

2

DTIC FILE COPY

TECHNICAL REPORT RD-AS-87-18



HYBRID GRAY SCALE OPTICAL/DIGITAL CORRELATOR

AD-A196 300

S. Richard F. Sims
John L. Johnson
Advanced Sensors Directorate
Research, Development, and
Engineering Directorate

DTIC
ELECTE
JUN 01 1988
S D

OCTOBER 1987



U.S. ARMY MISSILE COMMAND

Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.

88 6 1 045

DISPOSITION INSTRUCTIONS

**DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT
RETURN IT TO THE ORIGINATOR.**

DISCLAIMER

**THE FINDINGS IN THIS REPORT ARE NOT TO BE CONSTRUED AS AN
OFFICIAL DEPARTMENT OF THE ARMY POSITION UNLESS SO DESIGNATED
BY OTHER AUTHORIZED DOCUMENTS.**

TRADE NAMES

**USE OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES
NOT CONSTITUTE AN OFFICIAL INDORSEMENT OR APPROVAL OF
THE USE OF SUCH COMMERCIAL HARDWARE OR SOFTWARE.**

AD-A196300

REPORT DOCUMENTATION PAGE				Form Approved OMB No 0704-0188 Exp. Date Jun 30, 1986	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR-RD-AS-87-18			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Advanced Sensors Directorate Res, Dev, and Engr Ctr		6b. OFFICE SYMBOL (if applicable) AMSMI-RD-AS-SS	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State, and ZIP Code) Commander U.S. Army Missile Command ATTN: AMSMI-RD-AS-SS Redstone Arsenal, AL 35898-5253			7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State, and ZIP Code)			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
11. TITLE (Include Security Classification) Hybrid Gray Scale Optical/Digital Correlator					
12. PERSONAL AUTHOR(S) S. Richard F. Sims, John L. Johnson					
13a. TYPE OF REPORT FINAL		13b. TIME COVERED FROM Fall 85 TO Fall 87		14. DATE OF REPORT (Year, Month, Day) October 1987	
15. PAGE COUNT 112					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Optical/Digital Correlators, Pattern Recognition Incoherent Correlators, Image Processing, Hybrid Image Processing for Pattern Recognition		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The report describes the architecture and major components of a hybrid optical/digital correlator which makes use of the reference gray scale for image pattern recognition. The specifics of the electronic/optical design are presented in schematic diagrams. Software listings are included for system test and alignment including a host interface. Operational notes are included.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL S. Richard F. Sims			22b. TELEPHONE (Include Area Code) (205) 876-1648		22c. OFFICE SYMBOL AMSMI-RD-AS-SS

ACKNOWLEDGEMENT

The authors would like to recognize the significant contributions of Tom Stroud and Allen Davis in layout, fabrication, and debugging of this testbed.



Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification _____	
By _____	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION.....	1
II. SYSTEM ARCHITECTURE.....	1
A. Video Source.....	1
B. Acoustooptic Cell.....	1
C. Output Camera Video.....	13
D. A/D Converter Module.....	13
E. Reference Memory.....	13
F. Digital to Analog Driver (D/A), LED, Fiber Optics.....	13
III. GENERAL DESCRIPTION AND OPERATION.....	14
IV. ALIGNMENT OF LED GAIN AND BIAS.....	15
V. OPTICAL LAYOUT AND SPECS.....	16
VI. CONCLUSION.....	16
REFERENCE.....	21
APPENDIX A. BAR.FOR.....	A-1
APPENDIX B. BAR2.FOR.....	B-1
APPENDIX C. CDATEST.FOR.....	C-1
APPENDIX D. CFREQTEST.FOR.....	D-1
APPENDIX E. CMEMDUMP.FOR.....	E-1
APPENDIX F. CMEMTEST.FOR.....	F-1
APPENDIX G. CREADI.....	G-1
APPENDIX H. CRWIMAGE.FOR.....	H-1
APPENDIX I. DELTA.FOR.....	I-1
APPENDIX J. DELTA2.FOR.....	J-1
APPENDIX K. IMAGEH.FOR.....	K-1
APPENDIX L. IMAGEHDEMO.FOR.....	L-1
APPENDIX M. RWCMENT.FOR.....	M-1
APPENDIX N. SELDAT.FOR.....	N-1
APPENDIX O. SEQRAMP.FOR.....	O-1
APPENDIX P. SEQUENCE.FOR.....	P-1
APPENDIX Q. SHORTLINE.FOR.....	Q-1
APPENDIX R. SSEL DAT.FOR.....	R-1
APPENDIX S. SSEL DAT.FOR.....	S-1
APPENDIX T. SSEQUENCE.FOR.....	T-1
APPENDIX U. SWITCHTEST.FOR.....	U-1

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1	Hybrid gray scale optical/digital correlator-functional block diagram.....	2
2	Optical bench.....	3
3	Electronic rack (top view).....	4
4	Electronic rack (front view).....	5
5	Electronic rack layout with front panel.....	6
6	Digital control card (CORR).....	7
7	Digital control card (VERT).....	8
8	Digital control card (CORRCON).....	9
9	ADC/Video processing.....	10
10	Correlator reference memory.....	11
11	LED driver card.....	12
12	Incoherent correlator optics.....	17
13	Half frequency test.....	19
14	Maximum frequency test.....	19
15	Ramp test 0-2.55.....	19

I. INTRODUCTION

The two dimensional optical processing techniques using one dimensional input devices discussed by Demetri Psaltis [1,2] in several of his papers were used as a starting point for the following hybrid design. Graeme Duthie's knowledge of the Psaltis research instigated this work while he was at the U.S. Army Missile Command.

This design approach breaks out of the binary correlation world previously implemented in hybrid systems to use some of the dynamic range capabilities of the light emitting diodes and charge couple device (CCD) arrays. It also makes use of the strong points of both optical and digital correlators and bypasses some of their shortcomings.

II. SYSTEM ARCHITECTURE

The major components of the system are depicted in Figure 1. A photograph of the optical bench is presented in Figure 2. Top and front views of the electronic rack are shown in Figures 3 and 4. The electronic rack layout with a front panel view is shown in Figure 5, and schematics of the digital control cards are in Figures 6, 7 and 8. Schematics for ADC/Video processing, the correlator reference memory, and LED driver card are shown in Figures 9, 10, and 11. There are two CCD TV cameras used. The first camera is to image the input scene, and the other is for collecting and generating the output correlation surface. The acoustooptic cell (AOC) receives input from a fixture of 64 fiber optic filaments in a linear array and from the input camera video. The fiber optic filaments are attached to a light emitting diode (LED) array being modulated by the digital reference memory. Each LED transmits a single image line from the reference array. The reference memory is interfaced to a host computer for testing purposes. The video of the input camera, while in reference store mode, is typically used to generate a reference image via the analog to digital (A/D) converter in a single field time. In normal operation the input camera video is converted for input to the AO cell only.

A. Video Source

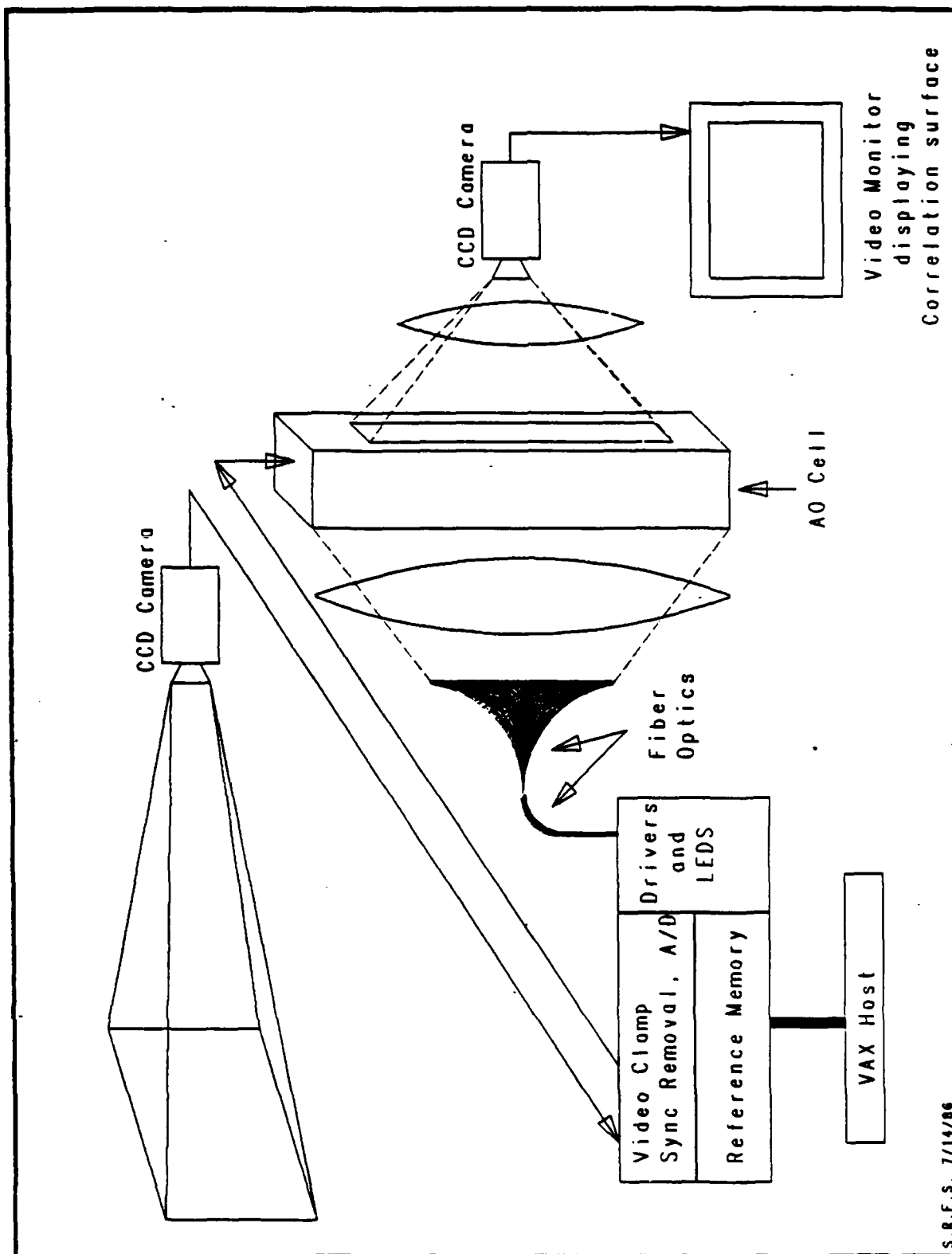
The input video camera is a NEC TI22AII/22PII CCD camera. The video is in standard RS170A format scanning 490 vertical and 384 horizontal picture elements (pixels). The video is not direct current restored, requiring additional circuitry prior to the analog to digital converter stage.

B. Acoustooptic Cell

The AO cell is made up of two major components, the modulator driver, and the large aperture high resolution Bragg cell. The Bragg cell aperture time is 70 microseconds (μs), approximately 7.5 μs longer than one TV line.

The AO cell needed a one-volt (V) analog source requiring a clipping circuit to takeoff the video sync. This circuit is shown in Figure 9.

The reference illumination over the AO cell window during every horizontal blanking time provides a convolution in the horizontal direction over each line in the source video image.



S.R.F.S. 7/14/86

Figure 1. Hybrid gray scale optical/digital correlator-functional block diagram.

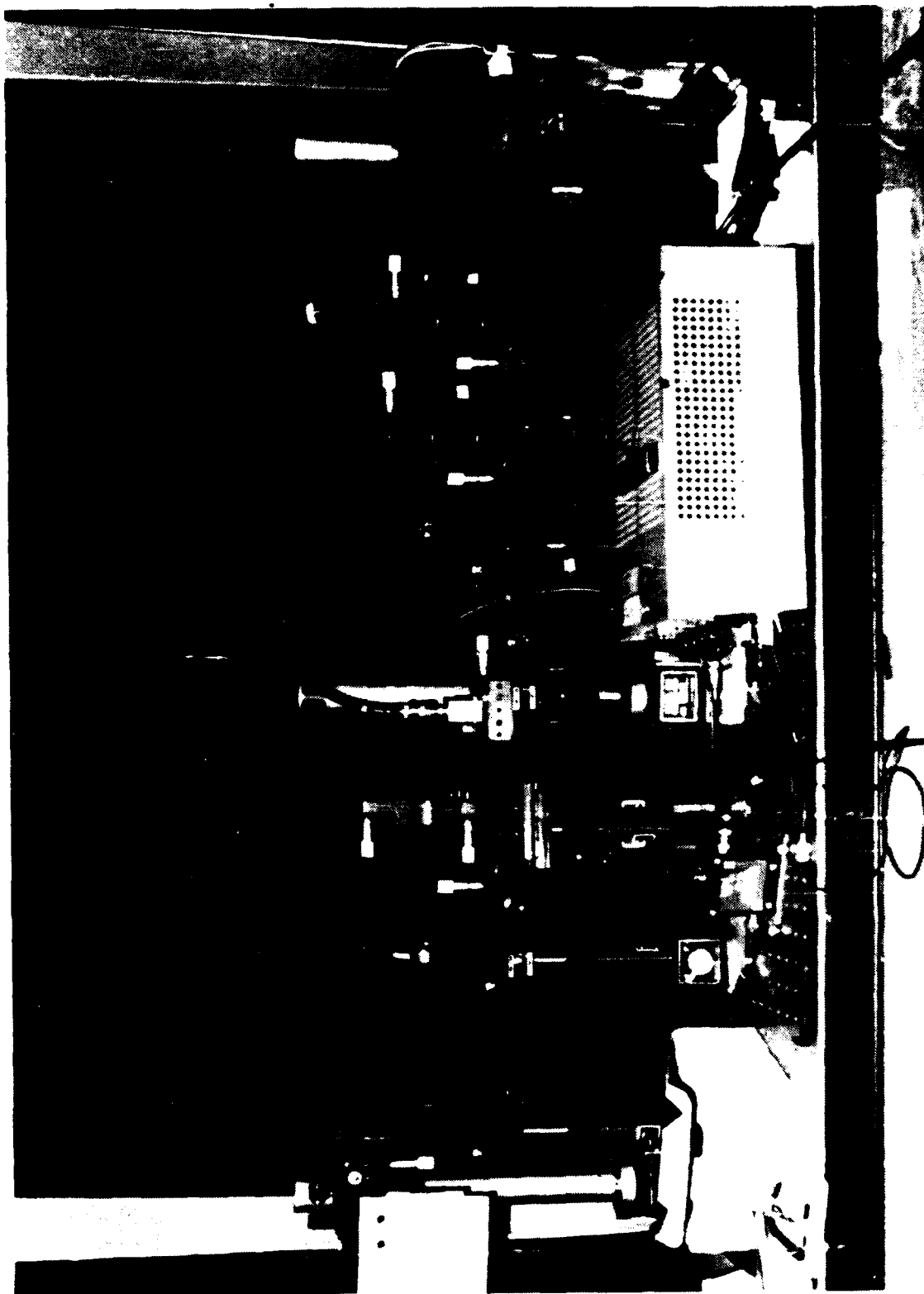


Figure 2. Optical bench.

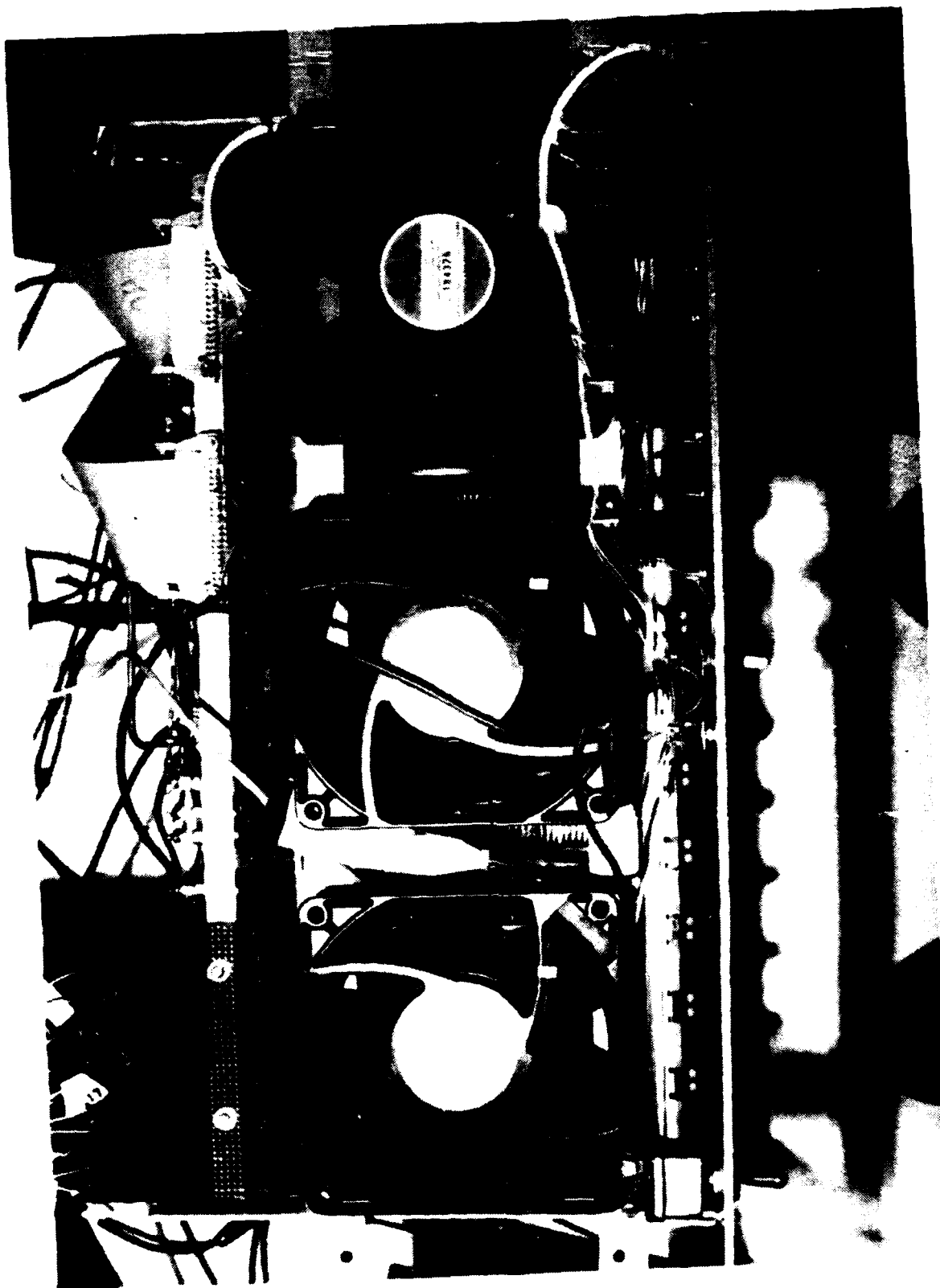


Figure 3. Electronic rack (top view).

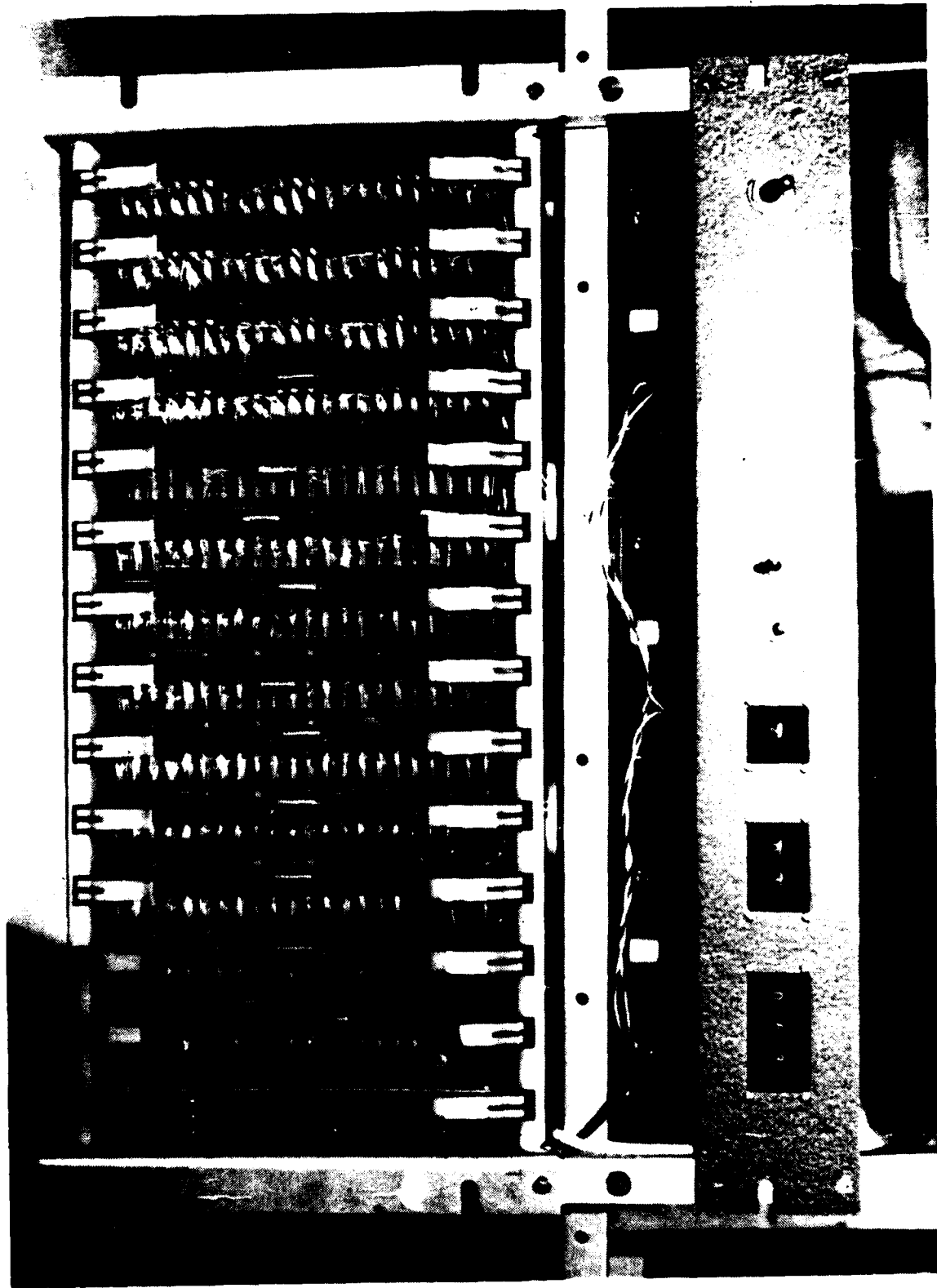


Figure 4. Electronic rack (front view).

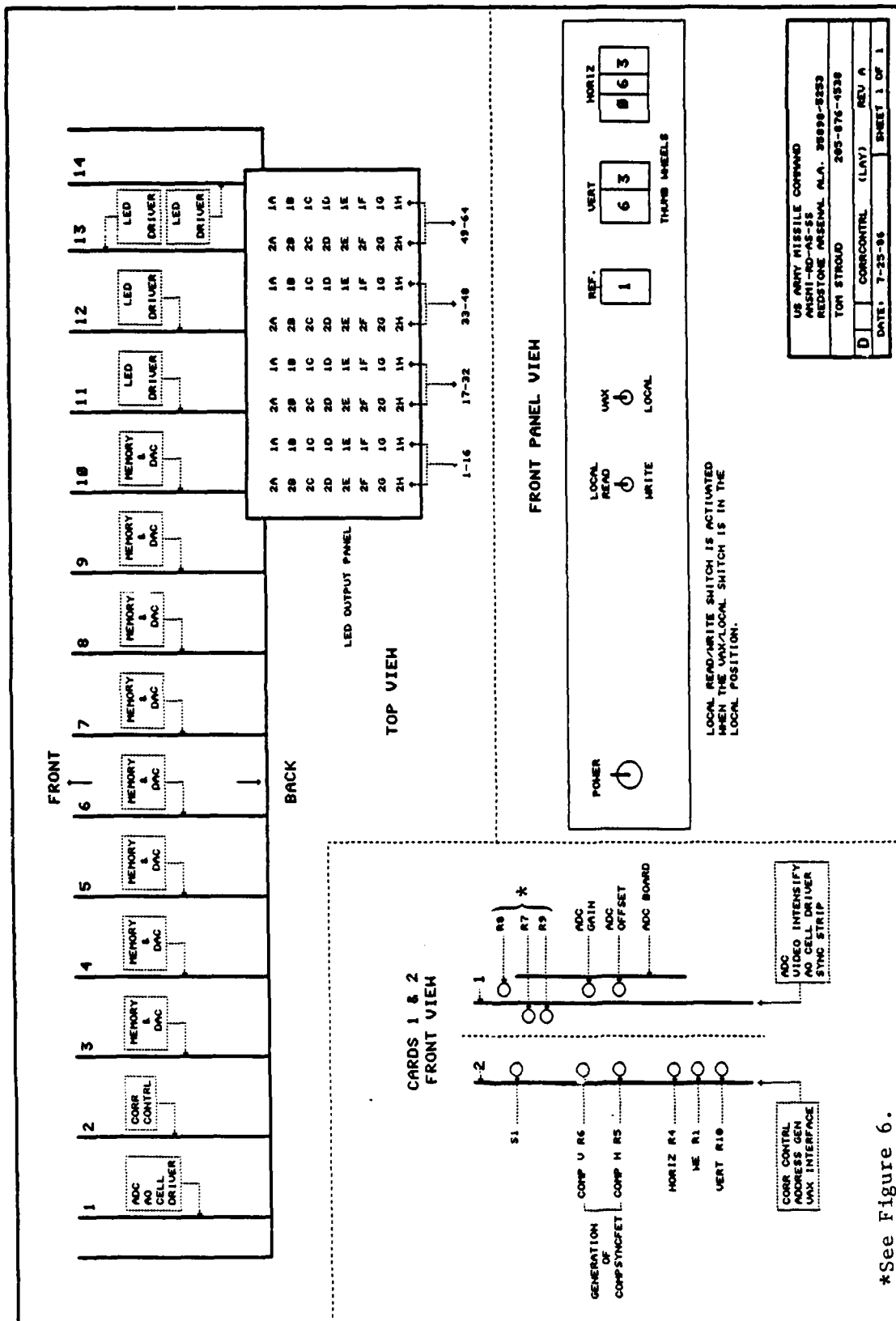
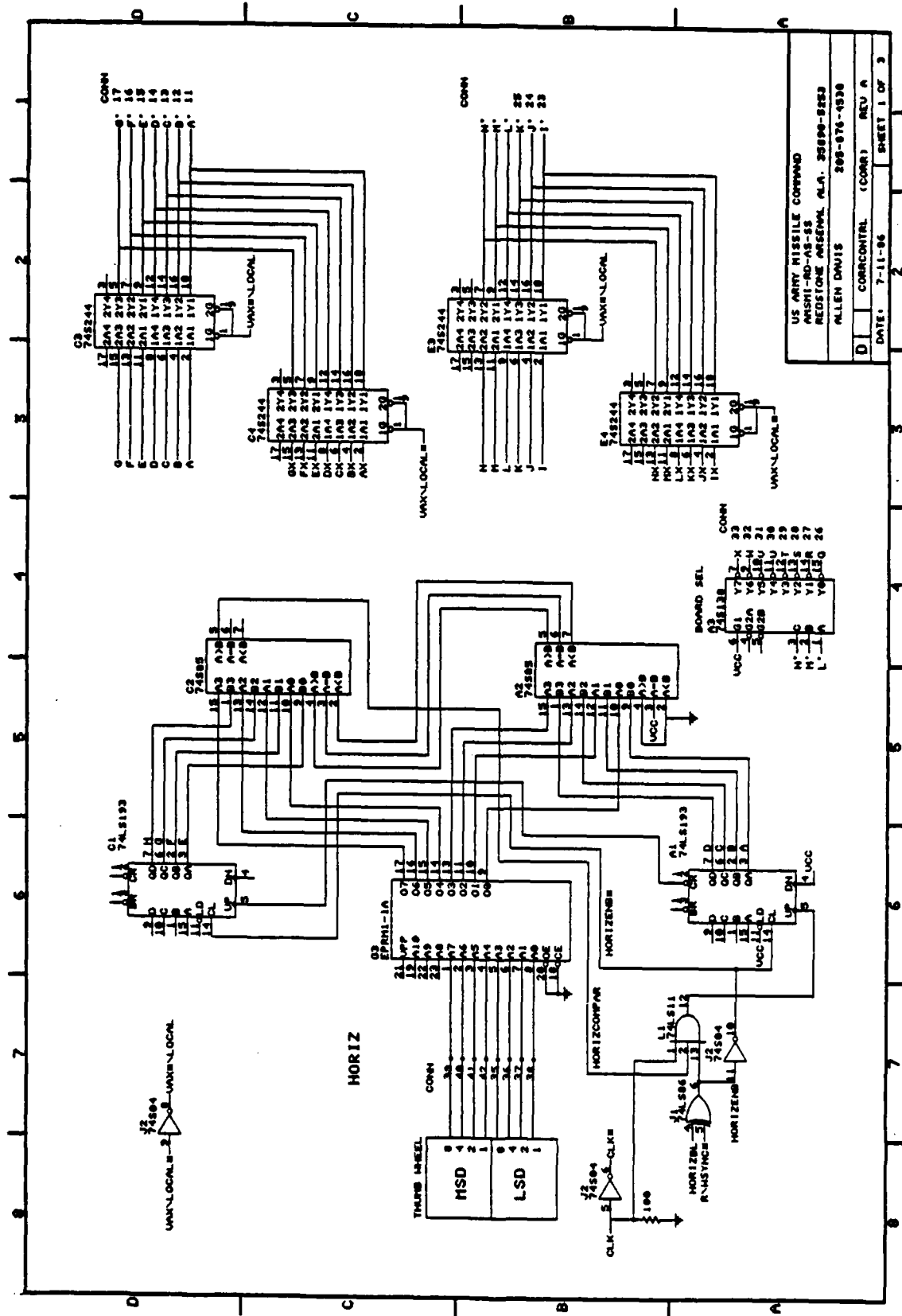
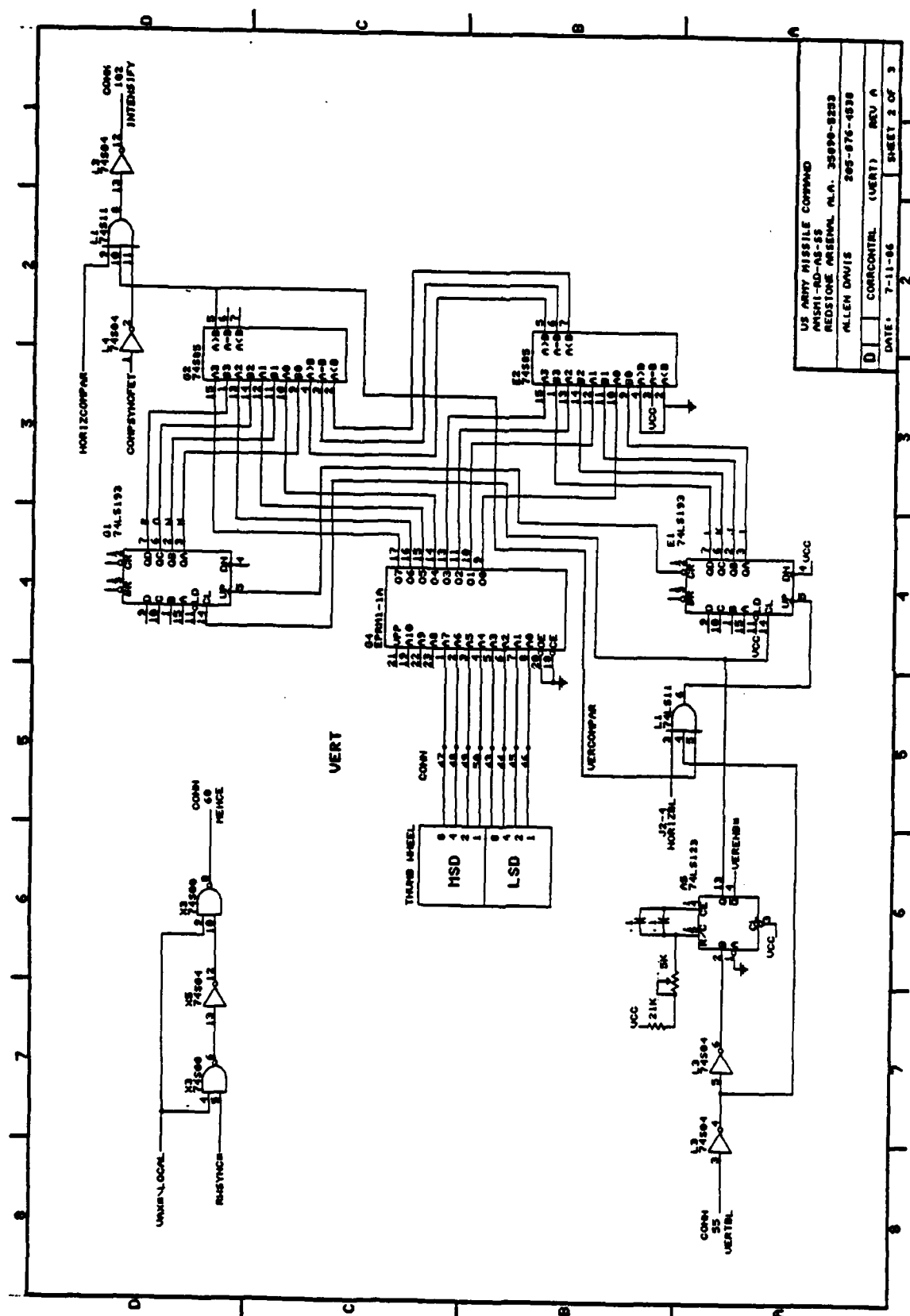


Figure 5. Electronic rack layout with front panel.





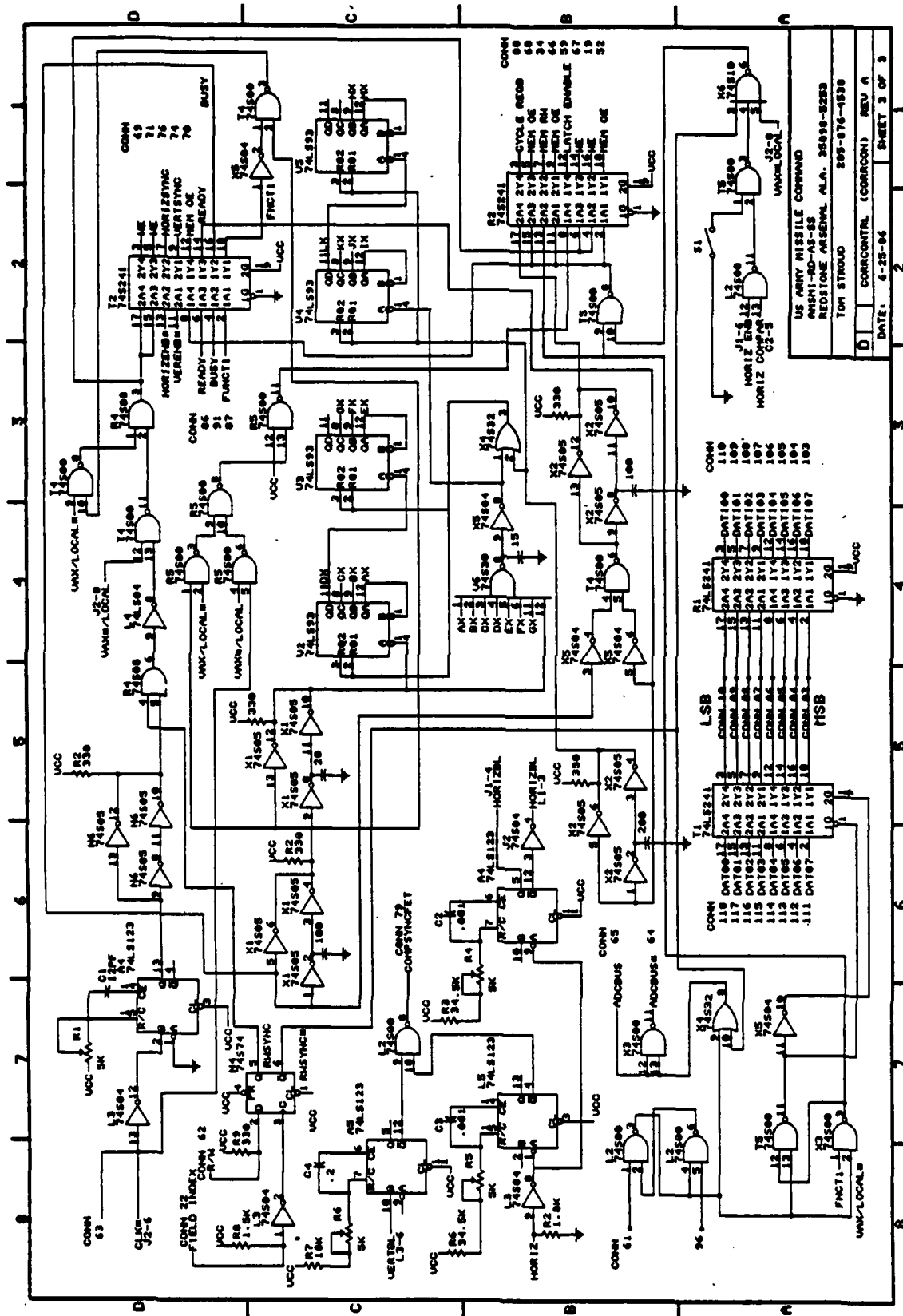


Figure 8. Digital control card (CORRCON).



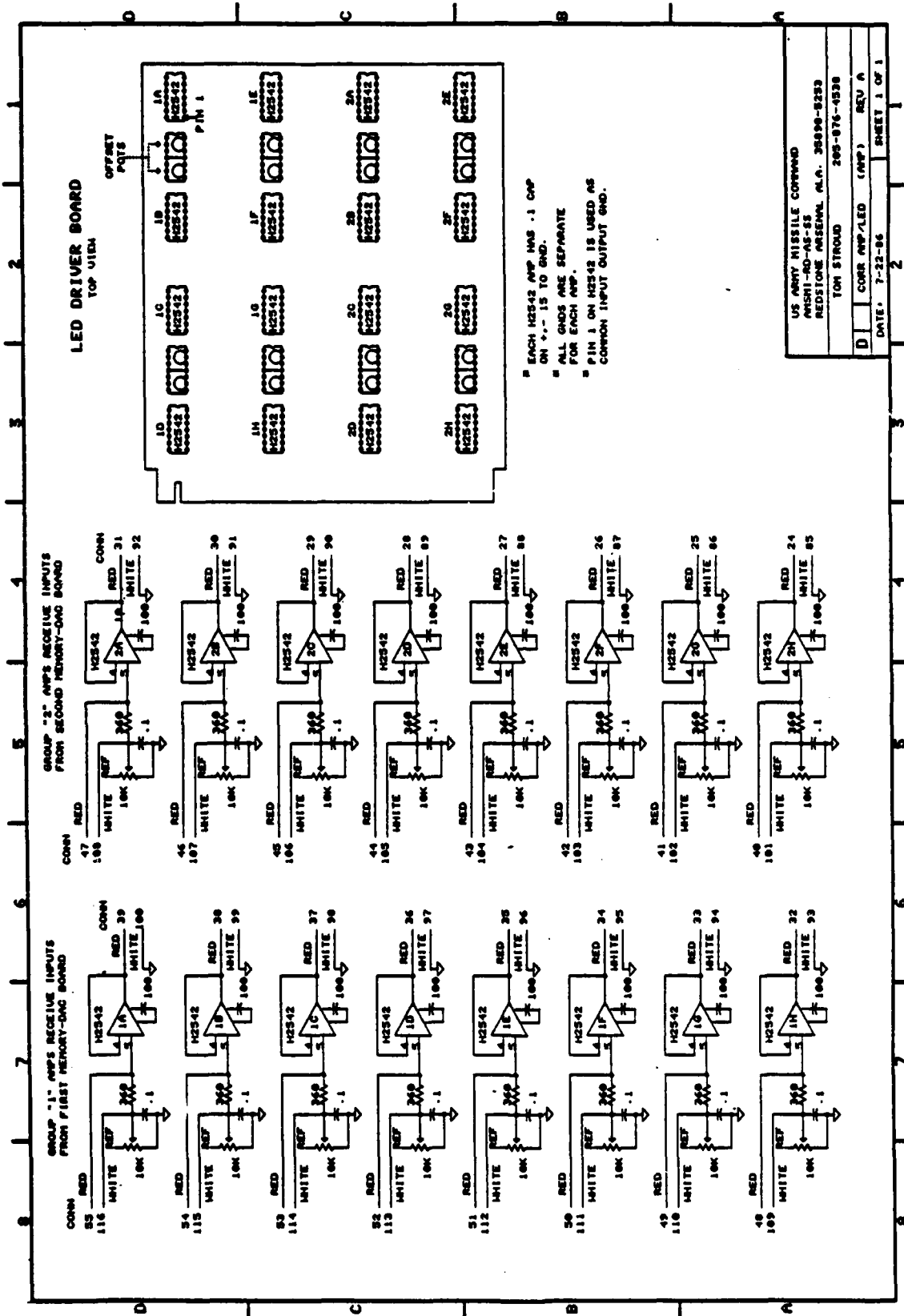


Figure 11. LED driver card.

C. Output Camera Video

The output video camera is a Fairchild CCD 3000 camera with an integral thermoelectric cooler. The camera provides NTSC television video with interlaced fields, scanning 488 lines and 380 pixels per line.

The camera receives its input from the AO cell's convoluted results on a per element basis. The camera was modified by disconnecting and then grounding the photogate clock input, thereby allowing an integration in the vertical scan direction producing the complete two-dimensional correlation surface.

D. A/D Converter Module

Prior to the A/D conversion the video from the NEC camera had to be dc clamped as shown in Figure 9. The A/D converter used was the TRW TDC1007P1C providing 8 bits per sample. The sample clock was the NEC CCD camera pixel clock of approximately 7.16 megahertz.

E. Reference Memory

The reference memory is configured into 64 1024 by 8 bit memories. Each 64 by 128 is selectable through a thumbwheel switch on the front panel. Eight separate references can be stored.

Memory Operation

In reference store mode the memory selected by the thumbwheel switch is enabled and stores the incoming digitized video pixels serially. The size of the input reference is dependant on two sets of thumb-wheel switches on the front panel. The maximum size of the reference is 64 lines and 73 pixels. The vertical dimension is limited by the number of LEDs and memory in that axis. The horizontal dimension is limited by the pixel clock from the NEC camera which can clock this finite number of pixels as a maximum during the horizontal blanking time. The reference can be smaller as desired by the operator. An enhanced area on the TV monitor shows the video area being used as the reference when storing into memory. In operational mode the selected reference memory is readout in parallel. All reference lines are readout simultaneously during every horizontal blanking interval.

F. Digital to Analog Driver (D/A), LED, Fiber Optics

Each memory chip is connected to a light emitting diode (LED) via a D/A converter buffered by a high bandwidth driver. The LED is bonded to a fiber optic filament. The 64 fiber optic filament ends are placed in a linear fixture aligned vertically.

III. GENERAL DESCRIPTION AND OPERATION

The lower right of Figure 5 shows the front panel with controls:

a. VAX (Host)/LOCAL switch

- o VAX - When in VAX mode the user can upload the reference memory to the host for display or storage, etc. The memory can also be downloaded for testing previously stored images or to test matched spatial filters.

- o LOCAL - When in LOCAL mode the system is either storing a reference image or correlating the selected reference with the input video. The LOCAL mode is referred to in the text as the operational mode.

b. LOCAL READ/WRITE switch is activated ONLY when the VAX/LOCAL switch is in the LOCAL position.

- o LOCAL READ - In this position the selected reference memory is being read for correlation.

- o LOCAL WRITE - In this position the selected reference memory is being written with the upper left-hand subimage of the input video. The area being stored is enhanced on the display.

c. REF - A thumbwheel switch to select the reference memory to use.

d. VERT - A set of thumbwheel switches to select the size of the reference image up to 64 lines. These switches should be set from 0 to 63.

e. HORIZ - A set of thumbwheel switches to select the size of the reference image up to approximately 73. This setting is defined by the clock reference divided into the horizontal blanking time as a maximum.

The upper part of Figure 5 shows the board layout in the electronic rack. The LEDs are inserted in the "LED OUTPUT PANEL" on the back of the rack as shown.

IV. ALIGNMENT OF LED GAIN AND BIAS

The setting of the dynamic range of the light emitting diodes is critical to the overall correlation performance. Each LED has its own characteristics which requires an individual gain bias setting. The following steps are required:

- a. Prior to removing or inserting any of the boards the power should be turned off and the correct orientation of the board in the rack must be observed.
- b. Each driver board should be put on an extender one at a time, leaving the other driver boards pulled out of the rack. There are 16 LED drivers per driver board. Figure 4 shows the driver boards in slots 11 through 14.
- c. Two individual memory boards are used per selected driver board, therefore one of these must be put on the extender at a time. Figure 5 shows the memory boards in slots 3 through 10. Memory boards 3 and 4 are used with driver boards in slots 3 through 10. Memory boards 3 and 4 are used with driver board 11, memory boards 5 and 6 are used with driver board 12, memory boards 7 and 8 are used with driver board 12, and memory boards 9 and 10 are used with driver board 14.
- d. The front panel LOCAL READ/WRITE switch should be set to READ.
- e. The VAX/LOCAL switch should be set to VAX.
- f. The SELDAT routine is used to load the memory with a constant of all bits high 377 octal. (Appendix A.)
- g. The VAX/LOCAL switch is now switched to LOCAL.
- h. Each LED driver output is examined to determine the voltage range across the LED. The GAIN potentiometer on the memory board and the bias potentiometer on the driver board are adjusted first to ensure that the voltage swing is always positive.
- i. The GAIN potentiometer can then be set to allow about a 300 millivolts pulse swing. The BIAS can then be set to where the light is just turned off on the LED output. This defines the turnoff bias point for the diode.
- j. The GAIN and BIAS are then set such that the LED output is a known constant brightness determined by the CCD camera video scan, and the turnoff level is set to the now known turnoff bias.
- k. In order to adjust adjacent LEDs the bias can be brought down until all diodes are adjusted then brought backup to the known turnoff bias point for operation.
1. All the above steps are repeated until each LED aligned.

V. OPTICAL LAYOUT AND SPECS

The optical system consists of two spherical and three cylindrical lenses as shown in Figure 12. The focal lengths of these five lenses are required to yield the relations. It accounts for interlace effects as discussed in reference [3].

$$\frac{7.68}{2.304} = \frac{10}{3} = \frac{f_1 f_3}{f_2 f_4}$$

and

$$\frac{f_5}{f_3} \text{ between } 0.294 \text{ and } 0.356.$$

Also, the total length of the optical train must be less than 48 in., the minimum element spacing must allow room for the optical mounting fixtures and varies from 60 millimeter (mm) for f_2 and f_5 to 100 mm for the others, and provision for adjustment must be included in the mounts.

VI. CONCLUSION

Discussion of Optical Effects, System Performance, and Laboratory Results

The incoherent correlator operates on optical intensity rather than amplitude, and thus all of the image processing is performed with positive real signals. This causes an unavoidable background signal bias level to be present, and this is a fundamental property of all incoherent optical processors. The actual correlation signal thus rides on a strong background signal which is a function of scene brightness, and the structure of the numerous cross-correlations of the reference target image, and other nontarget images present in the input scene. The experimental results showed that the highly structured cross-correlations were much stronger than the desired target correlation because the input scene was several times larger than the reference image, and had at least as much dynamic intensity range and structural scene elements. This severely degraded the ability of the incoherent correlator to discriminate the target from the nontarget scene elements.

Another major problem was the overall spacial resolution capability of the correlator. The finest detail that it could successfully process was on the order of ten pixels, and thus it was an extremely low spacial bandpass system. While the basic dynamic function of correlation was achieved, the low spacial bandpass prevented the system from being able to take advantage of the high spacial frequency content, and the observed correlations were extremely low, broad, and barely visible against the high bias background, and indistinguishable from the unwanted cross-correlations. The reason for this loss of spacial detail is that all acoustooptic optical systems have much poorer imaging properties than ordinary systems and coherent optical systems. The requirements are extremely severe for such optical systems to achieve even modest image quality. Since the incoherent correlator requires exact reimaging at the output CCD plane, but with markedly different scales

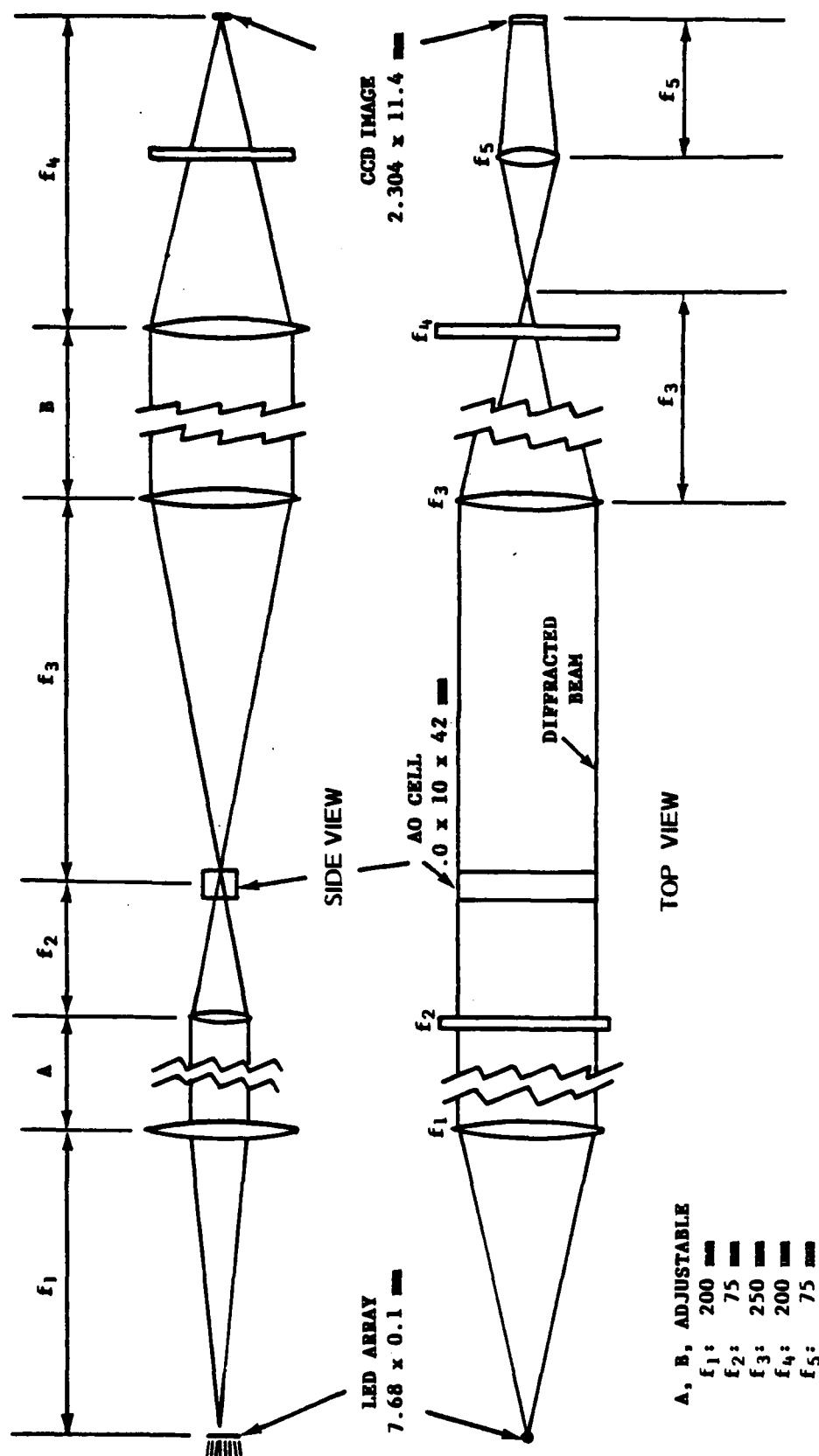


Figure 12. Incoherent correlator optics.

horizontally and vertically, it was not possible to achieve high spacial frequency transfer with the commercially available anamorphic optical lenses. The lenses required to meet this stringent image quality would have to be fully corrected, diffraction-limited, custom built optics, and their cost at the current time was far beyond the scope of this project.

One of the major objectives of the project was to determine the practicality of the optical system, and this result of image quality degradation showed that, while the acoustooptic incoherent correlator could be built and that the components could be attained with state-of-the-art techniques, it was not a cost-effective design in practical terms.

The results of the electronic design implemented is shown in Figures 13, 14, and 15. These signals produced the limits of the reference data spatial frequencies. The output was produced using a photodiode "looking" into the LED array. Several optimization ideas which would improve electronic performance were discussed, but in view of the previous discussion on the optical performance, were not implemented.

In addition, unpublished communications concerning two other research projects to build the same Psaltis architecture were received after this MICOM project was completed. Both of these projects, (Sandia National Laboratories and General Dynamics), reported unsuccessful results and indicated that their AO cells, which were identical to those used in the MICOM project, were at least partially at fault. Other problems were implied (an effort is being made to obtain further information) with this architecture. In view of the fact that three independent research projects, unknown to each other at the time, all obtained the same unsuccessful results when the original Psaltis architecture was implemented in hardware. It is concluded that this approach to a real time incoherent correlator with an undatable reference image, is unfeasible in practice. We recommend no further research on this particular architecture, and instead that other approaches be investigated.

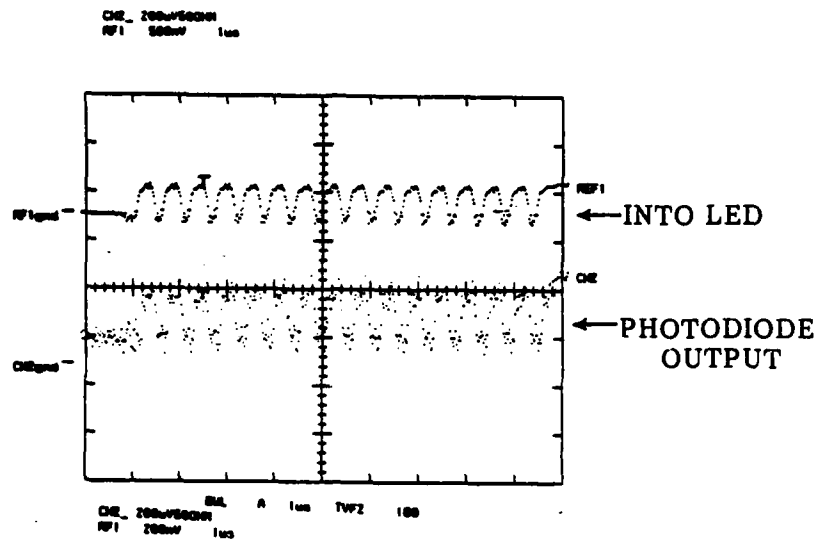


Figure 13. Half frequency test.

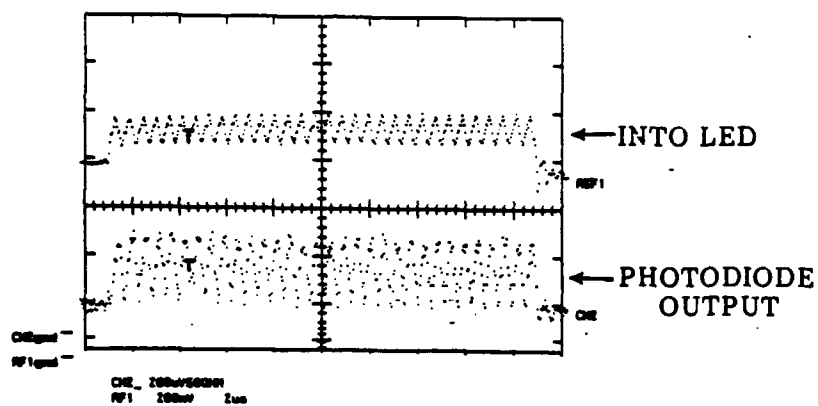


Figure 14. Maximum frequency test.

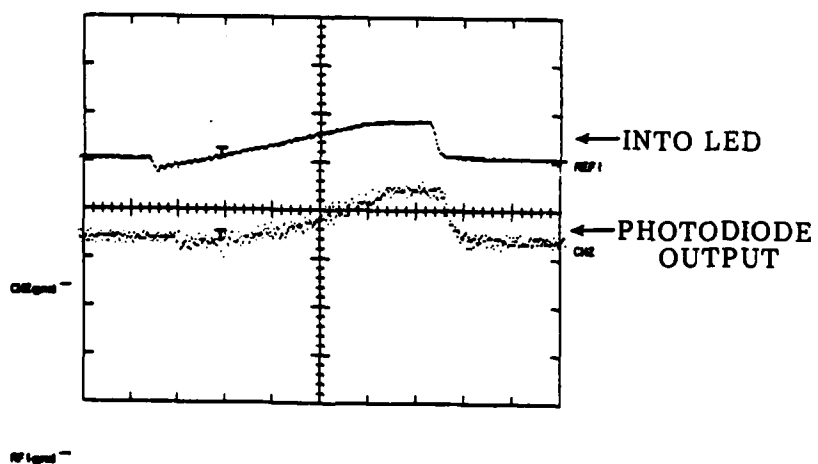


Figure 15. Ramp test 0-255.

REFERENCES

1. Psaltis, D., Opt. Eng., Vol 23, No. 1, January 1984, p. 12.
2. Psaltis, D., Proc. IEEE, Vol 72, No. 7, July 1984, p. 962.
3. Johnson, J. L., Technical Report RD-RE-86-3, U.S. Army Missile Command, Redstone Arsenal, AL, August 1986.

APPENDIX A

BAR.FOR

APPENDIX A

BAR.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C*****
*
  EXTERNAL IO$WRITEVBLK,IO$READVBLK
  INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
  INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
  BYTE DAC(64)
  CHARACTER*80 MSGBUF
  DATA ITIMES/0/
  ISTATUS=SYSS$ASSIGN('XTAO',XTCHAN,,)
  IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
  nbytes=2*nwords
  N=1
C155 TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
C   READ(5,33)DAC(N)
C33  FORMAT(I2)
C   DAC(N)=31
C   IF(DAC(N).GT.0)THEN
C   N=N+1
C   GO TO 155
C   ENDIF
C   N=N-1
      DO J=1,nwords
      BUF(J)='0'O
      ENDDO
      I=1
C   DO I=1,N
C   TYPE *, 'DAC NUMBER=',DAC(I)
C   DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
C   BUF(K)='100'O
C   ENDDO
C   ENDDO
C   DO K=1+((DAC(I)-1)*128),((128+((DAC(I)-1)*128))/2)-1
C   BUF(K)=0
C   ENDDO
C   DO K=((128+((DAC(I)-1)*128))/2)-1,128+((DAC(I)-1)*128)
C   BUF(K)=0
C   ENDDO

```

BAR.FOR

```
DO I=31,8192-95,128
BUF(I)='377'O
ENDDO
```

C BUF(4000-128)='377'O

```
ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(NBYTES),,,, )
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSSGETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,, )
END
```

APPENDIX B

BAR2.FOR

APPENDIX B

BAR2.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C*****
*
  EXTERNAL IOSWRITEVBLK,IOSREADVBLK
  INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
  INTEGER SYS$ASSIGN,XTCHAN,SYS$QIOW,SYS$GETMSG
  BYTE DAC(64)
  CHARACTER*80 MSGBUF
  DATA ITIMES/0/
  ISTATUS=SYS$ASSIGN('XTAO',XTCHAN,,)
  IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
  nbytes=2*nwords
  N=1
C155 TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
C   READ(5,33)DAC(N)
C33  FORMAT(I2)
C   DAC(N)=31
C   IF(DAC(N).GT.0)THEN
C   N=N+1
C   GO TO 155
C   ENDIF
C   N=N-1
      DO J=1,nwords
      BUF(J)='0'O
      ENDDO
      I=1
C   DO I=1,N
C   TYPE *, 'DAC NUMBER=',DAC(I)
C   DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
C   BUF(K)='100'O
C   ENDDO
C   ENDDO
C   DO K=1+((DAC(I)-1)*128),((128+((DAC(I)-1)*128))/2)-1
C   BUF(K)=0
C   ENDDO
C   DO K=((128+((DAC(I)-1)*128))/2)-1,128+((DAC(I)-1)*128)
C   BUF(K)=0
C   ENDDO

```

BAR2.FOR

```
DO I=31,8192-95,128
BUF(I)='377'O
BUF(I+1)='377'O
ENDDO
```

C BUF(4000-128)='377'O

```
ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(NBYTES),,,, )
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF=' '
ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,, )
END
```

APPENDIX C
CDACTEST.FOR

APPENDIX C

CDACTEST.FOR

```

EXTERNAL IOS$WRITEVBLK,IOS$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
c1  do 100 itest=1,5
1  continue
      DO J=1,nwords
      buf(j)=(j-1)*2
c      if(itest.eq.1)BUF(J)=J
c      if(itest.eq.2)buf(j)=0
c      if(itest.eq.3)buf(j)='377'o
c      if(itest.eq.4)buf(j)='125'o
c      if(itest.eq.5)buf(j)='052'o
      ENDDO
ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IOS$WRITEVBLK)),IOSB,,
1BUF(1),%VAL(NBYTES),,,)
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'I/O STATUS:',MSGBUF
IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IOS$READVBLK)),
1IOSB,,
1RBUF(1),%VAL(NBYTES),,,)
C      DO I=1,NWORDS
C      IF(IAND(BUF(I),'377'O).NE.IAND(RBUF(I),'377'O))THEN
C      WRITE(6,11)I,IAND(BUF(I),'377'O),IAND(RBUF(I),'377'O)
11  FORMAT('** MEMORY R/W ERROR AT',06,' INPUT-',1X,06,
1'      OUTPUT-',06)

```

CDATEST.FOR

```
C      ENDIF
C      ENDDO

100    continue
      ITIMES=ITIMES+1
C      type *, 'memory test complete with the above errors.', ITIMES,
C      1' ON', NWORDS, ' LOCATIONS'
      GO TO 1111
      END
```

APPENDIX D

CFREQTEST.FOR

APPENDIX D

CFEQTEST.FOR

```

EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
c1  do 100 itest=1,5
  1  continue
      DO J=2,nwords,2
      BUF(J-1)='377'O
      BUF(J)=0
C      buf(j)=(j-1)*2
C      if(itest.eq.1)BUF(J)=J
C      if(itest.eq.2)buf(j)=0
C      if(itest.eq.3)buf(j)='377'o
C      if(itest.eq.4)buf(j)='125'o
C      if(itest.eq.5)buf(j)='052'o
      ENDDO
ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(NBYTES),,,,)
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
TYPE *,'I/O STATUS:',MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,,)
C      DO I=1,NWORDS
C      IF(IAND(BUF(I),'377'O).NE.IAND(RBUF(I),'377'O))THEN
C      WRITE(6,11)I,IAND(BUF(I),'377'O),IAND(RBUF(I),'377'O)

```

CFREQTEST.FOR

```
11      FORMAT('** MEMORY R/W ERROR AT',06,' INPUT=',1X,06,
1'      OUTPUT=',06)
C      ENDIF
C      ENDDO

100     continue
        ITIMES=ITIMES+1
C      type *, 'memory test complete with the above errors.',ITIMES,
C      1' ON',NWORDS, ' LOCATIONS'
        GO TO 1111
        END
```

APPENDIX E
CMEMDUMP.FOR

APPENDIX E

CMEMDUMP.FOR

```

EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
  DATA ITIMES/0/
  ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
  IF(.NOT.ISTATUS)TYPE *, 'ERROR IN AMD XT CHANNEL ASSIGN'
  TYPE *, 'ENTER NUMBER OF LOCATIONS TO DUMP'
  ACCEPT*,NWORDS
  NBYTES=NWORDS*2
1111  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
  1IOSB,,,
  1RBUF(1),%VAL(NBYTES),,,, )
      DO I=1,NWORDS
C      IF(IAND(BUF(I),'377'O).NE.IAND(RBUF(I),'377'O))THEN
      IF(IAND(RBUF(I),'377'O).NE.0)THEN
11      WRITE(6,11)I,IAND(RBUF(I),'377'O)
      FORMAT('** MEMORY AT',O6,
1' =',O6)
C      ENDIF
      ENDIF
      ENDDO
100  continue
      END

```

APPENDIX F
-
CMEMTEST.FOR

APPENDIX F

CMEMTEST.FOR

```

EXTERNAL IOS$WRITEVBLK,IOS$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
INTEGER*2 ERROR,ITIMES,NWORDS
CHARACTER*80 MSGBUF
DATA ITIMES/0/,ERROR/0./
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
1 do 100 itest=1,5
  DO J=1,nwords
    if(itest.eq.1)BUF(J)=J
    if(itest.eq.2)buf(j)=0
    if(itest.eq.3)buf(j)='377'o
    if(itest.eq.4)buf(j)='125'o
    if(itest.eq.5)buf(j)='052'o
  ENDDO
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IOS$WRITEVBLK)),IOSB,,
  1BUF(1),%VAL(NBYTES),,,)
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
    ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
    TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
    TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
    TYPE *,'QIO WRITE PARAMETER STATUS:',MSGBUF
    MSGBUF=' '
    ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
    TYPE *,'I/O STATUS:',MSGBUF
    IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
    IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
  ENDIF
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IOS$READVBLK)),IOSB,,
  1RBUF(1),%VAL(NBYTES),,,)
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
    ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
    TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
    TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
    TYPE *,'QIO READ PARAMETER STATUS:',MSGBUF
    MSGBUF=' '

```

CNEMTEST.FOR

```

ISTATUS=SYSS$GETMSG (I%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS) TYPE *, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1)) TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
DO I=1, NWORDS
  IF(IAND(BUF(I), '377'O).NE.IAND(RBUF(I), '377'O)) THEN
    ERROR=ERROR+1.
C    WRITE(6,11) I, IAND(BUF(I), '377'O), IAND(RBUF(I), '377'O)
11  FORMAT('** MEMORY R/W ERROR AT ', I4, ' INPUT=', I4, 'O6,
1'    OUTPUT=', I4, 'O6)
    LOW=IAND(I, '177'O)
    ICHIP=ISHFT(IAND(I, '1600'O), -7)+1
    IBOARD=ISHFT(IAND(I, '16000'O), -10)+1
    IF(ICHIP.NE.IOLDCHIP)
1  TYPE *, 'BOARD, CHIP=', IBOARD, ICHIP
    IOLDBOARD=IBOARD
    IOLDCHIP=ICHIP
  ENDIF
ENDDO

100  continue
    ITIMES=ITIMES+1
    type *, 'memory tested with the above', ERROR,
1' errors.', ITIMES,
1' ON', NWORDS, ' LOCATIONS'
GO TO 1
END

```

APPENDIX G

CREADI

APPENDIX G

CREADI

```

EXTERNAL IOS$WRITEVBLK,IOS$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4),IMAGE(128,64)
  INTEGER*2 IXBW,IYBW
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
  EQUIVALENCE (BUF,IMAGE)
COMMON/BALLO/IXBW,IYBW
COMMON/CHAN/ITCHAN,IGCHAN
  DATA ITIMES/0/
  DATA IXBW/0/,IYBW/400/
  ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
  IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  ISTATUS=SYSS$ASSIGN('GRA0',IGCHAN,,)
  IF(.NOT.ISTATUS)TYPE *,'ERROR IN GRINNELL CHANNEL ASSIGN'
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCTEST IMAGECCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      DO I=1,64
C      DO J=1,128
C      IMAGE(J,I)=IAND(I*J,'377'O)
C      ENDDO
C      ENDDO
C
C      DO J=1,128
C      IMAGE(J,1)='377'O
C      IMAGE(J,64)='377'O
C      ENDDO
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCTEST IMAGECCCCCCCCCCCCCCCCCCCCCCCCCCCC
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IOS$READVBLK)),IOSB,,,
  1BUF(1),%VAL(16384),,,)
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'QIO PARAMETER STATUS:',MSGBUF
  MSGBUF=' '
  ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'I/O STATUS:',MSGBUF
  IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN XT OUTPUT'
  IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'

```

CREADI

```
ENDIF
C      WRITE(6,111)IMAGE
111    FORMAT(1X,10(1X,O6))
      DO I=1,64
      DO J=1,127
      IMAGE(J,I)=IAND(IMAGE(J,I),'377'O)
      ENDDO
      IMAGE(128,I)='34015'O
      ENDDO
      CALL PIX(IMAGE)
END
SUBROUTINE PIX(IBUFA)
IMPLICIT INTEGER*2 (A-Z)
EXTERNAL IOSWRITEVBLK
INTEGER SYSSQIOW,ITCHAN,IGCHAN,J,IOSWRITEVBLK
INTEGER*2 OUT(12),IBUFA(1)
COMMON/CHAN/ITCHAN,IGCHAN
COMMON/BALLO/IXBW,IYBW
DATA OUT/'17777'O,'120000'O,'107777'O,'24055'O,'26002'O,'44000'O,
1'50002'O,'54000'O,'64000'O,'74776'O,0,0/
OUT(11)=IXBW.OR.'44000'O
OUT(12)=IYBW.OR.'64000'O
J=SYSSQIOW(%VAL(1),%VAL(IGCHAN),%VAL(%LOC(IOSWRITEVBLK))),,,
1OUT(1),%VAL(24),,,,)
J=SYSSQIOW(%VAL(1),%VAL(IGCHAN),%VAL(%LOC(IOSWRITEVBLK))),,,
1IBUFA(1),%VAL(16384),,,,)
RETURN
END
```

APPENDIX H
CRWIMAGE.FOR

CRWIMAGE.FOR

H-1

CRWIMAGE.FOR

```

IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS) TYPE *, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1)) TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
DO I=1,64
DO J=63,127
IMAGE(J,I)=0
ENDDO
ENDDO
ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(16384),,,,
IF(.NOT.ISTATUS) TYPE *, 'ERROR ON WRITE BACK'
ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),IOSB,,,
1BUF(1),%VAL(16384),,,,
IF(.NOT.ISTATUS) TYPE *, 'ERROR ON SECOND READ'
C      WRITE(6,111) IMAGE
111    FORMAT(1X,10(1X,06))
      DO I=1,64
      DO J=1,127
      IMAGE(J,I)=IAND(IMAGE(J,I),'377'O)
      ENDDO
      IMAGE(128,I)='34015'O
      ENDDO
      CALL PIX(IMAGE)
END
SUBROUTINE PIX(IBUFA)
IMPLICIT INTEGER*2 (A-Z)
EXTERNAL IO$WRITEVBLK
INTEGER SYSSQIOW,ITCHAN,IGCHAN,J,IO$WRITEVBLK
INTEGER*2 OUT(12),IBUFA(1)
COMMON/CHAN/ITCHAN,IGCHAN
COMMON/BALLO/IXBW,IYBW
DATA OUT/'17777'O,'120000'O,'107777'O,'24055'O,'26002'O,'44000'O,
1'50002'O,'54000'O,'64000'O,'74776'O,0,0/
OUT(11)=IXBW.OR.'44000'O
OUT(12)=IYBW.OR.'64000'O
J=SYSSQIOW(%VAL(1),%VAL(IGCHAN),%VAL(%LOC(IO$WRITEVBLK)),,,,
1OUT(1),%VAL(24),,,,
J=SYSSQIOW(%VAL(1),%VAL(IGCHAN),%VAL(%LOC(IO$WRITEVBLK)),,,,
1IBUFA(1),%VAL(16384),,,,
RETURN
END

```

APPENDIX I

DELTA.FOR

APPENDIX I

DELTA.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C*****
*
EXTERNAL IOS$WRITEVBLK,IOS$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
BYTE DAC(64)
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
nbytes=2*nwords
N=1
C155 TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
C   READ(5,33)DAC(N)
33   FORMAT(I2)
      DAC(N)=31
C   IF(DAC(N).GT.0)THEN
C     N=N+1
C     GO TO 155
C   ENDIF
C   N=N-1
      DO J=1,nwords
      BUF(J)='0'O
      ENDDO
      I=1
C   DO I=1,N
      TYPE *, 'DAC NUMBER=',DAC(I)
      DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
      BUF(K)='377'O
      ENDDO
C   ENDDO
      DO K=1+((DAC(I)-1)*128),((128+((DAC(I)-1)*128))/2)-1
      BUF(K)=0
      ENDDO
      DO K=((128+((DAC(I)-1)*128))/2)-1,128+((DAC(I)-1)*128)
      BUF(K)=0
      ENDDO

```

DELTA.FOR

BUF(4000-128)='377'O

```
ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(NBYTES),,,,
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSSGETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,,
END
```

APPENDIX J

DELTA2.FOR

APPENDIX J

DELTA2.FOR

```

C*****
*
C
C
C
C
C
C
C*****
*
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
BYTE DAC(64)
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *, 'ERROR IN AMD XT CHANNEL ASSIGN'
NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
nbytes=2*nwords
N=1
C155 TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
C READ(5,33)DAC(N)
33 FORMAT(I2)
DAC(N)=31
C IF(DAC(N).GT.0)THEN
C N=N+1
C GO TO 155
C ENDIF
C N=N-1
      DO J=1,nwords
      BUF(J)='0'O
      ENDDO
      I=1
C DO I=1,N
      TYPE *, 'DAC NUMBER=',DAC(I)
      DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
      BUF(K)='100'O
      ENDDO
C ENDDO
      DO K=1+((DAC(I)-1)*128),((128+((DAC(I)-1)*128))/2)-1
      BUF(K)=0
      ENDDO
      DO K=((128+((DAC(I)-1)*128))/2)-1,128+((DAC(I)-1)*128)
      BUF(K)=0
      ENDDO

```

DELTA2.FOR

```
DO JJ=1,2
BUF(4000-128-2+JJ)='377'O
BUF(4000-256-2+JJ)='377'O
BUF(4000-384-2+JJ)='377'O
BUF(4000-512-2+JJ)='377'O
ENDDO

ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(NBYTES),,,,,)
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSSGETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,,,)
END
```

APPENDIX K

IMAGEH.FOR

APPENDIX K

IMAGE.FOR

```

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      This routine reads in the selected reference image from the
c      correlator memory and generates the gray scale histogram
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN'
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4),IMAGE(128,64)
      INTEGER*2 IXBW,IYBW
      INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
      integer*2 tb(65,64)
      integer hist(256)
      CHARACTER*80 MSGBUF
      EQUIVALENCE (BUF,IMAGE)
      COMMON/BALLO/IXBW,IYBW
      COMMON/CHAN/ITCHAN
      DATA ITIMES/0/
      DATA IXBW/0/,IYBW/256/
      data ncol,nrow/64,64/
      ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
      IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
      ISTATUS=SYSS$ASSIGN('GRA0',GRCHAN,,)
      IF(.NOT.ISTATUS)TYPE *,'ERROR IN GRINNELL CHANNEL ASSIGN'
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      DO I=1,64
c      DO J=1,128
c      IMAGE(J,I)=IAND(I*J,'377'O)
c      ENDDO
c      ENDDO
c      DO J=1,128
c      IMAGE(J,1)='377'O
c      IMAGE(J,64)='377'O
c      ENDDO
c      do i=1,64
c      do j=1,64
c      image(j,i)=iand(4*(j-1),'370'o)
c      enddo
c      enddo
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),IOSB,,

```

IMAGE.FOR

```

1BUF(1),%VAL(16384),,,)
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
  TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
  TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
  IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
  TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
  MSGBUF=' '
  ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
  IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
  TYPE *, 'I/O STATUS:',MSGBUF
  IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
  IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'

  ENDIF
C      WRITE(6,111)IMAGE
111    FORMAT(1X,10(1X,O6))
cccccccccccc
c      put out the histogram to the grinnell
c
      DO I=1,64
      DO J=1,127
      IMAGE(J,I)=IAND(IMAGE(J,I),'377'O)
      ENDDO
      ENDDO

do i=1,64
do j=1,64
tb(j,i)=image(j,i)
enddo
enddo
CALL GGHIST(tb)
      DO I=1,64
      IMAGE(128,I)='34015'O
      ENDDO
      CALL PIX(IMAGE)
END
SUBROUTINE PIX(IBUFA)
IMPLICIT INTEGER*2 (A-Z)
EXTERNAL IO$WRITEVBLK
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN'
INTEGER SYSS$QIOW,ITCHAN,J,IO$WRITEVBLK
INTEGER*2 OUT(12),IBUFA(1)
COMMON/CHAN/ITCHAN
COMMON/BALLO/IXBW,IYBW
DATA OUT/'17777'O,'120000'O,'107777'O,'24055'O,'26002'O,'44000'O,
1'50002'O,'54000'O,'64000'O,'74776'O,0,0/
OUT(11)=IXBW.OR.'44000'O
OUT(12)=IYBW.OR.'64000'O
J=SYSS$QIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(IO$WRITEVBLK)),,,
1OUT(1),%VAL(24),,,)
J=SYSS$QIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(IO$WRITEVBLK)),,,
1IBUFA(1),%VAL(16384),,,)
RETURN
END
SUBROUTINE GGHIST(IMAGE)
C *****
C

```

IMAGEH.FOR

```

C      FUNCTION:
C
C      THIS ROUTINE PERFORMS A HISTOGRAM ON THE IMAGE
C
C      *****
C
C      ARGUMENTS:
C
C      *****
C
C      MODULE NOTES:
C
C      *****
C
C      MODULE CREATION DATE:
C
C      DEC. 1985
C
C      *****
C
C      MODULE MODIFICATION HISTORY:
C
C      ***** EOH *****
C
C
C      EXTERNAL IOSWRITEVBLK
C      INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTLB.CMN'
C      INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN'
C      COMMON/IMGSOURCE/IMGSOURCE      !IMAGE SOURCE STRING COMMON TO SUBCON
SUBINIT
C      CHARACTER*5 IMGSOURCE
C      INTEGER*4 OPRINP
C      INTEGER*4 CHECKING                !DUPLICATE IMAGE CHECK FUNCTION
C      LOGICAL*1 DISKINPUT              !DISK/TAPE INPUT FLAG .TRUE.=DISK
C      .FALSE.=TAPE
C      DATA DISKINPUT/.TRUE./ !ASSUME DISK INPUT
C      CHARACTER*4 CHARIN                !OPERATER RESPONSE
C      INTEGER*2 IACODE                  !VARIABLE TO CONTAIN OPERATOR FUN REQ
C      INTEGER*4 DSPUPD
C      LOGICAL*1 IIFLG                  !REWRITE HEADER2 LOGICAL FLAG
C      LOGICAL*1 TAPEDUP,OLDIMG/.FALSE./
C      DIMENSION LISTSIZE(3),LISTSPEED(3),LISTFPOS(3)
C      DIMENSION LIST(8),ILIST(3),LISTMODE(2),LISTSPDT(2)
C      DIMENSION MLIST(5)
C      INTEGER*2 ERASEQ(15),IMAGE(NCOL+1,NROW),OUT(780)
C      INTEGER HIST(256)
C      DATA ERASEQ/'100000'O,'17777'O,'24240'O,'121040'O,
C      1'140001'O,'121000'O,'110000'O,'107777'O,'121000'O,
C      2'64000'O,'70377'O,'44000'O,'52377'O,'24040'O,'26000'O/
C      I=LIB$ERASEPAGE(1,1)             !ERASE SCREEN
C      IER = IFENCEY + IFENCEYSIZE - 1 !COMPUTE ENDING ROW FOR SUB IMAGE
C      IEC = IFENCEX + IFENCEXSIZE - 1 !COMPUTE ENDING COL FOR SUBIMAGE
C      IER=64
C      IEC=64

```

IMAGEH.FOR

```

C      CALL MESSAGE('ENTER HISTOGRAM PLOT QUADRANT')
C      CALL ACCEPT(IACODE)          !GET THE OPERATORS REQ KEY
C      IF(IACODE.EQ.'0031'X)THEN
ERASEQ(10)='64400'O      !LLA
ERASEQ(12)='44377'O      !LEA
C      ENDIF
C      IF(IACODE.EQ.'0032'X)THEN
ERASEQ(10)='64400'O
ERASEQ(12)='44000'O
C      ENDIF
C      IF(IACODE.EQ.'0033'X)THEN
ERASEQ(10)='64000'O
ERASEQ(12)='44000'O
C      ENDIF
C      IF(IACODE.EQ.'0034'X)THEN
ERASEQ(10)='64000'O
ERASEQ(12)='44400'O
C      ENDIF
C      I=SYSSQIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(ISO$WRITEVBLK)),,...
1ERASEQ,%VAL(30),,...)
IFENCEY=1
IFENCEX=1
DO ICOL=1,256
    HIST(ICOL)=0
ENDDO
DO IROW=IFENCEY,IER
DO ICOL=IFENCEX,IEC
C      PRINT *,'IMAGE(ICOL,IROW)=' ,IMAGE(ICOL,IROW),ICOL,IROW
IF(IMAGE(ICOL,IROW)+1 .GT. 256 .OR. IMAGE(ICOL,IROW)+1 .LT.1)
1TYPE *,'DATA ERROR IN INPUT IMAGE !!!!!!!!!!!',IMAGE(ICOL,IROW)+1
    HIST(IMAGE(ICOL,IROW)+1)=HIST(IMAGE(ICOL,IROW)+1)+1
ENDDO
ENDDO
LMAX=-1000000
LMIN=1000000
C      SKIP ZERO WHEN GETTING SCALE FOR HISTOGRAM (START AT 2)
MAXLOCATION=0
MINLOCATION=0
DO ILOOK=2,256
    IF(HIST(LOOK).GT.LMAX) THEN
        LMAX=HIST(LOOK)
        MAXLOCATION=LOOK
    ENDIF
    IF(HIST(LOOK).LT.LMIN) THEN
        LMIN=HIST(LOOK)
        MINLOCATION=LOOK
    ENDIF
ENDDO
    IF(LMAX.EQ.LMIN)THEN
TYPE *,'MAX IS THE SAME AS MIN.....!!!!!!=' ,LMAX
RETURN
ELSE
TYPE *,'MAX,MIN=' ,LMAX,LMIN
TYPE *,'MAXLOCATION=' ,MAXLOCATION
TYPE *,'MINLOCATION=' ,MINLOCATION
ENDIF
C      HIST(1)=(HIST(1)-LMIN)/(LMAX-LMIN)
IF(HIST(1).GT.255)HIST(1)=255      !THIS TAKES CARE OF ZERO

```

IMAGEH.FOR

```
DO ILOOK=2,256
HIST(ILOOK)=255.*(FLOAT(HIST(ILOOK))-FLOAT(LMIN))/
1(FLOAT(LMAX)-FLOAT(LMIN))
ENDDO
C      TYPE *,HIST
444    IOUT=2
      OUT(1)='50000'O
      DO ILOAD=1,256
      OUT(IOUT)=ERASEQ(12)+(ILOAD)
      OUT(IOUT+1)=ERASEQ(10)
      OUT(IOUT+2)='72000'O+HIST(ILOAD)
      IOUT=IOUT+3
      ENDDO
      IOUT=IOUT-1
      I=SYSSQIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(ISO$WRITEVBLK)),,,
      IOUT,%VAL(IOUT*2),,,,)
      RETURN
      END
```

APPENDIX L
IMAGEHDEMO.FOR

APPENDIX L

IMAGEHDEMO.FOR

```

cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c      This routine reads in the selected reference image from the
c      correlator memory and generates the gray scale histogram
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN'
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN'
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4),IMAGE(128,64)
INTEGER*2 IXBW,IYBW
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
integer hist(256)
CHARACTER*80 MSGBUF
EQUIVALENCE (BUF,IMAGE)
COMMON/BALLO/IXBW,IYBW
COMMON/CHAN/ITCHAN
DATA ITIMES/0/
DATA IXBW/0/,IYBW/256/
data ncol,nrow/64,64/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
ISTATUS=SYSS$ASSIGN('GRA0',GRCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN GRINNELL CHANNEL ASSIGN'
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      DO I=1,64
C      DO J=1,128
C      IMAGE(J,I)=IAND(I*J,'377'O)
C      ENDDO
C      ENDDO
C      DO J=1,128
C      IMAGE(J,1)='377'O
C      IMAGE(J,64)='377'O
C      ENDDO
C      do i=1,64
C      do j=1,64
C      image(j,i)=IAND(I*J,'377'O)
C      enddo
C      enddo
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C      ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),IOSB,,

```

IMAGEHDEMO.FOR

```

C      1BUF(1),%VAL(16384),,,)
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
C      ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
C      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
C      TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
C      MSGBUF= ' '
C      ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'I/O STATUS:', MSGBUF
C
C      IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
C      IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
C
C      ENDIF
C      WRITE(6,111)IMAGE
111    FORMAT(1X,10(1X,O6))
cccccccccccc
c      put out the histogram to the grinnell
c
      CALL GGHIST(IMAGE)
      DO I=1,64
      DO J=1,127
      IMAGE(J,I)=IAND(IMAGE(J,I),'377'O)
      ENDDO
      IMAGE(128,I)='34015'O
      ENDDO
      CALL PIX(IMAGE)
END
SUBROUTINE PIX(IBUFA)
IMPLICIT INTEGER*2 (A-Z)
EXTERNAL IO$WRITEVBLK
INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN'
INTEGER SYSS$QIOW,ITCHAN,J,IO$WRITEVBLK
INTEGER*2 OUT(12),IBUFA(1)
COMMON/CHAN/ITCHAN
COMMON/BALLO/IXBW,IYBW
DATA OUT/'17777'O,'120000'O,'107777'O,'24055'O,'26002'O,'44000'O,
1'50002'O,'54000'O,'64000'O,'74776'O,0,0/
OUT(11)=IXBW.OR.'44000'O
OUT(12)=IYBW.OR.'64000'O
J=SYSS$QIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(IO$WRITEVBLK)),,,
1OUT(1),%VAL(24),,,)
J=SYSS$QIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(IO$WRITEVBLK)),,,
1IBUFA(1),%VAL(16384),,,)
RETURN
END
SUBROUTINE GGHIST(IMAGE)
C *****
C
C      FUNCTION:
C
C      THIS ROUTINE PERFORMS A HISTOGRAM ON THE IMAGE
C
C      *****
C
C      ARGUMENTS:

```

IMAGEHDEMO.FOR

```

C
C *****
C
C      MODULE NOTES:
C
C *****
C
C      MODULE CREATION DATE:
C
C      DEC. 1985
C
C *****
C
C      MODULE MODIFICATION HISTORY:
C
C ***** EOH *****
C
C      EXTERNAL IO$WRITEVBLK
C      INCLUDE 'DISK$USERDISK:[SUBIMAGE]IMGTBL.CMN'
C      INCLUDE 'DISK$USERDISK:[SUBIMAGE]IOTBL.CMN'
C      COMMON/IMGSOURCE/IMGSOURCE      !IMAGE SOURCE STRING COMMON TO SUBCON
SUBINIT
C      CHARACTER*5 IMGSOURCE
C      INTEGER*4 OPRINP
C      INTEGER*4 CHECKING                !DUPLICATE IMAGE CHECK FUNCTION
C      LOGICAL*1 DISKINPUT              !DISK/TAPE INPUT FLAG .TRUE.=DISK
C      .FALSE.=TAPE
C      DATA DISKINPUT/.TRUE./ !ASSUME DISK INPUT
C      CHARACTER*4 CHARIN                !OPERATER RESPONSE
C      INTEGER*2 IACODE                  !VARIABLE TO CONTAIN OPERATOR FUN REQ
C      INTEGER*4 DSPUPD
C      LOGICAL*1 IIFLG                  !REWRITE HEADER2 LOGICAL FLAG
C      LOGICAL*1 TAPEDUP,OLDIMG/.FALSE./
C      DIMENSION LISTFSIZE(3),LISTSPEED(3),LISTFPOS(3)
C      DIMENSION LIST(8),ILIST(3),LISTMODE(2),LISTSPDT(2)
C      DIMENSION MLIST(5)
C      INTEGER*2 ERASEQ(15),IMAGE(NCOL+1,NROW),OUT(780)
C      INTEGER HIST(256)
C      DATA ERASEQ/'100000'O,'17777'O,'24240'O,'121040'O,
C      1'140001'O,'121000'O,'110000'O,'107777'O,'121000'O,
C      2'64000'O,'70377'O,'44000'O,'52377'O,'24040'O,'26000'O/
C      I=LIB$ERASEPAGE(1,1)              !ERASE SCREEN
C      IER = IFENCEY + IFENCEYSIZE - 1 !COMPUTE ENDING ROW FOR SUB IMAGE
C      IEC = IFENCEX + IFENCEXSIZE - 1 !COMPUTE ENDING COL FOR SUBIMAGE
C      IER=64
C      IEC=64
C      CALL MESSAGE('ENTER HISTOGRAM PLOT QUADRANT')
C      CALL ACCEPT(IACODE)                !GET THE OPERATORS REQ KEY
C      IF(IACODE.EQ.'0031'X)THEN
C      ERASEQ(10)='64400'O !LLA
C      ERASEQ(12)='44400'O !LEA
C      ENDIF
C      IF(IACODE.EQ.'0032'X)THEN

```

IMAGEDEMO.FOR

```

C      ERASEQ(10)='64400'O
C      ERASEQ(12)='44000'O
C      ENDIF
C      IF(IACODE.EQ.'0033'X)THEN
C      ERASEQ(10)='64000'O
C      ERASEQ(12)='44000'O
C      ENDIF
C      IF(IACODE.EQ.'0034'X)THEN
C      ERASEQ(10)='64000'O
C      ERASEQ(12)='44400'O
C      ENDIF
C      I=SYSSQIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(ISO$WRITEVBLK)),,,,
1ERASEQ,%VAL(30)),,,
C      IFENCEY=1
C      IFENCEX=1
C      DO ICOL=1,256
C      HIST(ICOL)=0
C      ENDDO
C      DO IROW=IFENCEY,IER
C      DO ICOL=IFENCEX,IEC
C      PRINT *, 'IMAGE(ICOL,IROW)=', IMAGE(ICOL,IROW), ICOL, IROW
C      HIST(IMAGE(ICOL,IROW)+1)=HIST(IMAGE(ICOL,IROW)+1)+1
C      ENDDO
C      ENDDO
C      LMAX=-1000000
C      LMIN=1000000
C      SKIP ZERO WHEN GETTING SCALE FOR HISTOGRAM (START AT 2)
C      DO ILOOK=2,256
C      IF(HIST(ILOOK).GT.LMAX)LMAX=HIST(ILOOK)
C      IF(HIST(ILOOK).LT.LMIN)LMIN=HIST(ILOOK)
C      ENDDO
C      IF(LMAX.EQ.LMIN)THEN
C      TYPE *, 'MAX IS THE SAME AS MIN.....!!!!!! ERROR.....'
C      RETURN
C      ELSE
C      TYPE*, 'MAX,MIN=', LMAX, LMIN
C      ENDIF
C      HIST(1)=(HIST(1)-LMIN)/(LMAX-LMIN)
C      IF(HIST(1).GT.255)HIST(1)=255 !THIS TAKES CARE OF ZERO
C      DO ILOOK=2,256
C      HIST(ILOOK)=255.*(FLOAT(HIST(ILOOK))-FLOAT(LMIN))/
1(FLOAT(LMAX)-FLOAT(LMIN))
C      ENDDO
C      IOUT=2
C      OUT(1)='50000'O
C      DO ILOAD=1,256
C      OUT(IOUT)=ERASEQ(12)+(ILOAD-1)
C      OUT(IOUT+1)=ERASEQ(10)
C      OUT(IOUT+2)='72000'O+HIST(ILOAD)
C      IOUT=IOUT+3
C      ENDDO
C      IOUT=IOUT-3
C      I=SYSSQIOW(%VAL(1),%VAL(GRCHAN),%VAL(%LOC(ISO$WRITEVBLK)),,,,
1OUT,%VAL(IOUT*2)),,,
C      RETURN
C      END

```

APPENDIX M

RWCMENT . FOR

APPENDIX M

RWCMENT.FOR

```

EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
  INTEGER*2 ERROR,ITIMES,NWORDS
CHARACTER*80 MSGBUF
  DATA ITIMES/0/,ERROR/0./
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
1  do 100 itest=1,5
      DO J=1,nwords
        if(itest.eq.1)BUF(J)=J
        if(itest.eq.2)buf(j)=0
        if(itest.eq.3)buf(j)='377'o
        if(itest.eq.4)buf(j)='125'o
        if(itest.eq.5)buf(j)='052'o
      ENDDO
      ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
      1BUF(1),%VAL(NBYTES),,,)
      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
        ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
        TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
        TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'QIO WRITE PARAMETER STATUS:',MSGBUF
        MSGBUF=' '
        ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
        IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
        TYPE *,'I/O STATUS:',MSGBUF
        IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN XT OUTPUT'
        IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
      ENDIF
100  continue
      ITIMES=ITIMES+1
      type *,'memory tested with the above',ERROR,
      1' errors.',ITIMES,
      1' ON',NWORDS, ' LOCATIONS'
      GO TO 1
    END

```

APPENDIX N

SELDAT.FOR

APPENDIX N

SELDAT.FOR

```

EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTAO',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
  TYPE *,'ENTER DATA PATTERN TO BE USED (377)'
  READ(5,55)IDAT
55  FORMAT(O3)
      DO J=1,nwords
      buf(j)=IDAT
      ENDDO
C    SET FIRST AND LAST IN EACH LINE TO ZERO
      DO J=1,NWORDS,128
      BUF(J)=0
      ENDDO
      DO J=64,NWORDS,128
      BUF(J)=0
      ENDDO
1    ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),
  1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,)
  1RBUF(1),%VAL(NBYTES),,,)
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
    ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
    TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
    TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
    TYPE *,'QIO WRITE PARAMETER STATUS:',MSGBUF
    MSGBUF=' '
    ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
    IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
    TYPE *,'I/O STATUS:',MSGBUF
    IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
    IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
    ENDIF
    ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),IOSB,,)
    1RBUF(1),%VAL(NBYTES),,,)
    IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
      ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)

```

SELDAT.FOR

```
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
TYPE *, ' ISTATUS=', ISTATUS, ' IOSB(1)=', IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO READ PARAMETER STATUS:', MSGBUF
MSGBUF=' '
ISTATUS=SYS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS) TYPE *, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1)) TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
DO I=1,NWORDS
  IF(IAND(BUF(I), '377'O).NE.IAND(RBUF(I), '377'O)) THEN
    WRITE(6,11) I, IAND(BUF(I), '377'O), IAND(RBUF(I), '377'O)
11  FORMAT('** MEMORY R/W ERROR AT ', I4, ' INPUT=', I4, 'O6,
1'    OUTPUT=', O6)
  ENDIF
ENDDO

ITIMES=ITIMES+1
type *, 'memory test complete with the above errors.', ITIMES,
1' ON', NWORDS, ' LOCATIONS'
GO TO 1
END
```

APPENDIX O

SEQRAMP.FOR

APPENDIX O

SEQRAMP.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C*****
*
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
BYTE DAC(64)
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
nbytes=2*nwords
N=1
155 TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
    READ(5,33)DAC(N)
33  FORMAT(I2)
    IF(DAC(N).GT.0)THEN
        N=N+1
        GO TO 155
    ENDIF
    N=N-1
        DO J=1,nwords
            BUF(J)='0'O
        ENDDO
    DO I=1,N
        TYPE *, 'DAC NUMBER=',DAC(I)
        DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
            buf(K)=(K-1)*2
            BUF(K)='377'O
        ENDDO
    ENDDO

C
ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
1BUF(1),%VAL(NBYTES),,,)
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)

```

SEQRAMP.FOR

```
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:', MSGBUF
MSGBUF= ' '
ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS) TYPE *, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1)) TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSS$QIOW(%VAL(1), %VAL(XTCHAN), %VAL(%LOC(IO$READVBLK)),
      1IOSB,,,
      1RBUF(1), %VAL(NBYTES),,,, )
END
```

APPENDIX P

SEQUENCE . FOR

APPENDIX P
SEQUENCE.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C*****
*
  EXTERNAL IO$WRITEVBLK,IO$READVBLK
  INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
  INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
  BYTE DAC(64)
  CHARACTER*80 MSGBUF
  DATA ITIMES/0/
  ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
  IF(.NOT.ISTATUS)TYPE *, 'ERROR IN AMD XT CHANNEL ASSIGN'
  NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
  nbytes=2*nwords
  N=1
155  TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
    READ(5,33)DAC(N)
33   FORMAT(12)
    IF(DAC(N).GT.0)THEN
      N=N+1
      GO TO 155
    ENDIF
    N=N-1
      DO J=1,nwords
        BUF(J)='0'O
      ENDDO
    TYPE *, 'ENTER DATA FOR DAC'
    READ(5,555)IDATA
555  FORMAT(O3)
    DO I=1,N
      TYPE *, 'DAC NUMBER=', DAC(I)
      DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128) -
        BUF(K)=IDATA
      ENDDO
    ENDDO
    DO J=1,NWORDS,128
      BUF(J)=0
    ENDDO
    DO J=64,NWORDS,128
      BUF(J)=0

```

SEQUENCE.FOR

ENDDO

```

ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(10$WRITEVBLK)),IOSB,,,
1RBUF(1),%VAL(NBYTES),,,,
IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSSGETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSSGETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:',MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(10$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,,
END

```

APPENDIX Q
SHORTLINE.FOR

APPENDIX Q

SHORTLINE.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C*****
*
  EXTERNAL IOS$WRITEVBLK,IOS$READVBLK
  INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
  INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
  BYTE DAC(64)
  CHARACTER*80 MSGBUF
  DATA ITIMES/0/
  ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
  IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
  nbytes=2*nwords
  N=1
C155 TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
C   READ(5,33)DAC(N)
33   FORMAT(I2)
     DAC(N)=31
C   IF(DAC(N).GT.0)THEN
C     N=N+1
C     GO TO 155
C   ENDIF
C   N=N-1
       DO J=1,nwords
       BUF(J)='0'O
       ENDDO

  I=1
C   DO I=1,N
C     TYPE *, 'DAC NUMBER=',DAC(I)
C       DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
C         BUF(K)='100'O
C       ENDDO
C     ENDDO
C       DO K=1+((DAC(I)-1)*128),((128+((DAC(I)-1)*128))/2)-1
C         BUF(K)=0
C       ENDDO
C       DO K=((128+((DAC(I)-1)*128))/2)-1,128+((DAC(I)-1)*128)
C         BUF(K)=0
C       ENDDO
C

```

SHORTLINE.FOR

```

C      DO I=31,8192-95,128
        DO I=35,40
          BUF(I)='377'O
        ENDDO

C      BUF(4000-128)='377'O

        ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),
        1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
        1BUF(1),%VAL(NBYTES),,,, )
        IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
          ISTATUS=SYSSGETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
          TYPE *, ' ISTATUS=',ISTATUS,'  IOSB(1)=' ,IOSB(1)
          TYPE *, ' ISTATUS=',ISTATUS,'  IOSB(1)=' ,IOSB(1)
          IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
          TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
          MSGBUF=' '
          ISTATUS=SYSSGETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
          IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
          TYPE *, 'I/O STATUS:',MSGBUF
          IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
          IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
        ENDIF
1111  ISTATUS=SYSSQIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
        1IOSB,,,
        1RBUF(1),%VAL(NBYTES),,,, )
        END

```

APPENDIX R
SSELDAT.FOR

APPENDIX R

SSELDAT.FOR

```

EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
  TYPE *,'ENTER DATA PATTERN TO BE USED (377)'
  READ(5,55)IDAT
55  FORMAT(O3)
      DO J=1,nwords
      buf(j)=IDAT
      ENDDO
C
1  SET FIRST AND LAST IN EACH LINE TO ZERO
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),
  1%VAL(%LOC(IO$WRITEVBLK)),IOSB,,
  1RBUF(1),%VAL(NBYTES),,,)
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'QIO WRITE PARAMETER STATUS:',MSGBUF
  MSGBUF=' '
  ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'I/O STATUS:',MSGBUF
  IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN XT OUTPUT'
  IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
  ENDIF
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),IOSB,,
  1RBUF(1),%VAL(NBYTES),,,)
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=',IOSB(1)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'QIO READ PARAMETER STATUS:',MSGBUF
  MSGBUF=' '
  ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)

```

SSELDAT.FOR

```
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS) TYPE *, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1)) TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
DO I=1,NWORDS
  IF(IAND(BUF(I), '377'O).NE.IAND(RBUF(I), '377'O)) THEN
    WRITE(6,11) I, IAND(BUF(I), '377'O), IAND(RBUF(I), '377'O)
11  FORMAT('** MEMORY R/W ERROR AT ', I4, ' INPUT=', 1X, O6,
1'      OUTPUT=', O6)
  ENDIF
ENDDO

ITIMES=ITIMES+1
type *, 'memory test complete with the above errors.', ITIMES,
1' ON', NWORDS, ' LOCATIONS'
GO TO 1
END
```

APPENDIX S

SSELDAT.FOR

APPENDIX S

SSELDAT.FOR

```

EXTERNAL IOS$WRITEVBLK,IOS$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTAO',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
  TYPE *,'ENTER DATA PATTERN TO BE USED (377)'
  READ(5,55)IDAT
55  FORMAT(O3)
      DO J=1,nwords
      buf(j)=IDAT
      ENDDO
C
1  SET FIRST AND LAST IN EACH LINE TO ZERO
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),
  1%VAL(%LOC(IOS$WRITEVBLK)),IOSB,,,
  1RBUF(1),%VAL(NBYTES),,,,
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'QIO WRITE PARAMETER STATUS:',MSGBUF
  MSGBUF=' '
  ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'I/O STATUS:',MSGBUF
  IF(.NOT.ISTATUS)TYPE*,'PARAMETER ERROR IN XT OUTPUT'
  IF(.NOT.IOSB(1))TYPE *,'I/O ERROR IN XT OUTPUT'
  ENDIF
  ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IOS$READVBLK)),IOSB,,,
  1RBUF(1),%VAL(NBYTES),,,,
  IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
  ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
  TYPE *,' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
  IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
  TYPE *,'QIO READ PARAMETER STATUS:',MSGBUF
  MSGBUF=' '
  ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)

```

SSELDAT.FOR

```
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:', MSGBUF
IF(.NOT.ISTATUS) TYPE *, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1)) TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
DO I=1,NWORDS
  IF(IAND(BUF(I), '377'O).NE.IAND(RBUF(I), '377'O)) THEN
    WRITE(6,11) I, IAND(BUF(I), '377'O), IAND(RBUF(I), '377'O)
11  FORMAT('** MEMORY R/W ERROR AT ', I4, ' INPUT=', 1X, 06,
1'      OUTPUT=', 06)
  ENDIF
ENDDO

ITIMES=ITIMES+1
type *, 'memory test complete with the above errors.', ITIMES,
1' ON', NWORDS, ' LOCATIONS'
GO TO 1
END
```

APPENDIX T
SSEQUENCE.FOR

APPENDIX T

SSEQUENCE.FOR

```

C*****
*
C
C   THIS ROUTINE IS USED TO SELECT WHICH LEDS OR LINES IS TO BE TURNED
C   ON. SEVERAL CAN BE SELECTED. ENTER AS MANY AS YOU WANT THE ENTER
C   A NEGATIVE NUMBER.
C
C   THIS ROUTINE IS THE SAME AS SEQUENCE EXCEPT THAT IT DOESN'T ZERO OUT
C   THE LAST VALUE IN EACH ROW ON EACH LED
C*****
*
EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
BYTE DAC(64)
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTA0',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *, 'ERROR IN AMD XT CHANNEL ASSIGN'
NWORDS=8192      ! NUMBER OF MEMORY LOCATIONS PER REFERENCE 128X64
nbytes=2*nwords
N=1
155  TYPE*, 'ENTER DAC NUMBER TO TURN ON, ENTER NEGATIVE TO COMPLETE'
    READ(5,33)DAC(N)
33   FORMAT(I2)
    IF(DAC(N).GT.0)THEN
        N=N+1
        GO TO 155
    ENDIF
    N=N-1
        DO J=1,nwords
            BUF(J)='0'0
        ENDDO
    TYPE *, 'ENTER DATA FOR DAC'
    READ(5,555)IDATA
555  FORMAT(O3)
    DO I=1,N
        TYPE *, 'DAC NUMBER=',DAC(I)
        DO K=1+((DAC(I)-1)*128),128+((DAC(I)-1)*128)
            BUF(K)=IDATA
        ENDDO
    ENDDO
    ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,,
    1BUF(1),%VAL(NBYTES),,,,,)

```

SSEQUENCE.FOR

```

IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
TYPE *, ' ISTATUS=',ISTATUS,' IOSB(1)=' ,IOSB(1)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
MSGBUF=' '
ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
IF(.NOT.ISTATUS) TYPE *, 'ERROR IN CALL TO $GETMSG'
TYPE *, 'I/O STATUS:',MSGBUF
IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
ENDIF
1111 ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(ISO$READVBLK)),
1IOSB,,,
1RBUF(1),%VAL(NBYTES),,,, )
END

```

APPENDIX U
SWITCHTEST.FOR

APPENDIX U

SWITCHTEST.FOR

```

EXTERNAL IO$WRITEVBLK,IO$READVBLK
INTEGER*2 RBUF(8192),BUF(8192),IOSB(4)
INTEGER SYSS$ASSIGN,XTCHAN,SYSS$QIOW,SYSS$GETMSG
CHARACTER*80 MSGBUF
DATA ITIMES/0/
ISTATUS=SYSS$ASSIGN('XTAO',XTCHAN,,)
IF(.NOT.ISTATUS)TYPE *,'ERROR IN AMD XT CHANNEL ASSIGN'
  type *,'Enter number of memory locations to test.'
  accept*,nwords
  nbytes=2*nwords
c1  do 100 itest=1,5
1  continue
      DO J=1,nwords
      buf(j)=(j-1)*2
c      if(itest.eq.1)BUF(J)=J
c      if(itest.eq.2)buf(j)=0
c      if(itest.eq.3)buf(j)='377'o
c      if(itest.eq.4)buf(j)='125'o
c      if(itest.eq.5)buf(j)='052'o
      ENDDO
C      ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$WRITEVBLK)),IOSB,,
C      1BUF(1),%VAL(NBYTES),,,)
C      IF(.NOT.ISTATUS.OR..NOT.IOSB(1))THEN
C      ISTATUS=SYSS$GETMSG (%VAL(ISTATUS), MSGLEN, MSGBUF,,)
C      TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
C      TYPE *, ' ISTATUS=',ISTATUS, ' IOSB(1)=' ,IOSB(1)
C      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'QIO PARAMETER STATUS:',MSGBUF
C      MSGBUF=' '
C      ISTATUS=SYSS$GETMSG (%VAL(IOSB(1)), MSGLEN, MSGBUF,,)
C      IF(.NOT.ISTATUS) TYPE *,'ERROR IN CALL TO $GETMSG'
C      TYPE *, 'I/O STATUS:',MSGBUF
C
C      IF(.NOT.ISTATUS)TYPE*, 'PARAMETER ERROR IN XT OUTPUT'
C      IF(.NOT.IOSB(1))TYPE *, 'I/O ERROR IN XT OUTPUT'
C
C      ENDIF
1111 ISTATUS=SYSS$QIOW(%VAL(1),%VAL(XTCHAN),%VAL(%LOC(IO$READVBLK)),
1IOSB,,
1RBUF(1),%VAL(NBYTES),,,)
      DO I=1,NWORDS
      IF(IAND(BUF(I),'377'O).NE.IAND(RBUF(I),'377'O))THEN
      WRITE(6,11)I,IAND(BUF(I),'377'O),IAND(RBUF(I),'377'O)

```

SWITCHTEST.FOR

```
11      FORMAT('** MEMORY R/W ERROR AT',O6,' INPUT=',1X,O6,
1'      OUTPUT=',O6)
        ENDIF
        ENDDO

100     continue
        ITIMES=ITIMES+1
        type *, 'MEMORY SWITCH test complete with the above errors.',
1' ON',NWORDS, ' LOCATIONS'
        GO TO 1111
        END
```

```

CONP      EQU      0AF83H
PORA      EQU      0AF80H
PORB      EQU      0AF81H
PORC      EQU      0AF82H

DATA      SEGMENT PUBLIC 'DATA'
TEMP      DW      4
DATA      ENDS

DGROUP    GROUP     DATA
CODE      SEGMENT   'CODE'
ASSUME    CS:CODE,DS:DGROUP,SS:DGROUP
PUBLIC    WRITEC
WRITEC    PROC      FAR
          PUSH      BP
          MOV       BP,SP
          CLI

          MOV       DX,CONP
          MOV       AL,10      ; RESET READ. PIN 12,PC5
          OUT       DX,AL
          MOV       AL,8      ; RESET READY. PIN 13
          OUT       DX,AL

          MOV       DX,PORA

          LES       BX,DWORD PTR [BP+10]      ; ADDRESS OF PARAM 1 ES:[BX]
          MOV       CX,ES:[BX]
          MOV       TEMP,CX

          LES       BX,DWORD PTR [BP+6]      ; ADDRESS OF PARAM 2 ES:[BX]

          MOV       CX,0

NEXT:
          MOV       AL,ES:[BX]                ; VALUE OF PARAM 1 IN AL

          OUT       DX,AL
          INC       BX
          INC       BX
          INC       BX
          INC       BX                      ; INCREMENT DATA ADDRESS
          INC       CX
          CMP       CX,TEMP
          JNE       NEXT

          MOV       DX,CONP
          MOV       AL,9                ; SET READY PIN 13
          OUT       DX,AL

          MOV       SP,BP
          POP       BP
          RET       08H
WRITEC    ENDP
CODE      ENDS
END

```

```

CONP      EQU      0AF83H
PORA      EQU      0AF80H
PORB      EQU      0AF81H
PORC      EQU      0AF82H

```

```

DATA      SEGMENT PUBLIC 'DATA'
TEMP2     DW      0
DATA      ENDS

```

```

DGROUP    GROUP     DATA
CODE      SEGMENT   'CODE'
ASSUME    CS:CODE,DS:DGROUP,SS:DGROUP
PUBLIC    READC
PROC      FAR
PUSH      BP

```

```

MOV       BP,SP
CLI
MOV       DX,CONP

```

```

MOV       AL,11      ; SET READ. PIN 12
OUT       DX,AL
MOV       AL,8        ; reSET READY LINE. PIN 13
OUT       DX,AL
MOV       AL,9
OUT       DX,AL
MOV       CX,0

```

WAIT:

```

INC       CX
CMP       CX,25
JB        WAIT

```

```

LES       BX,DWORD PTR [BP+10] ; ADDRESS OF 1ST PARAM
MOV       CX,ES:[BX]
MOV       TEMP2,CX             ;STORE NUMBER TO READ
LES       BX,DWORD PTR [BP+6]  ; ARRAY ADDRESS
MOV       CX,0                 ; INITIALIZE COUNTER
MOV       AL,8
OUT       DX,AL

```

CHECK:

```

MOV       DX,PORC

```

CHECK2:

```

IN        AL,DX

```

```

CMP       AL,32

```

```

JZ        CHECK2

```

FIRST:

```

MOV       DX,PORB
IN        AL,DX

```

```

MOV       ES:[BX],AL

```

```

INC       BX
INC       BX
INC       BX
INC       BX
INC       CX
CMP       CX,TEMP2
JB        CHECK

```

```

MOV       DX,CONP

```

```
MOV    DX,0
MOV    AH,0
```

```
MOV    SP,BP
POP    BP
RET    08H
```

```
READC ENDP
CODE ENDS
END
```

```

CONP      EQU      0AF83H
PORA      EQU      0AF80H
PORB      EQU      0AF81H
PORC      EQU      0AF82H

```

```

DATA      SEGMENT PUBLIC 'DATA'
DATA      ENDS

```

```

DGROUP    GROUP     DATA
CODE      SEGMENT   'CODE'
          ASSUME CS:CODE,DS:DGROUP,SS:DGROUP

```

```

PUBLIC    INITC

```

```

INITC     PROC      FAR
          PUSH      BP

```

```

          MOV       BP,SP
          CLI
          MOV       DX,CONP
          MOV       AL,167
          OUT       DX,AL

```

```

          MOV       SP,BP
          POP       BP
          RET       00H

```

```

          INITC ENDP
          CODE ENDS
          END

```

```

DATA    SEGMENT PUBLIC 'DATA'
DATA    ENDS

DGROUP  GROUP          DATA
CODE    SEGMENT        'CODE'
        ASSUME CS:CODE,DS:DGROUP,SS:DGROUP

PUBLIC  IAND
IAND    PROC          FAR
        PUSH         BP

        MOV          BP,SP
        LES          BX,DWORD PTR [BP+6]
        MOV          AL,ES:[BX]
        MOV          AH,0
        MOV          DX,0
        MOV          SP,BP
        POP          BP
        RET          04H

IAND    ENDP
CODE    ENDS
END

```

```

CONP      EQU      0AF83H
PORA      EQU      0AF80H
PORB      EQU      0AF81H
PORC      EQU      0AF82H

```

```

DATA      SEGMENT PUBLIC 'DATA'
DATA      ENDS

```

```

DGROUP    GROUP     DATA
CODE      SEGMENT   'CODE'
          ASSUME CS:CODE,DS:DGROUP,SS:DGROUP

```

```

PUBLIC    LOOP
LOOP

```

```

PROC      FAR
PUSH      BP

MOV       BP,SP
CLI
MOV       DX,CONP
MOV       AL,11      ; SET READ. PIN 12
OUT       DX,AL
MOV       AL,9       ; SET READY LINE. PIN 13
OUT       DX,AL

MOV       DX,PORB
IN        AL,DX

```

```

CHECK:

```

```

MOV       DX,PORC

IN        AL,DX
CMP       AL,48
JZ        CHECK

MOV       DX,0
MOV       AH,0

```

```

MOV       SP,BP
POP       BP
RET       08H

```

```

LOOP     ENDP
CODE     ENDS
END

```

```

#include:'\forintf.h'
PROGRAM KEYPEAK
INTEGER*2 IV,IN,ic,iy,ix,ival
integer*2 x1,x2,y1,y2,val,ipeak
integer*4 value,buffer(256),posx(100),posy(100)
character BUF(256),TEMPC
INTEGER*2 TEMP
INTEGER*2 MAP,COLOR,START,LENGTH
EQUIVALENCE (TEMP,TEMPC)
DATA MAP,COLOR,START,LENGTH/3,2,0,256/
i=init(620)
call chan(2)
C call auto
call sync(1)
C call snap(1)
CALL CGRAB(1)
CALL LUTM(MAP)
1 write(*,'(40H ENTER MAXIMUM MAPPING THRESHOLD.(0-255))')
READ(*,*)IPEAK
do 209 iy=1,256
IF(IY-1.LT.IPEAK)THEN
TEMP=IY-1
BUF(IY)=TEMPC
ELSE
TEMP=255
BUF(IY)=TEMPC
ENDIF
209 continue

CALL LUTD(MAP,COLOR,START,LENGTH,BUF)
GO TO 1
C call pexit
C stop
end

```

```

$include: '\forintf.h'
$LARGE
PROGRAM CREADI
EXTERNAL INITC,READC
INTEGER INITC,READC
INTEGER*2 IV,IN,ic,iy,ix,ival
integer*2 x1,x2,y1,y2,val,ipeak
integer*4 value,buffer(256),posx(100),posy(100)
INTEGER DAT(8192),INUM
I=INITC()
INUM=8192
I=READC(INUM,DAT)
WRITE(*,'(22H AFTER REFERENCE READ.)')
i=init(620)
ival=2
call chan(ival)
ival=1
call sync(ival)
call quadm(1)
call dquad(0)
X1=0
Y1=0
X2=511
Y2=511
CALL RECT(X1,Y1,X2,Y2)
DO 109 IY=1,64
DO 108 IX=1,128
VALUE=DAT(IX+(IY-1)*128)
call pixw((ix-1)*2,(iy-1)*2,value)
call pixw((ix-1)*2,((iy-1)*2)+1,value)
call pixw(((ix-1)*2)+1,((iy-1)*2)+1,value)
call pixw(((ix-1)*2)+1,(iy-1)*2,value)
108 continue
109 continue
write(*,'(18H IMAGE ON DISPLAY.)')

call pexit
stop
end

```

```

$include: '\\forintf.h'
$LARGE

PROGRAM STOREIMG
EXTERNAL INITC,READC,ITODSK
INTEGER INITC,READC
INTEGER*2 ITODSK,ISTATUS
INTEGER*2 IV,IN,ic,iy,ix,ival
integer*2 x1,x2,y1,y2,val,ipeak
INTEGER*2 BSIZE,QUAD
CHARACTER*13 FNAME
CHARACTER WORKBUFFER(4096)
integer*4 value,buffer(256),posx(100),posy(100)
INTEGER DAT(8192),INUM
DATA BSIZE/4096/
I=INITC()
INUM=8192
I=READC(INUM,DAT)
WRITE(*,'(22H AFTER REFERENCE READ.)')
i=init(620)
ival=2
call chan(ival)
ival=1
call sync(ival)
call quadm(1)
call dquad(0)
X1=0
Y1=0
X2=511
Y2=511
CALL RECT(X1,Y1,X2,Y2)
DO 109 IY=1,64
DO 108 IX=1,128
VALUE=DAT(IX+(IY-1)*128)
call pixw(ix-1,iy-1,value)
108 continue
109 continue
write(*,'(18H IMAGE ON DISPLAY.)')
FNAME='REFER.IMG'
ISTATUS=ITODSK(BSIZE,QUAD,FNAME,WORKBUFFER)
call pexit
stop
end

```

```

C.....
C
C      THIS ROUTINE IS THE CORRELATOR REFERENCE MEMORY READ/WRITE
C      TEST ROUTINE
C.....
$LARGE
      PROGRAM RWCMEM
      PROGRAM MAIN
      EXTERNAL READC,WRITEC,IAND,INIT
      INTEGER INUM,LOOP,READC,WRITEC,IAND,INIT
      INTEGER DAT(8192),DAT2(8192)

      DO 2 I=1,8192
      DAT(I)=I
C      DAT(I)=255
2      CONTINUE

      I=INITC()
      INUM=8192
1      CONTINUE
      I=WRITEC(INUM,DAT)
      I=READC(INUM,DAT2)
      DO 100 I=1,INUM
      II=IAND(I)
100     IF(DAT2(I).NE.II)WRITE(*,'(1X,2I6)')II,DAT2(I)
      CONTINUE
      WRITE(*,'(36H COMPLETE WITH THE ABOVE DIFFERENCES)')
      GO TO 1

      END

```

```
$include: '\forintf.h'
```

```
$LARGE
```

```
PROGRAM DOWNIMG  
EXTERNAL INITC,READC,ITODSK,IFRDSK,WRITEC  
INTEGER INITC,READC,IFRDSK,WRITEC  
INTEGER*2 ITODSK,ISTATUS  
INTEGER*2 IV,IN,ic,iy,ix,ival  
integer*2 x1,x2,y1,y2,val,ipeak  
INTEGER*2 BSIZE,QUAD  
CHARACTER*13 FNAME  
CHARACTER WORKBUFFER(4096)  
integer*4 value,buffer(256),posx(100),posy(100)  
INTEGER DAT(8192),INUM,IMAGE(128,64)  
EQUIVALENCE(DAT,IMAGE)  
DATA BSIZE/4096/,QUAD/0/
```

```
i=init(620)  
ival=2  
call chan(ival)  
ival=1  
call sync(ival)  
call quadm(1)  
call dquad(0)  
X1=0  
Y1=0  
X2=511  
Y2=511  
CALL RECT(X1,Y1,X2,Y2)  
FNAME='REFER.IMG'  
ISTATUS=IFRDSK(BSIZE,QUAD,FNAME,WORKBUFFER)  
DO 109 IY=1,64  
DO 108 IX=1,128  
VALUE=IpixR(ix-1,iy-1)  
IMAGE(IX,IY)=VALUE  
108 continue  
109 continue  
write(*,'(19H DOWNLOADING IMAGE.)')  
I=INITC()  
write(*,'(13H AFTER INITC.)')  
INUM=8192  
I=WRITEC(INUM,DAT)  
WRITE(*,'(19H DOWNLOAD COMPLETE.)')  
end
```

```
*include: '\forintf.h'  
PROGRAM RTIME  
INTEGER*2 STATE, CHANNEL, MODE  
STATE=1  
MODE=1  
CHANNEL=2  
CALL CGRAB(MODE)  
CALL SYNC(MODE)  
CALL CHAN(CHANNEL)  
END
```

```

C .....
C
C THIS ROUTINE READS CORRELATOR REFERENCE MEMORY AND COMPARES IT TO
C AND INCREASING SEQUENCE
C .....

```

```

PROGRAM RCMEM
EXTERNAL INITC,READC
INTEGER INUM,INITC,READC
INTEGER DAT(8192)

I=INITC()
INUM=5
READ(*,'(I4)')INUM
write(*,'(11H Starting..)')
1 CONTINUE
I=READC(INUM,DAT)
DO 3 I=1,INUM
c IF(I.LT.5)WRITE(*,'(1X,6H**OK**',2I6)')I,DAT(I)
II=IAND(I)
IF(DAT(I).NE.II) WRITE(*,'(1X,2I6)')II,DAT(I)
3 CONTINUE
write(*,'(1X,36H complete with the above differences)')
GO TO 1

END

```

```
#include:'\forintf.h'
```

```
C  
C  
C  
C
```

```
THIS ROUTINE IS USED TO STOP THE FRAME BUFFER IMAGE FROM BEING  
WRITTEN CONTINUOUSLY
```

```
PROGRAM STOP  
INTEGER*2 STATE  
STATE=0  
CALL CGRAB(STATE)  
call snap(1)  
END
```

```
*include: 'Nforintf.h'
PROGRAM HISTO
integer*4 value,buffer(256)
i=init(620)
call chan(2)
call sync(1)
call snap(1)
value=ihisto(buffer)
call dhisto(value,30.470,200.0,0,0,buffer)
call pexit
stop
end
```

```

$include: '\forintf.h'
PROGRAM PEAK
INTEGER*2 IV,IN,ic,iy,ix,ival
integer*2 x1,x2,y1,y2,val,ipeak
integer*4 value,buffer(256),posx(100),posy(100)
character*1 box
218 format(a1)
i=init(620)
call chan(2)
call auto
call sync(1)
call snap(1)

C      value=0
C      do 109 iy=1,512
C      do 108 ix=1,512
C      call pixw(ix,iy,value)
C108 continue
C109 continue
C      IX=256
C      IY=256
C      VALUE=100
C      CALL PIXW(IX,IY,VALUE)
C      write(*,'(16H AFTER WRITING...)')

      ipeakcnt=1
      ipeak=-255
      do 209 iy=1,512
      do 208 ix=1,512
      ival=ipixr(ix,iy)
      if(ival.gt.ipeak)then
      ipeak=ival
      posx(ipeakcnt)=ix
      posy(ipeakcnt)=iy
      endif
C      write(*,*) ix,ival
208 continue
209 continue
      write(*,*)ipeak,posx(1),posy(1)
      call pexit
      stop
      end

```

```

C.....
C
C   THIS ROUTINE IS USED TO WRITE AN INCREASING SEQUENCE TO THE
C   CORRELATOR REFERENCE MEMORY
C.....

```

```

      PROGRAM WCMEM
      EXTERNAL WRITEC,INITC
      INTEGER INUM,WRITEC,INITC
      INTEGER DAT(8192)

      DO 2 I=1,8192
      DAT(I)=I
C      DAT(I)=255
2      CONTINUE

      I=INITC()
      INUM=8192
1      CONTINUE
      I=WRITEC(INUM,DAT)
      WRITE(*,'(1X,/,3(1X,I6))')I,INUM,DAT(1)
      GO TO 1

      END

```

```

C.....
C
C      This routine is used for loading a constant into correlator reference
C      memory.
C.....

```

```

      PROGRAM SELDAT
      EXTERNAL WRITEC,INITC
      INTEGER INUM,WRITEC
      INTEGER DAT(8192)
      write(*,'(41H ENTER DATA VALUE TO BE LOADED (0 - 255).)')
      read(*,'(I3)')IDAT
      DO 2 I=1,8192
      DAT(I)=IDAT
C      DAT(I)=255
2      CONTINUE

      I=INITC()
      INUM=8192
1      CONTINUE
      I=WRITEC(INUM,DAT)
      WRITE(*,'(1X,/,3(1X,I6))')I,INUM,DAT(1)
      GO TO 1

      END

```

DISTRIBUTION LIST

	<u>Copies</u>
Director U.S. Army Research Office SLCRO-PH P. O. Box 12211 Research Triangle Park, NC 27709-2211	1
Director U.S. Army Research Office SLCRO-ZC P. O. Box 12211 Research Triangle Park, NC 27709-2211	1
Headquarters Department of the Army DAMA-ARR Washington, DC 20310-0632	1
Headquarters OUSD&E ATTN: Dr. Ted Berlincourt The Pentagon Washington, DC 20310-0632	1
Defense Advanced Research Projects Agency Defense Sciences Office Electronics Systems Division ATTN: Dr. John Neff 1400 Wilson Boulevard Arlington, VA 22209	1
Commander U.S. Army Foreign Science and Technology Center AIAST-RA 220 Seventh Street NE Charlottesville, VA 22901-5396	1
Commander U.S. Army Strategic Defense Command DASD-H-V P. O. Box 1500 Huntsville, AL 35807-3801	1
Director, URI University of Rochester College of Engineering and Applied Science The Institute of Optics Rochester, NY 14627	1

DISTRIBUTION LIST (Concluded)

	<u>Copies</u>
Director, JSOP University of Arizona Optical Science Center Tucson, AZ 85721	1
US Army Materiel System Analysis Activity ATTN: AMXSY-MP Aberdeen Proving Ground, MD 21005	1
IIT Research Institute ATTN: GACIAC 10 W. 35th Street Chicago, IL 60616	1
AMSMI-RD, Dr. W. McCorkle	1
Dr. R. Rhoades	1
-RD-RE, Dr. R. Hartman	1
Dr. J. Bennett	1
-RD-RE-OP, Dr. R. Johnson	5
-RD-AS, Dr. R. Powell	1
-RD-AS-SS, Mr. R. Farmer	1
Mr. R. Sims	5
-RD-CS-R	15
-RD-CS-T, Record Copy	1
-GC-IP, Mr. Bush	1

END

DATE

FILMED

DTIC

9-88