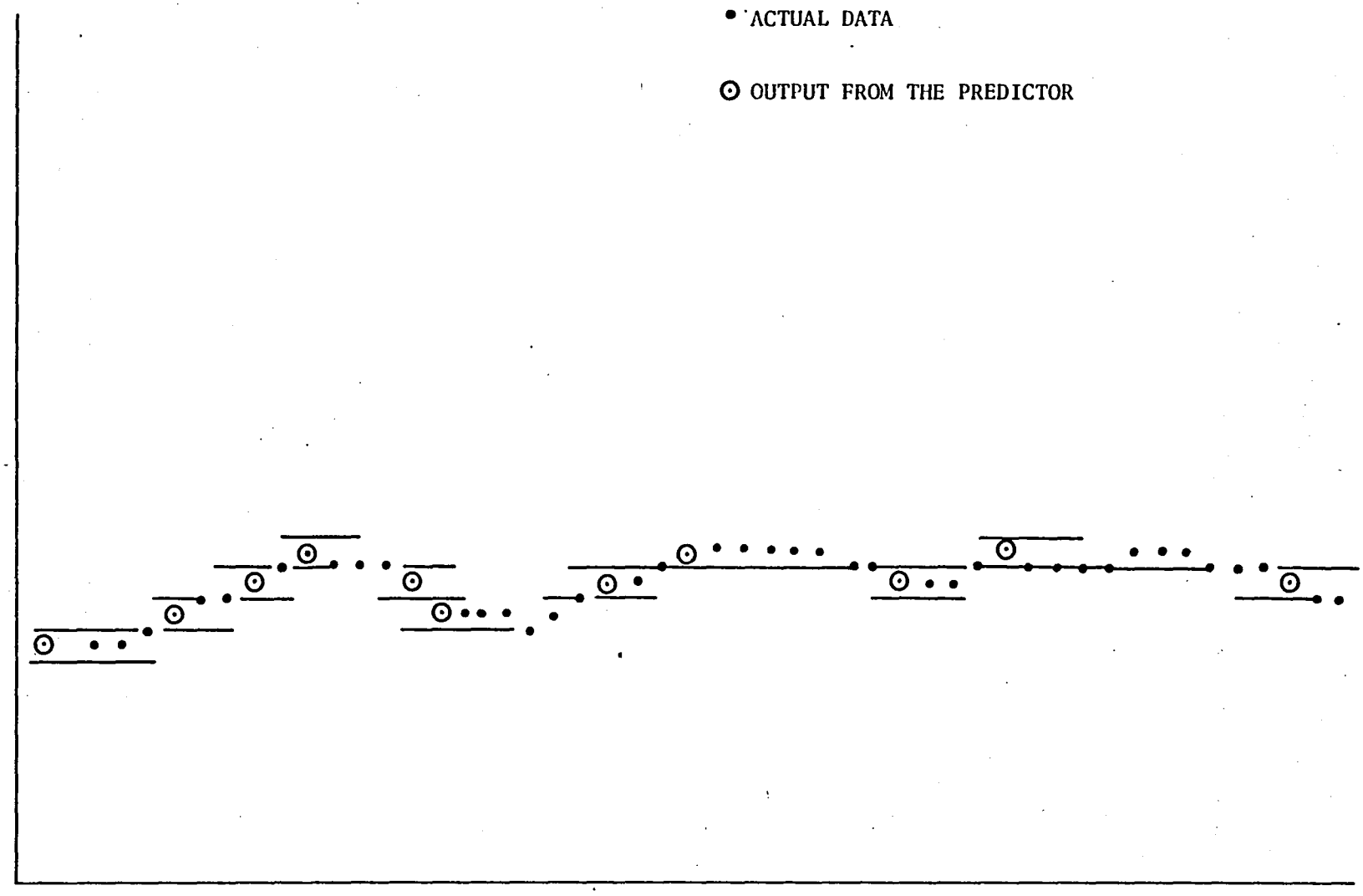


Figure B3. Zero order predictor.

where ζ_i^n denotes the nth value of data component i. Note that the dimension of the covariance matrix is K × K rather than N × N. With the covariance matrix available, its eigenvectors must be computed. The process is normally long and complex. It has been shown (refs. 11, 12) that, if the covariance matrix is bisymmetric, it can be partitioned into submatrices of smaller dimension. When such a partition is possible, the eigenvector computation time is reduced by a factor of four (ref. 11). If a partition is possible, along with the reduced dimension, the eigenvector computation time can be reduced significantly.

Once the basis set is determined, the system is ready to begin transforming each data vector. This process computes the transform coefficients α_i's by equation (B12). This computation requires MK multiplications, a reduction of M(N-K) operations. Therefore, depending on the data structure and on the order of polynomial predictor used, if K is minimized without introducing any error, the dimensionality of the KLT is reduced significantly.

Verification

Verification of the proposed system was carried out on the DEC-10 general purpose computer in FORTRAN. The objective was to verify proper overall operation of the solution proposed as well as to show that the predictor preceding the KLT does not adversely affect the transform's performance. The signal used for the verification is a video signal resulting from an oculometer, which when displayed by a television normally appears as one of the images of Figure B4. The oculometer is a vision-monitoring device. Its function is to determine a person's lookpoint on a rectangular plane at a fixed distance away by projecting infrared light (IR) into one of the subject's eyes. An IR-sensitive video camera images the pupil and corneal reflections resulting from the subject's eye.[1] (For further detail see references 14 and 15).

---

[1]The Flight Management Branch at NASA/LaRC uses the oculometer to determine an aircraft pilot's lookpoint on the instrument panel during landing conditions. This study will help them design future aircraft instrument panels that are better suited to the pilot.

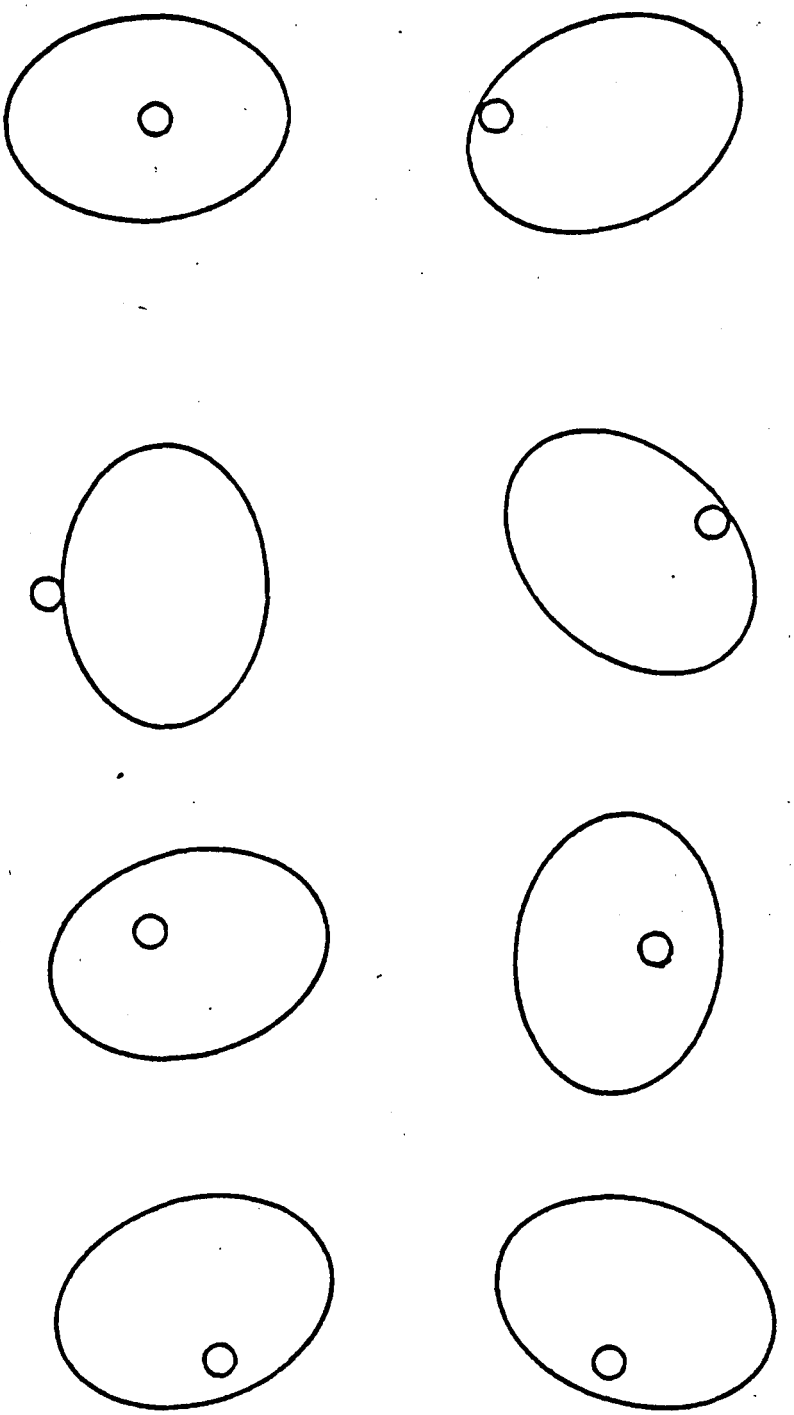


Figure B4. Pupil and corneal reflections corresponding to different look-points.

Two performance measures were used to evaluate the results: the correlation coefficient, ρ, defined by equations (B18) to (B23), and the mean square error, ε, between the input sequence $\{\zeta(n)\}$ and the output sequence $\{\hat{\zeta}(n)\}$, defined by equation (B24).

$$\rho = \frac{\sigma^2_{IO}}{\sigma_I \sigma_O} \tag{B18}$$

$$\sigma_I = \left[\frac{1}{K}\sum_{i=1}^{K} (\zeta_i - \overline{\zeta})^2\right]^{1/2} \tag{B19}$$

$$\sigma_o = \left[\frac{1}{K}\sum_{i=1}^{K} (\hat{\zeta}_i - \hat{\zeta})^2\right]^{1/2} \tag{B20}$$

$$\sigma^2_{IO} = \frac{1}{K}\sum_{i=1}^{K} (\zeta_i - \overline{\zeta})(\hat{\zeta}_i - \hat{\overline{\zeta}}) \tag{B21}$$

$$\overline{\zeta} = \frac{1}{K}\sum_{i=1}^{K} \zeta_i \tag{B22}$$

$$\hat{\overline{\zeta}} = \frac{1}{K}\sum_{i=1}^{K} \hat{\zeta}_i \tag{B23}$$

$$\varepsilon = \left[\frac{1}{K}\sum_{i=1}^{K} (\zeta_i - \hat{\zeta}_i)^2\right]^{1/2} \tag{B24}$$

Due to limited computer storage available, rather than process the entire image, only the region of the image which contained the desirable information was tested. Two tests were conducted: one using a zero-order predictor with a floating aperture and one using a smaller region of the image with a zero-order predictor whose tolerance aperture was zero. Although the floating tolerance aperture was expected to introduce an error to the signal which would not be acceptable to the algorithm, the test was carried out for comparison of results. Figure B5 shows a plot of the position vector corresponding to the reduced amplitude vector at the output of the predictor with the floating aperture. Two video fields were used to compute the covariance matrix under the conditions described. Only two eigenvectors were necessary to form the transformation matrix in order to represent the data vector within a mean square error of 0.4851

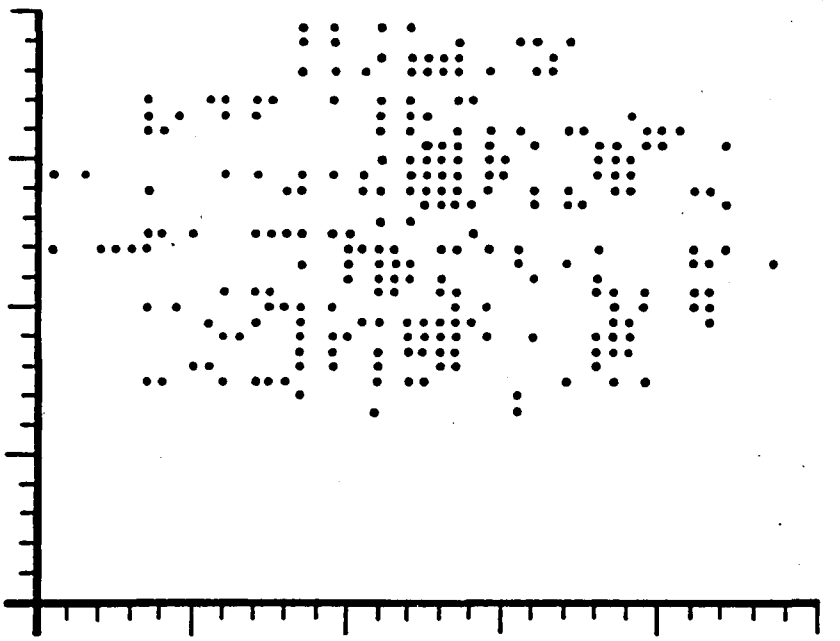


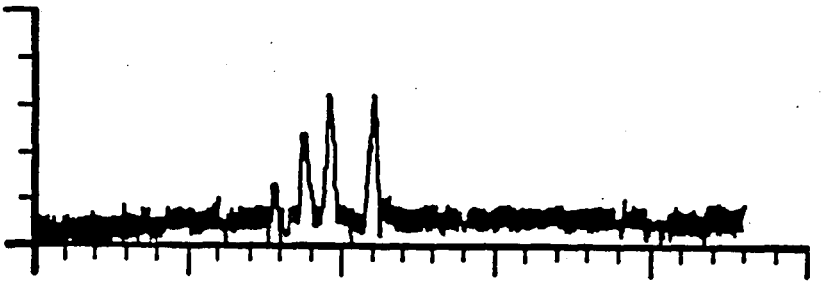
Figure B5. Plot of the position vector corresponding to the reduced amplitude vector at the output of the predictor with the floating aperture.

percent and a correlation coefficient of 1.00. The input to the transform and its corresponding reproduced vector are displayed in Figures B6(a) and (b), respectively. A difference curve for the two curves of Figure B6 is displayed by Figure B7, along with correlation coefficient ρ mean square error ε. The two eigenvectors composing the transformation matrix are shown in Figures B8(a) and (b). Both vectors share characteristics similar to the amplitude vector. Since a wide tolerance aperture was used by the predictor in order to reduce the dimension to within the limits of the available computer storage capacity, it was expected that a large mean square error would result at the output of the inverse predictor. The error was large, but the correlation coefficient was 0.84115, which could be acceptable for some applications.

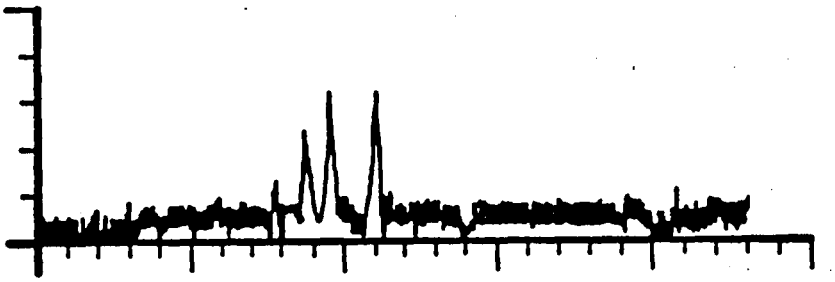
A second test was conducted using a zero-tolerance aperture predictor. Also, the size of the region was reduced in order to reduce the dimension to within the limits of available computer storage capacity. In addition to reducing the region size, the original data were smoothed by a digital filter to remove much of the high-frequency noise in the data. The covariance matrix for this set of data was computed, where each entry was defined by

$$\sigma_{ij}^2 = \sigma_{ji}^2 = \frac{1}{3}\sum_{m=1}^{2} \zeta_i^m \zeta_j^m - \bar{\zeta}_i \bar{\zeta}_j \qquad \text{(B25)}$$

where ζ_i^m denotes the mth value of data component i. A null vector was assumed in the calculation of the covariance matrix and therefore a division by three was necessary in equation (B25). When the null vector was not assumed, results were not satisfactory. Again, two eigenvectors were necessary to represent the data within a 5.13597 percent mean square error and a correlation coefficient of 0.994362. The input amplitude vector and corresponding reconstructed vector are displayed by Figures B9(a) and (b), respectively. The difference curve of the two Figures is displayed by Figure B10, and the eigenvectors used for this transformation matrix are shown in Figures B11(a) and (b). They share similar characteristics with the eigenvectors of Figures B8(a) and (b), and also with the input amplitude vector. The mean square error and the



(a)



(b)

Figure B6. Input to the transform (a) and its corresponding vector (b).

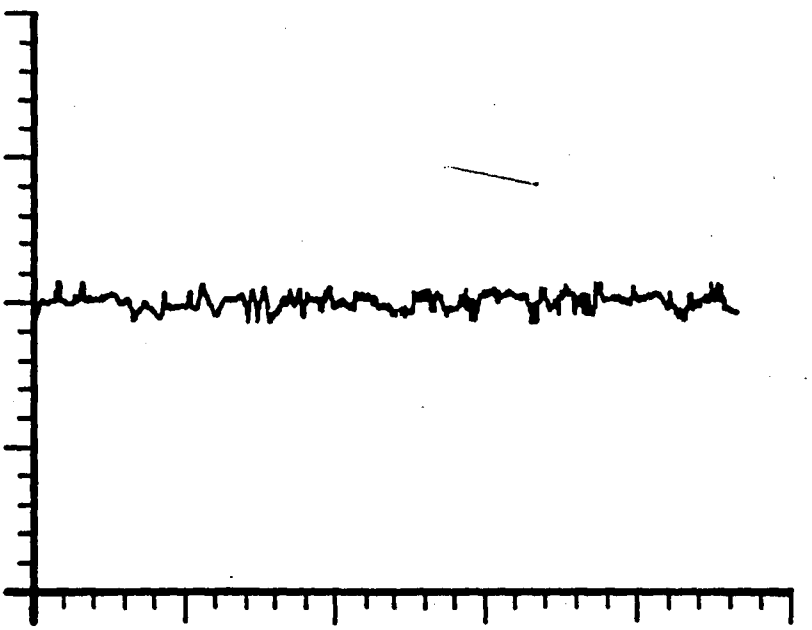
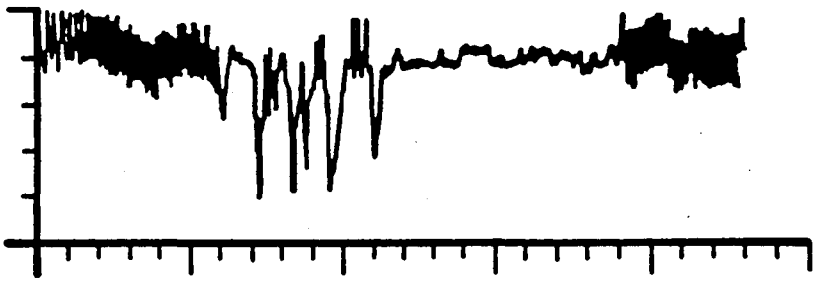
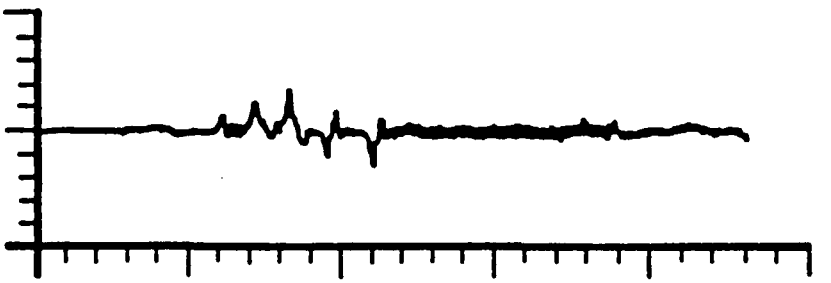


Figure B7. Difference curve for the curves shown in Figures B6(a) and (b).



(a)



(b)

Figure B8. Eigenvectors composing the transformation matrix.

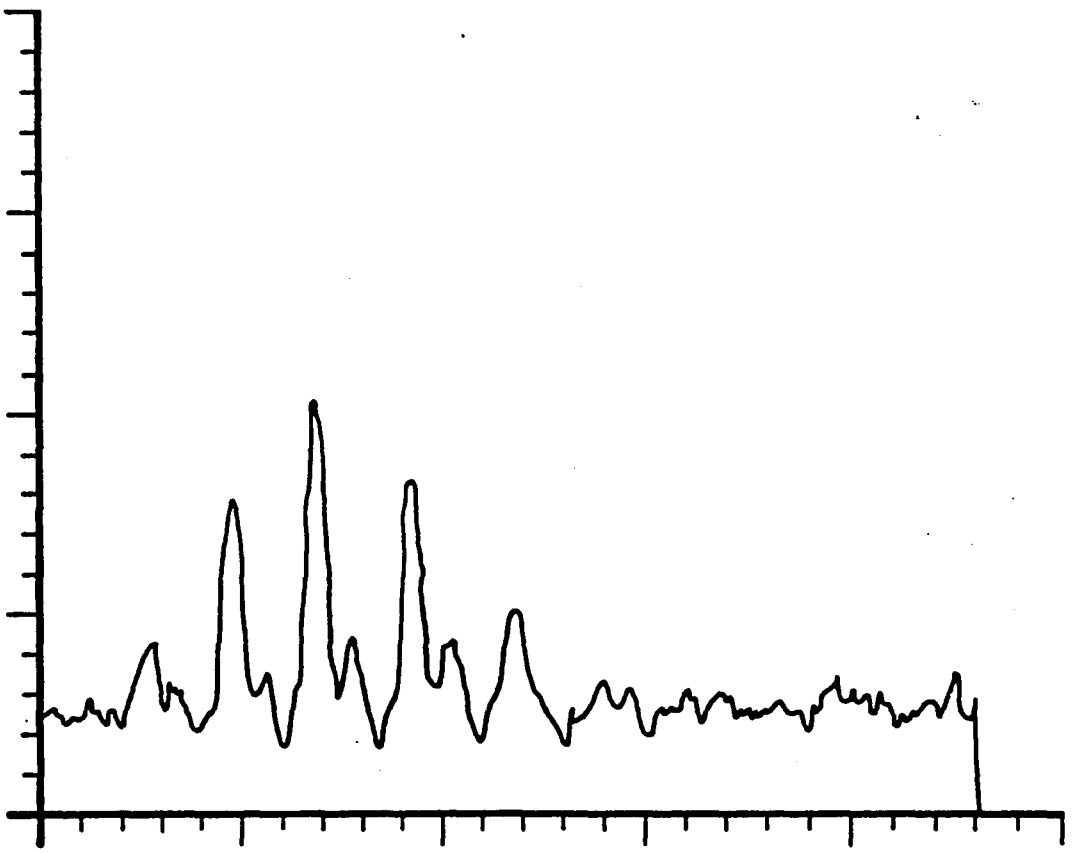
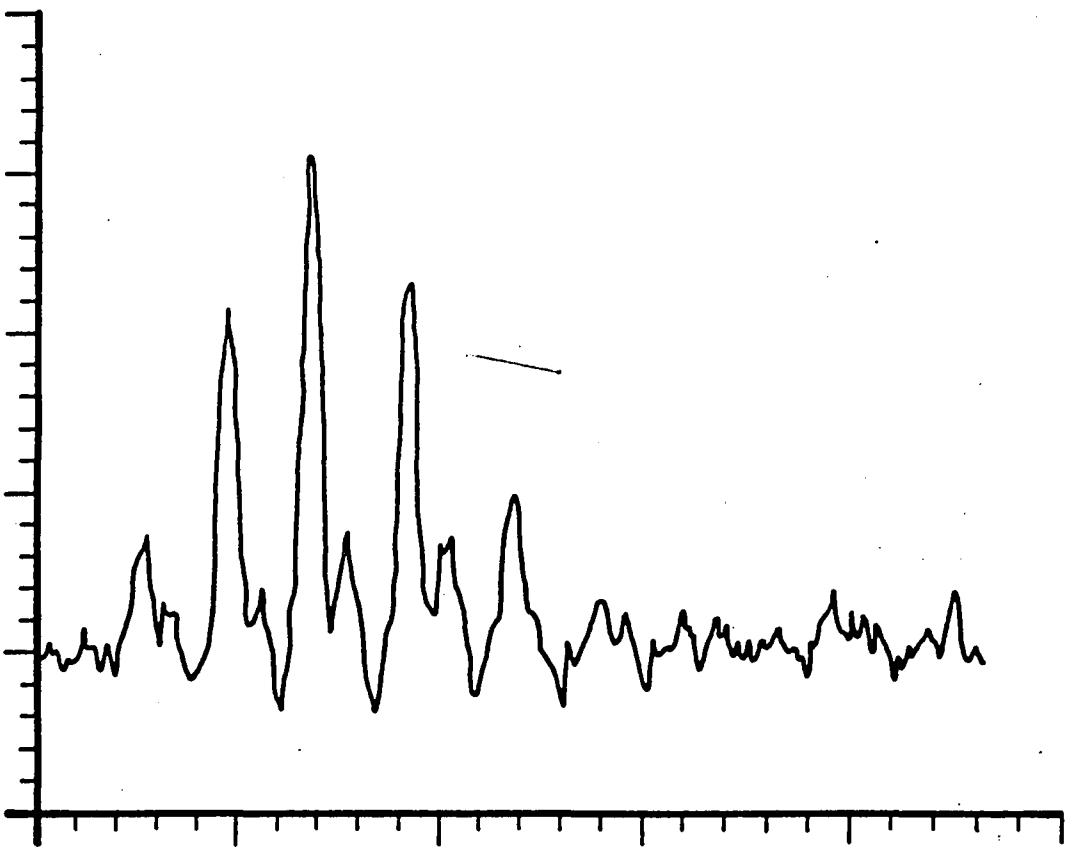


Figure B9. Input amplitude vector (a) and corresponding reconstructed vector (b). (Continued).



(b)

Figure B9. (Concluded).

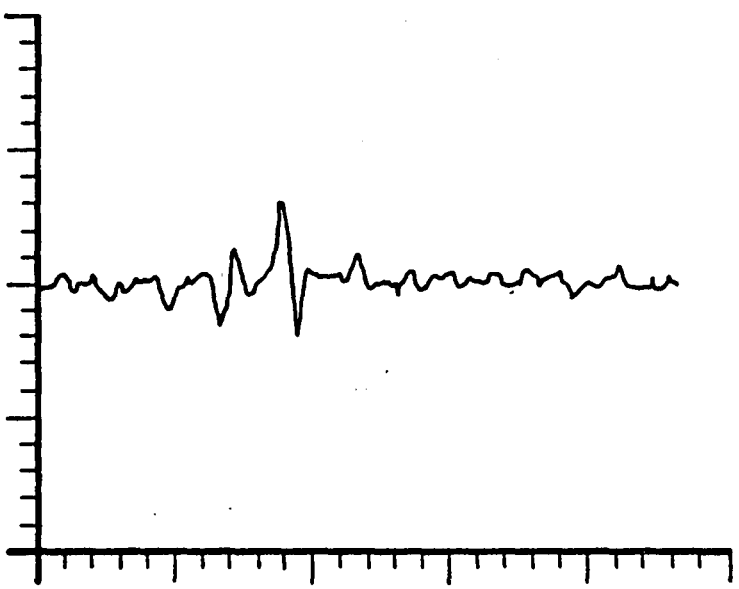
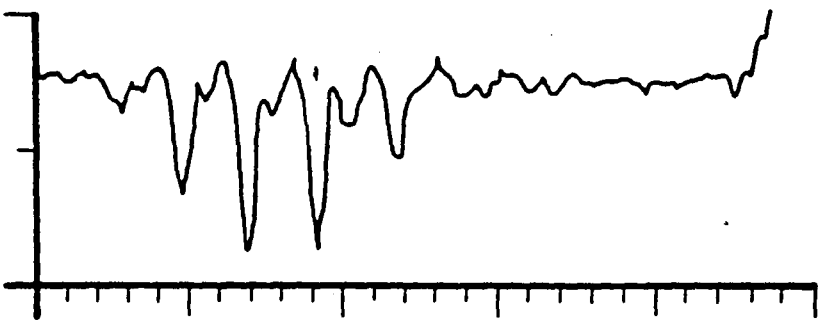
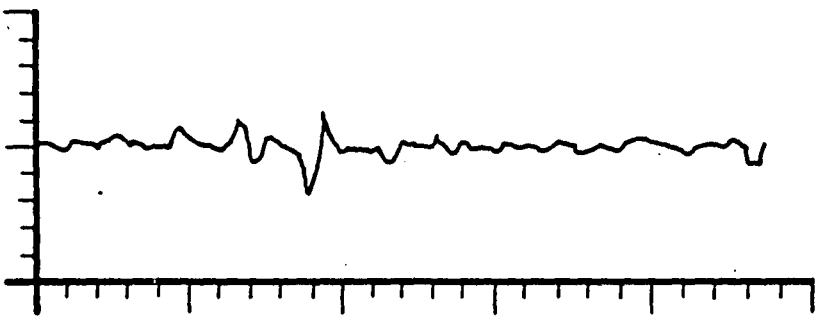


Figure B10. Difference curve for the curves shown in Figures B9(a) and (b).



(a)



(b)

Figure B11. Eigenvectors composing the transformation matrix.

corelation coefficient between the output from the inverse predictor and the input to the predictor were computed and were 4.7206 percent and 0.9945967, respectively. Therefore, from these results, it is believed that the zero-order predictor with zero-tolerance aperture does not affect the error introduced by the KLT.

Conclusions

The goal established for this research was to determine a viable implementation for the Karhunen-Loève Transform. To do this required reduction of the covariance matrix computation time, the eigenvector computation time, and the transformation time. It has been demonstrated that the proposed system meets these goals. One disadvantage to the proposed system is that, in addition to the transformation coefficients, the position vector at the output of the predictor must be kept for synchronization. Therefore, the reduction ratio is lower than when the KLT is used alone. Future research may be directed toward determining whether a set of basis vectors can be computed which would transform the position vector and therefore increase the overall ratio.

Apart from the proposed algorithm, it is strongly believed that a fast implementation for the KLT for general application can be found by studying the properties of the covariance matrix. Since fast implementations to other transforms result by factoring the transformation matrix into Kronecker products of sparse matrices, it is felt that one should concentrate on determining orthogonal similarity transformations to diagonalize the covariance matrix. These similarity transformations should be factored into Kronecker products of sparse matrices and should be easy to determine. With this approach, both the eigenvectors and a fast implementation would be available simultaneously. Until such an algorithm can be established, however, the algorithm proposed by this research offers a possible alternative.

ACKNOWLEDGMENTS

The authors would like to acknowledge the National Aeronautics and Space Administration for the financial support of this research under grant NSG 1379 to Old Dominion University Research Foundation. We would

also like to acknowledge David Livingston and Wallace Harrison for their help in designing and implementing a data acquisition system and also William Dalton for his software expertise.

References

1. Pratt, W.K.: A Comparison of Digital Image Transforms. Proc. Melvin J. Kelley Commun. Conf., 1970, pp. 17-4-1 to 17-4-5.

2. Habibi, Ali; and Wintz, P.A.: Image Coding by Linear Transformation and Block Quantization. IEEE Transactions on Communication Technology, Vol. 19, 1971, pp. 50-62.

3. Campanella, S.J.; and Robinson, Guner S.: A Comparison of Orthogonal Transformations for Digital Speech Processing. IEEE Transactions on Communication Technology, Vol. 19, No. 6, Dec. 1971, pp. 1045-1049.

4. Kortman, C.M.: Redundancy Reduction—A Practical Method of Data Compression. Proc. IEEE, Vol. 55, No. 3, Mar. 1967, pp. 253-263.

5. Dettman, John W.: Mathematical Methods in Physics and Engineering. McGraw Hill Book Co. (N.Y.), 1969.

6. Andrews, Harry C.: Computer Techniques in Image Processing. Academic Press (N.Y.), 1970, pp. 73-179.

7. Ahmed, N.; and Rao, K. R.: Orthogonal Transforms for Digital Signal Processing. Springer-Verlag (N.Y.), 1975.

8. Beauchamp, K.G.: Walsh Functions and Their Applications. Academic Press, 1975.

9. Jain, Anil K.: A Fast Karhunen-Loève Transform for a Class of Random Signals. IEEE Transactions on Communications, Sept. 1976, pp. 1023-1029.

10. Jain, Anil K.: A Fast Karhunen-Loève Transform for Digital Restoration of Images Degraded by White and Colored Noise. IEEE Transactions on Computers, Vol. C-26, June 1977, pp. 560-571.

11. Shanmugan, K.; and Harlick, R.M.: A Computationally Simple Procedure for Imagery Data Compression by the Karhunen-Loève Method. IEEE Transactions on Aerospace and Electronic Systems, Vol. AE59, 1973, p. 813.

12. Ray, W.D.; and Driver, R.M.: Further Decomposition of the Karhunen-Loève Series, Representation of a Stationary Random Process. IEEE Proceedings on Information Theory, Vol. IT-16, No. 6, Nov. 1970.

13. Good, I.J.: The Interaction Algorithm and Practical Fourier Analysis. J. Roy. Statist. Soc. (London), Vol. B20, 1958, p. 361.

14. Lawrence, L.R.; and Sheena, D.: Survey of Eye Movement Recording Methods. Behavior Research Methods and Instrumentation. Vol. 7(5), 1975, pp. 397-429.

15. The Honeywell Mark 3 Remote Oculometer Operating and Maintenance Manual, Vols. 1, 2, 3. Honeywell Radiation Center (Lexington, Mass.), March 1976.

16. Livingston, D.L.: A High-Speed Algorithm Processor for Signal Processing Applications. Master's Thesis, Dept. of Electrical Engineering, Old Dominion Univ., 1978.

17. Rao, K.R.; and Ahmed, N.: The Generalized Transform. Proceedings 1971 Symposium on Applications of Walsh Functions, pp. 60-67.

18. Weber, D.R.: A Synopsis on Data Compression. IEEE Proceedings of National Telemetry Conference, 1965, pp. 9-16.

19. Andrews, C.A.; Davies, J.M.; Schwarz, G.R.: Adaptive Data Compression. Proceedings of the IEEE, Vol. 55, No. 3, Mar. 1967, pp. 267-277.

20. Davisson, Lee D.; and Gray, Robert M.: Data Compression. Dowden, Hutchinson & Ross, Inc. (Stroudberg, Penn.), 1976.

21. Cooper, George R.; and McGillem, Clare D.: Probabilistic Method of Signal and System Analysis. Holt, Rinehart and Winston, Inc. (N.Y.), 1971.

APPENDIX C

CIRCUIT DIAGRAMS

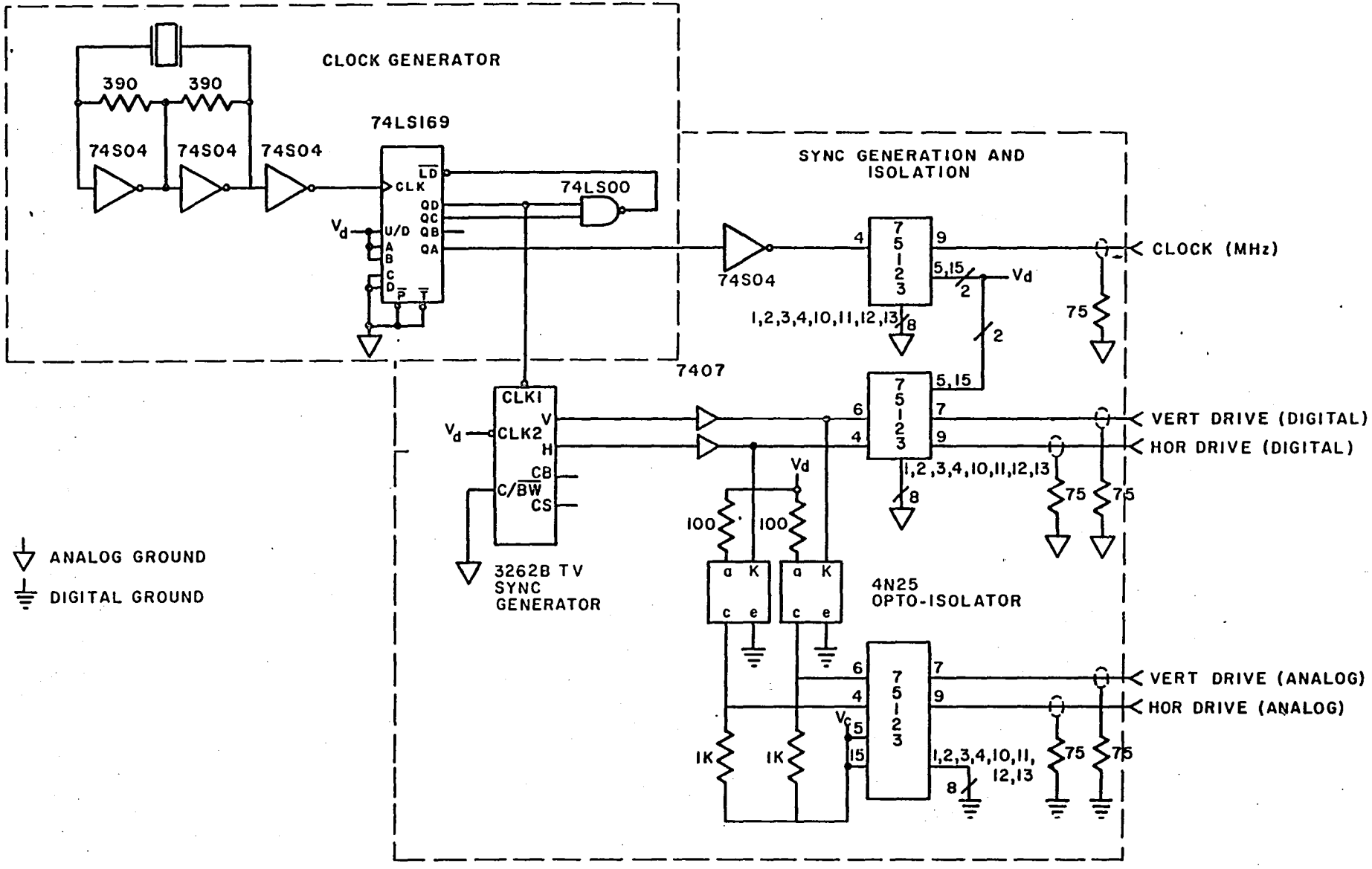


Figure C1. Synchronization subsystem.

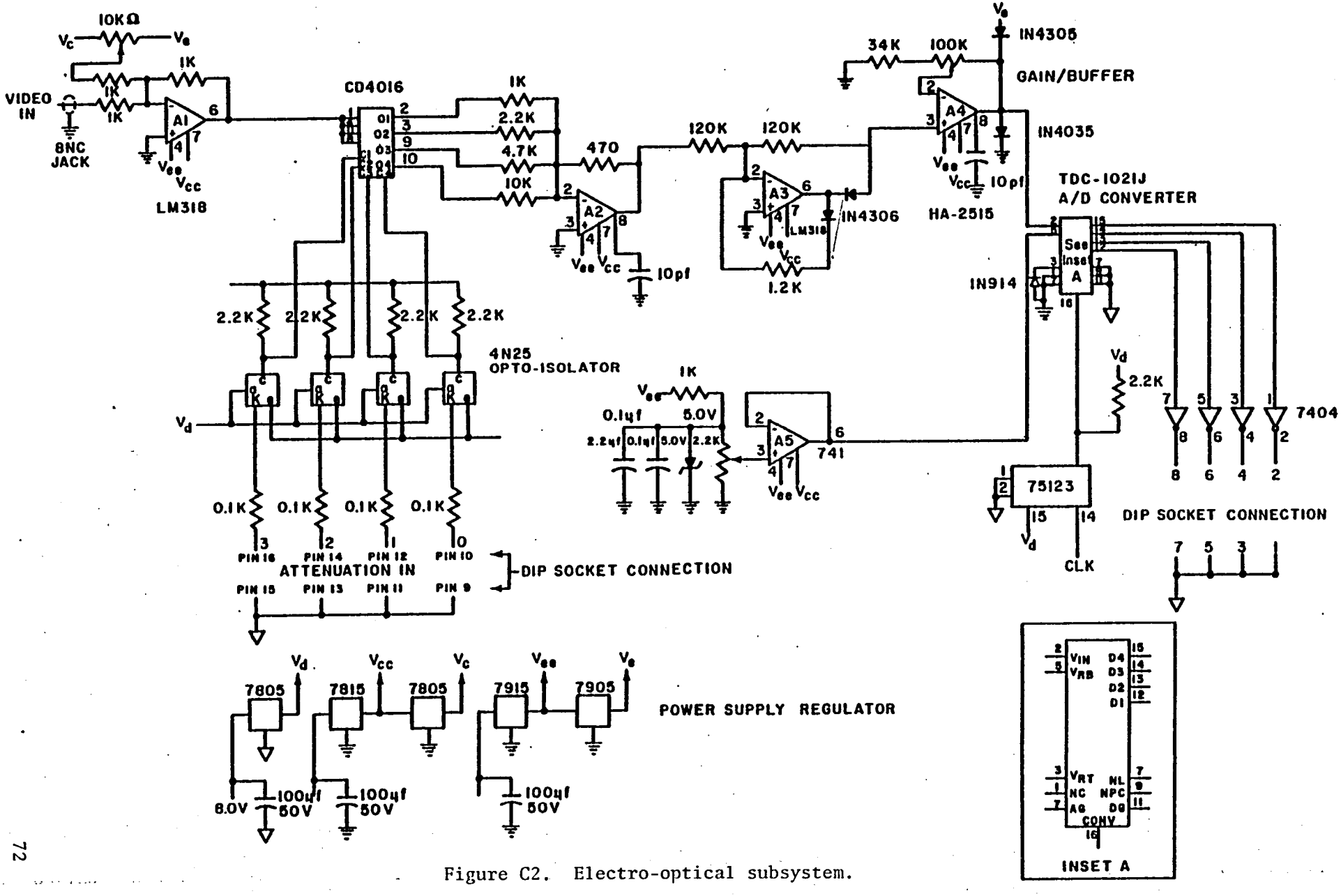


Figure C2. Electro-optical subsystem.

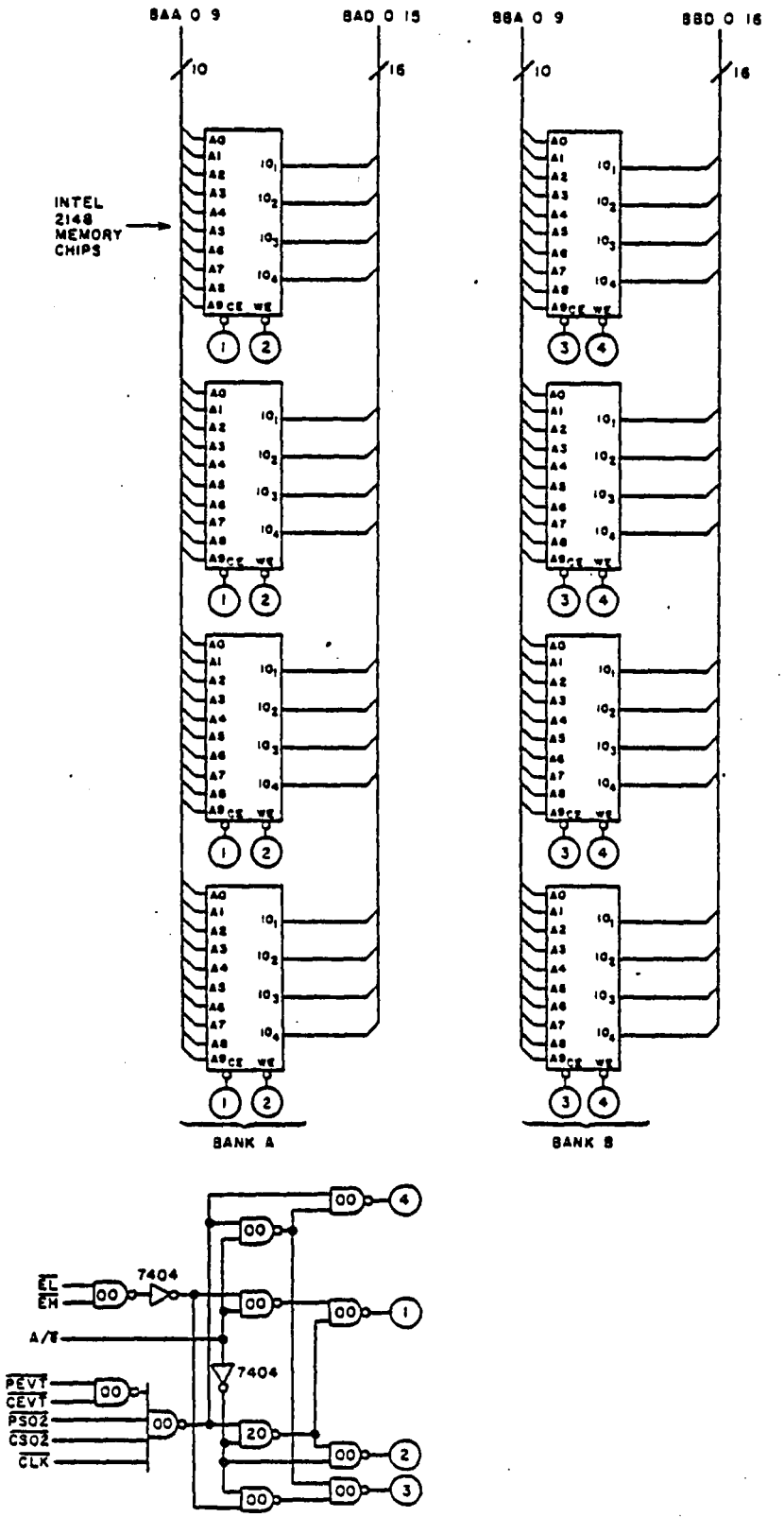


Figure C3. Memory block and memory cycle address logic.

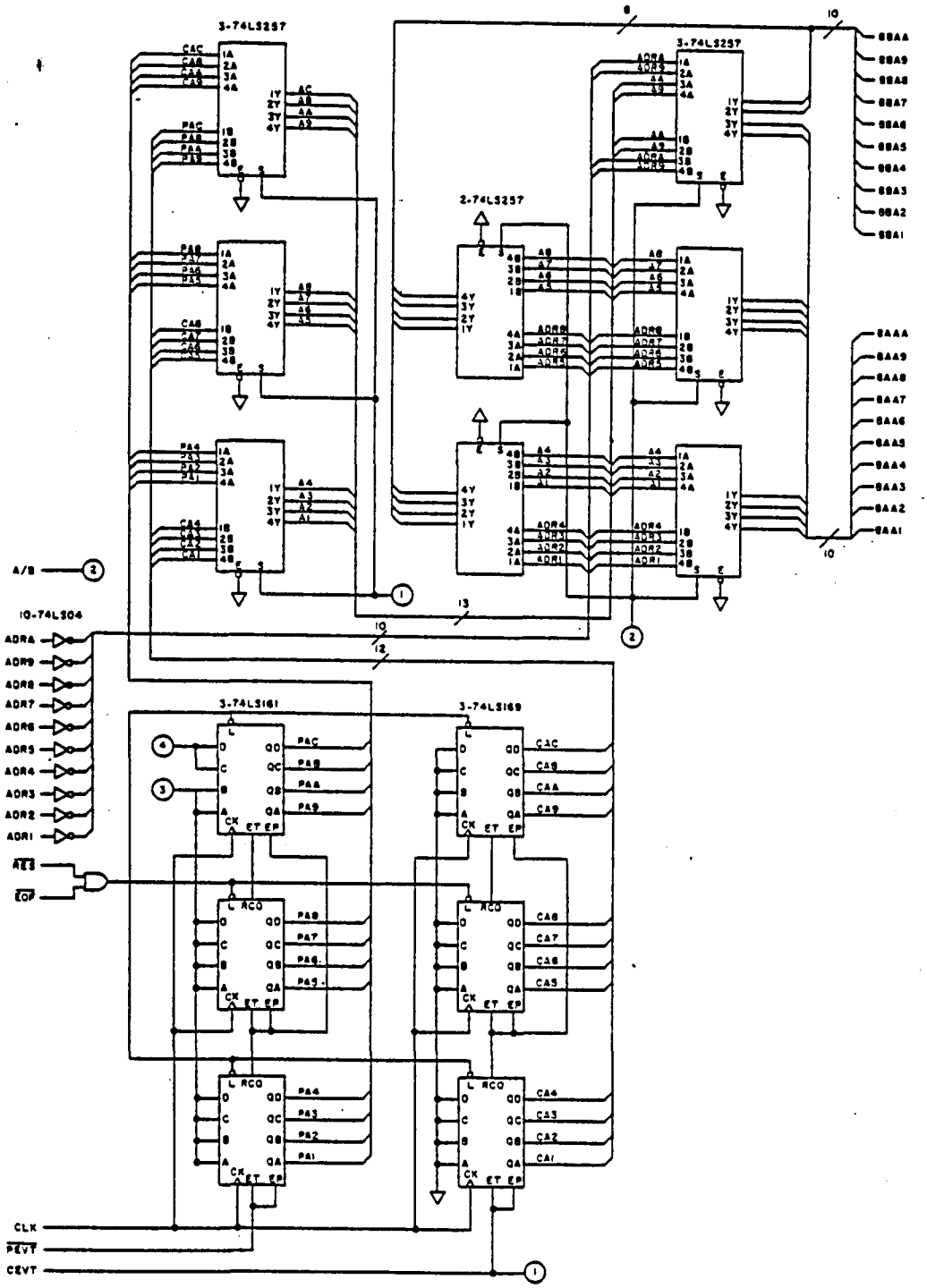


Figure C4. Select 2 and DMA address generator.

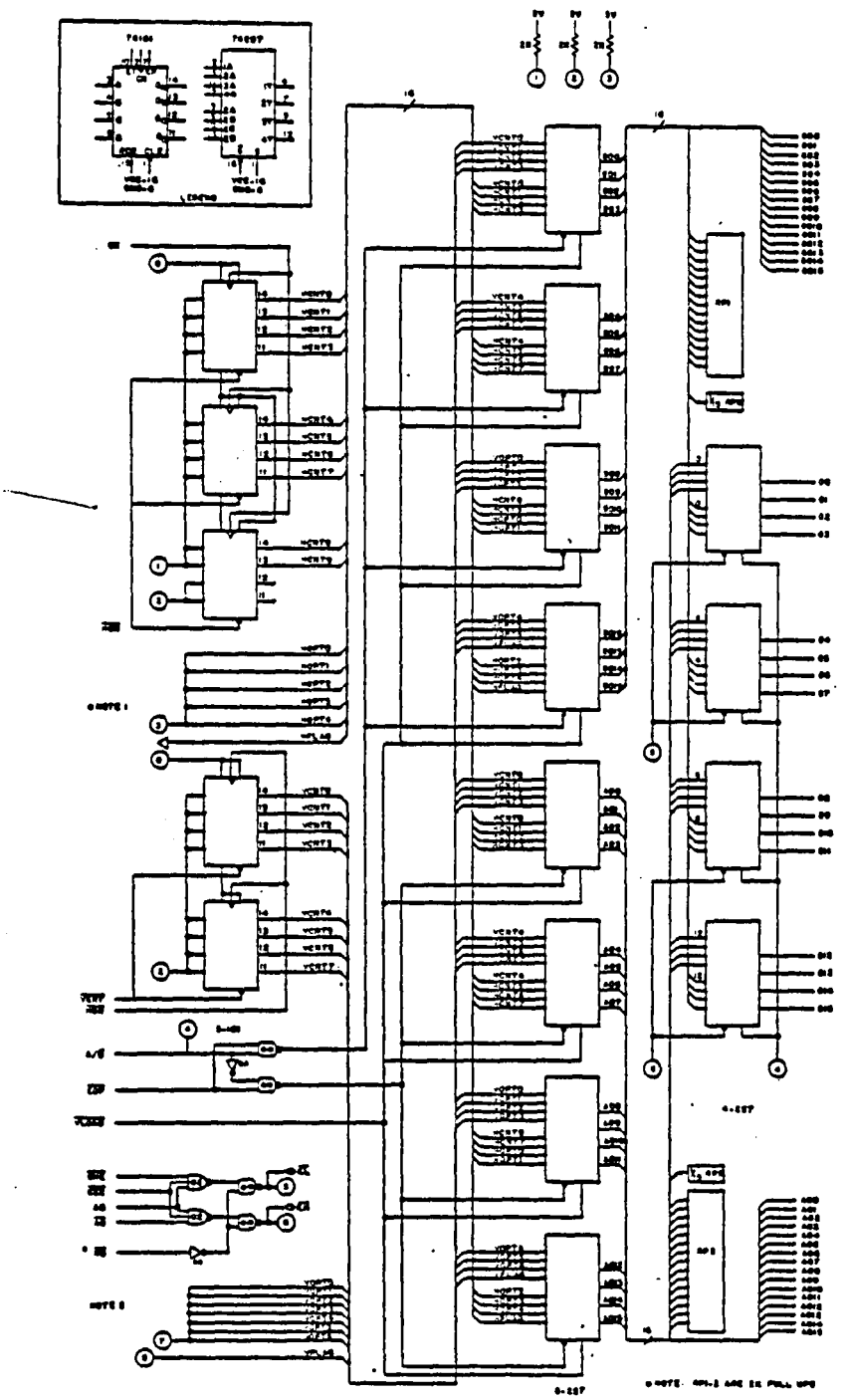


Figure C5. Horizontal and vertical counters and select 1 module.

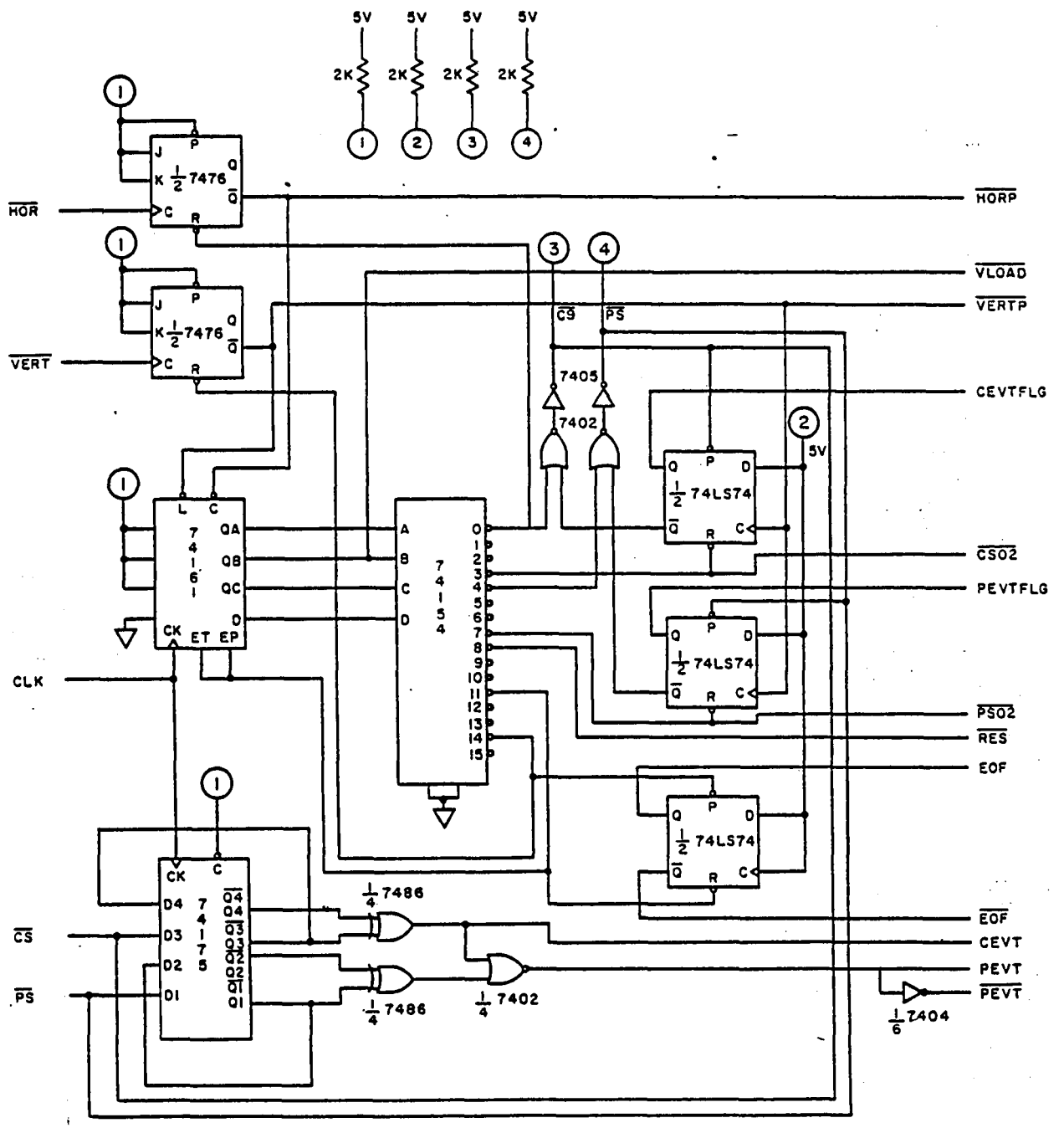


Figure C6. EOL/EOT and event generators.

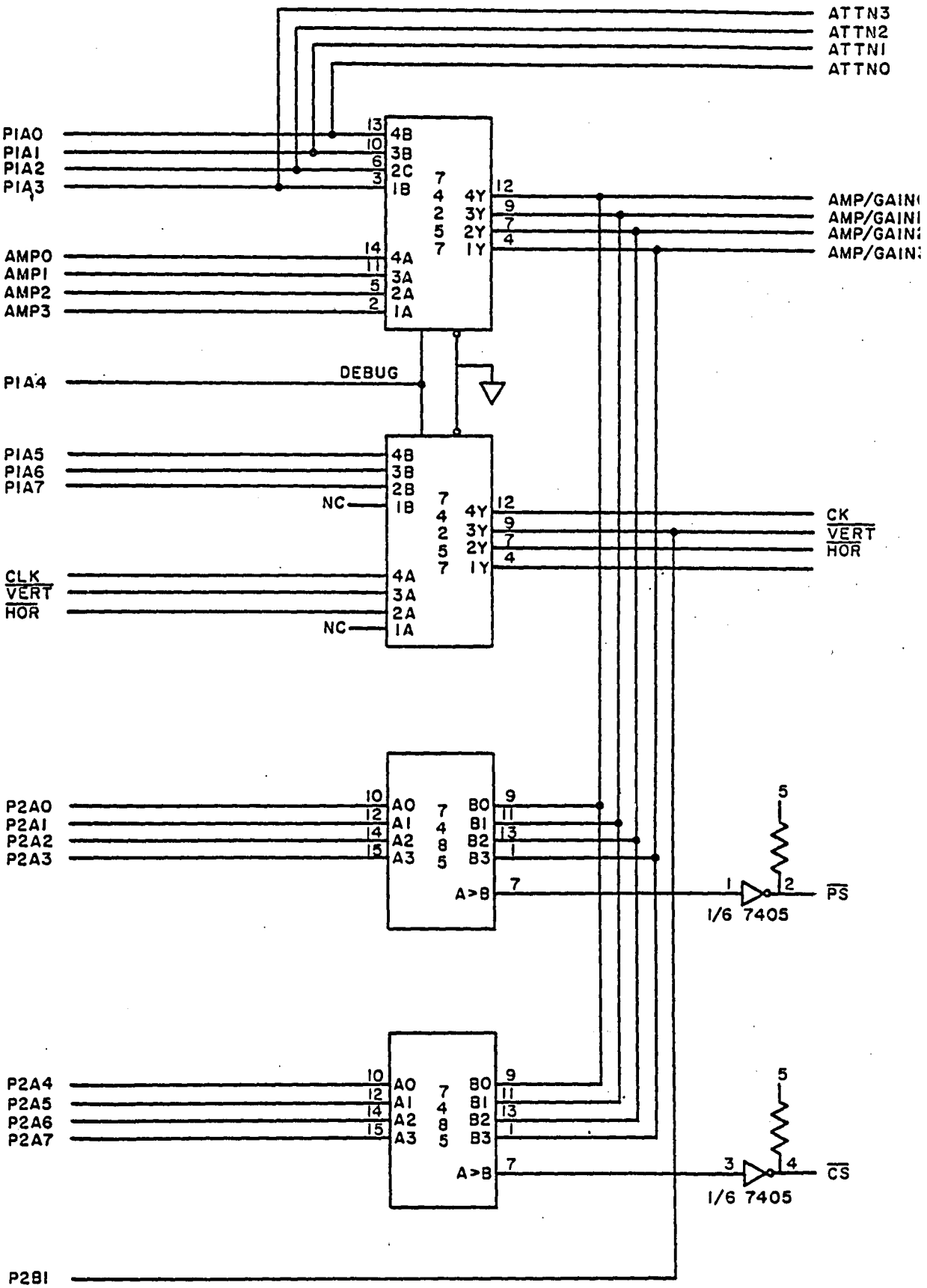


Figure C7. Debug control and threshold detection.

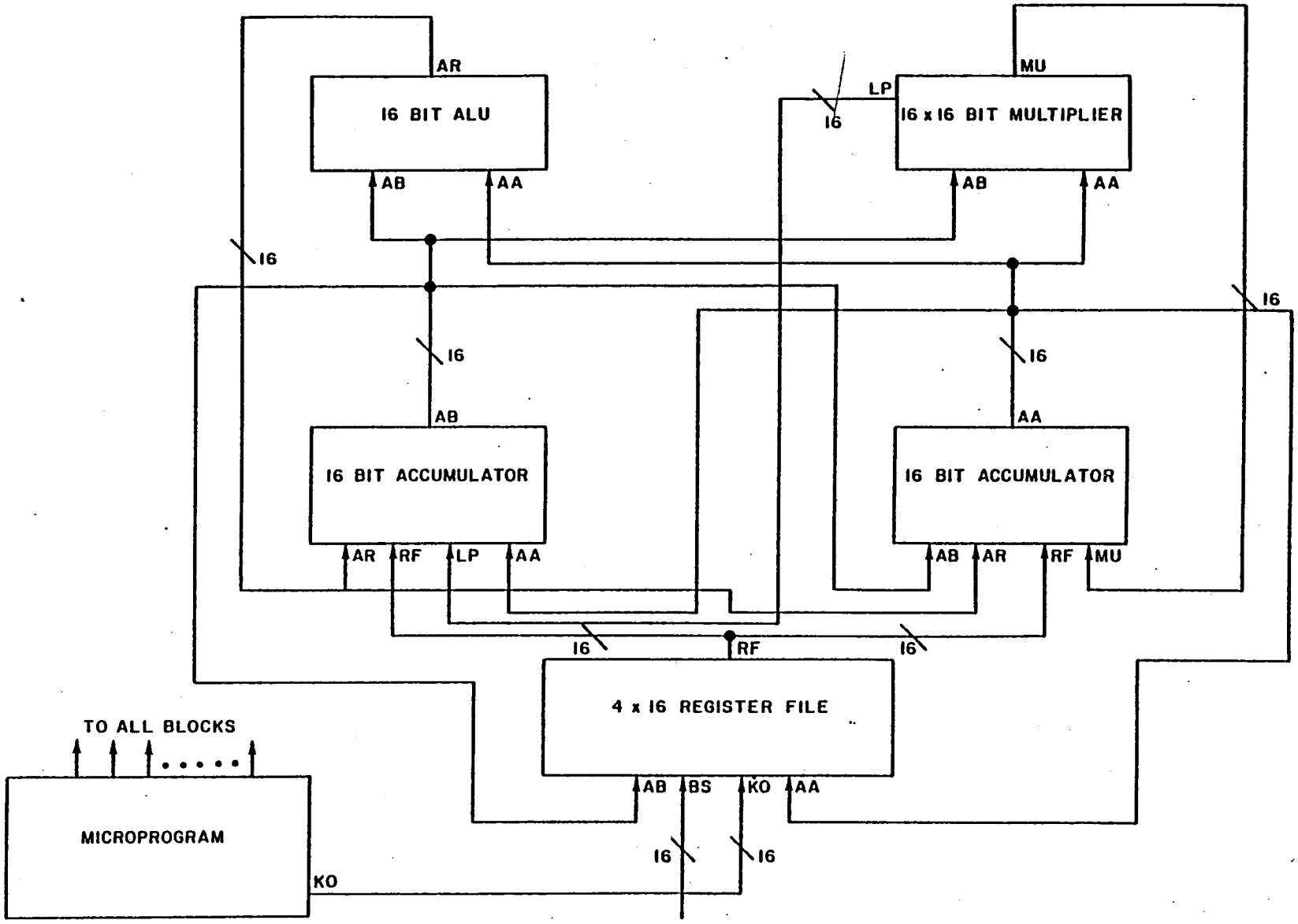


Figure C8. Block structure of high-speed ALU.

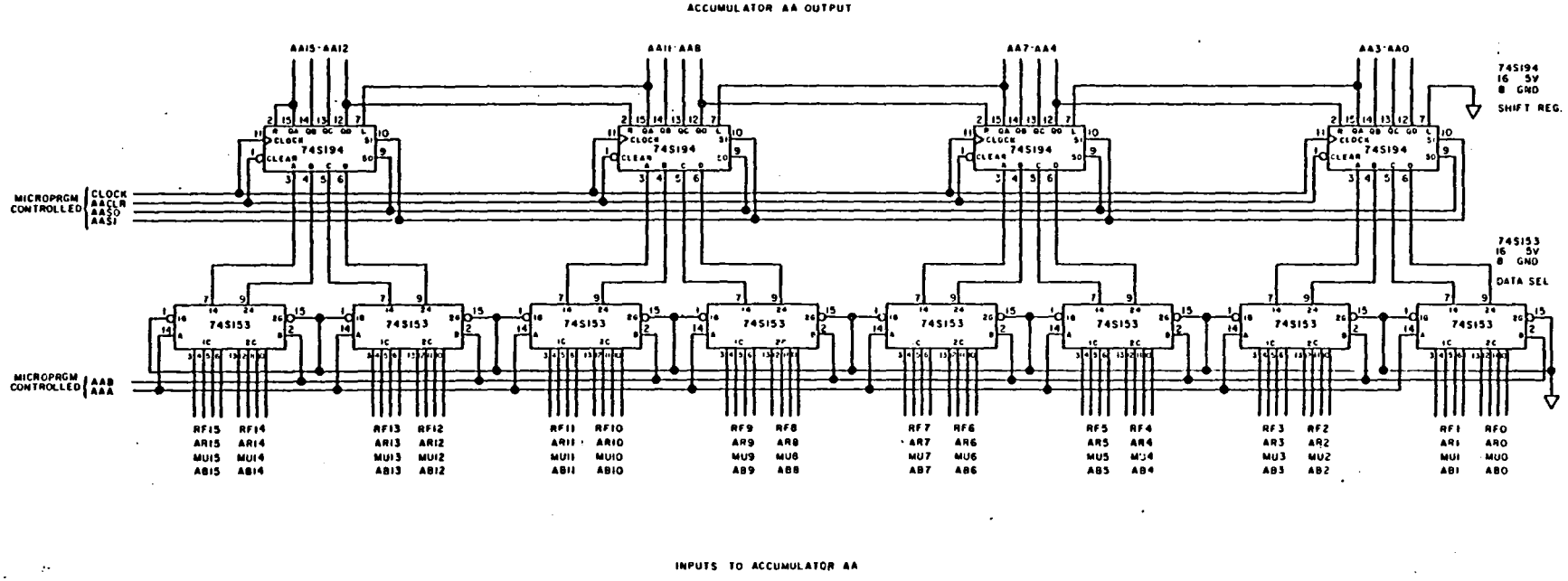


Figure C9. Accumulator AA.

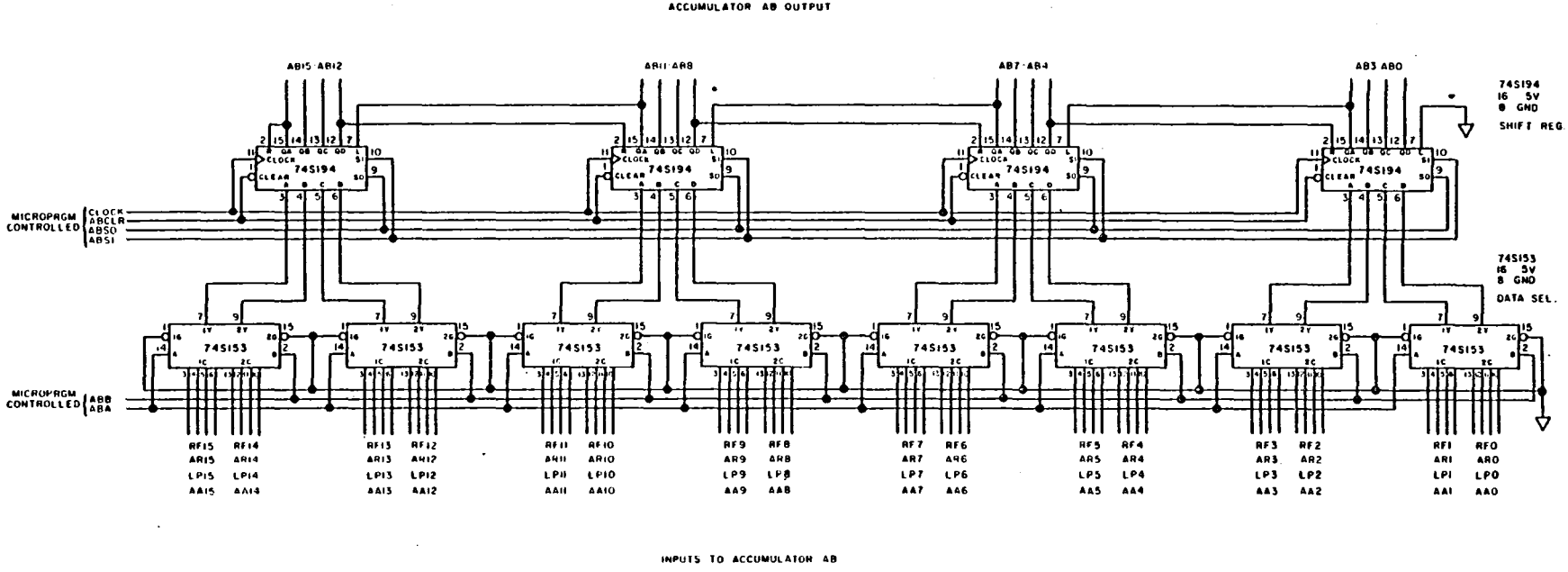


Figure C10. Accumulator AB.

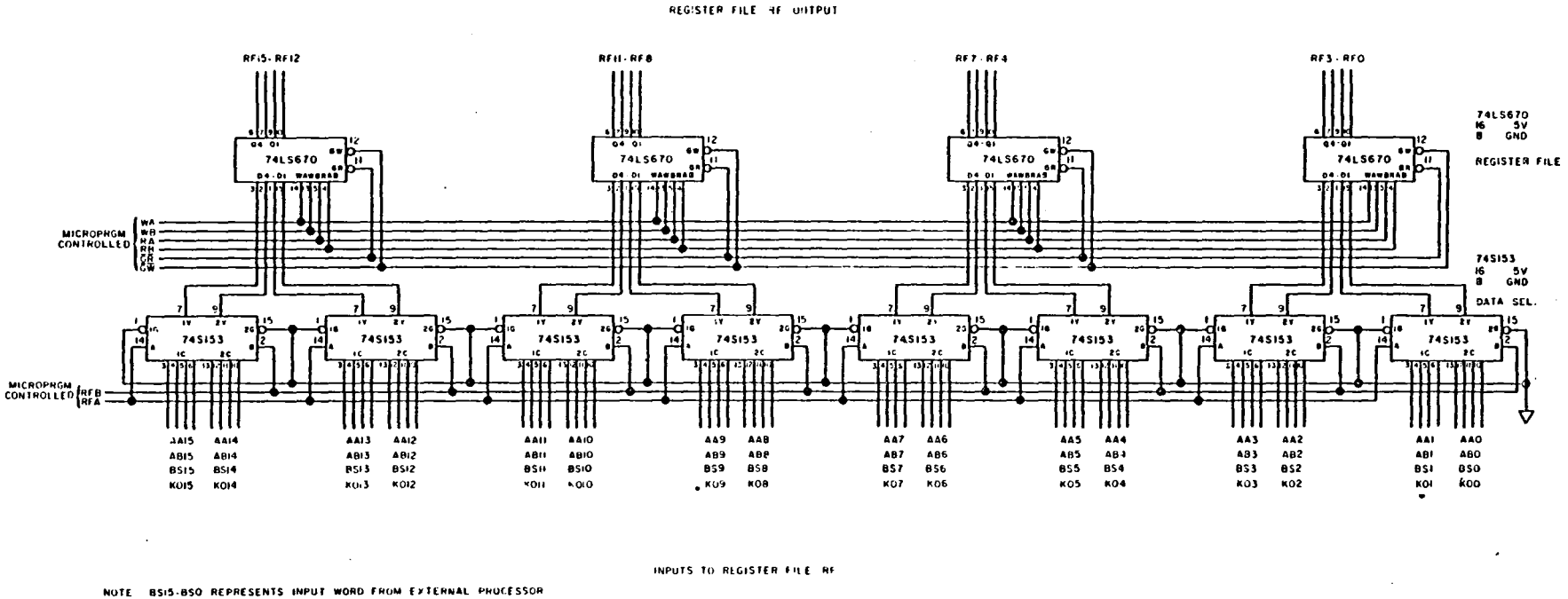


Figure C11. Register file.

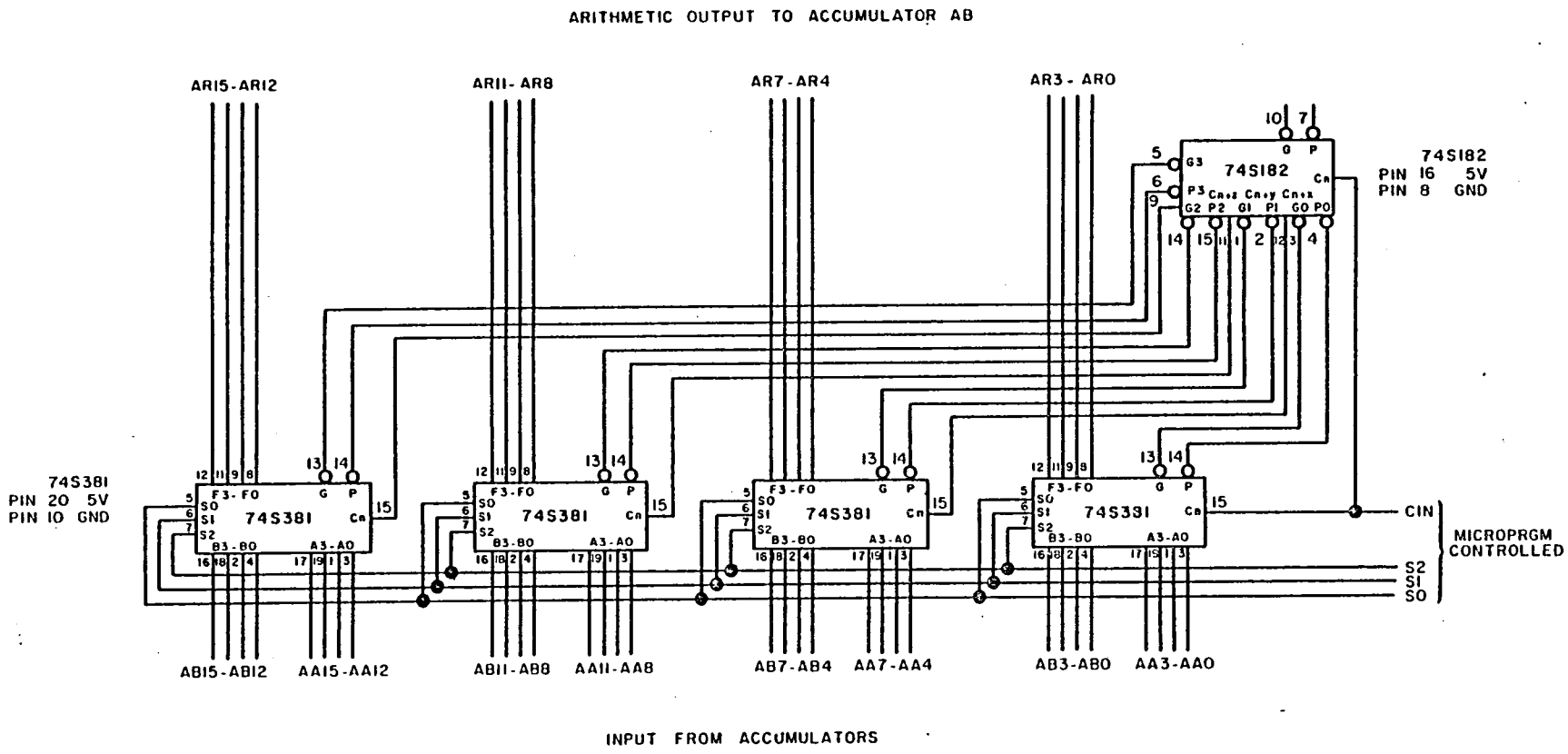


Figure C12. ALU.

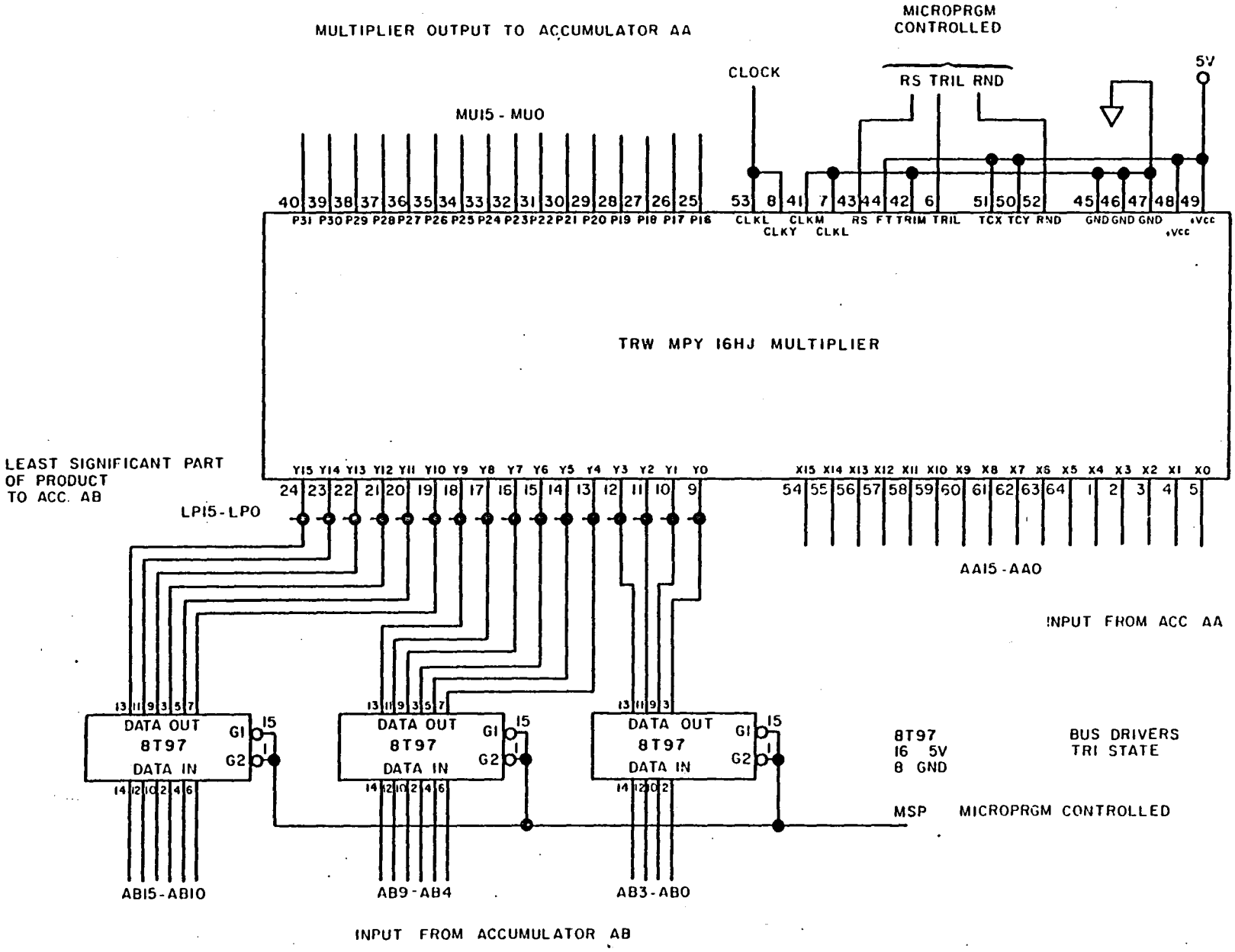


Figure C13. Multiplier.

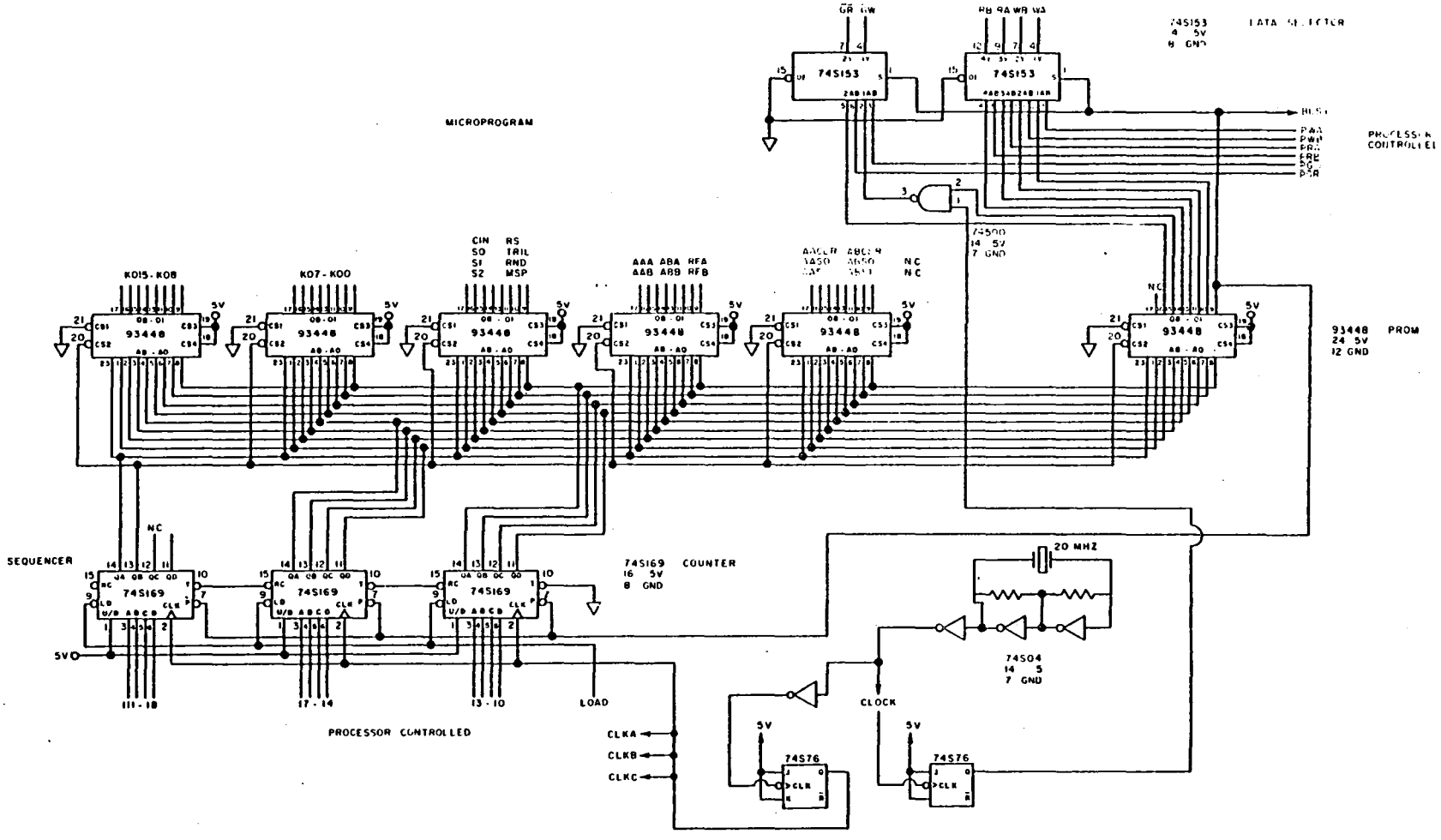


Figure C14. Controller.

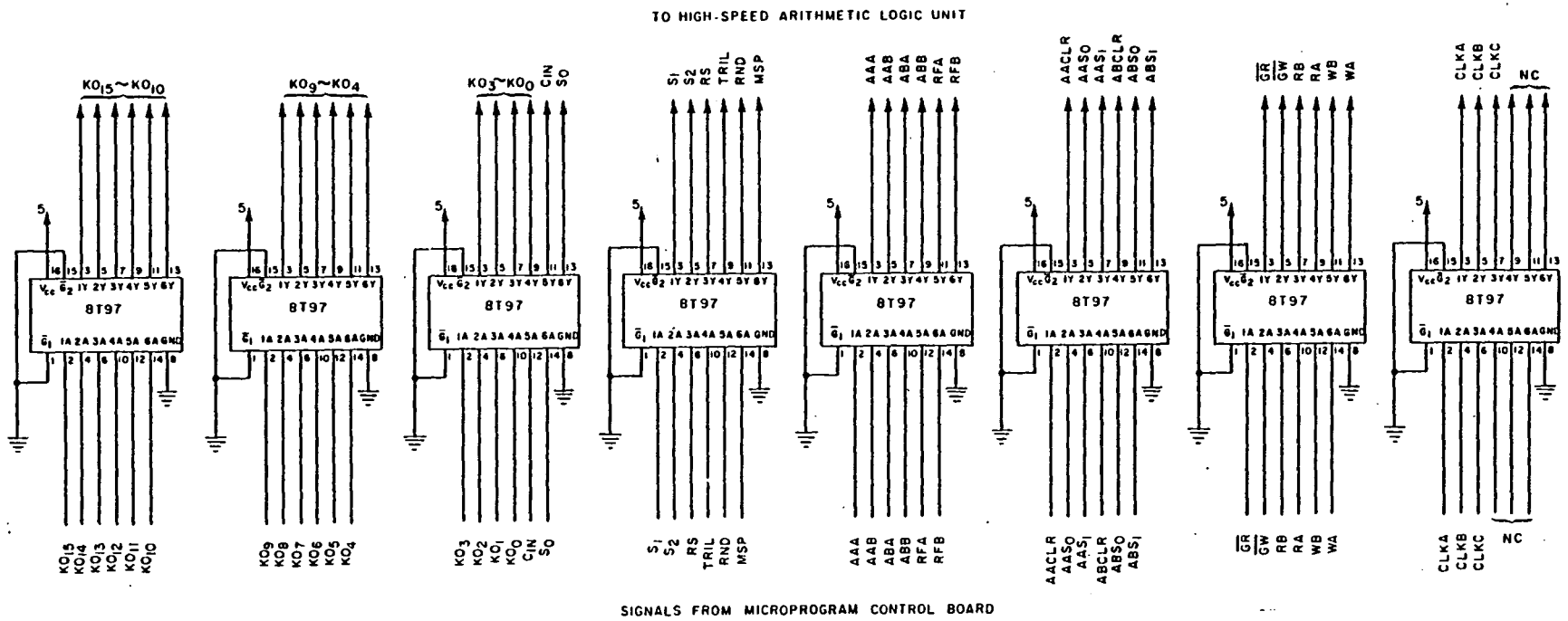


Figure C15. Buffers.

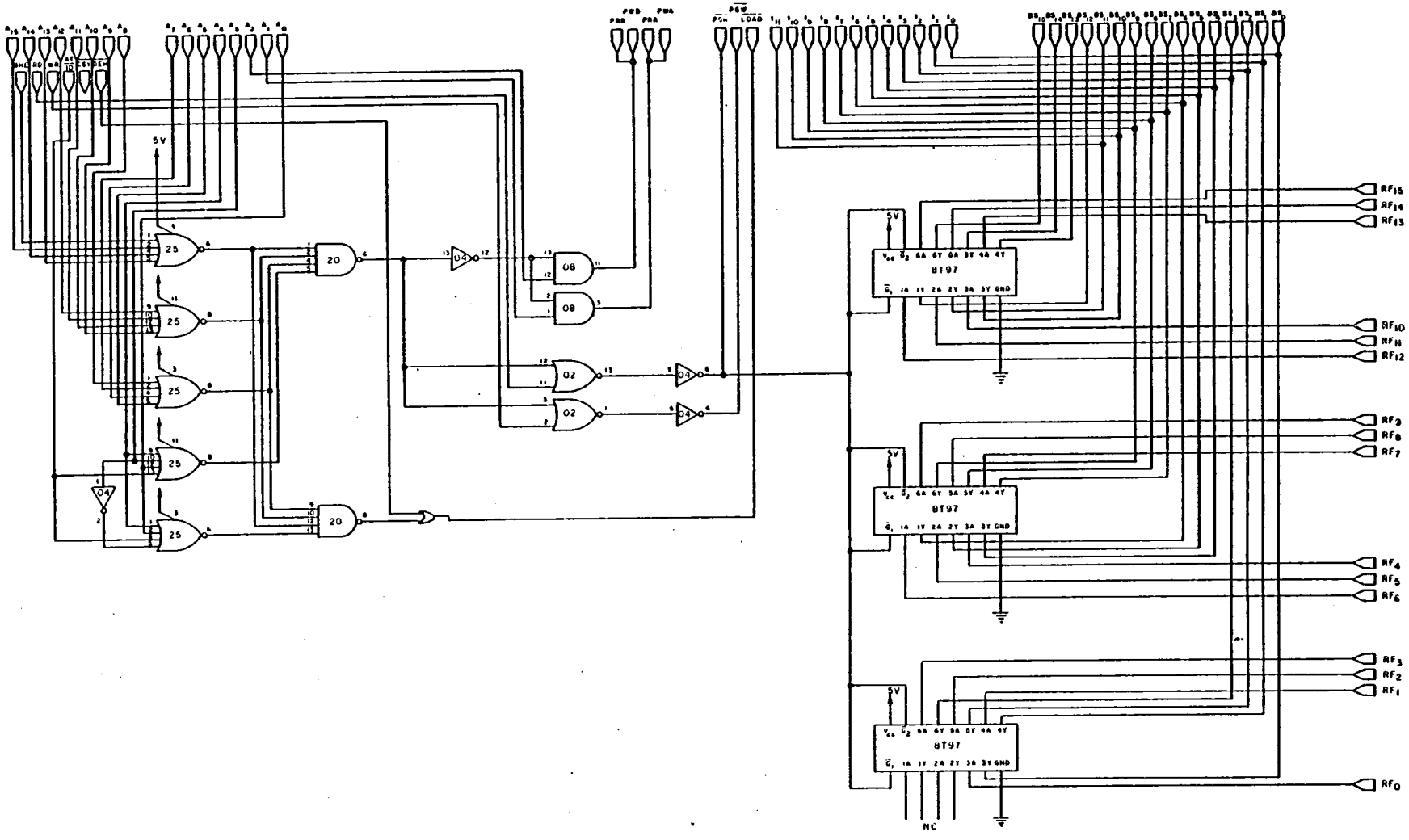


Figure C16. HSAP Interface.

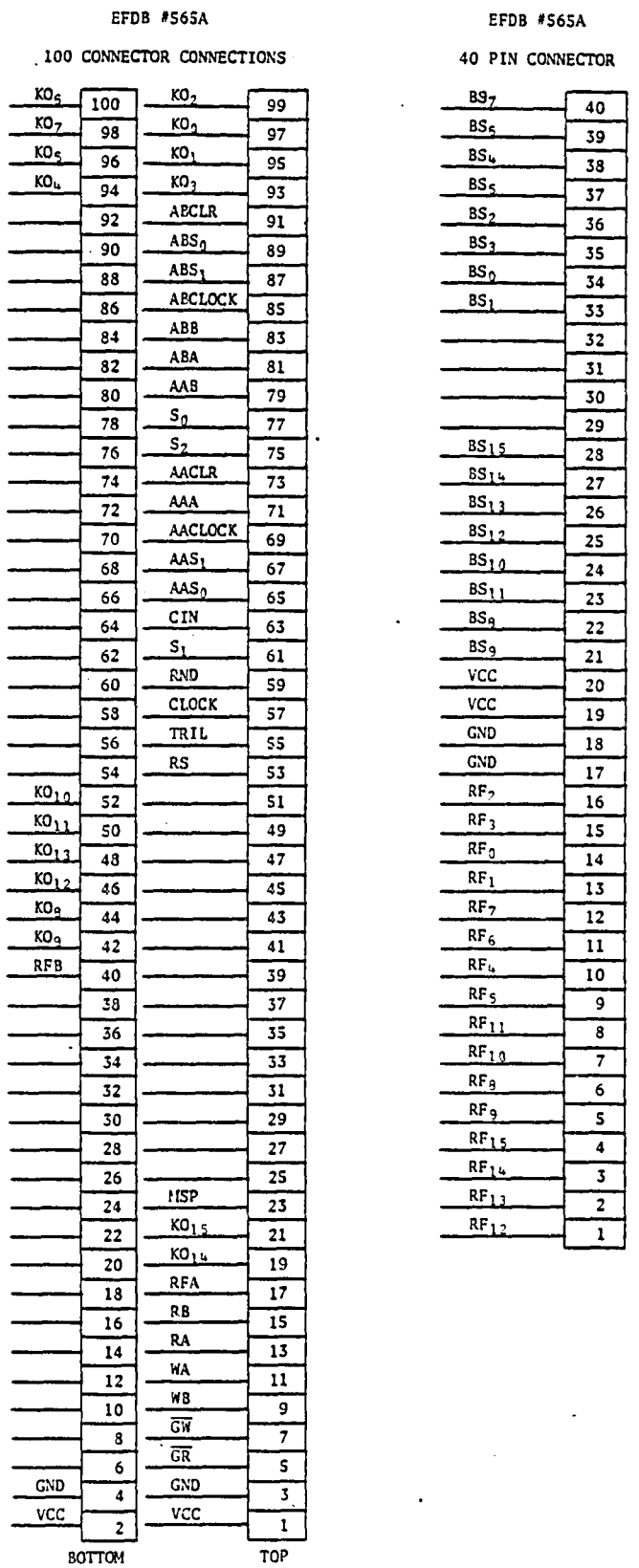


Figure C17. Edge connections.

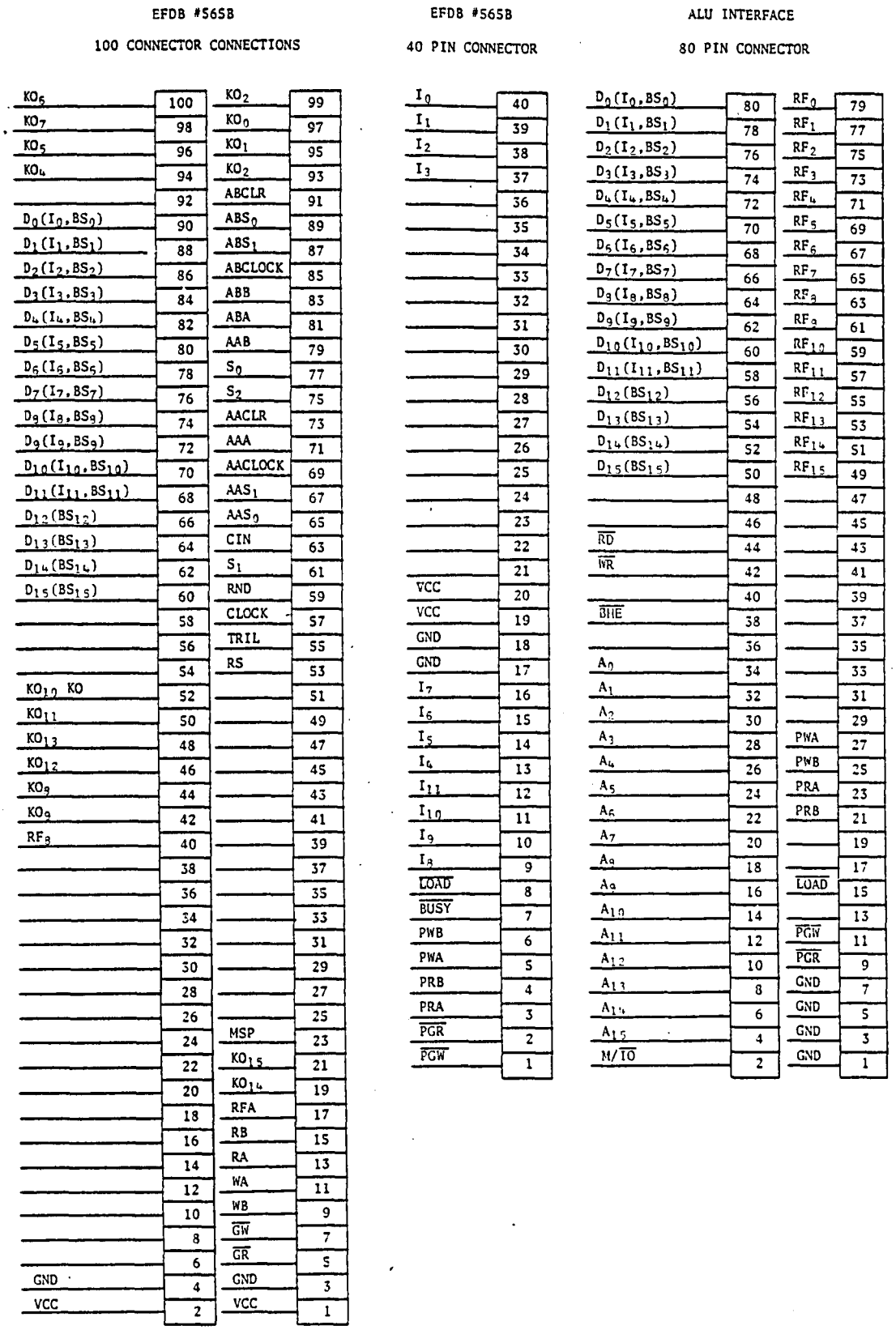


Figure C18. Edge connections.

	K O K O K O K O K O K O K O K O 15 14 13 12 11 10 9 8	K O K O K O K O K O K O K O K O 7 6 5 4 3 2 1 0	C IN S_0 S_1 S_2 R S T R L R N N I S P	A A A B B A B B R F A R F B N C N C	$\overline{A}$ A A $\overline{B}$ B B N C N C C L R S_0 S_1 C U R S_0 S_1	N C $\overline{G}$ R G W R B R A W B N A $\overline{B}$	$S_0S_1S_2$	
A ← B			0 0	0 0	1 1 Ø Ø	1 0 0	0 0 0	AR = $\overline{0}$
A ← u			+ 0 + 0	1 0	1 1 Ø Ø	1 0 0	0 0 1	AR = B - A
A ← R(n)			0 0	1 1	1 1 Ø Ø	0 0 * * 0	0 1 0	AR = A - B
A ← AR			* * * * 0 0	0 1	1 1 Ø Ø	1 0 0	0 1 1	AR = A + B
B ← A			0 0	0 0	Ø Ø 1 1	1 0 0	1 0 0	AR = A ⊕ B
B ← L			+ 1 + 1	1 0	Ø Ø 1 1	1 0 0	1 0 1	AR = A or B
B ← R(n)			0 0	1 0	Ø Ø 1 1	0 0 * * 0	1 1 0	AR = A ↯ B
B ← AR			* * * * 0 0	0 1	Ø Ø 1 1	1 0 0	1 1 1	AR = T
R(n) ← A			0 0	1 1	Ø Ø Ø Ø	1 1 * * 0		
R(n) ← B			0 0	0 1	Ø Ø Ø Ø	1 1 * * 0		
R(n) ← BUS			0 0	1 0	Ø Ø Ø Ø	1 1 * * 0		
R(n) ← EMIT	* * * * * * * *	* * * * * * * *	0 0	0 0	Ø Ø Ø Ø	1 1 * * 0	S_1S_0	NOP
A ← SR(A)			0 0		1 0 Ø Ø	1 0 0	0 0	RS
A ← SL(B)			0 0		0 1 Ø Ø	1 0 0	0 1	LS
B ← SR(B)			0 0		Ø Ø 1 0	1 0 0	1 0	LD
B ← SL(B)			0 0		Ø Ø 0 1	1 0 0	1 1	
HLT			0 0	1 0	0 0 0 0	1 0 1		

Figure C19. Micro instructions.

End of Document