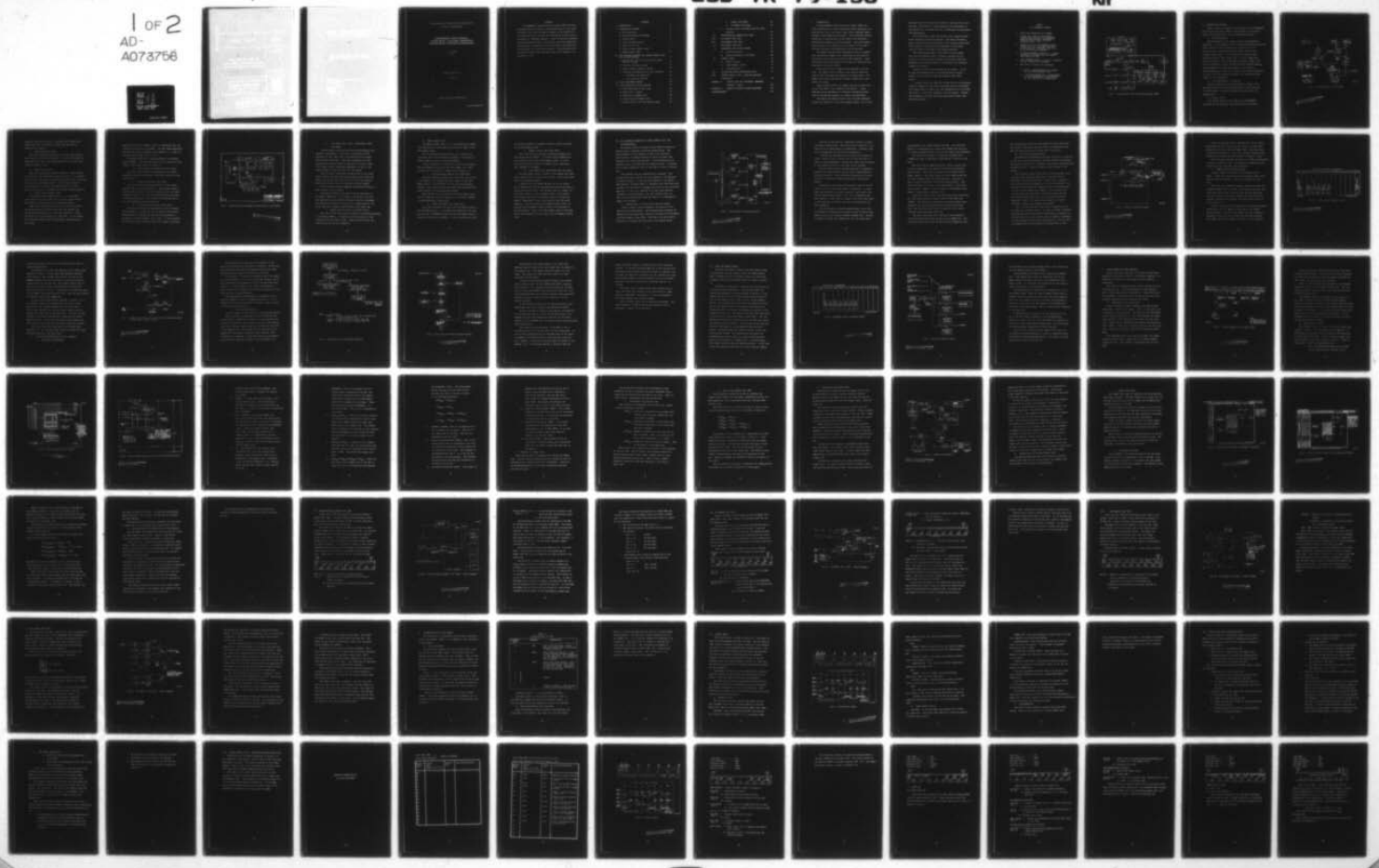
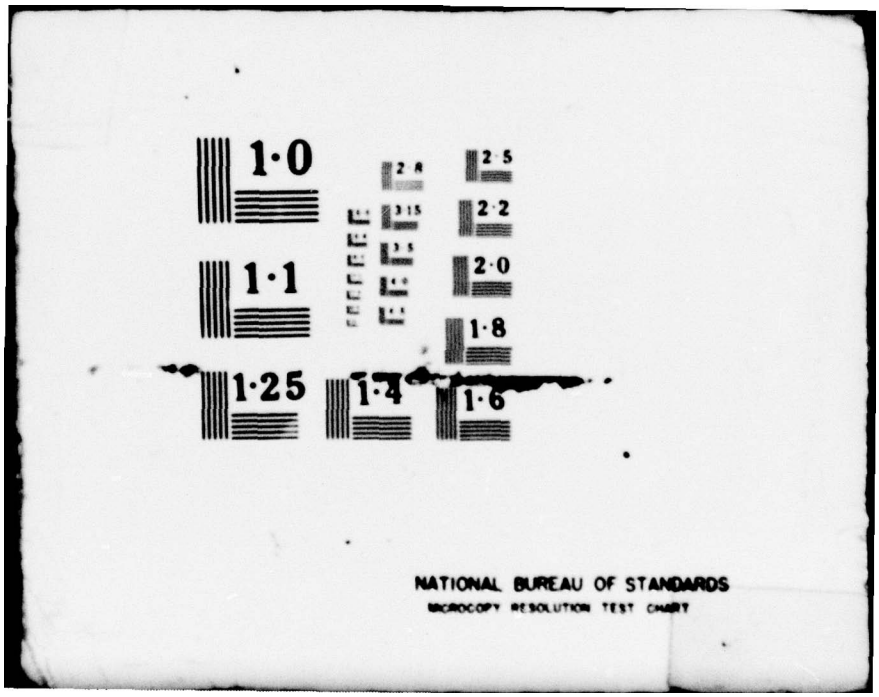
AD-A073 756 MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB F/G 9/2
PROGRAMMABLE VIDEO DIGITIZING SYSTEM (PVDS) - FUNCTIONAL DESCRI--ETC(U)
JUN 79 L E EATON F19628-78-C-0002
UNCLASSIFIED ETS-46 ESD-TR-79-156 NI

1 OF 2
AD-
A073756




1·0
1·1
1·25
1·4
1·6
2·5
2·2
2·0
1·8
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

ETS-40

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

PROGRAMMABLE VIDEO DIGITIZING SYSTEM (PVDS) – FUNCTIONAL DESCRIPTION AND PRELIMINARY SOFTWARE CONSIDERATIONS

L. E. EATON

Group 94

PROJECT REPORT ETS-46

5 JUNE 1979

LEXINGTON MASSACHUSETTS

ABSTRACT

A Programmable Video Digitizing System (PVDS) has been developed that will digitize selected portions of the return TV analog video from the Ebsicon camera at the GEODSS ETS. The digitized video can then be processed by either a microprocessor or a Modcomp Computer. From the data, the software may then determine object centroiding, night sky background calculations, signal/noise ratios and object rate/direction calculations for closed loop tracking. The system may also be used to process digitized data from future sensors, i.e., infrared or CCD.

CONTENTS

I.	INTRODUCTION	1
II.	SYSTEM BLOCK DIAGRAM	5
	A. Microprocessor$_A$	5
	B. Microprocessor$_B$ and Modcomp	5
	C. Control Panel	5
	D. Video Processing Logic	7
	E. Video A/D and D/A	7
	F. System Control Basket (SCB)	7
	G. Memory Basket (MB)	11
III.	THE INTERFACE BETWEEN THE SYSTEM CONTROL LOGIC AND MICROPROCESSOR$_A$	13
	A. Address Bus and Bi-Directional Tri-State Data Bus Design	13
	B. Priority Interrupt Design	17
	C. Common Register Address Decode	19
	D. Software Considerations for the Interface	21
	1. Accessing the Registers	21
	2. The Interrupt Handler	23
IV.	FOUR PORT MEMORY DESIGN	28
V.	VIDEO FORMAT AND VIDEO SAMPLING	32
	A. PVDS Definition of the Video	32
	B. Definition of Terms	34
	C. Defining the Sample Area	40
	D. Controlling the Sample Area	43
	E. Sending Data to the Fast Memory Cards	45

1. Summed Field Mode 46
2. Un-Summed Field Mode 46
F. Software Timing Considerations for Data Sampling 50
VI. MICROPROCESSOR$_A$ MEMORY PORT (MAP) 52
VII. A/D MEMORY PORT (ADP) 56
VIII. FAST MEMORY CARD (FMC) 60
IX. BOX SAMPLE CARD (BSC) 63
X. INTERRUPTS USED IN THE SYSTEM 67
A. Interrupt Levels 67
B. Enabling/Disabling Interrupts 68
XI. CONTROL PANEL 70
A. Memory Control 70
B. BOX1/CURSOR1 Control 72
C. Considerations 73
XII. INITIAL AND FINAL SYSTEM OBJECTIVES 75
XIII. SYSTEM CONTROL LOGIC - DETAILED REGISTER DESCRIPTION 79
APPENDIX A: INITIALIZING THE PERIPHERAL INTERFACE ADAPTER'S (PIA'S) 129
APPENDIX B: EXAMPLE PROGRAM TO ACCESS REGISTERS 132
ACKNOWLEDGEMENT 134

I. INTRODUCTION

The Programmable Video Digitizing System (PVDS) was first conceived as a signal processing system operating on a selected portion of the return video from an EBSICON camera mounted on the GEODSS ETS telescope. The preliminary design called for a fixed portion of the video to be digitized, stored and processed by a microprocessor. The fixed sample area would be a 16 element by 16 line area, on the video, nominally placed at the boresight of the TV picture.

The design has since been expanded to allow a variable size and position sample area, selectable by the operator, the size of which can be as large as 4096 elements. Each video pixel will be digitized to an eight bit word with the digitizing rate being 1Ø Mhz.

There can be up to 3 sample areas defined at any one time. The sample area is drawn on the video so that the operator may view where the sampling is taking place. The sample area can be defined with either a drawn box on the video or a cursor.

Under program control the sample area can be read into one of four 4096 x 8 bit memories (4K binary). These memories can be switched to an external processing device such as the Modcomp computer or another microprocessor.

The memory and memory control logic has been designed around the concept of a four port memory system. Up to four

different data sources can have access to the four 4K x 8 bit memories. The control of this access will be maintained in the software of the System Control & Bookkeeping Microprocessor ($Microprocessor_A$).

The PVDS design, then, has evolved into a sophisticated buffer switching and storage system. The buffer switching, memory control and data processing is all software controlled. This allows for flexibility and expansion of the system. Listed in Table 1 are some of the design features.

The first design phase of the PVDS will only handle data from the EBSICON video. Software routines will be written to do such things as object centroiding, night sky background, and rate/directional calculations for automatic closed loop tracking.

Figure 1 gives a block diagram of the system. It can be seen that expandability of the system would include a memory port for the CCD video and possibly some other sensor.

A caution should be observed in that this is a preliminary report as opposed to the final functional use of this system. The reader should be aware that the implementation of the PVDS may differ, in the final analysis, from the report. However, the philosophy of design and logic description should stay relatively stable.

TABLE 1

PVDS HARDWARE FEATURES

1. DATA FLOW CONTROLLED BY SOFTWARE

2. DIGITAL BUS SWITCH ALLOWS MEMORY'S ADDRESS AND DATA BUS TO BE SWITCHED UNDER PROGRAM CONTROL (HARDWARE SWITCHING TIME ≈ 1μS)

3. BLOCKS OF 4K X 8 BIT MEMORIES CAN BE LOADED WITH REAL TIME DATA AND THEN BE SWITCHED TO A PROCESSOR UNIT (i.e., MICRO-PROCESSOR$_B$ OR MODCOMP)

4. BY TOGGLING MEMORY BLOCKS (i.e., DOUBLE BUFFERING) PROCESSOR CAN HAVE LONGER DATA REDUCTION TIME

5. VIDEO SAMPLING AREA (i.e, SIZE & POSITION) CAN BE CONTROLLED BY SOFTWARE

6. TWO TYPES OF DIGITAL BUS SWITCHING

   a. ON/OFF - ALLOWS ACCESS TO A COMMON BUS BY VARIOUS MEMORY PORTS

   b. 3 POSITION/CENTER OFF - ALLOWS MEMORY TO BE ACCESSED WHILE DATA TRANSFER IS STILL HAPPENING ON COMMON BUS

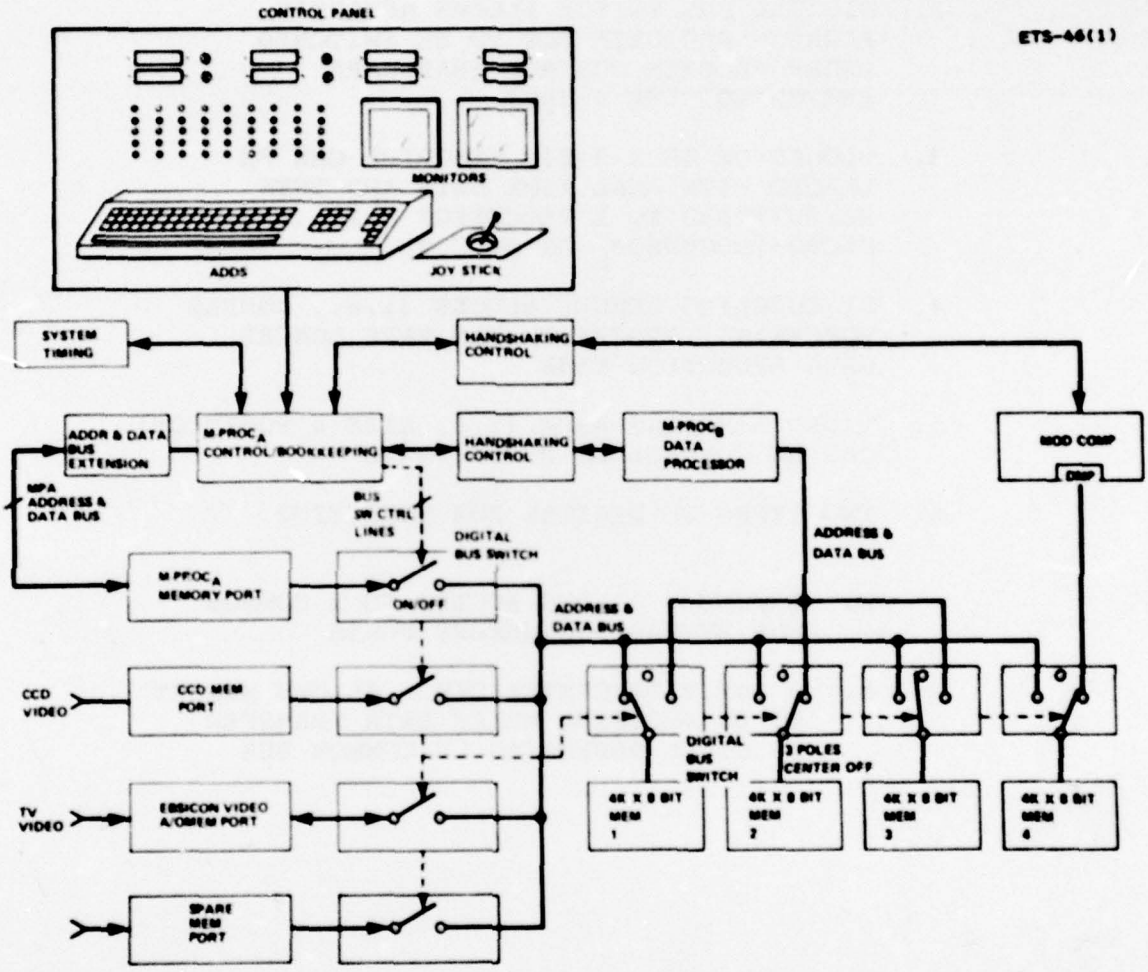


Fig.1. Programmable Video Digitizing System (PVDS).

II. SYSTEM BLOCK DIAGRAM

Figure 2 shows a hardware block diagram of the Programmable Video Digitizing System (PVDS). This section of the report gives a brief description of the blocks within Figure 2.

A. $Microprocessor_A$

$Microprocessor_A$ is the system control and bookkeeping processor. It will handle all of the operator/system interaction, logic interface, task coordination, all the I/O to the Control Panel, including monitoring switch closures and lighting system LEDS. It will control the memory buffer switching and the video sample area. It will communicate with the operator via an alphanumeric Terminal (ADDS), and will be the master control for synchronizing $Microprocessor_B$ and the Modcomp for doing data sampling and reduction. $Microprocessor_A$ will also have access to the sampling memories via one of the four memory ports which can be used for data reduction and diagnostics.

B. $Microprocessor_B$ and Modcomp

$Microprocessor_B$ will be the designation given to the processor which will solely be dedicated to data reduction. It will be able to access the memories which contain the sampled data.

C. Control Panel

The control panel will be made up of programmable switches and LED displays. The switches will be used as

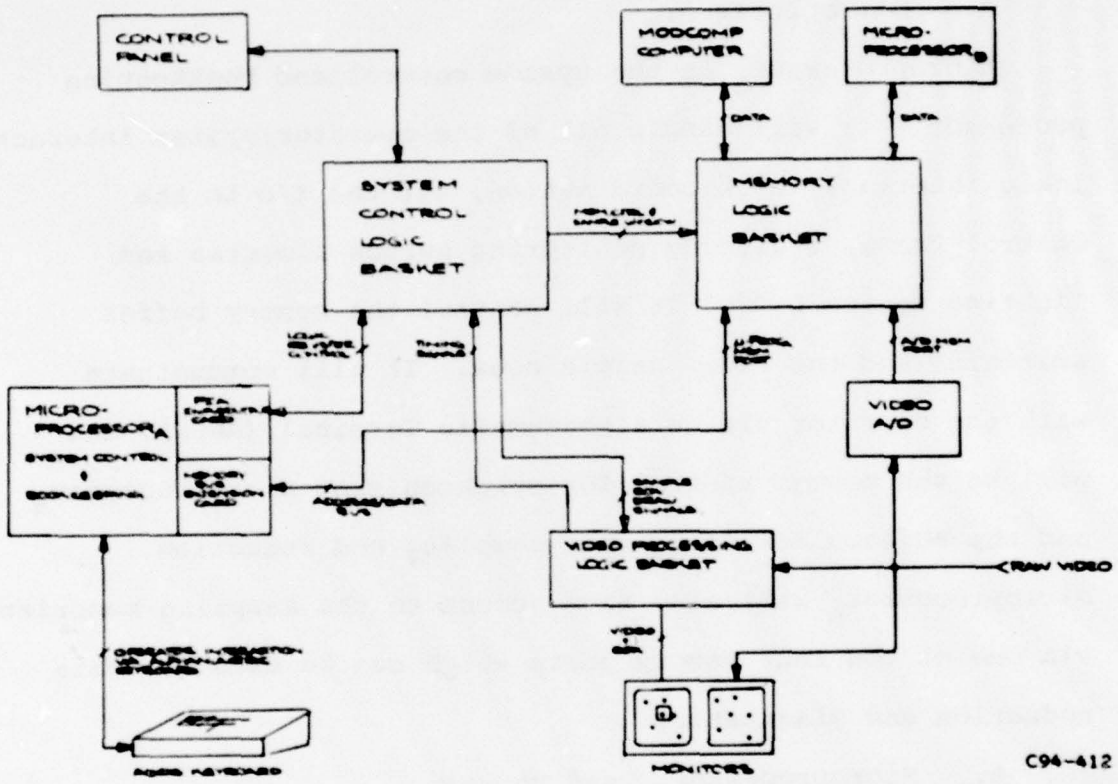


Fig.2. PVDS Hardware Block Diagram.

operator interaction control. The LED's will display the sample area size and position, along with the centroid position of an object.

D. Video Processing Logic

The video processing logic will do video conditioning, sync generation and video switching. It will also house the Phase Lock Loop (PLL) logic which generates the 10.2375 Mhz video A/D sample signal.

E. Video A/D and D/A

This unit contains the 20 Mhz Tektronix video A/D and D/A along with the associated interface control. It receives the raw video along with the sample clock and sends a digitized 8 bit word to the memory basket. The D/A within the box will be used to re-convert the digitized data to video. This can be used as a confirmation of proper system operation.

F. System Control Basket (SCB)

The System Control Basket (SCB) is a 13 card AUGAT logic basket that houses the logic used for system control. This basket interfaces directly to $Microprocessor_A$.

Each card in the SCB has access to an 8 bit tri-state bi-directional data bus along with an 8 bit address bus. The address bus defines 256, 8 bit data registers. There can be up to 16 such registers on each card. It is through this interface scheme that $Microprocessor_A$ communicates to the system.

Each card is given a mnemonic which is associated with its function along with an SCB number. The SCB number describes the slot the card is assigned in the basket, e.g., SCB-1 is System Control Basket, slot 1 of 13.

At present there are 7 cards assigned to the System Control Basket. A brief description of the function of each card is given below (refer to Figure 3).

1. Control Panel Interface Card (CPI) SCB-1

This card interfaces Microprocessor_A to the system switches, LEDs and the operator controlled joystick on the Control Panel.

2. System Timing Card (STC) SCB-3

The System Timing Card generates all of the timing signals used in the video sampling. These timing signals are generated from the video sync signals within the Video Processing Basket. This card also generates the interrupts to Microprocessor_A which facilitates the synchronizing of the video sampling and data reduction.

3. PIA Conditioning Card-X (PCCX) SCB-4

This card interfaces Microprocessor_A with the System Control Basket. It makes up the tri-state bi-directional data bus and the address bus from the microprocessor's Peripheral Interface Adapter (PIA) chips. It also contains an 8 level priority interrupt design for system interrupts.

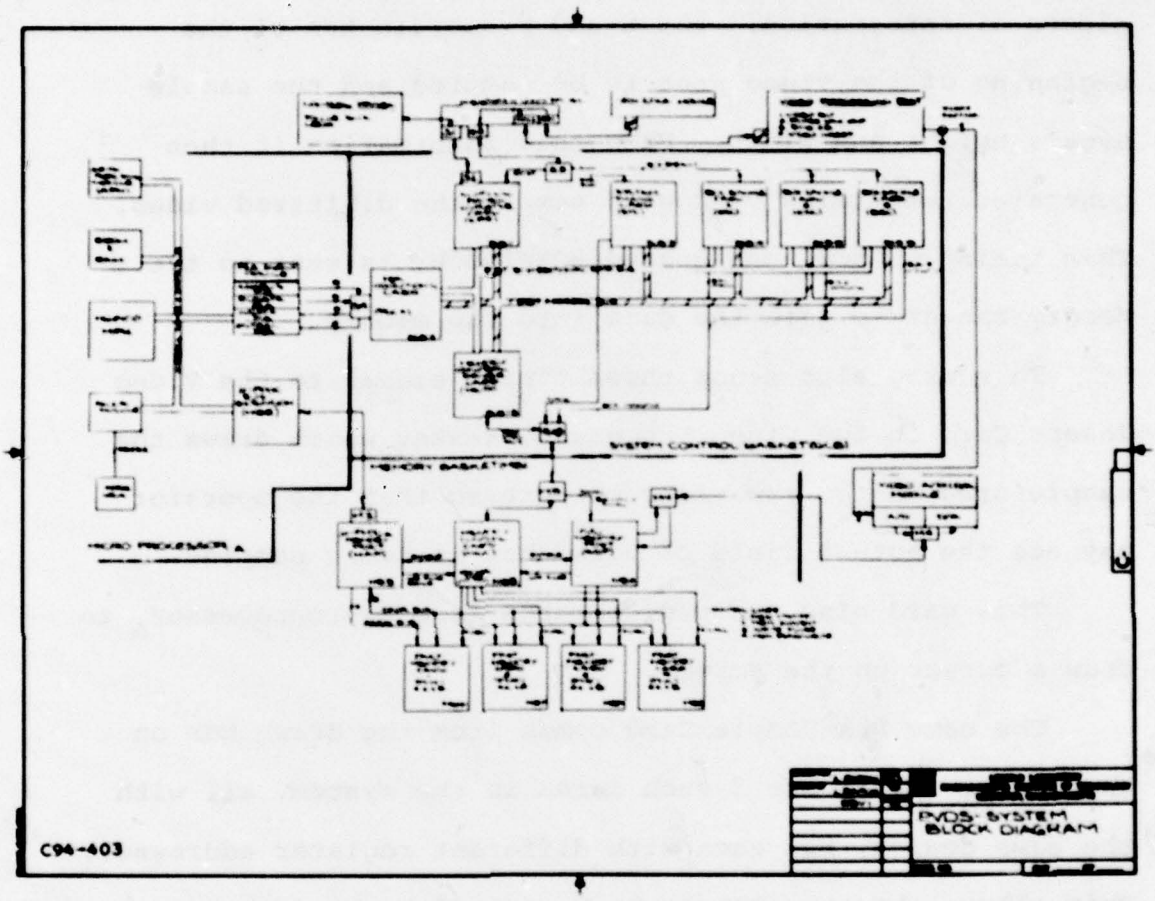


Fig.3. Functional Block Diagram Including Logic Card Layout.

4. Box Sample Card 1 thru 3 (BSC1-BSC3) SCB-5 thru SCB-7

The Box Sample Card receives from Microprocessor$_A$ four pieces of information: the x and y coordinates of the beginning of the video area to be sampled and the sample area's height and width. From this information it then generates the timing window to sample the digitized video. This timing window, designated BOXWINDOW, is sent to the Memory Basket to gate the data into the memory.

This card also sends these timing signal to the Video Insert Card in the Video Processing Basket which draws the sample area on the raw video picture so that the operator may see the actual field of view that is being sampled.

This card also accepts commands from Microprocessor$_A$ to draw a cursor on the screen.

The name Box Sample Card comes from the drawn box on the video. There are 3 such cards in the system, all with the same design, but each with different register addresses. This allows the operator to have up to 3 boxes drawn on the screen and thus 3 sample areas defined at the same time.

5. Memory Control Driver Card (MCD) SCB-13

This card is the communication interface from Microprocessor$_A$ to the Memory Basket. It sends the control signals to the 4 memory ports and memory cards. These signals control the buffer switching and data sampling.

G. Memory Basket (MB)

The Memory Basket (MB) is a 13 card AUGAT logic basket that contains the 4 port memory design and 4, 4096 X 8 bit Fast Memory Cards.

The 4 port memory design allows up to 4 sources to access any one of the 4K blocks of memory. The access is under software control via Microprocessor_A.

A brief description of the logic cards is given below. The card mnemonic along with the slot number (e.g., MB-1 thru MB-13) is also given. (Refer also to Figure 3.)

1. Microprocessor_A Memory Port (MAP) MB-3:

The Microprocessor_A Memory Port interfaces the CPU bus of the M6800 to the address/data bus common to the Fast Memory Cards. This interface is tied to Microprocessor_A through a card called the Microprocessor Bus Extension Card (MBE) housed in the microprocessor module.

When the MAP is given access to the Fast Memory Cards, then the selected card will appear as a 4K block of memory in Microprocessor_A's memory map.

2. Video A/D Memory Port (ADP) MB-4:

This card interfaces the data from the Video A/D Box to the Fast Memory Card. It generates an address for the data and receives a BOXWINDOW signal from the Box Sample Cards to enable the data into the memory. It also contains the logic

for allowing summed or unsummed fields of video to be sent to the Fast Memory Cards.

3. Memory Control Card (MCC) MB-5:

This card receives commands from $Microprocessor_A$ via the System Control Basket. It controls which memory port has access to one of the 4 Fast Memory Cards. It also controls the state of each Fast Memory Card, those being OFF, INTERNAL or EXTERNAL.

4. Fast Memory Card (FMC1-FMC4) MB-6 thru MB-9:

The Fast Memory Card is a 4096 x 8 bit memory card that has an access write or read time of 30 ns. There are 4 such cards in the Memory Basket.

A memory may be accessed through one of two routes. One is labeled an internal address/data bus and the other is called the external bus. The internal bus is a common tri-state bus, common to all the memory ports in the system. The external bus allows an external device to access the memory, independent of any other activity with the other memories. The bus switching and memory access control is handled from the Memory Control Card via the software in $Microprocessor_A$. Each of the Fast Memory Cards is the same design with separate control coming from the Memory Control Card.

III. THE INTERFACE BETWEEN THE SYSTEM CONTROL LOGIC AND MICROPROCESSOR$_A$.

The system control interface for the PVDS is based on a design called a Register Configured System Design. The system appears to the software as 256, 8 bit registers. Each register is uniquely defined as being an Output Type Register or an Input Type Register. This determination of register types is dictated by the hardware needs. A detailed list of each register's function and type is given in Section XIII.

Each register may be accessed by the software. The software does so by issuing an 8 bit address to the Register Configured System. If that selected register is an Output Type Register (microprocessor to hardware) then that particular register will be alerted to accept data from the bi-directional data bus. If the 8 bit address is an Input Type Register (hardware to microprocessor) then that register puts its data on the bi-directional data bus where it is subsequently read by the software.

A. Address BUS & Bi-Directional Data Bus Design

Figure 4 shows a block diagram of the design for the Register Configured System. The microprocessor accesses the address bus and input/output data bus via Motorola's Peripheral Interface Adapter Chips (PIA). The PIA is the chip used by Motorola to interface the CPU chip to the outside world.

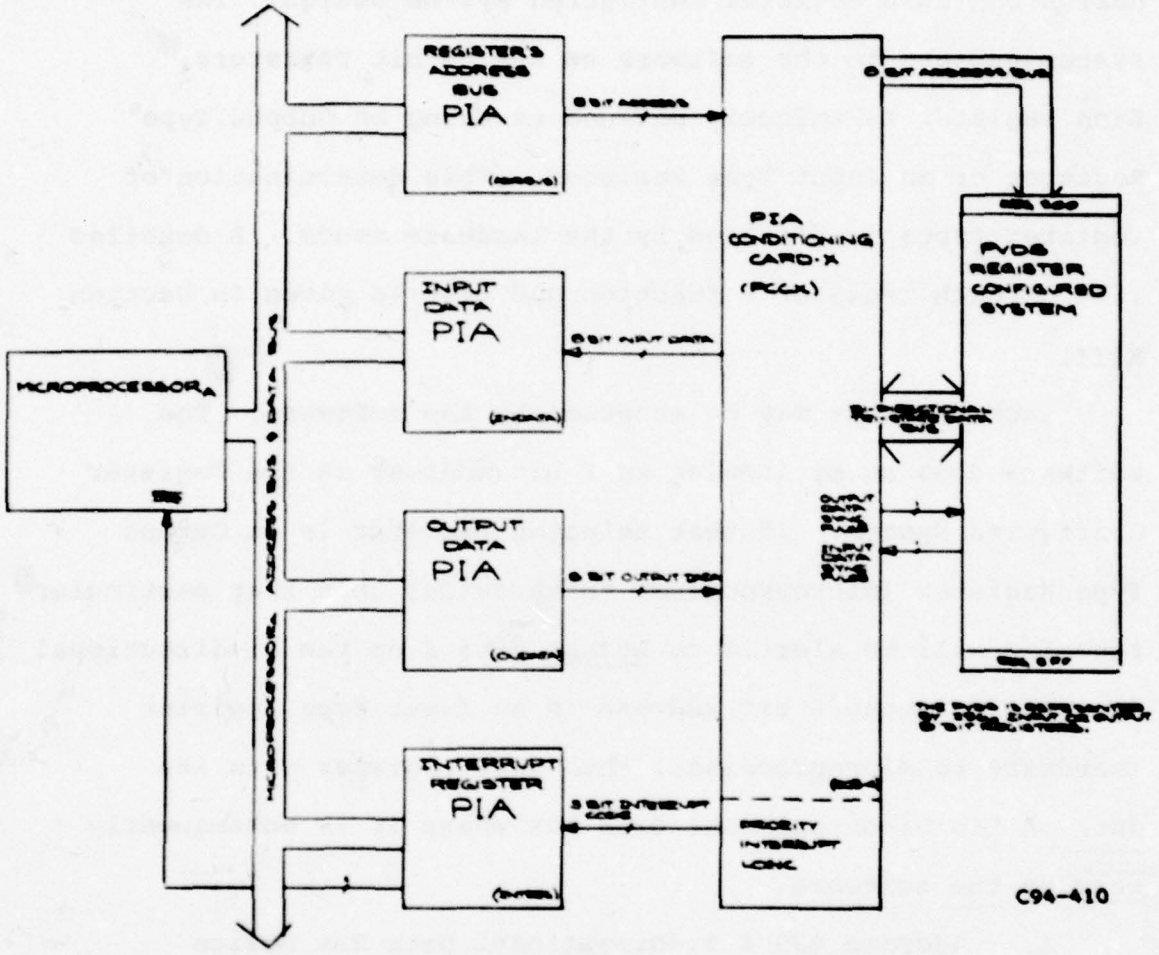


Fig.4. The Register Configured System.

Each one of the PIAs is a dedicated location in microprocessor$_A$'s memory map. Thus, data may be stored to a PIA from one of Microprocessor$_A$'s accumulators. Likewise, data may be loaded from the PIA into an accumulator.

A PIA may be configured to allow data to be output or input. There is also a control line on each PIA that is used for data handshaking. Various PIA modes can be defined by the software writing to the PIAs control register.

The PIA initialization program described in Appendix A shows how the configuration of the PIAs is determined for use with the PVDS. A more detailed description of the PIA is given in Motorola's M6800 Microprocessor Application Manual.

The Register Configured System Design uses all of the PIAs in the so called "pulse mode" of operation. In this mode, any time data are either output or input from a PIA, the Microprocessor$_A$'s phase 2 ($\phi 2$) clock accompanies the data transfer. This pulse is termed a Data Sync Pulse (DSP) to the logic and is used by the logic to synchronize the data transfers.

For the Register Configured System Design one of the PIA channels is defined as an output PIA and is used as the source of the 8 bit register address (ADRBUS PIA). Another is used as an input PIA (INDATA PIA) and data from Input

Type Registers are loaded through this PIA. The third PIA channel is configured as an output PIA (OUDATA PIA) and data are stored through this PIA to any of the Output Type Registers.

The fourth PIA channel is called the Interrupt PIA (INTREG PIA) and is described in more detail in Section IIIB below.

The data from the Address Bus PIA, Input & Output Data PIA are interfaced to the system via the PIA Conditioning Card X (PCCX). This card is located in the System Control Basket (SCB-4). This card creates an 8 bit address bus and an 8 bit bi-directional tri-state data bus. This bus is sent to every card within the System Control Basket. Each card interfaces to the bus in the same fashion. This front end interface, common to all the cards in the System Control Basket, is described in Section IIIC.

Also sent to each card is the Data Sync Pulse associated with the Output Data PIA (OUDSP) and the Input Data PIA (INDSP). The Output Data Sync Pulse is used by Output Type Registers to latch the data from the bi-directional data bus. The Input Data Sync Pulse is used only as a timing reference during hardware check out.

The PCCX card steers data from Input Type Registers, off the bi-directional data bus and to the INDATA PIA. The card also steers the 8 bits of data from the OUDATA PIA onto

the bi-directional data bus and passes along the Output Data Sync Pulse. The Output Data Sync Pulse is regenerated such that the data are solid on either edge of the pulse.

B. Priority Interrupt Design

Because the microprocessor only has one level of interrupt to the CPU, an 8 level priority interrupt scheme was designed. This design allows up to 8 external interrupts to asynchronously interrupt the CPU. Figure 5 shows a block diagram of the design.

An interrupt occurs on the high to low going edge of an interrupt level from the PVDS logic. This will set an associated interrupt flip/flop, if that level is enabled. This interrupt will be passed on to a Priority Encoder. The Priority Encoder makes the highest priority interrupt level a 3 bit code and passes this number to the Interrupt PIA. It also pulses this PIA which in turn interrupts the CPU. The software then inputs the 3 bit code that caused the interrupt. This 3 bit code will indicate one of eight interrupt handlers the software may execute.

Bit 7 of the Interrupt PIAs control register will be set when any interrupt occurs. If there are other peripheral chips within the system that may also cause an interrupt on the CPU's $\overline{IRQ}$ line, then this bit may be tested to determine if the interrupt came from the Interrupt PIA. This bit will be automatically cleared when the Interrupt PIA is read.

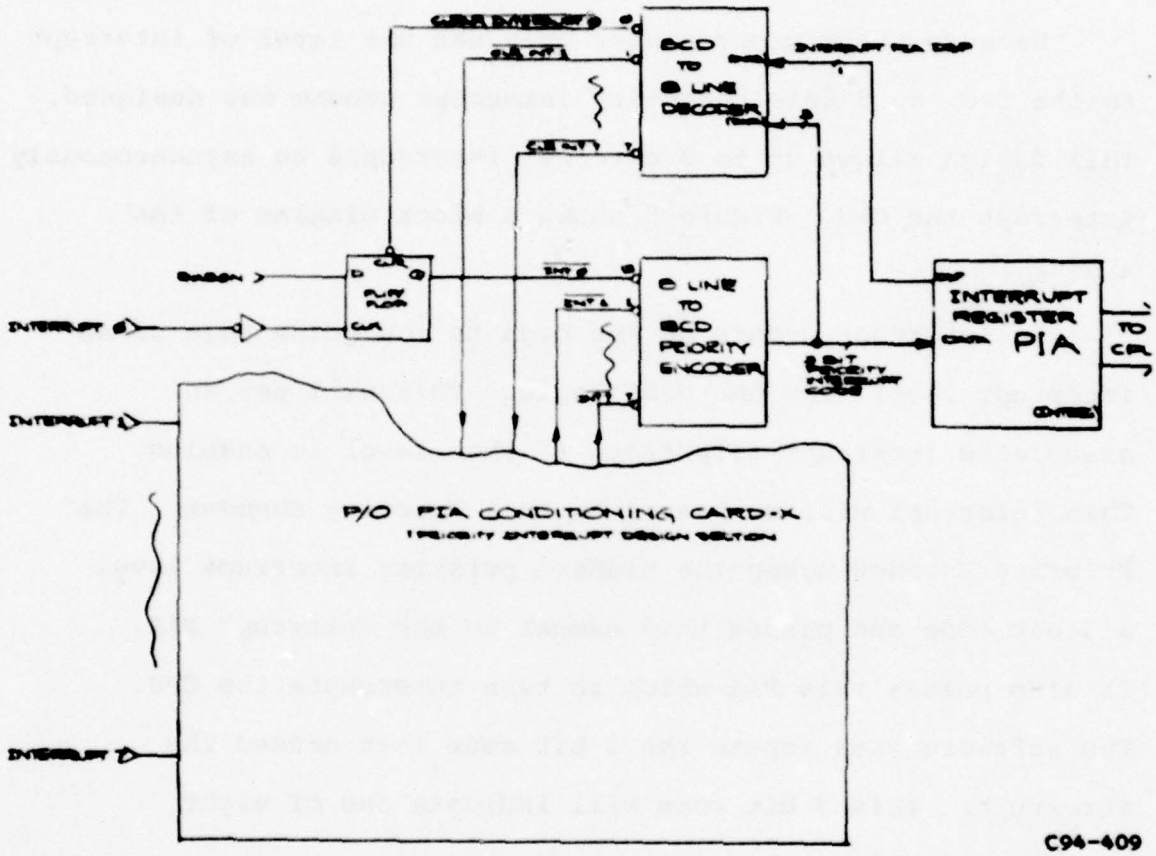


Fig.5. Priority Interrupt Design.

Since the Interrupt PIA is configured in the so called pulse mode and as an input type PIA, there is a sync pulse associated with the Interrupt number read. This sync pulse initiates the sequence to clear the interrupt flip/flop that caused the interrupt and allow another interrupt to occur.

The interrupts that will be used in the PVDS are described in detail in Section IIID2. The software considerations for these particular interrupts are also given.

C. Common Register Address Decode

Figure 6 shows a front view of the System Control Basket. Shown also are the card slots presently used, their function and the registers that are accessible on each card.

Each card slot (whether presently used or not) has the register address bus, tri-state bi-directional data bus, and output data sync pulse. Each card also has the same front end logic design to decode the address bus with the normal register configuration allowing each card to have up to 16, 8 bit registers.

Figure 6 shows the block diagram of the register address decode logic that is common to each card in the System Control Basket. From the 8 bit address bus the 4 MSBs are decoded to select one of 16 cards that is being accessed, the 4 MSBs and the four LSBs will select 1 of 16 registers

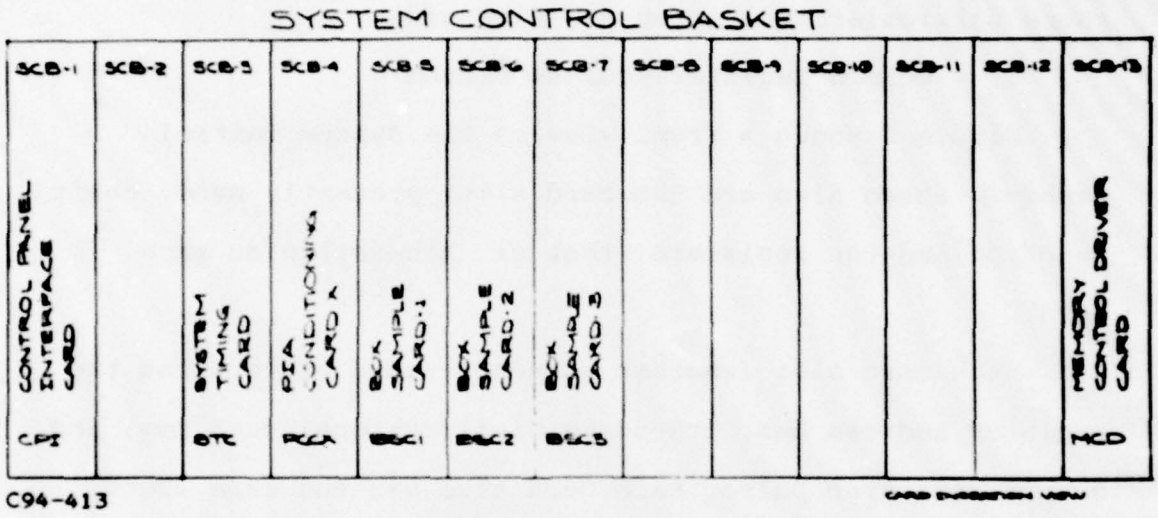


Fig.6. System Control Basket Layout.

on that card. Each register will be defined as an Input or Output Type Register.

An example of an Input Type Register and an Output Type Register is shown. For the Input Type Register (Register $Ø in Figure 7) the software has put an 8 bit code on the ADRBUS PIA which has filter through to this card giving an enable signal for register $Ø. This enable signal turns on the 8 bit tri-state drivers associated with register $Ø and puts the data on the bus. These data are transferred back to the INDATA PIA through the PCCX card. They are then read by the software through the INDATA PIA.

For the Output Type Register example in Figure 7, an 8 bit address is put on the address bus, the first 4 bits being a $F. The "enable register $F" line goes high. 8 bits of data are then sent to the bi-directional data bus through the OUDATA PIA and the PCCX card. The data are thus present at the Output Type Register along with the Output Data Sync Pulse. Since the load input to the register has been enabled, the Output Data Sync Pulse loads the register.

The Output Data Sync Pulse goes to all Output Type Registers but only the particular register enabled will pass the pulse through to the load input.

D. Software Considerations for the Interface

1. Accessing the Registers.

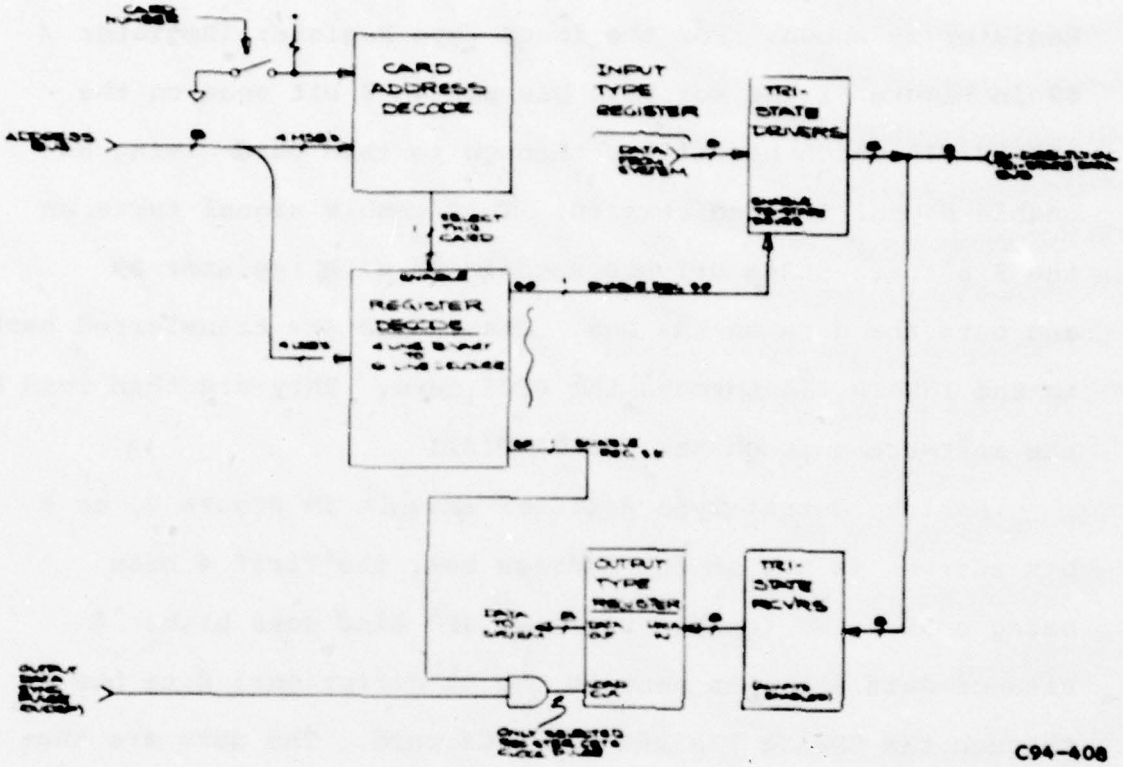


Fig.7. Common Logic Design for the Register Address Decoding in the System Control Basket.

The software may access any 8 bit register in the system by following the flow chart in Figure 8. Notice that the interfacing PIAs have been assigned labels with these labels being the actual core location of the PIA.

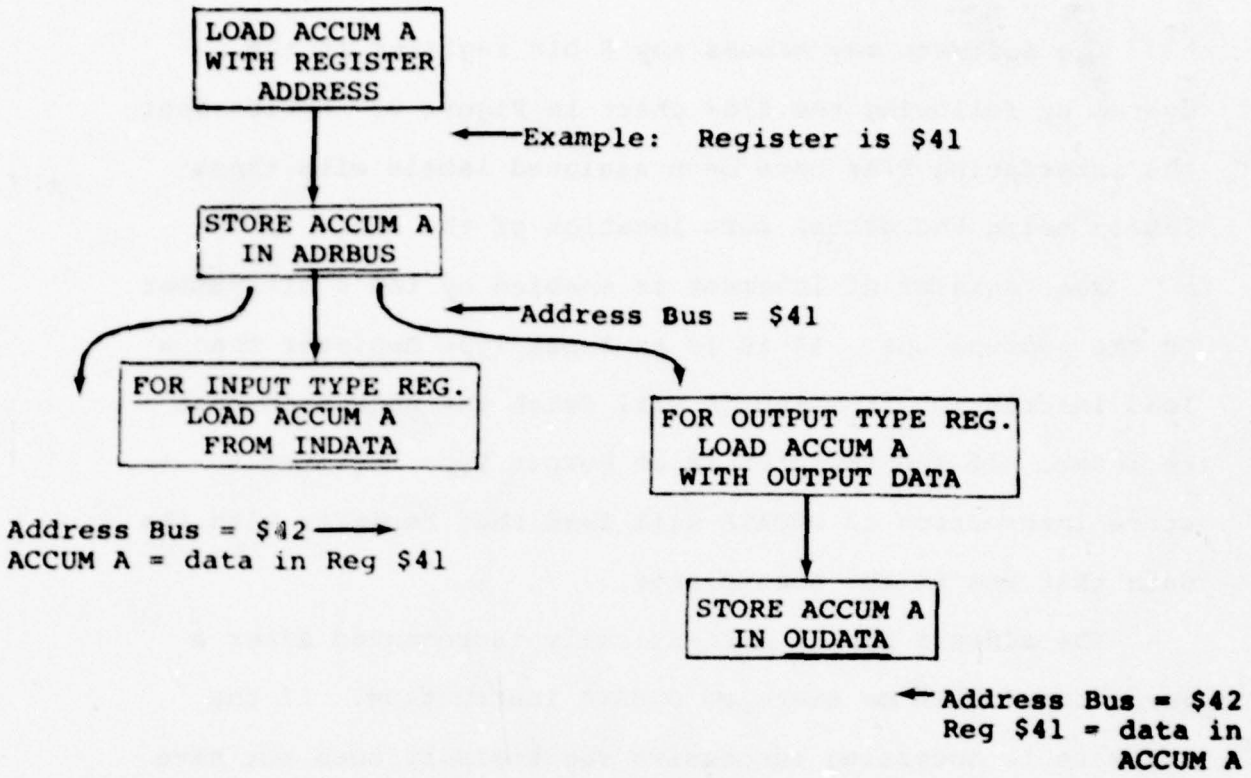
The register of interest is enabled by the 8 bit number on the address bus. If it is an Input Type Register then a load instruction from INDATA will fetch the data from that register. If the register is an Output Type Register a store instruction to OUDATA will load that register with the data that was in the accumulator.

The address bus is automatically incremented after a load from INDATA or store to OUDATA instruction. If the software is accessing successive registers it does not have to re-define the address bus.

2. *The Interrupt Handler*

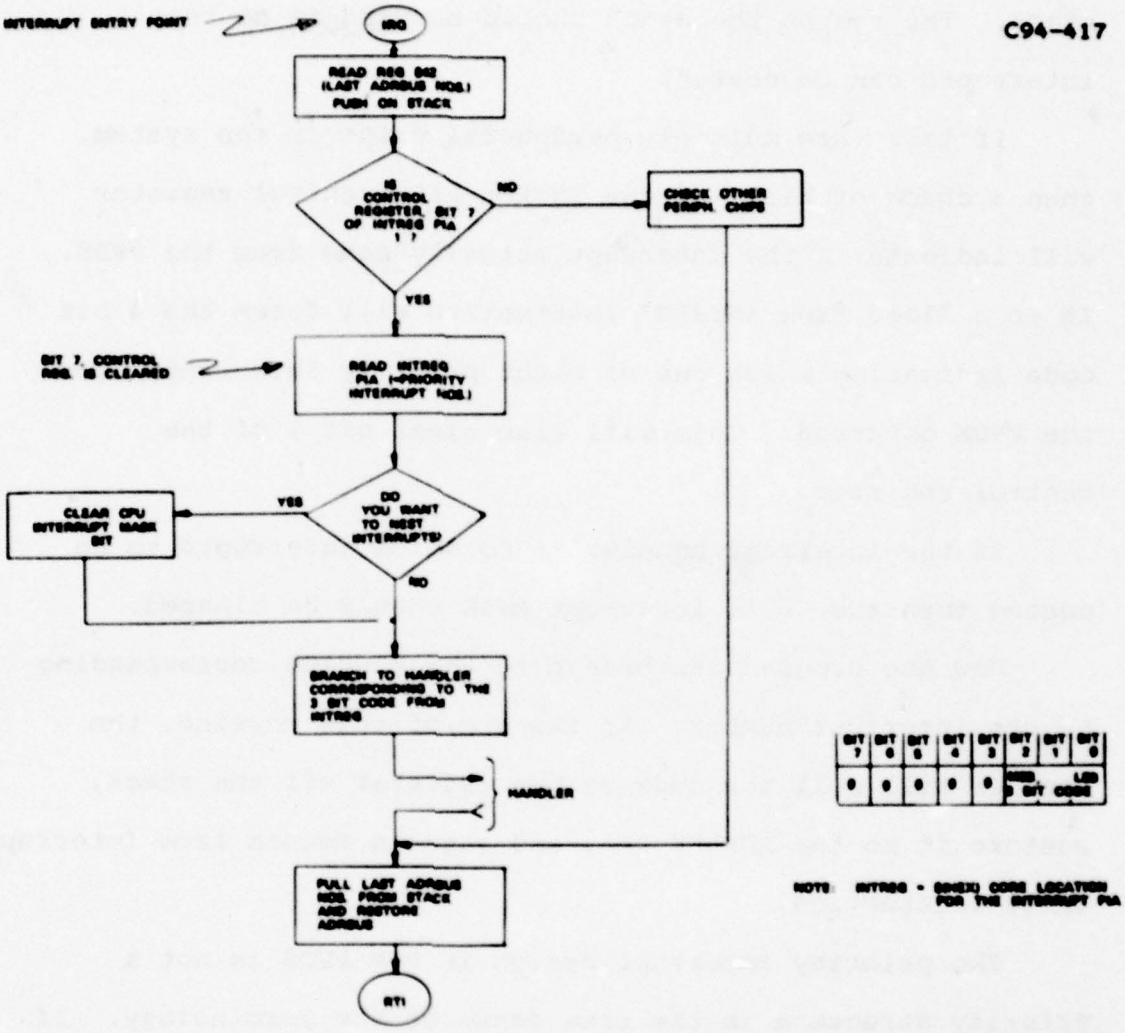
Figure 9 shows the Flow Chart for the Interrupt handler.

Upon receiving an interrupt via the CPU $\overline{IRQ}$ line the microprocessor automatically stacks its own registers and vectors the program to the starting location of the handler. The first thing the handler should do is save the data held in the address bus. This number will be restored to the address bus just before the Return from Interrupt (RTI) instruction is executed. Saving and restoring the address bus allows the interrupt handler to use the PVDS's I/O.



NOTES: $ = HEX symbol
ADRBUS = $ CORE location of PIA that accesses the Register's Address Bus.
INDATA = $ CORE location of Input Data PIA
OUDATA = $ CORE location of Output Data PIA

Fig. 8. Flow Chart to Access PVDS Registers.



C94-417

Fig.9. Flow Chart for PVDS Interrupt Handler.

PVDS register $42 ($=HEX symbol) is an Input Type Register and will contain the last number that is present on the address bus. This number should be pushed onto the stack. The reason the stack should be used is so that interrupts can be nested.

If there are multiple peripheral chips in the system, then a check of bit 7 of the INTREG PIAs control register will indicate if the interrupt actually came from the PVDS. If so a "load from INTREG" instruction will fetch the 3 bit code indicating which one of eight priority interrupts from the PVDS occurred. This will also clear bit 7 of the control register.

If the interrupt handler is to allow interrupts to be nested then the CPU's interrupt mask should be cleared.

Now the program can branch to the routine corresponding to the interrupt number. At the end of this routine, the handler will pull the address bus register off the stack, restore it to the ADRBUS PIA, and issue a Return from Interrupt (RTI) instruction.

The priority interrupt design in the PVDS is not a Priority Structure in the true sense of the terminology. If two or more interrupts occur at the same time (if the level is enabled) the highest priority interrupt will reach the CPU. However, if the CPU's interrupt mask is cleared in the handler (e.g., if interrupt nesting is desired) then any

other interrupt, higher or lower priority, will interrupt the CPU. If the CPU interrupt mask bit is not cleared until the Return from Interrupt instruction (RTI) is executed then no other interrupt will interrupt the CPU. Pending interrupts are stacked, however, with the highest priority interrupts getting to the CPU as soon as the interrupt mask bit is cleared.

The above point is made because some systems allow a given interrupt level handler to be interrupted by higher priority interrupts. If needed, this can be implemented with some hardware and software changes.

Each interrupt level can be enabled and disabled. The enable register is $4Ø, an Output Type Register and is discussed in detail in Section XIV.

IV. FOUR PORT MEMORY DESIGN

The Four Port Memory Interface and Fast Memory Cards are housed in one logic basket, namely the Memory Basket (MB). A hardware layout is shown in Figure 10. The block diagram of the Four Port Memory design is shown in Figure 11.

The design is based upon a digital bus switch which is controlled by $Microprocessor_A$. This design allows the controlling software in $Microprocessor_A$ to switch a 12 bit address bus and an 8 bit bi-directional data bus from any one of four memory ports to any (or all) of four memory blocks. The software may also switch the address and data bus of the memory blocks to an external processing unit. Processing can go on without disturbing any activity on the common four port bus. The memory blocks are 4096 (4K binary) X 8 bits and are termed Fast Memory Cards (FMC). The Fast Memory Card connotation reflects the 30 ns speed with which the memories may be operated. This section of the PVDS may be better described as a programmable buffer switching design. $Microprocessor_A$ has complete control of the data flow and data access. The expansion of the system can accommodate any sensor that will present digitized data, with a 12 bit address to a memory port. $Microprocessor_A$ will handle the data flow and buffer switching. In the case of the CCD, where the data may be 16 bits long per sample,

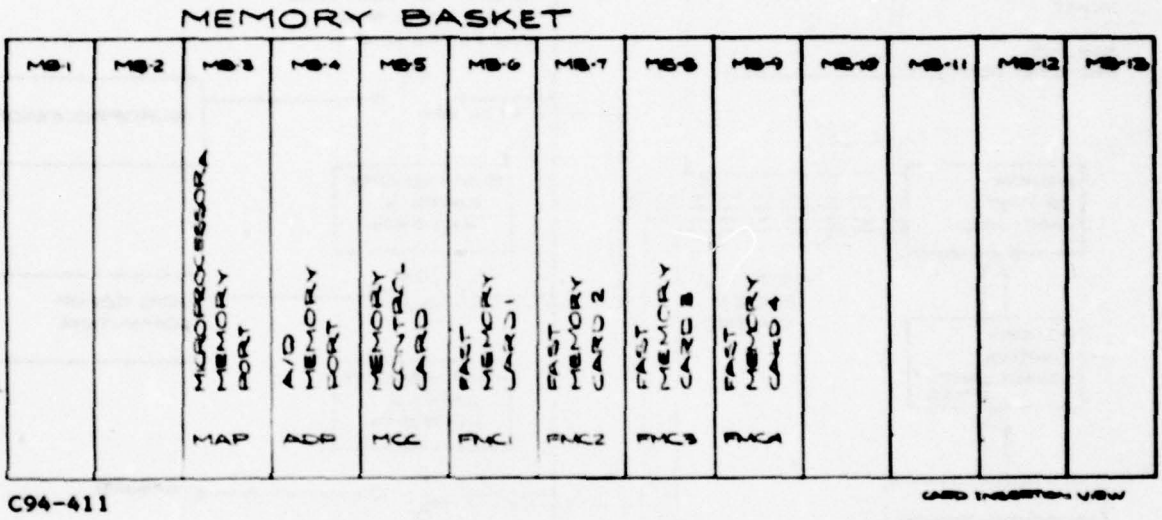


Fig.10. Hardware Layout of Memory Basket.

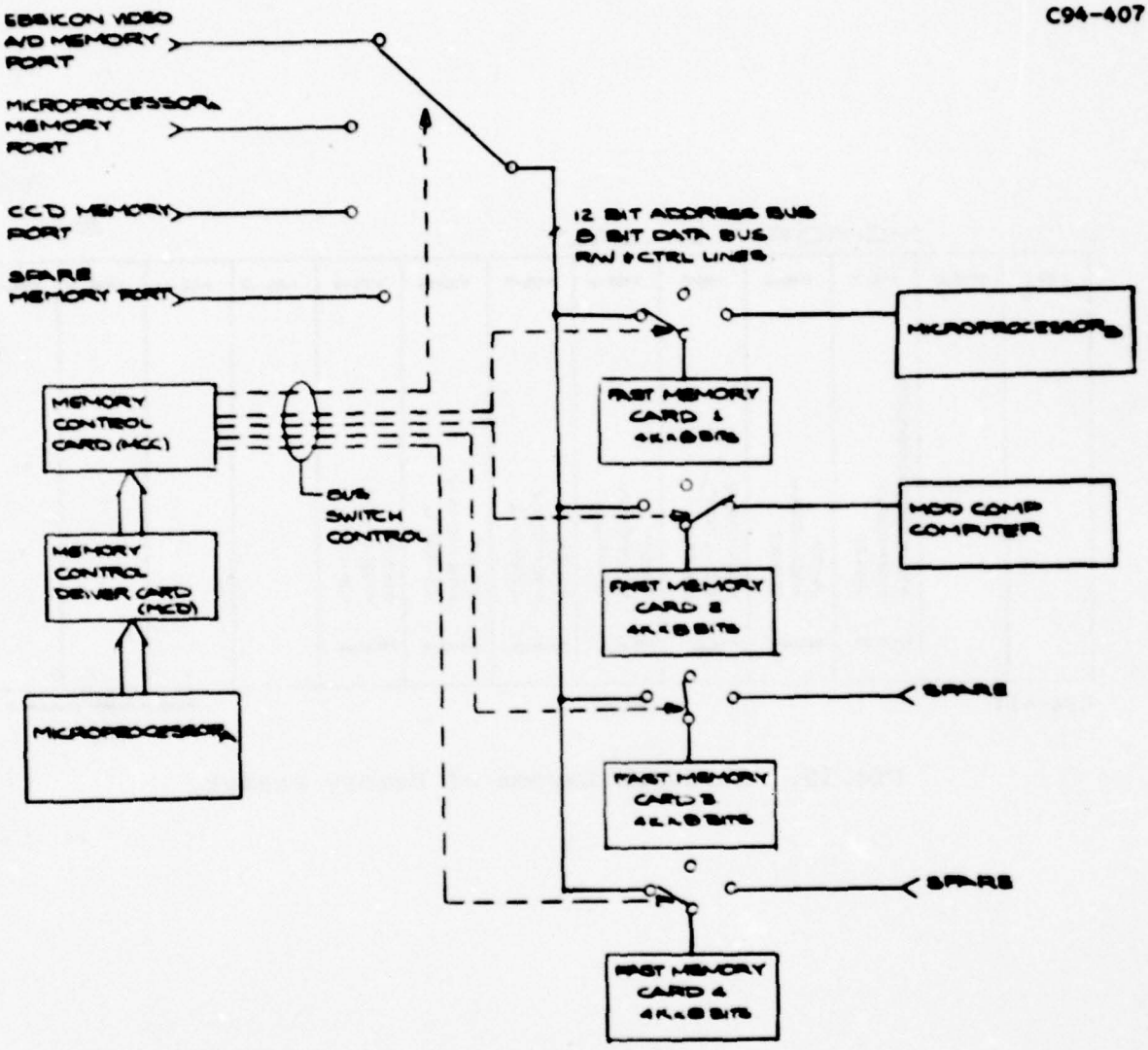


Fig.11. Four Port Memory Design.

the CCD Memory Port interface would load 2, 8 bit words into the Fast Memory Cards for each sample.

Microprocessor$_A$ has control over enabling each memory port. Each memory port sends its read/write line to the Memory Control Card which is sent to the appropriate Fast Memory Cards. To illustrate the flexibility of the design a couple of scenarios are given below.

Microprocessor$_A$ can connect the A/D Memory Port to two Fast Memory Cards. When the video's sample window area is enabled, digitized video will be written into both memories. The end of this sample time will interrupt Microprocessor$_A$ which will then switch one Fast Memory to the Modcomp and the other to Microprocessor$_B$. They can both do data reduction on the same data.

Microprocessor$_A$ can connect its own memory bus to any one of the 4K blocks of memory via the Microprocessor$_A$ Memory Port Card. Any one of the 4K blocks will appear as a 4K section of Microprocessor$_A$'s memory map. Microprocessor$_A$ can perform memory diagnostics or data reduction.

The four port control and Fast Memory Card control is interfaced to Microprocessor$_A$ as a group of registers within the Register Configured System Design. The details of these registers are found in Section XIV.

V. VIDEO FORMAT AND VIDEO SAMPLING

Since the initial objective of the PVDS involves data reduction on digitized video from the Ebsicon cameras it is important to define the video format and the sampling procedures.

The video is standard TV video, 30 Hz refresh rate with interlacing lines. For PVDS purposes the video sampling area is given some specific definitions. These definitions are discussed in this section of the report along with the software considerations for sampling the video.

A. PVDS Definition of the Video

Figure 12 shows a timing diagram of the PVDS defined video area. There are 525 lines. Line 1 is the first line of retrace for the odd field. This line is also designated as RO1 (Retrace Odd Field 1). For PVDS purposes the sample area consists of 241 displayed lines during the odd field (DO1 - DO241) and 241 displayed lines during the even field (DE1 - DE241). There are 21 retrace lines for the odd field (RO1 - RO21) and 22 retrace lines for the even field (RE1 - RE21).

The displayed area of one line is divided into sample areas of 97.68 ns each. There are 512 of these elements across one line. Each element is digitized into an 8 bit word.

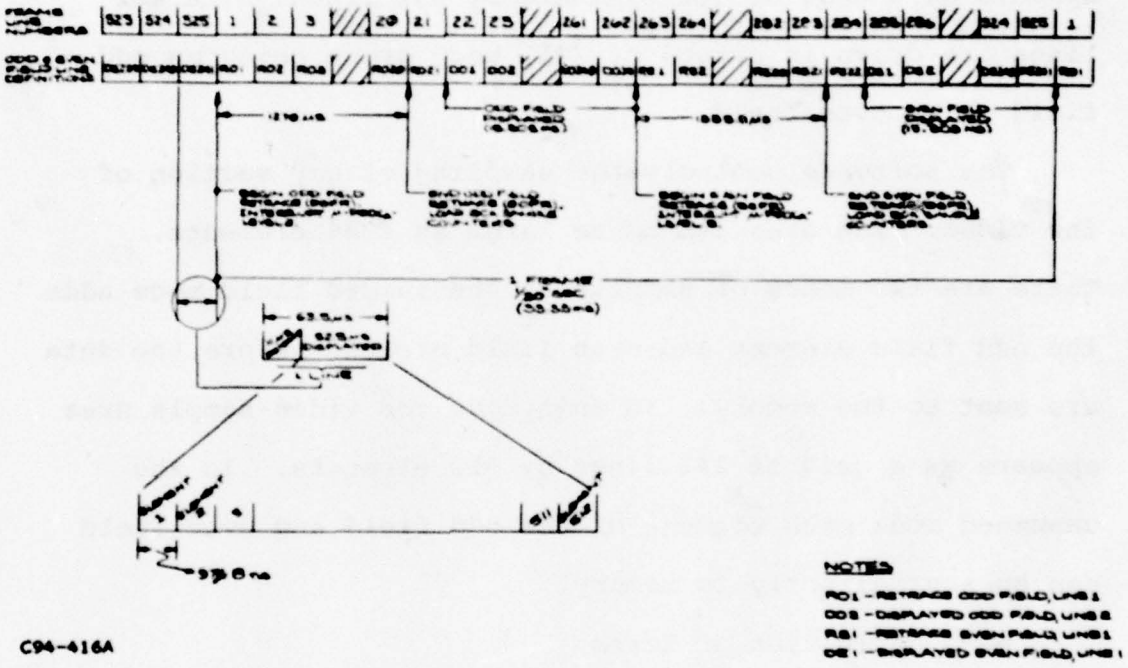


Fig.12. Timing Diagram of One Video Frame.

Figure 13 shows the displayed area of the video with the PVDS defined format. Notice that the circles represent the odd field displayed area and the squares represent the even field. The two fields are interlaced.

Thus, the total sample area available to the software appears as a grid of 512 elements by 482 lines. The 482 lines are drawn in groups of 241, each group being an odd field or an even field.

The software controls the sampling of any section of the video. The area can be as large as 4096 elements. There are two modes of sampling. The summed field mode adds the odd field element and even field element before the data are sent to the memory. In this mode the video sample area appears as a grid of 241 lines by 512 elements. In the unsummed mode each element of the odd field and even field can be sent directly to memory.

B. Definition of Terms

The software, with its interaction with the PVDS, has complete control of the video data that are sampled. To discuss what this control will be requires some definition of terms used in the PVDS. Figure 14 shows a sample area that is drawn on the video along with a cursor. Refer to this figure for the following definitions.

1. BOX - The sample area which will be drawn on the video and will indicate to the

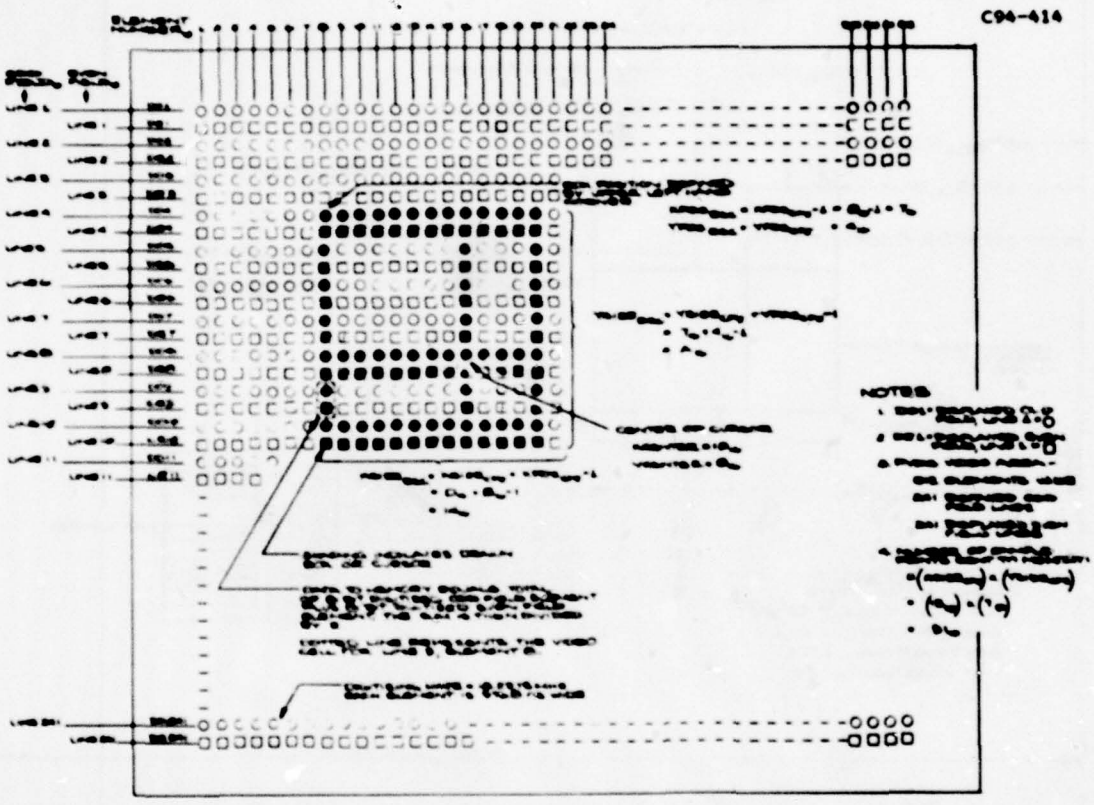


Fig. 13. PVDS Displayed Area of Video.

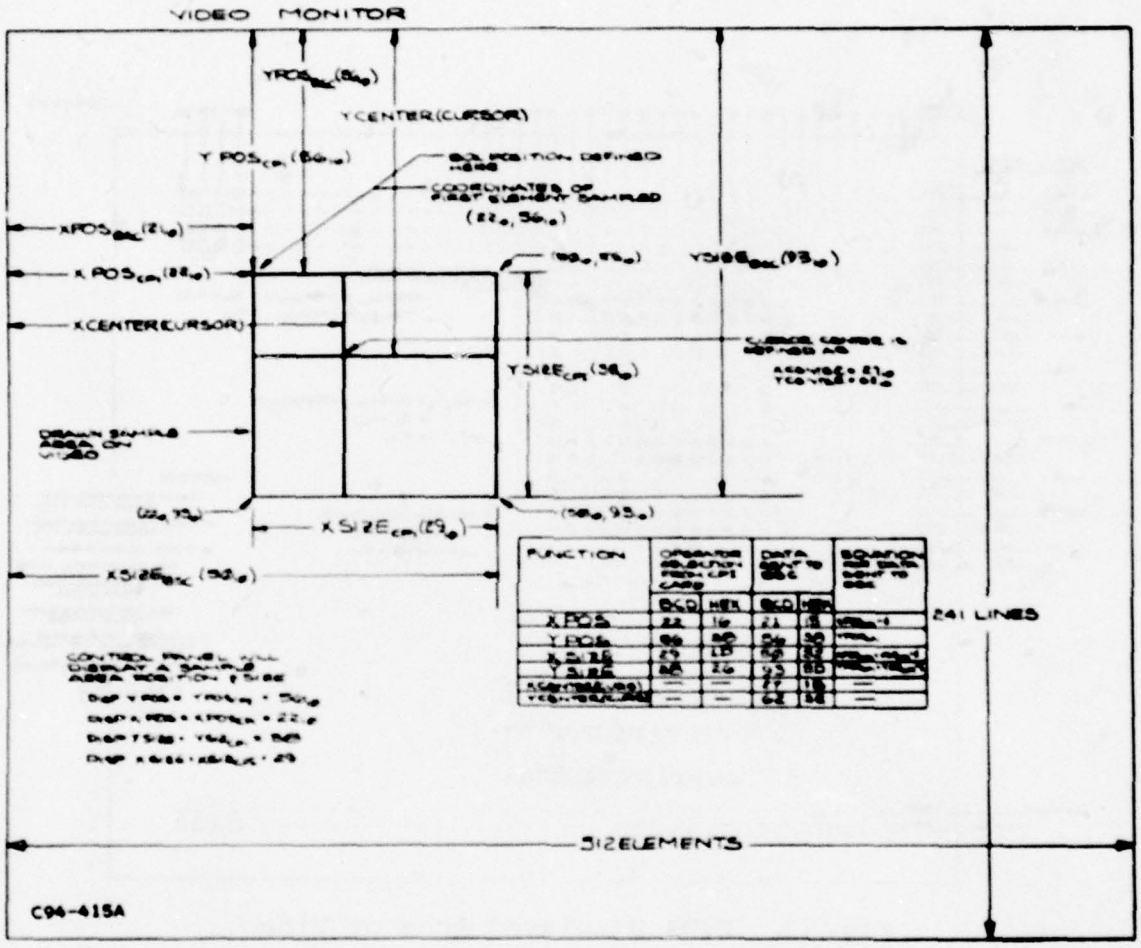


Fig.14. Drawn Sample Area plus Cursor with Term Definitions.

operator what area is being sampled. The software may enable or disable the drawing of the BOX.

2. CURSOR - A cursor may also be drawn on the video inside the box. The cursor may be blanked or unblanked as is the case with the box. The x, y center of the cursor is controlled by the software.

3. BSC - Mnemonics for the Box Sample Card. This card contains the logic for drawing both the box and the cursor, the control of blanking or unblanking and for generating the BOXWINDOW signal. There are three of these cards (BSC1, BSC2, BSC3) which allows up to 3 sample areas to be independently defined. A subscript on any of the box parameters (e.g., $XPOS_{BSC}$) indicates these are the data going to the Box Sample Card.

4. CPI - The mnemonic for the Control Panel Interface Card. This card contains the operator controlled data for defining the parameters of the sample area. A subscript on any of the box parameters (e.g., $XPOS_{CPI}$) would indicate that these are data from the CPI card.

5. BOXWINDOW - This is the mnemonic for the timing signal, generated by the BSC, that enables the digitized video to the memory. This signal is gated onto the Fast Memory Cards if so enabled by the software. There are three such signals (BOXWINDOW1 - 3), one for each BSC. They are all independently controlled.

6. $XPOS_{CPI}$,$YPOS_{CPI}$ - This is the x and y position of the box that is generated by the CPI card. These 9 bit numbers define the upper left hand corner of the sample area. The numbers can be changed by the operator via the joystick. These same numbers will be converted to BCD and displayed on the control panel for the operator.

7. $XSIZE_{CPI}$,$YSIZE_{CPI}$ - These two 9 bit numbers come from the CPI card, are controlled by the operator and will be displayed on the control panel in BCD. They define the sample area size.

8. $XPOS_{BSC}$,$YPOS_{BSC}$,$XSIZE_{BSC}$,$YSIZE_{BSC}$ - These are the actual 9 bit numbers sent to the BSC to allow the logic to draw the box and generate

the BOXWINDOW signal. The relationship between the data received from the CPI card and that sent to the BSC is given in the following equations.

$$XPOS_{BSC} = XPOS_{CPI} - 1$$

$$YPOS_{BSC} = YPOS_{CPI}$$

$$XSIZE_{BSC} = XPOS_{CPI} + XSIZE_{CPI} - 1$$

$$YSIZE_{BSC} = YPOS_{CPI} + YSIZE_{CPI} - 1$$

9. XCENTER, YCENTER = This is the definition of the x,y center of the cursor. Each is a 9 bit number sent to the BSC. They are not generated on the CPI card.
10. Begin Odd Field Retrace (BOFR) - This is the timing signal that is generated by the logic to indicate the beginning of the retrace for the odd field of the video. This happens at the beginning of line 1. It is one of the interrupts to $Microprocessor_A$. The software should send the box sampling information to the BSCs at this time.
11. End Odd Field Retrace (EOFR) - This signal is

generated at the beginning of line 21 and is used to load the sample area information, sent by the software, into the BSC logic. This signal will happen approximately 1270 μs after BOFR. This means the software has this much time to react to the BOFR interrupt.

12. Begin Even Field Retrace (BEFR) - This signal happens at line 263 and is one of the interrupts to $Microprocessor_A$. It indicates the beginning of the retrace for the even field.
13. End Even Field Retrace (EEFR) - This signal is generated at the beginning of line 284 approximately 1333 μs after BEFR. It is used to load the sampling data sent from the software into the BSC logic.
14. End of Box (EOB) - This signal will be an interrupt to $Microprocessor_A$ and is generated when the last sample element has been sent to the memory.

C. Defining the Sample Area

There are two sets of information to define the sample area. One set of BOX coordinates is the information that is displayed on the Control Panel for the operator. This will be the BCD equivalent of the binary information received from the CPI card.

The second set of sample area coordinates is that needed by the BSC to generate the proper BOXWINDOW timing signal and to actually draw the BOX on the video. Refer to Figure 14 for a discussion of these definitions.

1. Coordinates from the CPI Card

The 4 pieces of information from the CPI card, needed to define the sample area are:

$XPOS_{CPI}$: Starting x coordinate of the sample area. This will be the first element per line to be sent to memory.

$YPOS_{CPI}$: Starting y coordinate of the sample area. This will be the first line of data that are sent to memory.

$XSIZE_{CPI}$: Number of elements per line that will be sent to memory. This is the size of the box in the x direction, i.e., width.

$YSIZE_{CPI}$: Number of lines to be sent to memory. This is the height of the sample box.

These binary 9 bit numbers are contained in the registers on the CPI card. The bit pattern and register numbers are found in Section XIV of this report (System Control Logic, Detailed Register Description). These 4 binary numbers should be converted to BCD and displayed in the Control Panel LEDs.

2. Data to Box Sample Card (BSC)

The information needed by the BSC to generate the proper timing signals for the memory (BOXWINDOW) and for the BOX display has to be reformatted from that received from the CPI card.

The relationship to the data from the CPI card is given in the following equations and is also shown in Figure 13.

$$XPOS_{BSC} = XPOS_{CPI} - 1$$
$$YPOS_{BSC} = YPOS_{CPI}$$
$$XSIZE_{BSC} = XPOS_{CPI} + XSIZE_{CPI} - 1$$
$$YSIZE_{BSC} = YPOS_{CPI} + YSIZE_{CPI} - 1$$

The reason for the reformatting is described in the BSC logic description found in this report, Section X. From these 4 pieces of information the BSC will draw the sample area on the video. The BSC also contains the logic to draw a cursor on the video. The size of the cursor will be determined by the size of the sample box. The center of the cursor is determined by two more 9 bit words sent to the BSC. These are XCENTER and YCENTER and define the center of the cursor.

The bit pattern and register information for communicating to the BSC can be found in Section X of this report.

D. Controlling the Sample Area

The best way to describe how the sample area is controlled would be to give a scenario from the time the operator selects the sample area to the time the data are actually being loaded into the Fast Memory Cards. Refer to Figure 15.

The operator selects the position and size of the sample area via a joystick and push buttons. These controls increment or decrement four counters on the CPI card. These counters are the 9 bit registers which describe the X and Y position and the X and Y size of the sample area.

These data are read by the $Microprocessor_A$ software which converts to BCD and writes to the control panel LED displays. The data are also reformatted per the equations given in *Section VIIC above*. This reformatted position and size information is sent to the 4, 9 bit registers on the BSC.

The BSC will then generate a BOXWINDOW signal which does two things. It is sent to the video insertion logic which draws a box on the video. It also enables the Fast Memory Card to accept the digitized video at the correct sample time.

The BOX that is drawn on the video represents the sample area. To prevent flicker the BOX is drawn on both the odd field and the even field. The actual data that are

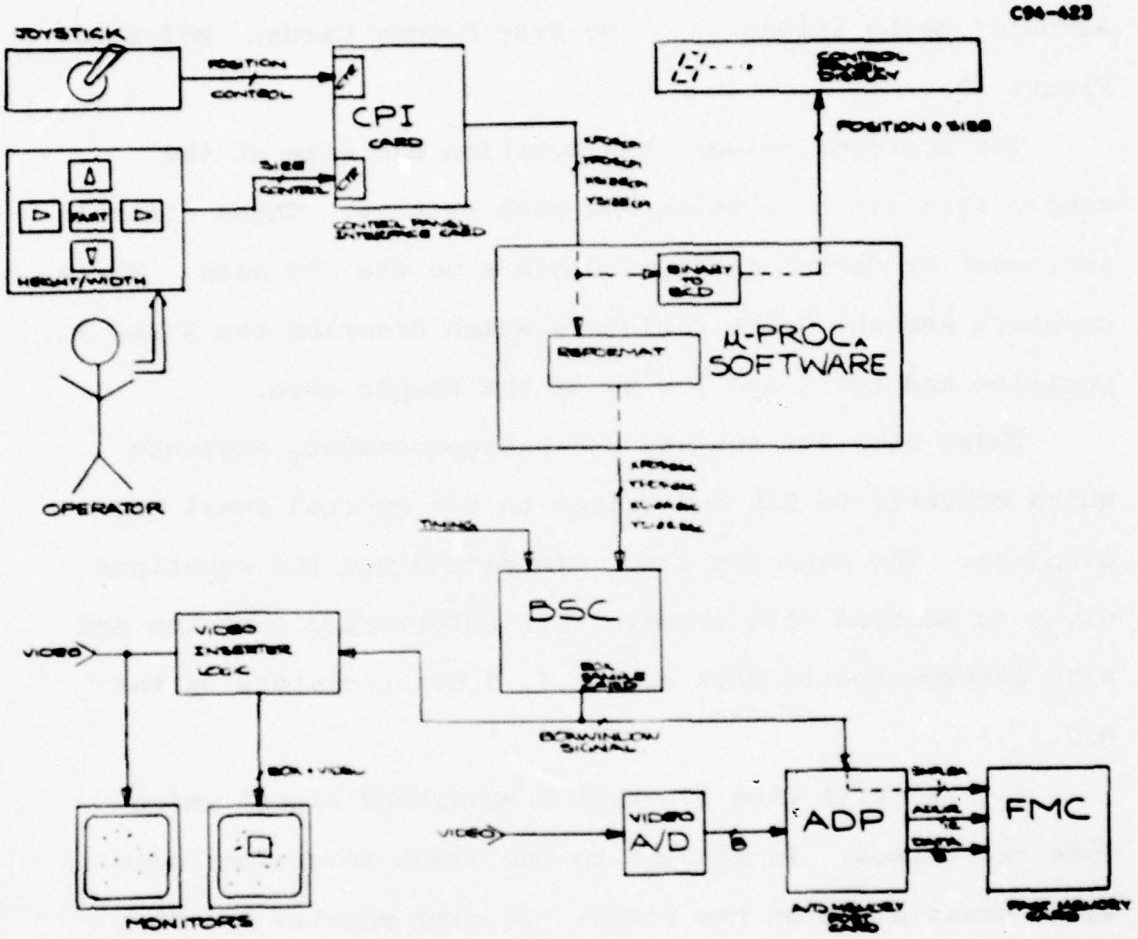


Fig.15. Sample Area Control.

sampled and sent to the Fast Memory Cards are determined by one of two modes selected by the software. These being either summed or unsummed field modes and these are described in the next section.

To draw the sample area on the video, the video appears as a grid of 241 lines by 512 elements. Referring to Figure 13 it can be seen how the drawing of the sample area actually takes place. The operator selected Y position ($YPOS_{CPI}$) is 4. The hardware automatically draws the sample area on line 4 of the odd field and line 4 of the even field. The starting X position ($XPOS_{CPI}$) is element 9. This element is illuminated for both the odd and even field.

The selection of the sample area is actually hardware independent of the data coming from the CPI card. One of the modes of operation will be to have the above scenario where the operator controls the BOX via the joystick. However, the software itself can automatically control the sample area size and position. This automatic control may be used in scanning the field or narrowing down on a particular object if the initial sample area is bigger than needed.

E. Sending Data to the Fast Memory Cards

The sampled digitized data that are sent to the Fast Memory Cards (FMC) can be in two formats, the summed field and unsummed field, and are described below.

1. Summed Field Mode

The summed field mode of formatting the digitized data means the following. The same element, on a given line, for the odd field and even field is summed and divided by two before being sent to the FMC.

This mode makes the video area appear as a grid of 241 lines by 512 elements. The smallest Y size will be 1 line. This mode will be used when the data reduction is concerned with signature analysis and possibly centroiding.

Figure 16 shows a sample area with the actual memory locations in the Fast Memory Card where the given element would be stored. The elements are sequentially sent to memory locations.

Notice that the operator displayed position of the box is the same as the positional data from the CPI card. The register that controls this mode and the bit pattern is described in A/D Memory Port Section VIII.

2. Un-Summed Field Mode

The un-summed mode allows the data for the odd field and even field to be independently sent to the FMC. This format is shown in Figure 17. This makes the video appear as a grid of 482 lines by 512 elements. The smallest Y size sample area will be 2 lines.

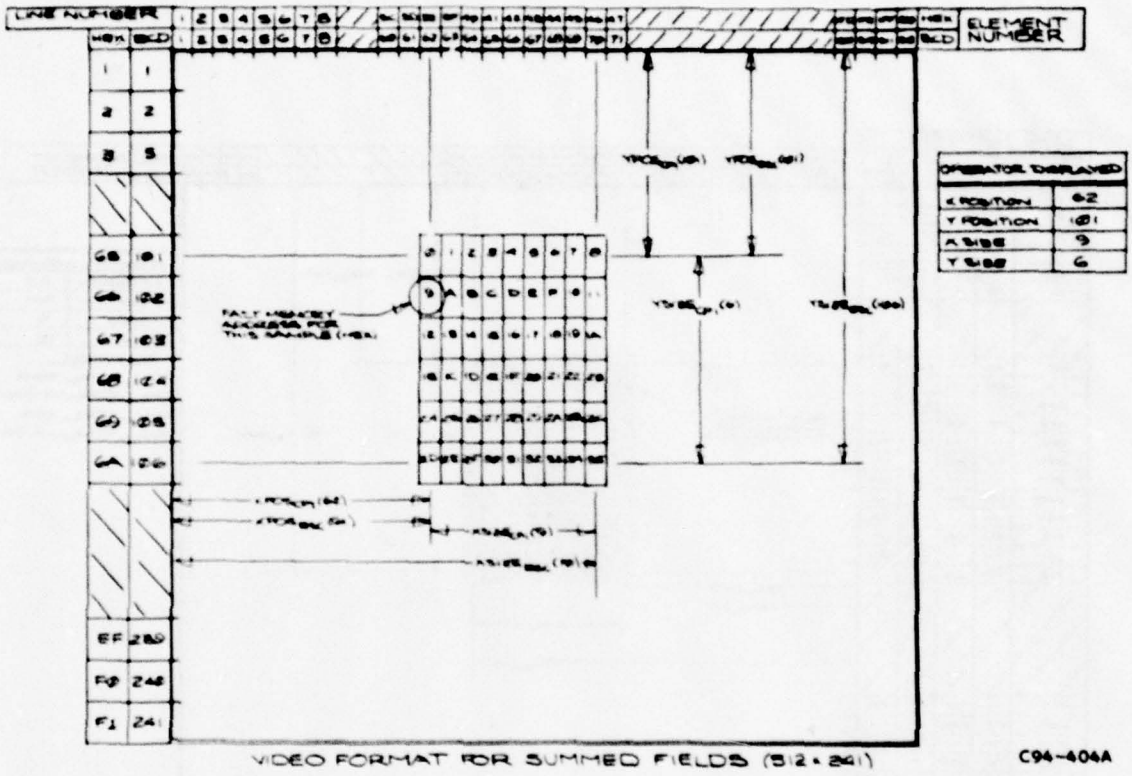


Fig. 16. Data and Video Format for Summed Field Mode.

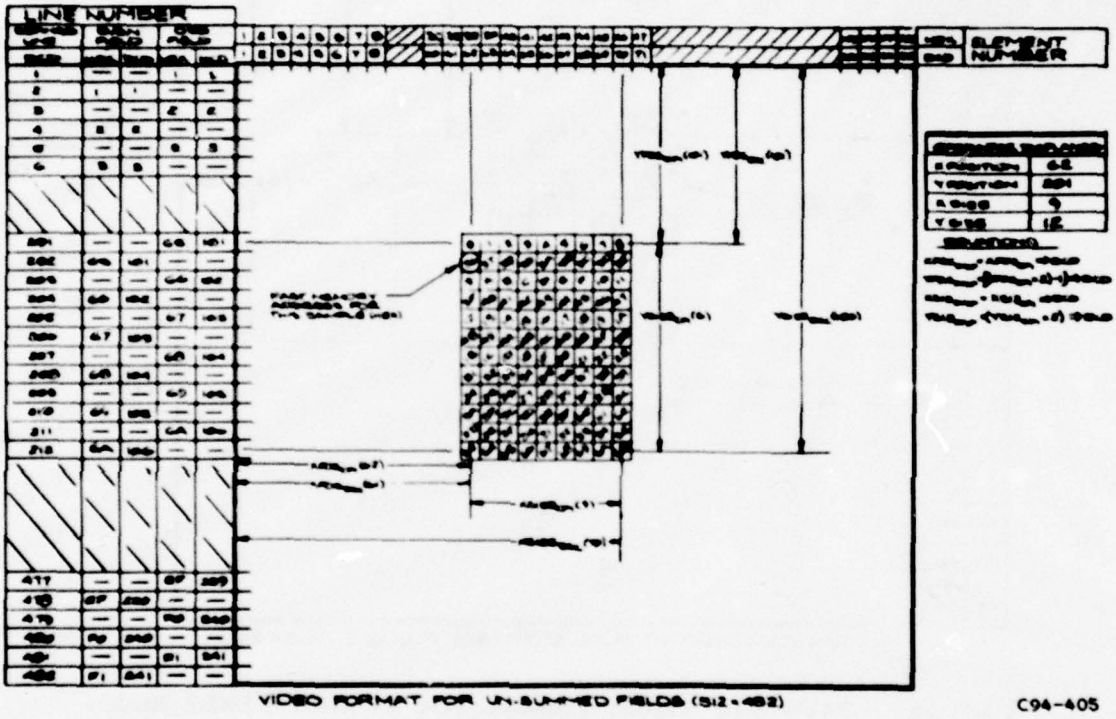


Fig. 17. Data and Video Format for Un-Summed Field Mode.

Shown in Figure 17 is the HEX location in the FMC of each sampled element. Notice that the second line of sampled elements is not located sequentially after the first line. The software will have to unpack the data for proper orientation of the sample area.

This mode will be useful when doing calibration processes as the Y direction of the screen is now divided into 482 segments rather than 241.

In the un-summed mode the Control Panel display will show the BOX position as being:

$$XPOS_{DISPLAYED} = XPOS_{CPI} = \text{BCD.}$$

$$YPOS_{DISPLAYED} = (YPOS_{CPI} \times 2)-1 = \text{BCD.}$$

$$XSIZE_{DISPLAYED} = XSIZE_{CIP} = \text{BCD.}$$

$$YSIZE_{DISPLAYED} = (YSIZE_{CPI} \times 2) = \text{BCD.}$$

The data that goes to the BSC will still have the same relationship with the CPI data, but to the operator the video area will appear to have twice as much resolution.

When the operator tells the CPI card to move the BOX one line, the Control Panel Display will actually show a two line increment. The same relationship will hold true for increasing the height. For example, in Figure 17 the Y position of the BOX is given by line 201. The operator can

only move the BOX to line 203. In this mode the smallest sample area has to be 2 lines, line 204 will be accessed, and thus sampled.

The Register and bit pattern information for this mode is described in the A/D Memory Port Card, Section VIII.

F. Software Timing Considerations for Data Sampling

The Box Sample Card description gives the detail of how the video sampling takes place. However, some overall timing consideration should be brought to light here.

Interrupt level 2 is the Begin Odd Field Retrace (BOFR) signal. Refer to Figure 12. All sampling parameters should be set up at this time for the Box Sample Card. The numbers for the BOX position, size and cursor center coordinates should be sent to the BSC.

At the End Odd Field Retrace (EOFR) time the data sent to the BSC is automatically loaded into the counters that determine the BOXWINDOW timing. The time between the Beginning and End of the Odd field retrace is approximately 1270 μs. This will be the maximum time the software has to set the video sample parameters. During this time the software should also enable the A/D Memory Port and the appropriate Fast Memory Cards.

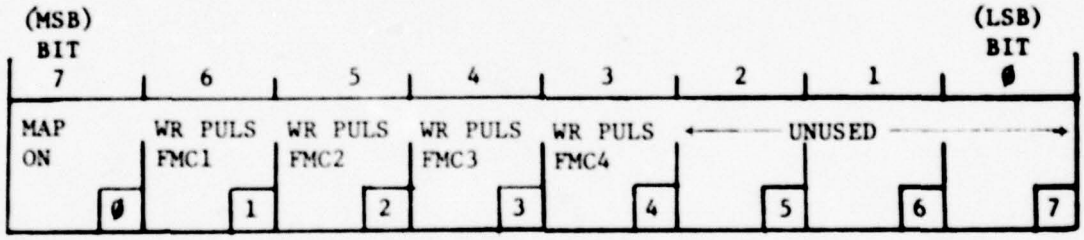
The sampling parameters need not be set again unless they are to be changed. The hardware will continue to use the original parameters until given a new set.

The important timing consideration is for all bus switching and BSC parameters to be set up during the video retrace.

VI. MICROPROCESSOR$_A$ MEMORY PORT (MAP)

Figure 18 is a block diagram of the Microprocessor$_A$ Memory Port (MAP). The MAP enables Microprocessor$_A$ access to any one of the 4K FMC memory blocks as though they were part of Microprocessor$_A$'s memory map.

The control for enabling the MAP is through the System Control Basket and in particular is register $D2. By enabling the MAP and one of the FMCs, Microprocessor$_A$ can read and write to the memory. This capability allows a hardware diagnostic for the Fast Memory Cards as well as some capability for data processing. It can also be a means of data transfer between Microprocessor$_A$ and the Modcomp or Microprocessor$_B$ since they will have access to the FMCs via the external port. Shown below is Register $D2 for accessing the MAP.



MAP ON: (Bit 7)
1 = The Microprocessor$_A$'s memory address and data bus is connected to one of the 4K blocks of memory.

Ø = Microprocessor$_A$ is disconnected from the common FMC bus.

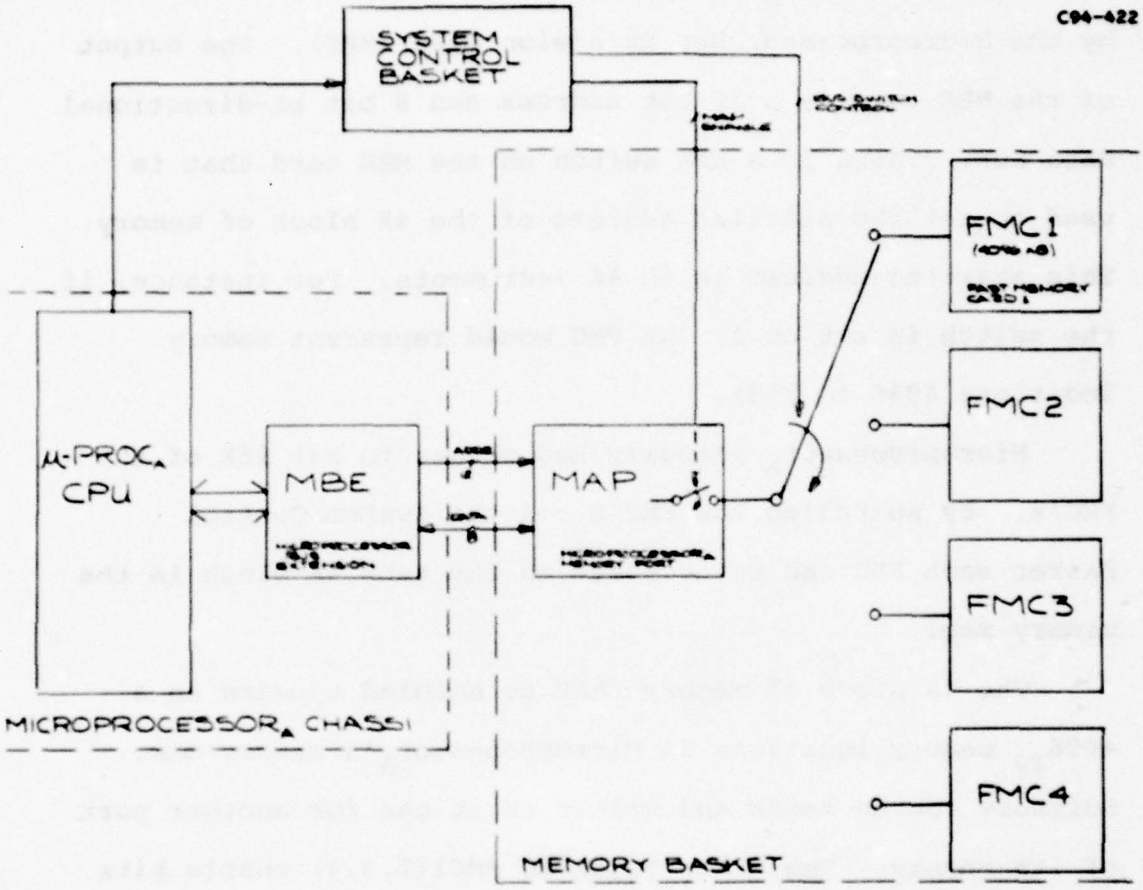


Fig.18. Microprocessor$_A$ Memory Port (MAP) - Block Diagram.

WR PULS FMC1(2,3,4): 1 = A write pulse will be sent to the specified FMC (during write cycle)
(Bits 6 - 3)

Ø = No write pulse sent.

Microprocessor$_A$'s memory bus is interfaced to the MAP by the Microprocessor Bus Extension Card (MBE). The output of the MBE card is a 12 bit address and 8 bit bi-directional data bus. There is a HEX switch on the MBE card that is used to set the starting address of the 4K block of memory. This starting address is in 4K increments. For instance, if the switch is set at 1, the FMC would represent memory locations 4096 to 8191.

Microprocessor$_A$ actually has access to all 16K of the FMC's. By switching the FMC's via the System Control Basket each FMC can be accessed as the same 4K block in the memory map.

The 4K block of memory that is enabled appears as a 4096_{10} memory locations in Microprocessor$_A$'s memory map. Software can do reads and writes as it can for another port of its memory. The Write Pulse to FMC1(2,3,4) enable bits is a result of the hardware 4 port design. This allows, up to all 4 FMCs to be written to at the same time. To have a FMC appear as a 4K block of memory, the particular FMC card must be enabled as well as the Write Pulse bit. In this mode the memory can be read as well as written to, just as any standard block of memory in Microprocessor$_A$'s memory map.

The Control Register Configuration to access FMC2 and to have it appear as a 4K memory block from $2000 to $2FFF, with the capability of being both read and written to, would be the following:

1. Set the switch on the MBE card to 2.
2. MAP Control Register, $D2 would have the following bit pattern.

   Bit 7 = 1 - ON/OFF

   Bit 6 = Ø - WR PULS FMC1

   Bit 5 = 1 - WR PULS FMC2

   Bit 4 = Ø - WR PULS FMC3

   Bit 3 = Ø - WR PULS FMC4

   Bits 2-Ø = X

3. Fast Memory Card 2 would be enabled and in the Internal Mode: Thus Register $DØ would have

   Bits 7,6 = Ø

   Bit 5 = 1 FMC2 ON/OFF

   Bit 4 = 1 FMC2 INT/EXT

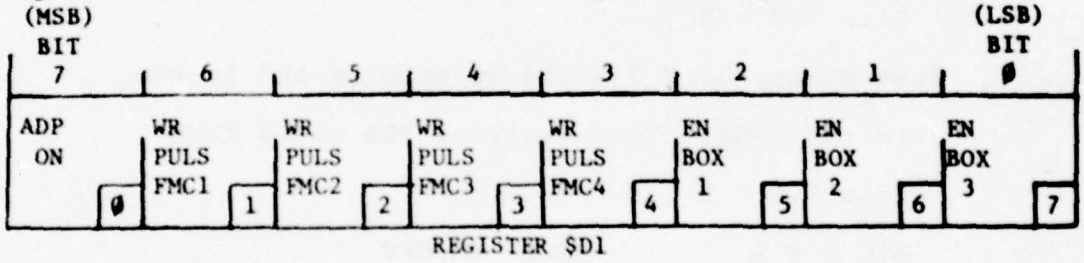
   Bits 3-Ø = Ø

VII. A/D MEMORY PORT (ADP)

Figure 19 shows a block diagram of the A/D Memory Port (ADP) Card. This card formats the digitized video for the Fast Memory Cards.

The digitized video is received by the ADP card and re-transmitted back to the 20 Mhz A/D, D/A box. Since the video is constantly being digitized the re-transmitted data can be converted back to analog and presented as one of the video displays. An operator comparison of this video and of the live video will give a verification of correct digitizing of the analog signal.

The control of the A/D Memory Port Card is through Register $D1 and $D3 with the bit patterns given below.



REGISTER $D1

ADP ON: (Bit 7) 1 = The A/D Memory Port has control of the common bus and can thus write to memory.

Ø = The A/D Port is disabled.

WR PULS FMC1(2,3,4): (Bits 5-3) 1 = Write pulses during the BOXWINDOW time will be sent to the appropriate (one or all) FMC's.

Ø = No writing is done to memory.

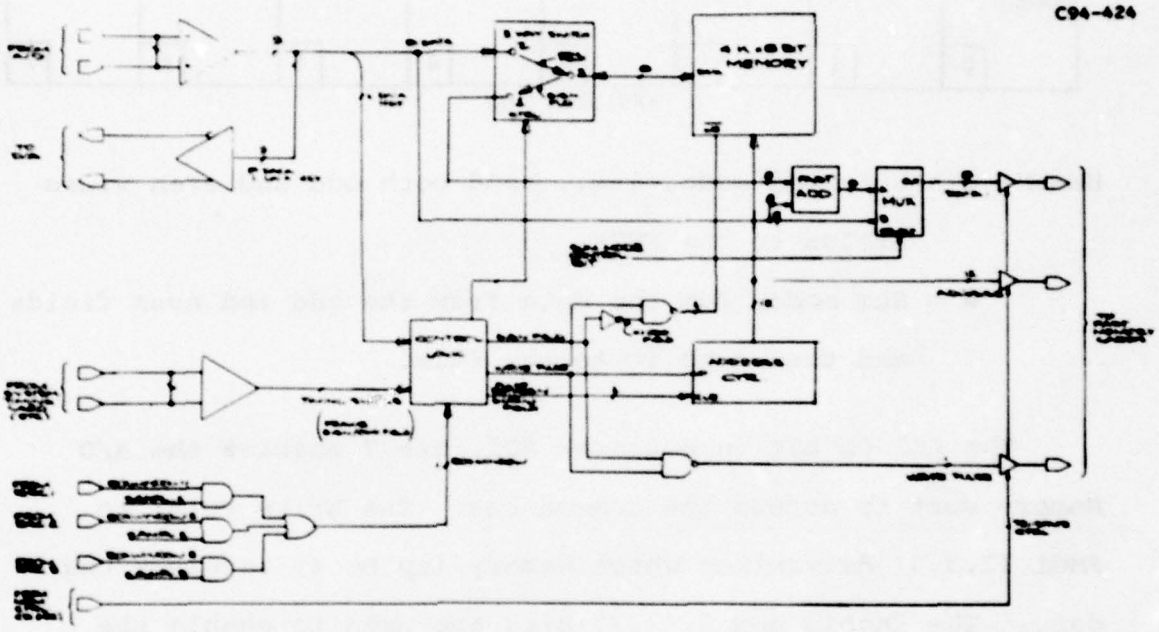
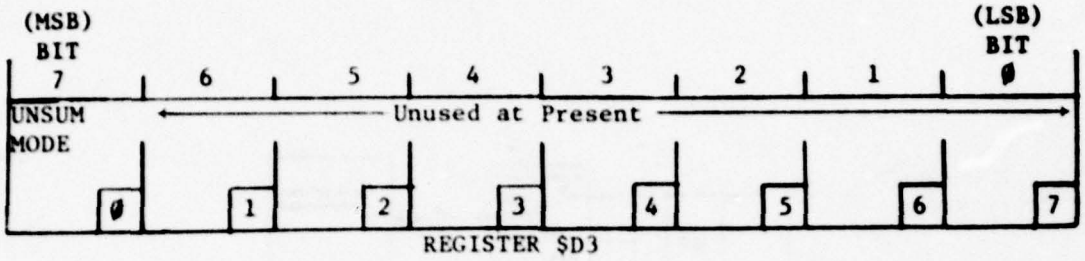


Fig.19. A/D Memory Port (ADP) - Block Diagram.

EN BOX1 (2,3): 1 = The appropriate sample box signal, BOXWINDOW, (Bits 2-Ø) will be enabled.

Ø = Disable BOXWINDOW1 (2,3).



UNSUM: 1 = Unsummed mode, i.e., send both odd and even video fields to the FMCs.

Ø = Sum mode, sum the data from the odd and even fields and then send it to the FMCs.

The ADP ON bit in Register $D1, bit 7 enables the A/D Memory Port to access the common bus. The Write Pulse to FMC1 (2,3,4) determines which memory (up to 4) receives the data. The Enable Box 1 (2,3) bits are used to enable the BOXWINDOW signal from each one of the Box Sample Cards (BSC1-3). This would depend on which sample area was to be sent to memory. The option is there, however, to enable up to all 3 sample areas to be sent to the FMCs.

If more than one FMC is to be loaded with data then the appropriate Write Pulse to FMC bits should be enabled along with the corresponding Fast Memory Cards. To enable the Fast Memory Cards the software accesses the FMC control

register, $DØ, a description of which is given in Section VIII.

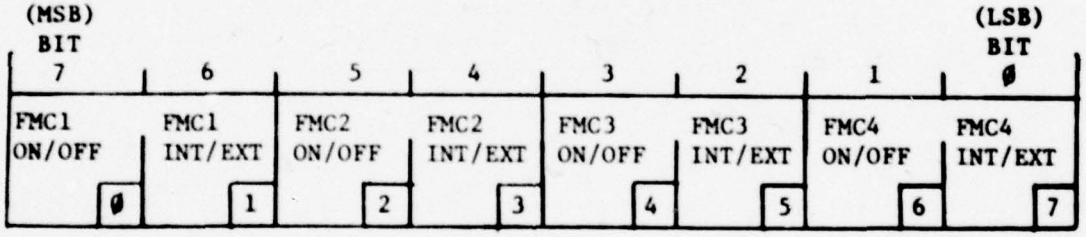
Register $D3 is used to define how the digitized video will be formatted before being sent to the FMCs. In the sum mode the corresponding elements of the odd and even fields of the sampled area will be summed before being sent to memory. In the un-summed mode the odd field will be sent to memory and then the even field. This mode will be used in star calibration type operations.

VIII. FAST MEMORY CARD (FMC)

There are four identical Fast Memory Cards (FMC) in the system. Each card is 4096 x 8 bits and can be accessed at a 20 Mhz rate. Figure 20 shows a block diagram of the card.

The memory has a 12 bit address, 8 bit data bus, read/write control, and a write pulse input. All these signals are connected to a 3-way digital switch. The memory can thus be accessed and controlled from either the common memory port bus or an external source. The third switch position is the off position where the memory is disconnected from both sources.

Register $DØ, an output register, is the control register for all four memories.



ON/OFF = 1 Memory is enabled and is connected to the common bus or connected to an external source depending on the state of the INT/EXT bit.

= Ø Memory is disconnected from system, INT/EXT bit is ignored.

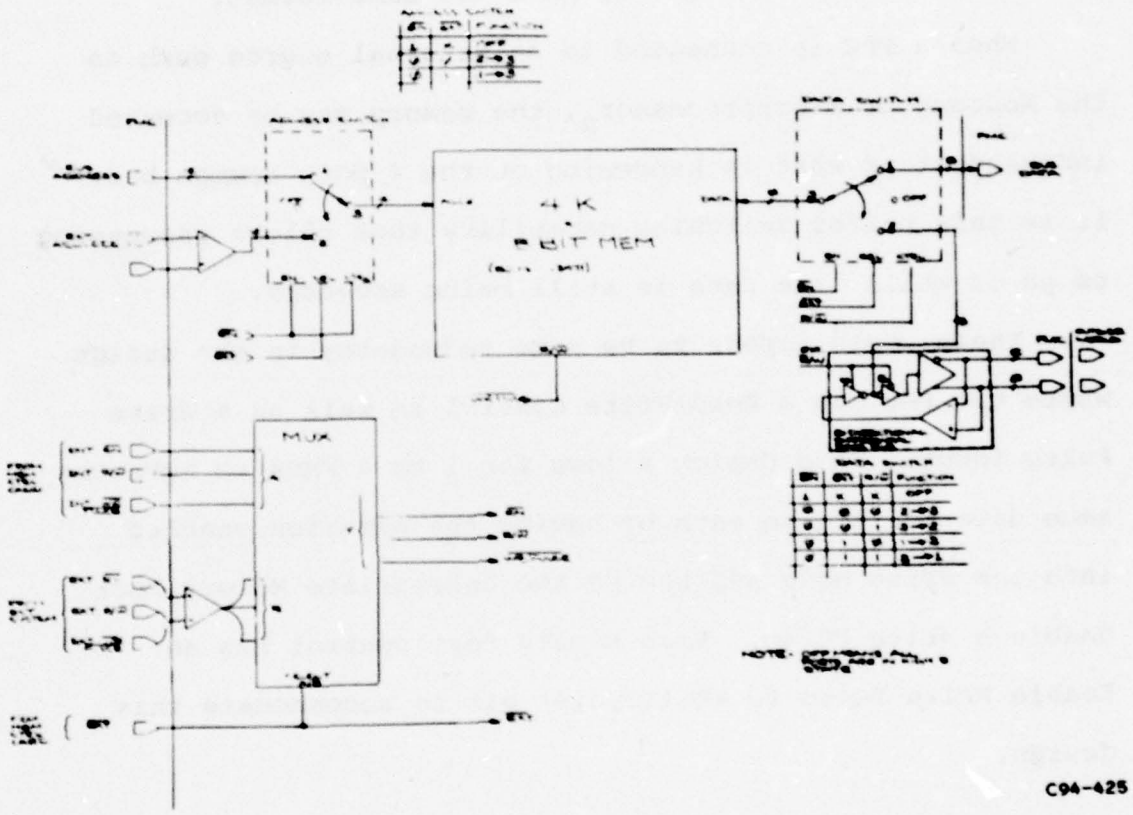


Fig.20. Fast Memory Card (FMC) - Block Diagram.

INT/EXT = 1 Memory is connected to common memory port system.

= Ø Memory is connected to an external source for independent access.

FMC2 - FMC4 (bits 5 - Ø) have the same format.

When a FMC is connected to an external source such as the Modcomp or Microprocessor_B, the memory may be accessed independent of what is happening on the 4 port common bus. It is this buffer switching capability that allows processing to go on while live data is still being accessed.

There would appear to be some redundancy in the design where the FMC has a Read/Write Control as well as a Write Pulse input. This design allows for 1 to 4 FMCs to have the same data written to each by having the memories enabled into the Write Mode and having the appropriate Memory Port enable a Write Pulse. Each Memory Port control has an Enable Write Pulse to FMC1(2,3,4) bit to accommodate this design.

IX. BOX SAMPLE CARD (BSC)

The Box Sample Card (BSC) contains the logic for generating the video sampling signal (called BOXWINDOW) from information received from Microprocessor$_A$. The BSC also generates the timing signal to draw the sample area and a cursor on the video screen. There are three separate BSCs thus allowing three separate sample areas to be defined.

There are six 9 bit words the BSC needs to define the sample area and the center of the cursor. They are:

XPOS YPOS XSIZE YSIZE	For the box
XCENTER YCENTER	For the cursor

The bit pattern and register numbers are found in the detailed logic description section of this report, Section XIII. Figure 21 shows a block diagram of the BSC.

The sample box area is generated only during the displayed portion of the video. The software has time to set and define the sample area during the vertical retrace. The Beginning of Odd Field Retrace (BOFR) signal interrupts Microprocessor$_A$ to alert it to send data to the BSC to define the box. These data are loaded into the holding registers in Figure 21. The information is then loaded into

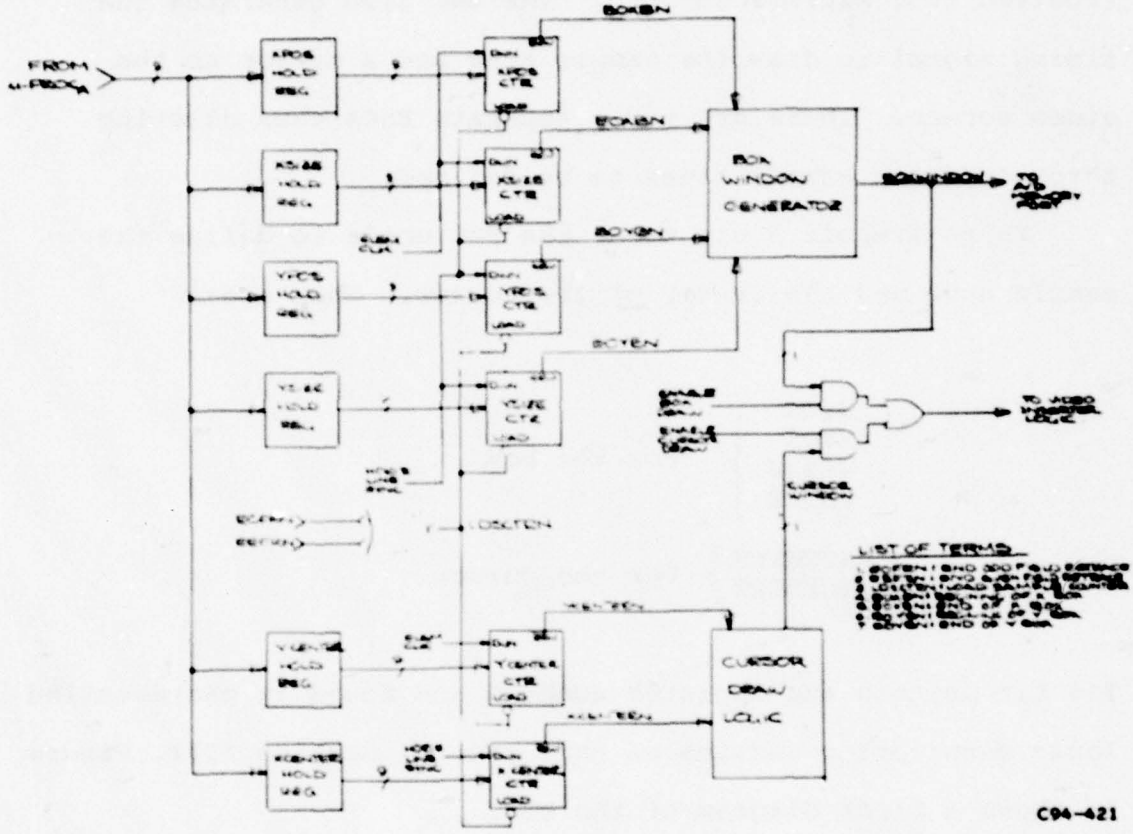


Fig.21. Box Sample Card (BSC) - Block Diagram.

the various box counters at the End of Odd Field Retrace (EOFR). The software has approximately 1270 μs to send this information to the holding registers before the vertical retrace time is over.

The XPOS number defines the left hand side of the sample area. The XSIZE number is the sum of the XPOS and the actual width of the box defined by the operator. These two counters are decremented by the element clock and are reloaded from the holding registers at the beginning of each line. Thus, when these two counters count down to zero, the borrow pulse from the XPOS counter defines the beginning of the sample box and the borrow pulse from the XSIZE counter defines the end of the box area (in the X direction).

The same philosophy holds for the Y counters. The Y direction counters are decremented by the line counter and are loaded at the end of each field retrace.

Thus the box is defined in the X direction and the Y direction. The logical AND of these signals generate the BOXWINDOW signal. This signal gates the digitized video into the FMC.

The BOXWINDOW signal is also used as the video insert signal to draw the box on the screen. The displaying of the sample area is controlled by the BOX Control Register with the bit pattern defined in Section XIII.

A cursor may also be drawn on the video. The height and width of the cursor is defined by the box size. The center of the cursor is defined by 2 more 9 bit registers called XCENTER and YCENTER.

The cursor is only used for display purposes. The X and Y center information is loaded into the holding registers at the beginning of the odd field retrace time. The Y center and X center counters behave identical to the X an Y position and size counter logic. The cursor may be displayed or blanked via a bit in the BOX Control Register.

The cursor may be used to show the centroid position of an object in the sample area. Or, if the box display is blanked and the cursor is enabled, the cursor could be used as a tagging character.

The cursor could also be made to look like the cross hairs now used at the ETS. By blanking the box display yet making its size to be as large as the screen, the cursor's X size and Y size would cover the whole screen. The software would now take the positional information from the CPI card as defining the center of the cursor. As the operator moves the joystick, the cross hairs would follow.

X. INTERRUPTS USED IN THE SYSTEM

The following is a discussion of the actual interrupts used in the system. Table 2 is a summary of the interrupts in the system.

A. Interrupt Levels

Level Ø is the highest priority interrupt and is used for the End of BOX (EOB) signal. This signal tells the software when the sampling has just finished. That is, when the last element of the sample area is loaded in memory, the EOB interrupt occurs. At this time the software can do the appropriate buffer switching to allow a processor to reduce the data.

Level 1 is the indication of the beginning of the odd field retrace (BOFR). This happens at video line 1. The software has approximately 1270 μs before the end of the odd field retrace occurs. At this time the sample counters are loaded with the sample parameters and the logic starts the video sampling process.

Level 2 is the beginning of the even field retrace (BEFR). This would probably be used for diagnostic or, in conjunction with the BOFR interrupt, it can be used as a 60 Hz timer.

TABLE 2

INTERRUPTS USED IN PVDS

INTERRUPT LEVEL	MNEMONIC	DESCRIPTION
ø	EOB	End of Box Interrupt - tells when the sample time is furnished (highest priority).
1	BOFR	Begin Odd Field Retrace - this is the beginning of the retrace for the odd field. All parameters for the sample are set up during this interrupt
2	BEFR	Begin Even Field Retrace - this is the beginning of the retrace for the even field. This will probably be used for diagnostics or as a 60 Hz timer.
3		
4		Unused
5		
6		
7	CI	Console Interrupt. This is from a switch on the Control Panel.

Interrupt Levels 3 - 6 are unassigned at present.

Interrupt level 7 is the Console Interrupt. This is activated by a momentary switch on the Control Panel. It will be used to alert the software to service the operator.

B. Enabling/Disabling the Interrupts

There are several levels of enabling and disabling the interrupts in the system. The first is in the CPU itself.

There is an Interrupt Mask that may be set or cleared under program control. The second is control register bit Ø of the INTREG PIA. A 1 in this bit enables the interrupt, a Ø disables it. The final level of control is through the 8 interrupts in the logic. The interrupts are enabled or disabled through output register $4Ø. The 3 LSBs are the interrupt number (high truth). Bit 3 is set to a 1 to enable the interrupt and a Ø to disable it.

XI. CONTROL PANEL

The Control Panel is shown in Figure 22. This panel is made up of programmable switches and LEDS. The two banks of LEDS (8 each) represent two output registers to the software and can be lit or extinguished under program control. The two sets of 8 toggle switches are two input type registers and their states can be interrogated by the software. Normally the LED above each switch will be lit, by the software, when the operator pushes a switch up. This will provide verification that the software received the correct switch information.

Also shown in Figure 22 is the Sample Area's position and size display as well as the centroid position of the object. These HEX LEDS are driven by the software.

The detailed bit pattern and register information for the Control Panel LED's and switches as well as the sample area display information is given in Section XII. This section of the report gives a qualitative description of the functions on the Control Panel.

A. Memory Control

The top row of switches defines the memory control. This includes the access to the Fast Memory via the A/D Memory Port (ADP) or the Microprocessor_A Memory Port (MAP).

AUTO/MAN: This switch defines whether the software will control the Memory (AUTO) or if in the manual (MAN)

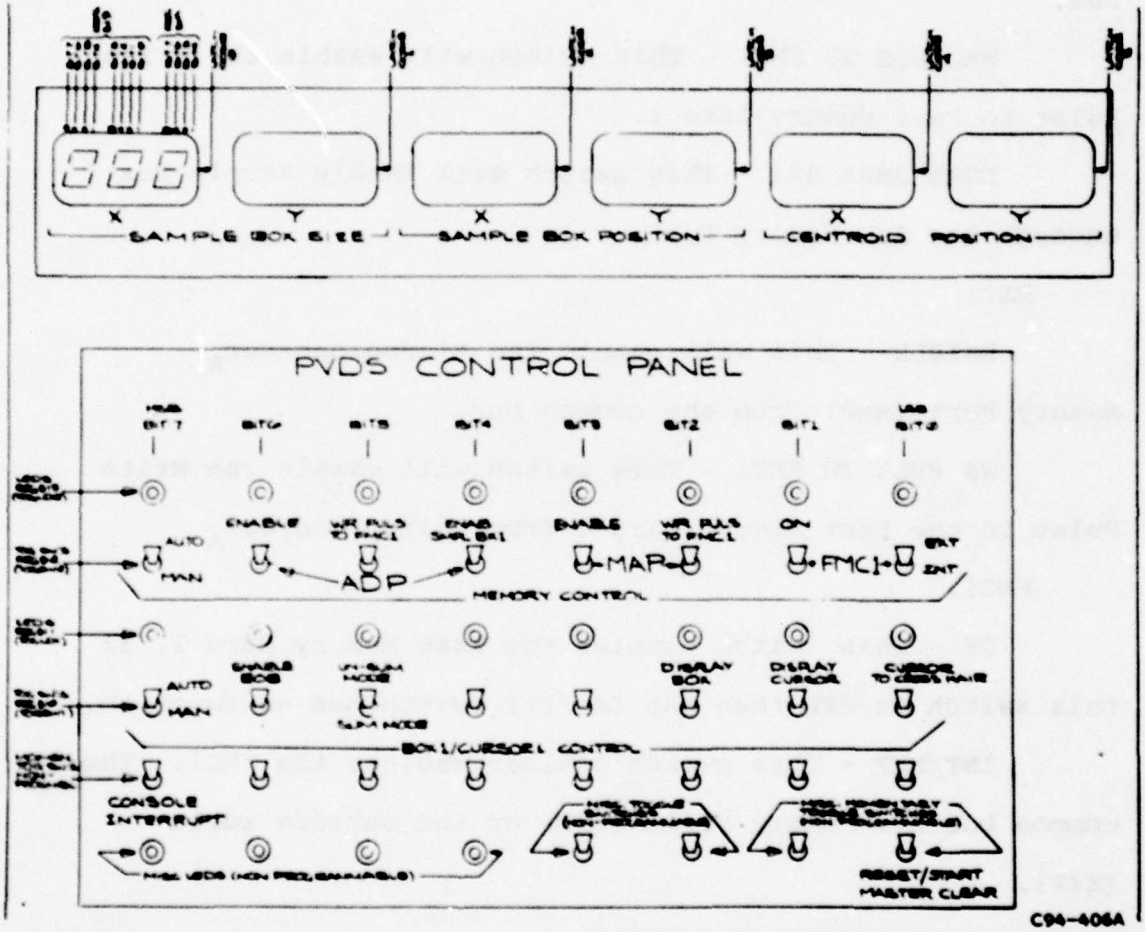


Fig.22. PVDS Control Panel.

mode, memory control will come from the operator via the switch settings.

ADP:

ENABLE - When this switch is up, the video A/D Memory Port is enabled and will have access to the common memory bus.

WR PULS TO FMC1 - This switch will enable the Write Pulse to Fast Memory Card 1.

ENAB SMPL BX1 - This switch will Enable Sample Box 1 through the A/D Memory Port.

MAP:

ENABLE - This will enable the Microprocessor$_A$ Memory Port (MAP) onto the common bus.

WR PULS TO FMC1 - This switch will enable the Write Pulse to the Fast Memory Card 1 from Microprocessor$_A$

FMC1:

ON - This switch enables the Fast Memory Card 1. If this switch is OFF then the INT/EXT switch has no function.

INT/EXT - This switch defines who has the FMC1. The common bus and Memory Ports (INT) or the outside world (EXT).

B. BOX1/CURSOR1 Control

AUTO/MAN - In the Auto Mode the software will control the conditions. The Manual Mode (MAN) will allow the operator to define the controls.

ENABLE EOB - This switch Enables the End of Box interrupt to be passed on to the Microprocessor$_A$.

UNSUM/SUM MODE - This defines what mode the digitized video is stored into memory: fields summed or unsummed before being sent to FMC1.

DISPLAY BOX & DISPLAY CURSOR - These switches will enable or disable the displaying of the Sample Box or Cursor on the video.

CURSOR TO CROSS HAIR - This switch allows the operator to define the cursor as a cross hair similar to the one now used at the GEODSS ETS.

The non-programmable switches used on the Control Panel are a CONSOLE INTERRUPT Switch and a RESET/START MASTER CLEAR switch.

The Console Interrupt is connected to interrupt number 7 of Microprocessor$_A$ and will be used by the operator to get Microprocessor$_A$'s attention for various tasks.

The Reset/Start Master Clear switch sends a master clear pulse to the logic as well as pulsing Microprocessor$_A$'s RESET line. This will cause it to go to the program initialization routine and restart the operating program.

C. CONSIDERATIONS

The Control Panel reflects a minimum configured PVDS system. There is only control for one Fast Memory Card

(FMC1) and one Box Sample Card (BSC1). The system is designed and built to handle 4 FMCs on 3 Sample Boxes but the initial software and hardware development will start with a minimum system as the Control Panel shows.

XII. INITIAL AND FINAL SYSTEM OBJECTIVES

Since the PVDS is a development system and we need to define and write some fundamental Bookkeeping and control software the initial system will be hardware configured in the following manner -

1. There will be 1 Fast Memory Card.
2. There will be 1 Sample Area (Box Sample Card).
3. Microprocessor_A will be both the bookkeeping and control processor as well as the unit for doing data reduction on the digitized video.

The initial software program development should include the following -

1. Communication to the Control Panel
   a. This would include reading the switches and lighting the LEDs behind the switches.
   b. This would also include getting BOX position and size information and displaying such in the HEX LEDs.
2. Program to control the Sample Area Size and Position
   a. Read data from CPI card.
   b. Format the data and send to BSC
   c. Enable the FMC and the ADP via instructions from the Control Panel.
   d. Convert position and size data to BCD display format for operator display.

e. Control the formatting depending on the state of the UN-SUM or SUM Field switch.

3. Calculation of Centroid

   a. The Sample Box will be positioned by the operator and the digitized data are snatched by the FMC.

   b. The centroid of the object is computed and displayed in the HEX LEDs on the Control Panel.

   c. The cursor is enabled with the center being over the centroid of the object.

   d. Initially this routine will not have to run at a real time rate, i.e., 30 Hz.

4. Make the cursor into a Cross Hair controlled via the joystick.

5. A PLC type Aid -

   This routine would allow the operator to put the sample box over a star. The system would snatch a frame of data in the un-summed field mode. The program would display the centroid of the object and give the RA and DEC position calculated relative to an earlier calibration sequence. In this mode the star of interest would not have to be boresighted each time to find its position. A routine could be written to map any non-linearities in the camera to allow the position of the star to be determined when it wasn't at boresight.

6. Fast Memory Diagnostics

   a. Connect the Microprocessor$_A$ Memory Port to the FMC1.

   b. Write data to the memory and then read it back for verification.

After these initial objectives are met the centroiding software routines will be transfered to Microprocessor$_B$ which will be connected to one or two Fast Memory Cards. The Modcomp will also be connected to a Fast Memory Card.

The hardware communications between Microprocessor$_A$ and both the Modcomp and Microprocessor$_B$ are already built and software can be written. The other 3 Fast Memory Cards and the two Box Sample Cards only have to be assembled. The controlling logic and cabling is already installed and checked out.

Thus, after the above initial software routines have been developed, the basic hardware checked out and a familiarity with the system accomplished the final system objectives will be -

1. Microprocessor$_A$ will control operator communications bookkeeping routines and system synchronization.
2. Microprocessor$_A$ will have access to the FMCs and will do centroiding algorithms or PLC type aiding routines.

3. The centroid of an object as well as x,y vector information will be passed to the Modcomp.
4. The Modcomp will have access to a FMC and will have the capability of doing data recording or reduction.

XIII. SYSTEM CONTROL LOGIC - DETAILED REGISTER DESCRIPTION

This section of the report describes, in detail, all the registers in the system. The bit pattern and functional description is given. The register address in hex ($=HEX symbol), register type, logic card name and location, software label and functional description are also given.

The number in the lower right hand corner of each register bit pattern layout represents the bit number that the hardware uses. Unfortunately, the hardware defines bit 7 as the LSB whereas Motorola defines bit 7 as the MSB. All bit pattern descriptions in this report refer to the software's interpretation of bit 7 being the MSB.

DETAILED DESCRIPTION OF

ALL LOGIC REGISTERS

LOGIC CARD NAME
LOGIC CARD LOCATION } UNUSED AT PRESENT

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$00			
$01			
$02			
$03			
$04			
$05			
$06			
$07			
$08			
$09			
$0A			
$0B			
$0C			
$0D			
$0E			
$0F			

LOGIC CARD NAME : Control Panel Interface Card (CPI)

LOGIC CARD LOCATION : System Control Basket 1 (SCB-1)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$1Ø	INPUT	BXCSW	BOX1/Cursor1 control switches
$11	OUTPUT	BXCLD	8 LEDs above 1st set of toggle switches
$12	INPUT	MCSW	Memory control toggle switches
$13	OUTPUT	MCLD	8 LEDs above 2nd set of toggle switches
$14	INPUT	XSZIN	8 MSBs of operator requested X sample size
$15	INPUT	YSZIN	8 MSBs of operator requested Y sample size
$16	INPUT	XPOSIN	8 MSBs of operator requested X sample area position
$17	INPUT	YPOSIN	8 MSBs of operator requested Y sample area position
$18	INPUT	XYLSBS	4 LSBs of operator requested X,Y size and X,Y position
$19	OUTPUT	DISP1	8 MSBs of the 12 bit HEX LED displays on control panel
$1A	OUTPUT	DISP2	4 MSBs of HEX LED data plus, 4 bit latch code
$1B			
$1C			
$1D			
$1E			

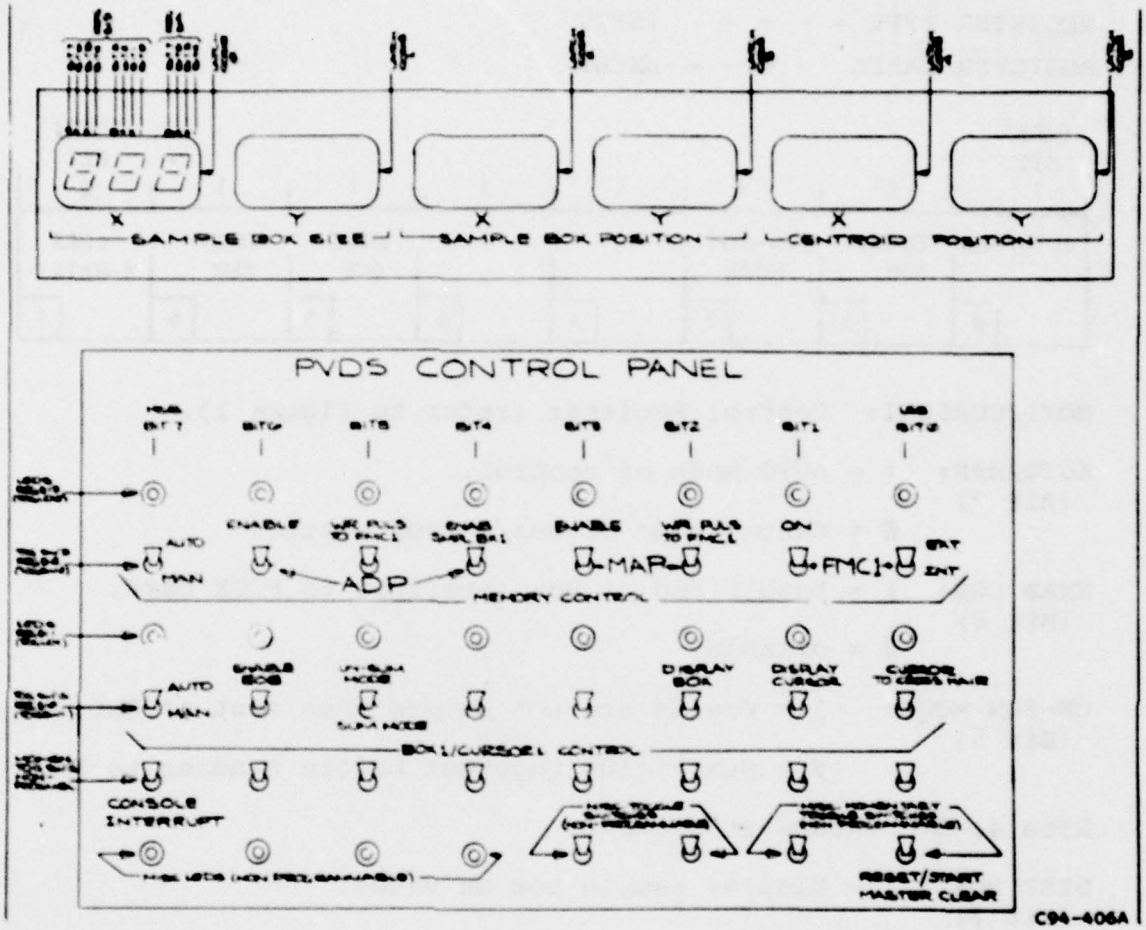


Fig.23. Control Panel.

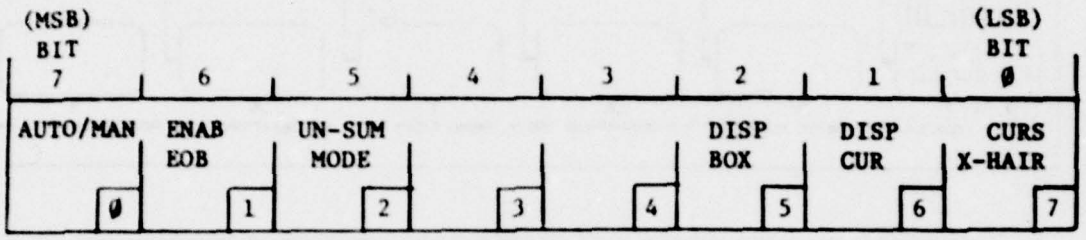
CARD NAME - - - - - - - - CPI

CARD LOCATION - - - - - - SCB-1

REGISTER ADDRESS - - - $1Ø

REGISTER TYPE - - - - - INPUT

REGISTER LABEL - - - - BXCSW



BOX1/CURSOR1: Control Register (refer to Figure 1).

AUTO/MAN: (Bit 7) 1 = AUTO mode of control.
Ø = Manual mode of box/cursor control.

ENAB EOB: (Bit 6) 1 = Enable End of Box interrupt to PCCX card.
Ø = Disable.

UN-SUM MODE: (Bit 5) 1 = Fields are not summed when sent to FMC.
Ø = Sum fields together before sending to FMC.

Bits 4, 3: Unused at present.

DISP BOX: (Bit 2) 1 = Display sample box on video.
Ø = Blank.

DISP CURS: (Bit 1) 1 = Display cursor on video.
Ø = Blank.

CURS X-HAIR: 1 = Make cursor into a crosshair and ignore bits 1 and 2.

Ø = Use bits 1 and 2 to determine box and cursor function.

This register reflects the operator requested mode of the BOX, CURSOR and the sample area. The actual control of the BOX and CURSOR is through registers $5Ø - $57. The sample box enable control is found in register $Dl.

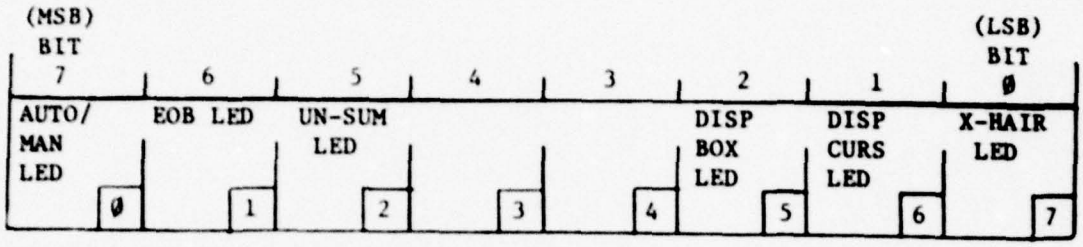
CARD NAME - - - - - - - - CPI

CARD LOCATION - - - - - - SCB-1

REGISTER ADDRESS - - - - $11

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - BXCLD

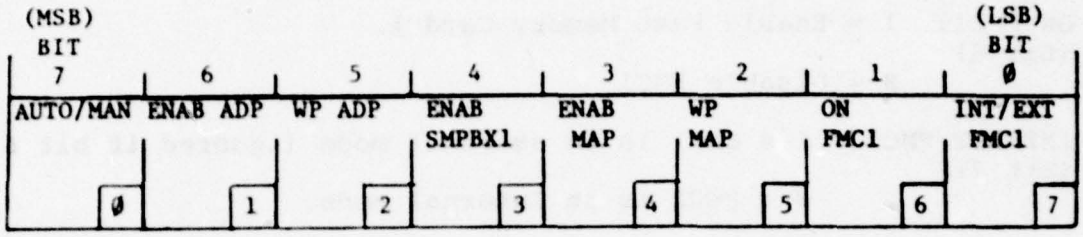


1 = Light LED

Ø = Extinguish LED

This register represents the 8 LEDS above the BOX1/CURSOR1 control toggle switches ($1Ø). Normal operation would call for the LEDs to be lit when the switches are up (i.e., bits = 1). Refer to Figure A.

CARD NAME - - - - - - - - CPI
CARD LOCATION - - - - - - SCB-1
REGISTER ADDRESS - - - $12
REGISTER TYPE - - - - - - INPUT
REGISTER LABEL - - - - - MCSW



Memory control switches (refer to Figure 23).

AUTO/MAN: (Bit Ø)
1 = Memory controlled via software routines.
Ø = Memory controlled manually, i.e., via these switches.

A/D Memory Port Control

ENAB ADP: (Bit 1)
1 = Enable A/D memory port on to common address bus.
Ø = Disable ADP.

WR ADP: (Bit 2)
1 = Allow the write pulse from the A/D memory port to be passed to Fast Memory Cards.
Ø = Disable write pulse.

ENAB SMPBX1: (Bit 3)
1 = Enable the BOXWINDOW signal from Sample BOX1 to the FMCs.

Microprocessor$_A$ Memory Port Control

ENAB MAP: (Bit 4)
1 = Enable Microprocessor$_A$ Memory Port onto common address bus.
Ø = Disable MAP

WP MAP: (Bit 5)
1 = Enable the write pulse from Microprocessor$_A$ to be passed on to Fast Memory Card.

Ø = Disable write pulse.

Fast Memory Card 1 Control

ON FMC1: (Bit 6)
1 = Enable Fast Memory Card 1.

Ø = Disable FMC1.

INT/EXT FMC1: (Bit 7)
1 = FMC1 is in external mode (ignored if bit 6 = Ø).

Ø = FMC1 is in Internal mode.

These switches are operator inputs to the controlling software and have the above definitions. The registers that actually control the A/D Memory Port, Microprocessor$_A$ Memory Port, the Fast Memory Cards, found in registers $DØ through $D3.

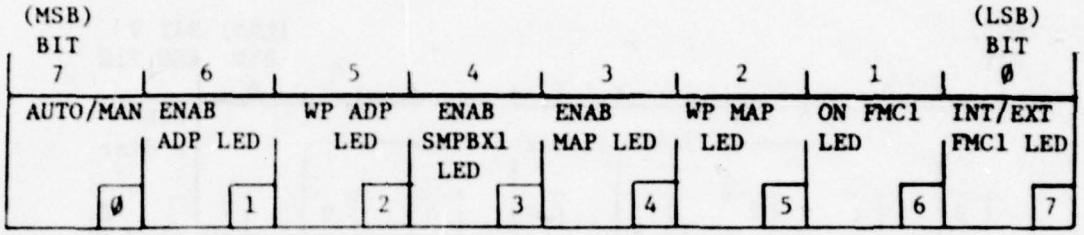
CARD NAME - - - - - - - - CPI

CARD LOCATION - - - - - - SCB-1

REGISTER ADDRESS - - - $13

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - MCLD



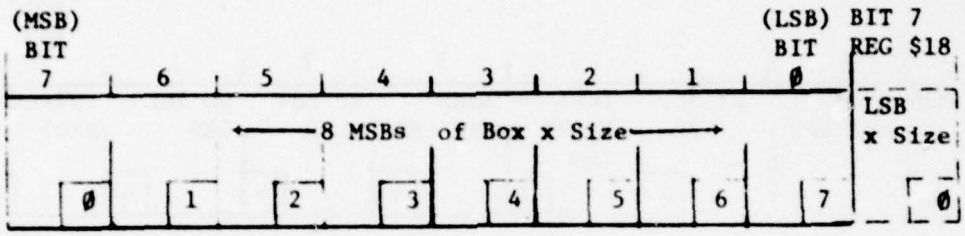
Memory Control LEDs.

1 = LED is lit.

Ø = LED is extinguished.

This register represents the LEDs above the Memory Control toggle switches. Normal operation, in the Manual Mode call for the LEDs to be lit when the switches are up (i.e., = 1). Refer to Figure A.

CARD NAME - - - - - - - - CPI
CARD LOCATION - - - - - - SCB-1
REGISTER ADDRESS - - - - $14
REGISTER TYPE - - - - - - INPUT
REGISTER LABEL - - - - - XSZIN



This is the 8 MSBs of the operators requested box sample size in the x direction, i.e., the box width. The box size is represented by a 9 bit binary number. The LSB is found in Register $18, bit 7. Figure 24 is a graphic representation of the sample area. The binary number is high truth. The representation of this term in Figure 24 is $XPOS_{CPI}$ and is equal to

$$XSize_{CPI} = \$01D = 0|0001|1101| = 29_{10}$$

This means the box sample width would be 29_{10} elements wide.

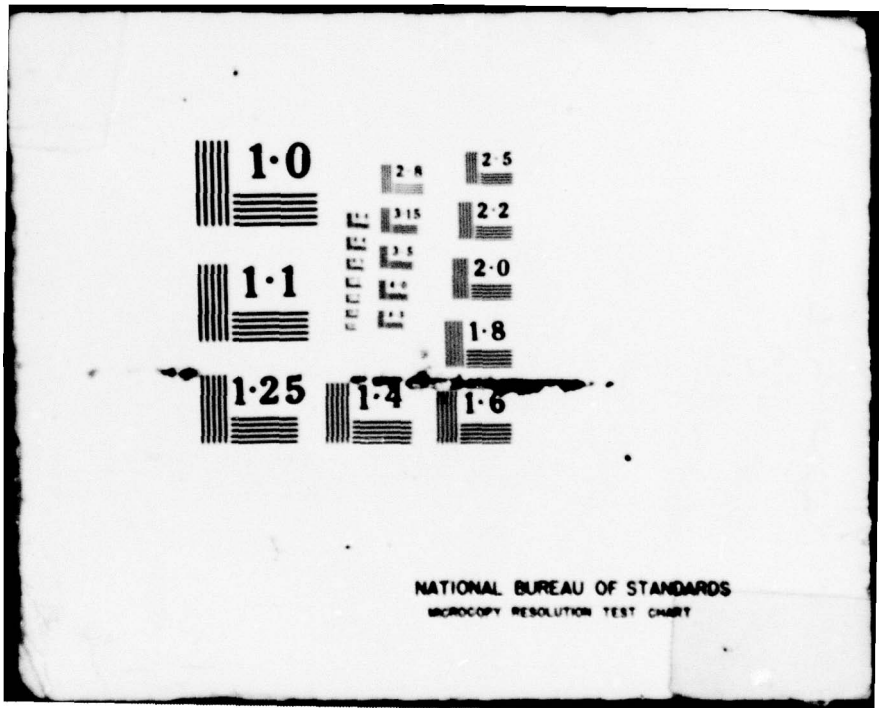
When reading this register, the software must always read Registers $14 through $18.

AD-A073 756 MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB F/G 9/2
PROGRAMMABLE VIDEO DIGITIZING SYSTEM (PVDS) - FUNCTIONAL DESCRI--ETC(U)
JUN 79 L E EATON F19628-78-C-0002

UNCLASSIFIED ETS-46 ESD-TR-79-156 NI

2 OF 2
AD-
A073756

END
DATE
FILMED
11-79
DDC


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

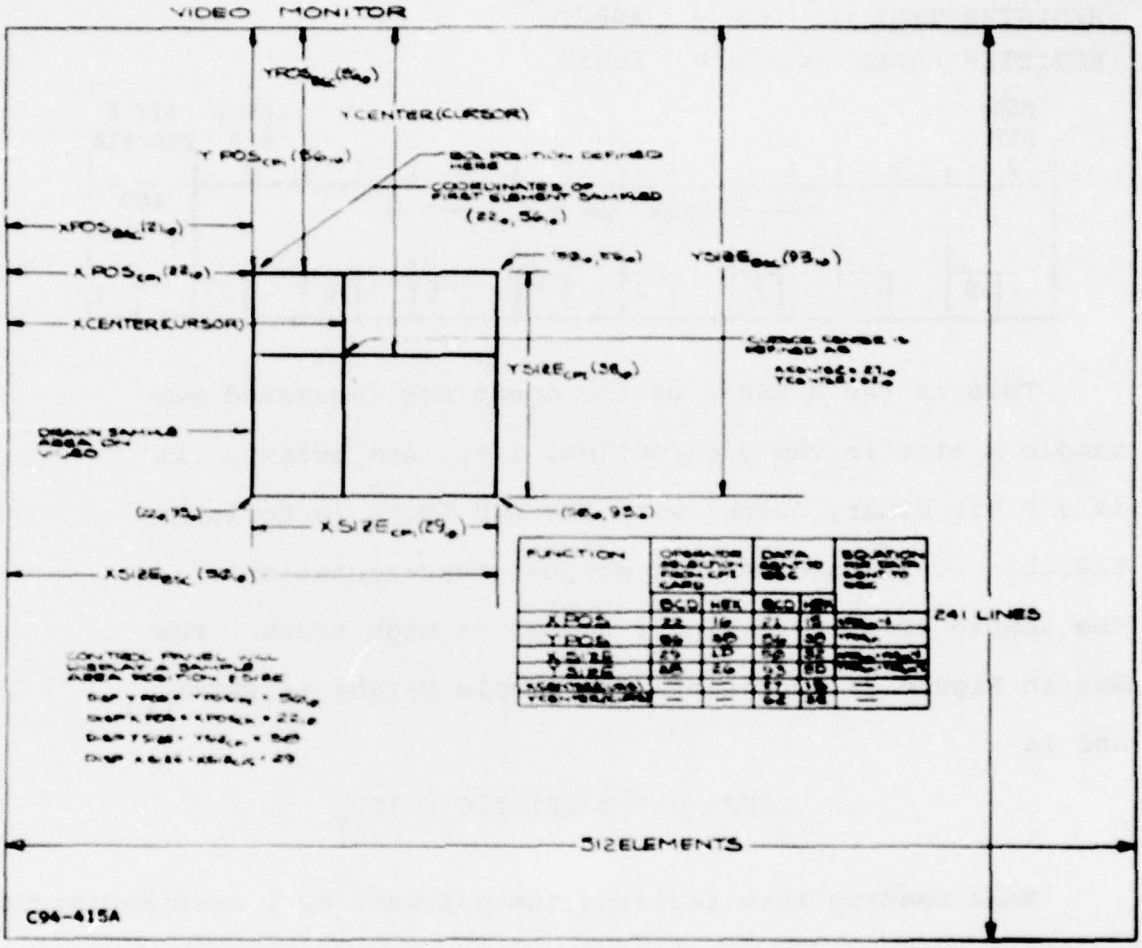


Fig.24. Sample Area Definitions.

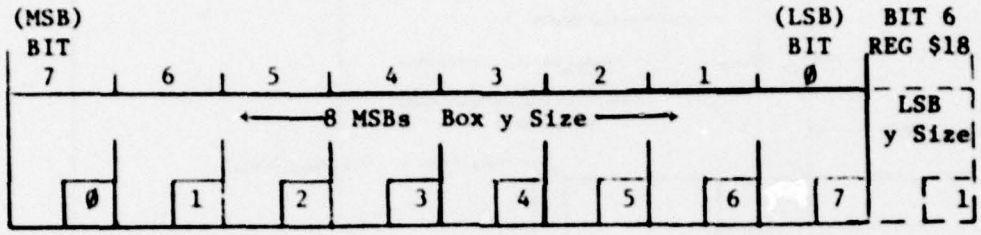
CARD NAME - - - - - - - - CPI

CARD LOCATION - - - - - - SCB-1

REGISTER ADDRESS - - - $15

REGISTER TYPE - - - - - - INPUT

REGISTER LABEL - - - - - YSZIN



This is the 8 MSBs of the operators requested box sample x size in the y direction, i.e., box height. It is a 9 bit binary number with the LSB found in Register $18, bit 6. Figure 24 is a graphic representation of the sample area. The binary number is high truth. The Box in Figure 24 represents the sample height as $ySize_{CPI}$ and is

$$\$Ø26 = 0|0010|0110 = 38_{1Ø}$$

When reading this register, the software must read Registers $14 through $18.

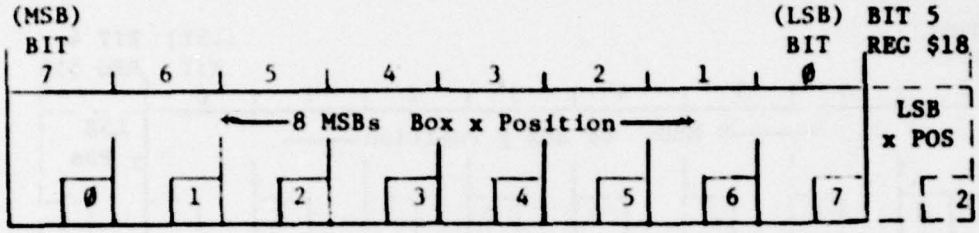
CARD NAME - - - - - - - - CPI

CARD LOCATION - - - - - - SCB-1

REGISTER ADDRESS - - - $16

REGISTER TYPE - - - - - - INPUT

REGISTER LABEL - - - - - XPOSIN



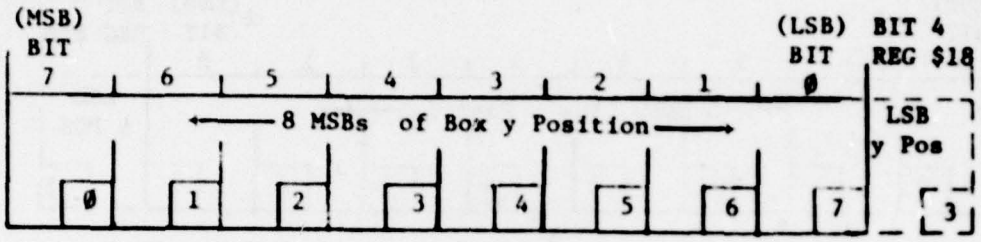
This is the 8 MSBs of the operators requested sample box position in the x direction. This number is the x co-ordinate of the corner of the sample box. The number is a bit high truth binary number with the LSB found in Register $18 bit 5. It is represented in Figure 24 by $xPOS_{CPI}$ and is equal to:

$$\$16 = 0|0001|0110| = 22_{10}$$

Thus, the first element sampled in the x direction would be number 22_{10}.

When reading this register, the software must always read Registers $14 through $18.

CARD NAME - - - - - - - - CPI
CARD LOCATION - - - - - - SCB-1
REGISTER ADDRESS - - - $17
REGISTER TYPE - - - - - - INPUT
REGISTER LABEL - - - - YPOSIN



This register is the 8 MSBs of the Y position of the box sample area. This is the y coordinate of the upper left hand corner of the box. The 9 bit number is a high truth binary number with the LSB found in bit 4 of Register $18. The number is represented in Figure 24 by $yPos_{CPI}$ and is equal to:

$$\$Ø38 = 0|0011|1000| = 56_{1Ø}$$

When reading this register, the software must always read Registers $14 through $18.

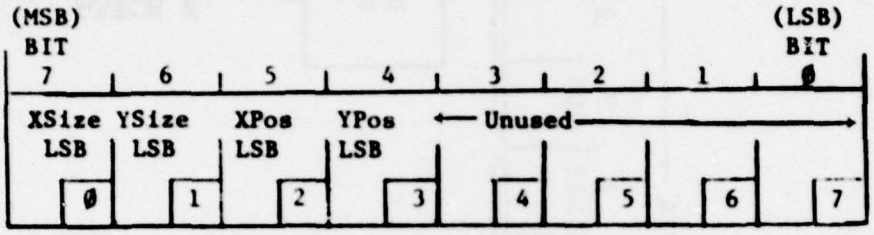
CARD NAME - - - - - - - - CPI

CARD LOCATION - - - - - - SCB-1

REGISTER ADDRESS - - - - $18

REGISTER TYPE - - - - - - INPUT

REGISTER LABEL - - - - - XYLSBS



This word contains the LSB at the 4, 9 bit. Words that define the operators requested sample box size and position.

When reading this register, the software must always read Registers $14 through $18.

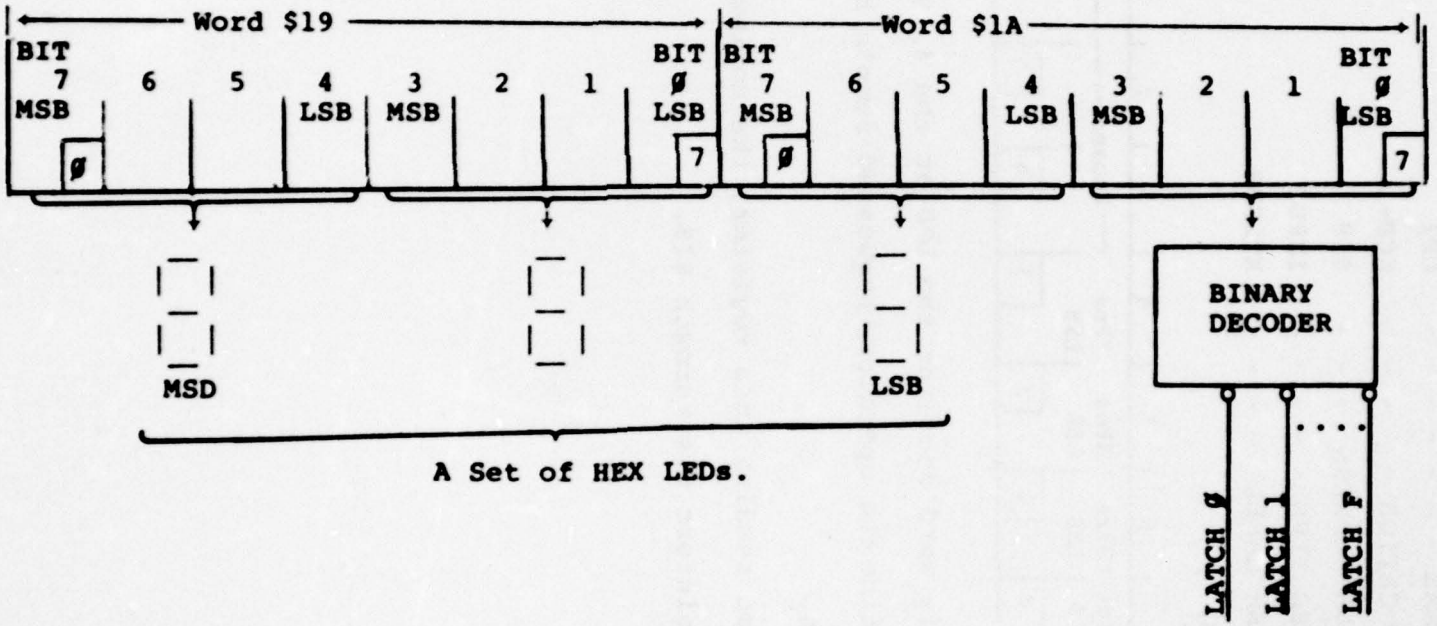
CARD NAME CPI

CARD LOCATION SCB-1

REGISTER ADDRESS $19 and $1A

REGISTER TYPE OUTPUT & OUTPUT

REGISTER LABEL DISP1 & DISP2



A Set of HEX LEDs.

Data are high truth binary.

The Control Panel (Figure 23) contains 6 sets of HEX LED Displays. Each set consists of 3 HEX LEDs. Each HEX LED requires a 4 bit binary number. For 3 LEDs, the 12 bits of data come from word $19 and the 4 MSBs of word $1A. The set of HEX LEDs that receives these data are determined by the 4 LSBs of word $1A and is shown in Table 2.

All sets of HEX LEDs (3 per set) on the Control Panel receive the same data. These data are packed in registers $19 and $1A. However, only one set of LEDs at a time receives a latch pulse which is decoded from the 4 LSBs of register $1A. Table 2 shows which set of LEDs receives a latch pulse for a given code.

TABLE 3

LATCH CODE FOR HEX LEDs

BITS 3 - Ø REG $1A	DATA HEX LEDS THAT ARE SENT TO
$Ø	BOX WIDTH
$1	BOX HEIGHT
$2	BOX X POSITION
$3	BOX Y POSITION
$4	CENTROID-X
$5	CENTROID-Y
$6	
$7	
$8	
$9	
$A	UNUSED AT PRESENT
$B	
$C	
$D	
$E	
$F	

Thus, if a BOX WIDTH of 128_{10} were to be displayed, the software would send a $12 to register $19 and then a $8Ø to register $1A. The $8 would be the last data number in 128 and the $Ø would be the latch code to send the 128 to the BOX WIDTH LEDs. Register $19 has to be loaded first and $1A second for proper operation.

The LEDs will display hexidecimal numbers. This may be useful for diagnostic purposes. However, for operator convenience the software should convert to a BCD format.

1. BOX WIDTH(X) - This will display the width, in BCD, of the sample area. The number is direct binary to BCD conversion of the data from Register $14 and $18.

2. BOX HEIGHT(Y) - This will display the height, in BCD of the sample area. The number is a direct binary to BCD conversion of the data from Register $15 and $18.

3. BOX POSITION(X) - This will display the BCD X position of the sample area's center. This is the upper left hand corner position required by the logic.

4. BOX POSITION(Y) - This will display the BCD Y position of the sample area's center. This is the upper left hand corner position required by the logic.

5. CENTROID POSITION(X) - This will be the calculated BCD X position of the centroid of an object within the box.

6. CENTROID POSITION(Y) - This will be the calculated BCD Y coordinate of the centroid of an object within the box.

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION SYSTEM CONTROL BASKET-2 (SCB-2)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$20			
$21			
$22			
$23			
$24			
$25			
$26			
$27			
$28			
$29			
$2A			
$2B			
$2C			
$2D			
$2E			
$2F			

LOGIC CARD NAME SYSTEM TIMING CARD (STC)

LOGIC CARD LOCATION SYSTEM CONTROL BASKET-3 (SCB-3)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$3Ø			
$31			
$32			
$33			
$34			
$35			
$36			
$37			
$38			
$39			
$3A			
$3B			
$3C			
$3D			
$3E			
$3F			

LOGIC CARD NAME PIA CONDITIONING CARD

LOGIC CARD LOCATION SCB-4

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$4Ø	OUTPUT	INTEN	Enable register for 8 level interrupt
$41	INPUT	ENNOS	Register that contains which interrupts are enabled/disabled
$42	INPUT	ADRLST	Last number on address bus
$43			
$44			
$45			
$46			
$47			
$48			
$49			
$4A			
$4B			
$4C			
$4D			
$4E			
$4F			

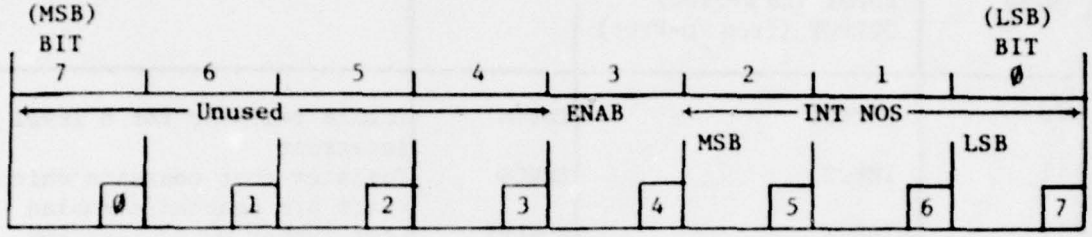
CARD NAME - - - - - - - - PCCX

CARD LOCATION - - - - - - SCB-4

REGISTER ADDRESS - - - $4Ø

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - INTEN



ENAB: (Bit 3) 1 = Enable interrupt represented by 3 bit high truth code in bits 2 - Ø.

Ø = Disable that interrupt.

INT NOS: (bits 2-Ø) High truth code for accessing interrupt level represented by that code for purposes of enabling or disabling.

This register is used to enable or disable the 8 priority interrupts on the PCCX card. Refer to Section XI for a more detailed description.

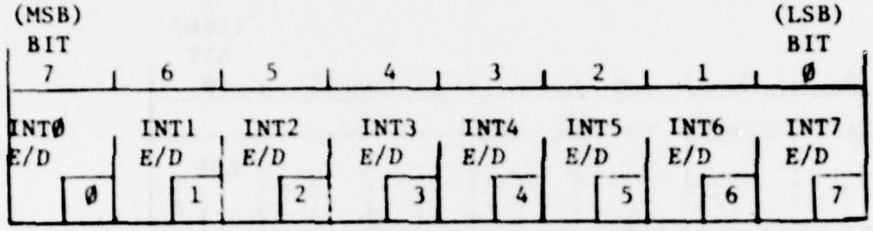
CARD NAME - - - - - - - - PCCX

CARD LOCATION - - - - - - SCB-4

REGISTER ADDRESS - - - $41

REGISTER TYPE - - - - - - INPUT

REGISTER LABEL - - - - - ENNOS



INTØ E/D (Bit 7): 1 = interrupt level Ø is enabled.
Ø = interrupt level Ø is disabled.

This register is an input register that the software can use to check the state of the interrupts being enabled or disabled. Caution - note that the status of the interrupt enable for interrupt number 7 is found in bit Ø of the word.

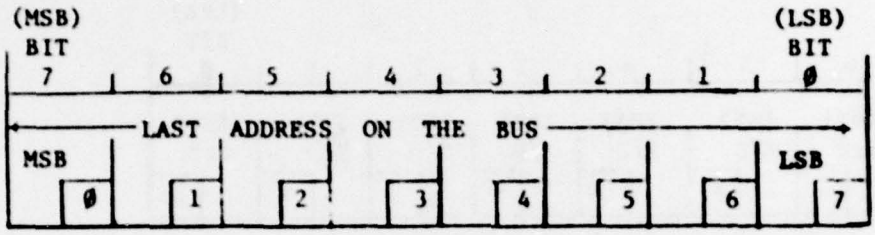
CARD NAME - - - - - - - - PCCX

CARD LOCATION - - - - - - SCB-4

REGISTER ADDRESS - - - $42

REGISTER TYPE - - - - - - INPUT

REGISTER LABEL - - - - - ADRLST



This register contains the last 8 bit address that was on the register address bus. It is used in the interrupt handler. For more detail refer to Section XI.

LOGIC CARD NAME BOX SAMPLE CARD 1 (BSC-1)

LOGIC CARD LOCATION SYSTEM CONTROL BASKET-5 (SCB-5)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$50	OUTPUT	XSZOT1	8 MSBs of Box 1 width
$51	OUTPUT	YSZOT1	8 MSBs of Box 1 height
$52	OUTPUT	XPSOT1	8 MSBs of Box 1 X Position (upper left hand corner)
$53	OUTPUT	YPSOT1	8 MSBs of Box 1 Y position (upper left hand corner)
$54	OUTPUT	XYLSB1	LSBs for Box 1 height, width, X position, Y position and X,Y center position of cursor
$55	OUTPUT	XCUR1	8 MSBs of cursor 1's center in X direction
$56	OUTPUT	YCUR1	8 MSBs of cursor 1's center in Y direction
$57	OUTPUT	BXCTL1	Control register for turning box draw logic on or cursor draw logic on.
$58			
$59			
$5A			
$5B			
$5C			
$5D			
$5E			
$5F			

LOGIC CARD NAME BOX SAMPLE CARD 2 (BSC-2)

LOGIC CARD LOCATION SYSTEM CONTROL BASKET-6 (SCB-6)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$6Ø	OUTPUT	XSZOT2	8 MSBs of Box 2 width
$61	OUTPUT	YSZOT2	8 MSBs of Box 2 height
$62	OUTPUT	XPSOT2	8 MSBs of Box 2 X Position (upper left hand corner)
$63	OUTPUT	YPSOT2	8 MSBs of Box 2 Y Position (upper left hand corner)
$64	OUTPUT	XYLSB2	LSBs for Box 2 height, width, X position, Y position and X, Y center position of cursor
$65	OUTPUT	XCUR2	8 MSBs of cursor 2's center in X direction
$66	OUTPUT	YCUR2	8 MSBs of cursor 2's center in Y direction
$67	OUTPUT	BXCTL2	Control Register for turning box draw logic on or cursor draw logic on
$68			
$69			
$6A			
$6B			
$6C			
$6D			
$6E			
$6F			

LOGIC CARD NAME BOX SAMPLE CARD 3 (BSC3)

LOGIC CARD LOCATION SYSTEM CONTROL BASKET 3 (SCB-3)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$70	OUTPUT	XSZOT3	8 MSBs of Box 3 width
$71	OUTPUT	YSZOT3	8 MSBs of Box height
$72	OUTPUT	XPSOT3	8 MSBs of Box 3 X position (upper left hand corner)
$73	OUTPUT	YPSOT3	8 MSBs of Box 3 Y position (upper left hand corner)
$74	OUTPUT	XYLSB3	LSBs for Box 3 height, width, X position, Y position and X, Y center position of cursor
$75	OUTPUT	XCUR3	8 MSBs of cursor 2's center in X direction
$76	OUTPUT	YCUR3	8 MSBs of cursor 3's center in Y direction
$77	OUTPUT	BXCTL3	Control Register for turning box draw logic on or cursor draw logic on
$78			
$79			
$7A			
$7B			
$7C			
$7D			
$7E			
$7F			

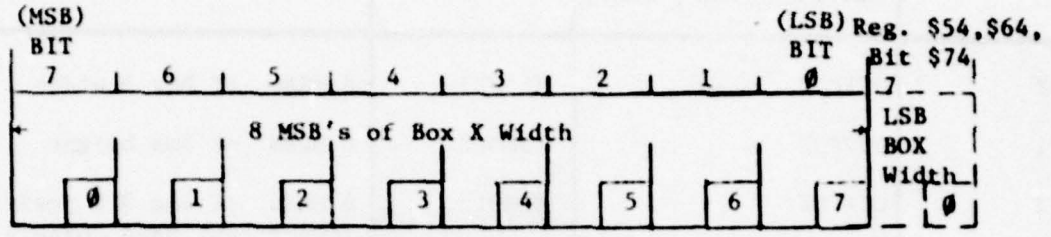
CARD NAME - - - - - - - - BSC1, BSC2, BSC3

CARD LOCATION - - - - - - SCB5, SCB6, SCB7

REGISTER ADDRESS - - - $5Ø, $6Ø, $7Ø

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - XSZOT1, XSZOT2, XSZOT3



This register is the 8 MSB's of the 9 bit word used to define the BOX Sample's size in the X direction (i.e., box width). The LSB is found in register $54, $64 or $74, bit 7 depending on which BSC the software is accessing (BSC1, BSC2, BSC3).

Figure 24 shows the definition of terms for the sample box shown. The term for this register is given by $XSIZE_{BSC}$. The equation for this number being sent to the BSC and the requested number from the CPI card is:

$$XSIZE_{BSC} = XPOS_{CPI} + XSIZE_{CPI} - 1$$

The actual number in Figure 24 is

$$\$32 = 0|0011|0010| = 5Ø_{1Ø}$$

Refer to Section V for more details.

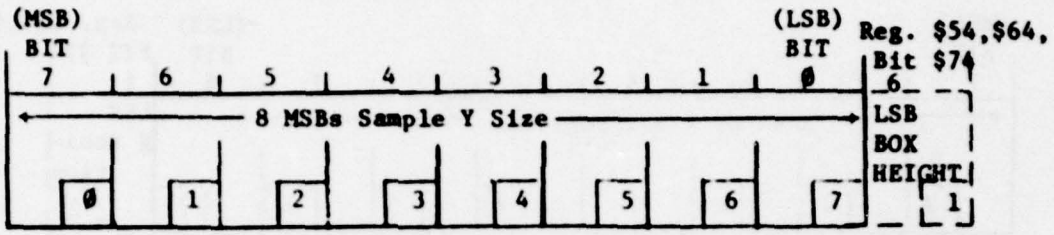
CARD NAME - - - - - - - - BSC1, BSC2, BSC3

CARD LOCATION - - - - - - SCB-5, SCB-6, SCB-7

REGISTER ADDRESS - - - $51, $61, $71

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - YSZOT1, YSZOT2, YSZOT3



This register is the 8 MSBs of the high truth 9 bit word used to define the BOX Sample's size in the Y direction (i.e., box height). The LSB is found in register $54, $64, or $74, bit 6 depending on which BSC the software is accessing (BSC1, BSC2, BSC3).

Figure 24 shows the definition of terms for the sample box shown. The term for this register is given by $YSIZE_{BSC}$. The equation for this number being sent to the BSC and the requested number from the CPI card is:

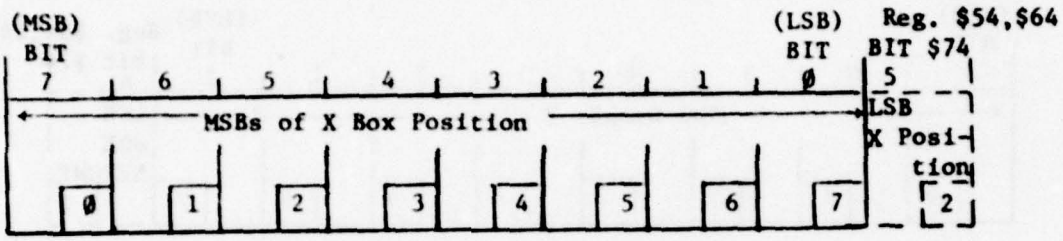
$$YSIZE_{BSC} = YPOS_{CPI} + YSIZE_{CPI} - 1$$

The actual number used in Figure 24 is:

$$\$5D = 0|0101|1101| = 93_{10}$$

Refer to Section V for more details.

CARD NAME - - - - - - - - BSC1, BSC2, BSC3
CARD LOCATION - - - - - - SCB5, SCB6, SCB7
REGISTER ADDRESS - - - $52, $62, $72
REGISTER TYPE - - - - - - OUTPUT
REGISTER LABEL - - - - - XPSOT1, XPSOT2, XPSOT3



This register is the 8 MSBs of the high truth 9 bit word used to define the BOX Sample's starting X position. The LSB is found in register $54, $64, or $74, bit 5 depending on which BSC the software is accessing (BSC1, BSC2, BSC3).

Figure 24 shows the definition of terms for the sample box shown. The terms for this register are given by $XPOS_{BSC}$. The equation for this number being sent to the BSC and the requested number from the CPI card is:

$$XPOS_{BSC} = XPOS_{CPI} - 1$$

For the example in Figure 24 this number is:

$$\$15 = 0|0001|0101| = 21_{10}$$

Refer to Section V for more details.

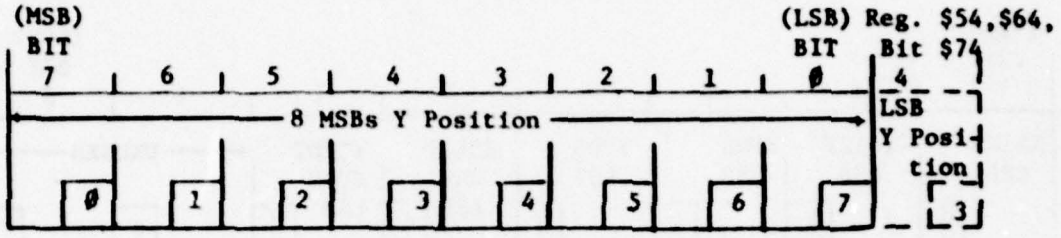
CARD NAME - - - - - - - - BSC1, BSC2, BSC3

CARD LOCATION - - - - - - SCB-5, SCB-6, SCB-7

REGISTER ADDRESS - - - $53, $63, $73

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - YPSOT1, YPSOT2, YPSOT3



This register is the 8 MSBs of the 9 bit word used to define the sample box's starting Y position. The LSB is found in register $54, $64, or $74, bit 4 depending on which BSC the software is accessing (BSC1, BSC2, BSC3).

Figure 24 shows the definition of terms for the sample box shown. The term for this register is given by $YPOS_{BSC}$. The equation for this number being sent to the BSC and the requested number from the CPI card is:

$$YPOS_{BSC} = YPOS_{CPI}$$

From Figure 24 this number is:

$$\$38 = 0|0011|1000| = 56_{10}.$$

Refer to Section V for more details.

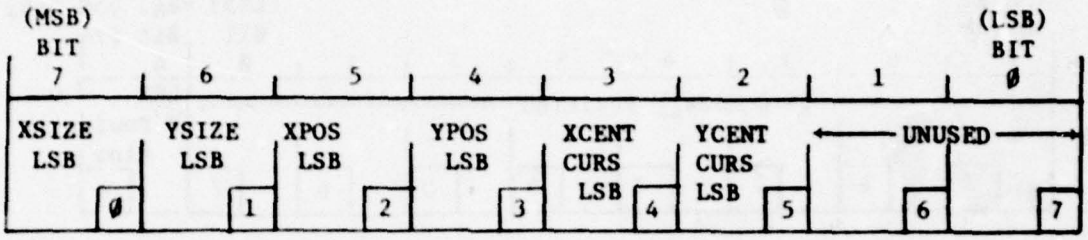
CARD NAME - - - - - - - - BSC1, BSC2, BSC3

CARD LOCATION - - - - - - SCB5, SCB6, SCB7

REGISTER ADDRESS - - - $54, $64, $74

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - XYLSB1, XYLSB2, XYLSB3



This register contains the LSB for each of the 9 bit words that describe the box's,

Width - X size

Height - Y size

Starting X position

Starting Y position

and when a cursor is drawn, the LSB for the cursors

X center

Y center

This is the same format for all 3 BSCs.

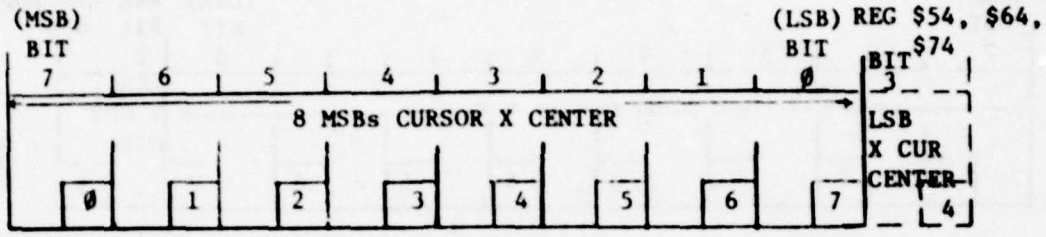
CARD NAME - - - - - - - - BSC2, BSC3

CARD LOCATION - - - - - - SCB5, SCB6, SCB7

REGISTER ADDRESS - - - SS55, $65, $75

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - XCUR1, XCUR2, XCUR3



This register has the same format for all 3 BSCs. It defines the X center of the drawn cursor and represents the 8 MSBs of the high truth 9 bit binary number. From Figure 24 this number is:

$$\$13 = 0|0001|1011| = 27_{10}$$

This means that the X center of the cursor will be at element 27 and that a vertical line the full Y height of the sample box will be drawn at this element.

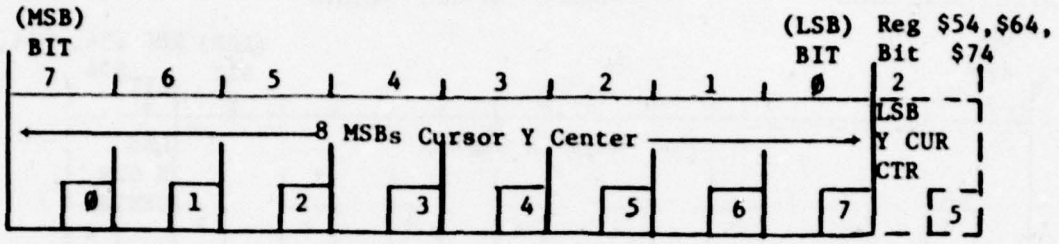
CARD NAME - - - - - - - - BSC1, BSC2, BSC3

CARD LOCATION - - - - - - SCB5, SCB6, SCB7

REGISTER ADDRESS - - - $56, $66, $76

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - YCUR1, YCUR2, YCUR3



This register has the same format for all 3 BSCs. It is the 8 MSBs of the high truth 9 bit binary word that describes the center of the cursor in the Y direction. A line will be drawn across the screen the width of the sample box. From Figure 24 this number is:

$$\$3E = 0|0011|1110| = 62_{10}$$

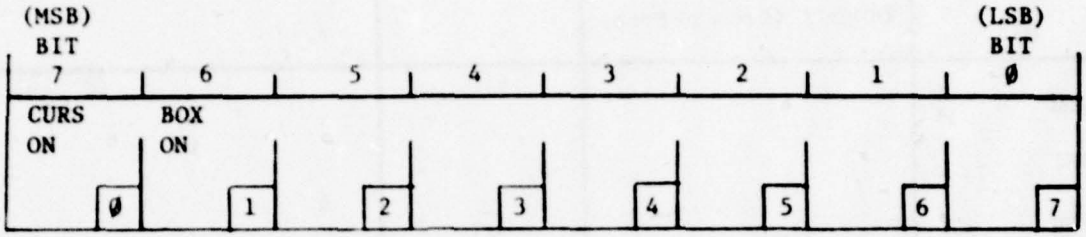
CARD NAME - - - - - - - - BSC1, BSC2, BSC3

CARD LOCATION - - - - - - SCB5, SCB6, SCB7

REGISTER ADDRESS - - - $57, $67, $77

REGISTER TYPE - - - - - OUTPUT

REGISTER LABEL - - - - BXCTL1, BXCTL2, BXCTL3



CURS ON: (Bit 7)
1 = Draw Cursor on Video
Ø = Blank Cursor Draw

BOX ON: (Bit 6)
1 = Draw Sample Box on Video
Ø = Blank Box Draw

This register controls the drawing of both the box sample box and the cursor. The sample signal (BOXWINDOW) still goes to the A/D Memory Port but the video blank and unblank control is here. Both the sample box and cursor can be drawn at the same time. If only the cursor is drawn the box size still has to be defined as it determines how large the cursor is.

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION SYSTEM CONTROL BASKET 8 (SCB8)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$8Ø			
$81			
$82			
$83			
$84			
$85			
$86			
$87			
$88			
$89			
$8A			
$8B			
$8C			
$8D			
$8E			
$8F			

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION SYSTEM CONTROL BASKET 9 (SCB-9)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$9Ø			
$91			
$92			
$93			
$94			
$95			
$96			
$97			
$98			
$99			
$9A			
$9B			
$9C			
$9D			
$9E			
$9F			

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION System Control Basket 1Ø (SCB-1Ø)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$AØ			
$A1			
$A2			
$A3			
$A4			
$A5			
$A6			
$A7			
$A8			
$A9			
$AA			
$AB			
$AC			
$AD			
$AE			
$AF			

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION System Control Basket 11 (SCB-11)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$BØ			
$B1			
$B2			
$B3			
$B4			
$B5			
$B6			
$B7			
$B8			
$B9			
$BA			
$BB			
$BC			
$BD			
$BE			
$BF			

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION System Control Basket 12 (SCB-12)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$CØ			
$C1			
$C2			
$C3			
$C4			
$C5			
$C6			
$C7			
$C8			
$C9			
$CA			
$CB			
$CC			
$CD			
$CE			
$CF			

LOGIC CARD NAME MEMORY CONTROL DRIVER CARD (MCD)

LOGIC CARD LOCATION System Control Basket 13 (SCB-13)

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$DØ	OUTPUT	FMCTRL	Controls the state of all 4 Fast Memory Cards
$D1	OUTPUT	ADPCTL1	Controls the state of the A/D Memory Port
$D2	OUTPUT	MAPCTL	Controls the state of the Microprocessor$_A$ Memory Port
$D3	OUTPUT	ADPCTL2	SUM/UNSUM MODE
$D4			
$D5			
$D6			
$D7			
$D8			
$D9			
$DA			
$DB			
$DC			
$DD			
$DE			
$DF			

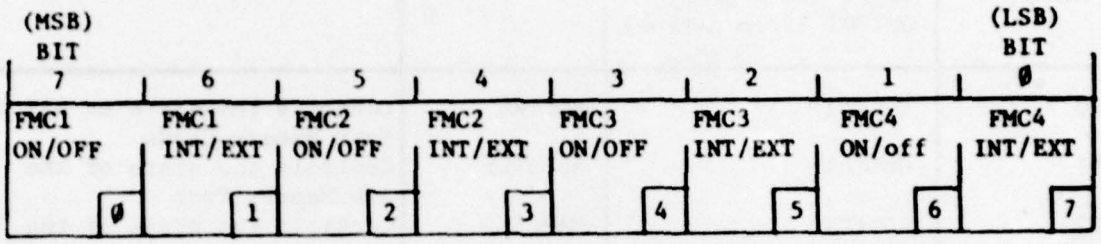
CARD NAME - - - - - - - - MCD

CARD LOCATION - - - - - - SCB-13

REGISTER ADDRESS - - - $DØ

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - FMCTRL



This register controls the access to the Fast Memory Cards (FMC). There are 2 bits for each FMC with the following pattern defining the control.

ON/OFF	EXT/INT	DESCRIPTION
0	X	No one can access the memory and it is disconnected from the common bus.
0	X	
1	0	Memory connected to common bus.
1	1	Memory connected to External Control

Notice that when a FMC is disabled, the External Port has no control, i.e., Memory is in the high impedance mode. Refer to Section IX for more detail.

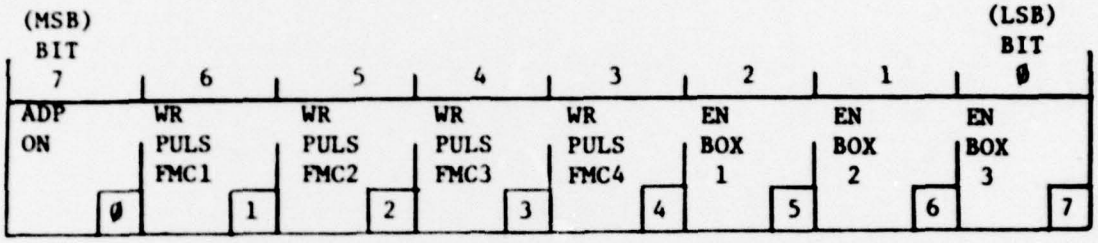
CARD NAME - - - - - - - - MCD

CARD LOCATION - - - - - - SCB-13

REGISTER ADDRESS - - - $D1

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - ADPCTL1



APD ON: (Bit 7) 1 = The A/D Memory Port has control of the common bus and can thus write to memory.

Ø = The A/D Port is disabled.

WR PULS FMC1(2,3,4): (Bits 5-3) 1 = Write pulses during the BOXWINDOW time will be sent to the appropriate (one or all) FMCs.

Ø = No writing is done to memory.

EN BOX1(2,3): (Bits 2-Ø) 1 = The appropriate sample box signal, BOXWINDOW, will be enabled.

Ø = Disable BOXWINDOW1(2,3).

Bit 7 closes the bus switch to give the A/D Port card the common memory bus system. Bits 6-3 define which Fast Memory Cards get the data from the A/D at the BOXWINDOW time. Note that the FMC control word (Register $DØ) must be in the proper mode for the A/D port to write to FMC1 along with bit 6 aboye being a 1.

Bits 2-Ø define where the sampling signal will come from BSC1, BSC2, or BSC3. All 3 BOXWINDOWS can be enabled at once and the selected memory will receive the sampled digitized video. Refer to Section VIII for more detail.

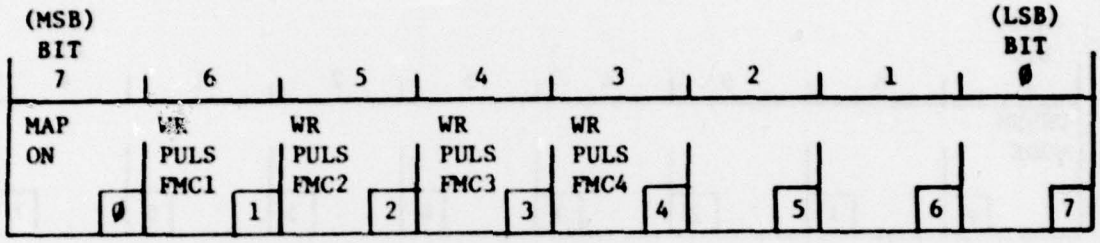
CARD NAME - - - - - - - - MCD

CARD LOCATION - - - - - - SCB-13

REGISTER ADDRESS - - - $D2

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - MAPCTL



MAP ON: (Bit 7) 1 = Microprocessor$_A$'s internal address/data bus is connected to the common address and data bus.

Ø = Microprocessor$_A$ is disconnected.

WR PULS FMC1(2,3,4): (Bits 6-3) 1 = A write pulse will be sent to the appropriate FMC (during write cycle), and the memory may be read by Microprocessor$_A$.

Ø = No write pulse is sent to the FMC card.

For more detail refer to Section VII.

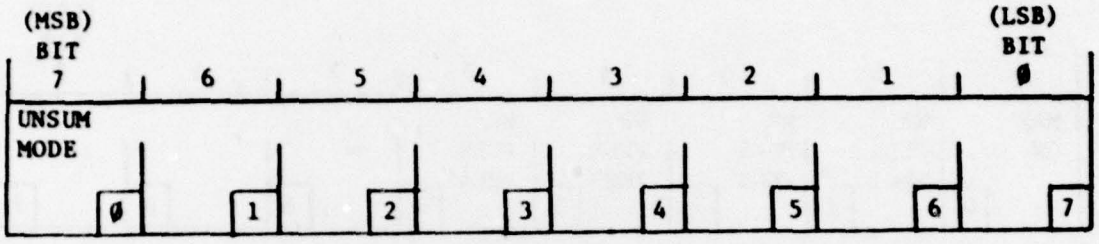
CARD NAME - - - - - - - - MCD

CARD LOCATION - - - - - - SCB-13

REGISTER ADDRESS - - - $D3

REGISTER TYPE - - - - - - OUTPUT

REGISTER LABEL - - - - - ADPCTL2



UNSUM MODE: (Bit 7)
1 = Send digitized video data to FMC without summing the odd and even fields.

Ø = Sum the digitized video odd and even field before the data are sent to the FMC.

For more detail refer to Section VIII.

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION UNUSED AT PRESENT

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$EØ			
$E1			
$E2			
$E3			
$E4			
$E5			
$E6			
$E7			
$E8			
$E9			
$EA			
$EB			
$EC			
$ED			
$EE			
$EF			

LOGIC CARD NAME UNUSED AT PRESENT

LOGIC CARD LOCATION UNUSED AT PRESENT

Register Address (HEX)	Register Type INPUT (to μ-Proc) OUTPUT (from μ-Proc)	Software Label	Functional Description
$FØ			
$F1			
$F2			
$F3			
$F4			
$F5			
$F6			
$F7			
$F8			
$F9			
$FA			
$FB			
$FC			
$FD			
$FE			
$FF			

APPENDIX A

INITIALIZING THE PERIPHERAL INTERFACE ADAPTER'S (PIA'S)

In the equate section of the program the PIA channels are defined as mnemonics with the appropriate memory location.

```
ADRBUS      EQU     $7FFE     HEX location of PIA that
                              defines the address bus.

INDATA      EQU     $7FF8     HEX location of PIA that
                              is used for input type
                              registers.

OUDATA      EQU     $7FFA     HEX location of PIA that
                              is used for sending data
                              to all output type
                              registers.

INTREG      EQU     $7FFC     HEX location of PIA used
                              to read interrupt number
                              from the PVDS.

PIAC1A      EQU     $7FF9     Control Register for
                              INDATA PIA

PIAC1B      EQU     $7FFB     Control Register for
                              OUDATA PIA

PIAC2A      EQU     $7FFF     Control Register for
                              INTREG PIA
```

The PIA's data direction registers are then set to define the peripheral registers as being either input or output. ADRBUS and OUDATA are output PIAs and INDATA and INTREG are input PIAs. To access the data direction register bit 2 of the control register should be Ø.

```
CLR         PIAC1A          Clear bit Ø2 of all
                            control registers

CLR         PIAC2A
```

```
CLR     PIAC1B
CLR     PIAC2B
LDAA    $FF         All 1s in the data direction
                    register define all 8 lines
                    as output.
STAA    OUDATA
STAA    ADRBUS
CLR     INDATA      All zeroes in these data
                    direction registers defines
                    them as input PIAs.
CLR     INTREG
```

The PIAs are used in the so called pulse mode. For this mode the control register should have

bit 5 = 1

bit 4 = ø

bit 3 = 1

The software will now always talk to the peripheral register

bit 2 = 1

For the INTREG PIA the interrupts will occur on the low to high transition of control line CB1 if:

bit 1 = 1

For the interrupts to be enabled:

bit ∅ = 1

Bits 6 and 7 are input bits only. The control registers should now have a $2E (to have the interrupts disabled) and a $2F (for the interrupts to be enabled). Only the Control Register for the INTREG PIA will ever have the interrupt enabled.

```
LDAA    #$2E
STAA    PIAC1A
STAA    PIAC2A      Set the control registers
STAA    PIAC2B
LDAA    #2F
STAA    PIAC2A
```

The interrupts can be enabled per the description in Section IX.

APPENDIX B

EXAMPLE PROGRAM TO ACCESS REGISTERS

A software label for each register address is used for ease in trouble shooting. Given below is an example program for reading an Input Type Register and for writing to an Output Type Register.

```
                          NAM       EXAMP
                TOGSW1    EQU       $1Ø       Toggle Sw's on tel Pan
                TOGLD1    EQU       $11       Leds above toggle switches
                ADRBUS    EQU       $7FFE     Address bus PIA
                OUDATA    EQU       $7FFA     Output data PIA
                INDATA    EQU       $7FFB     Input data PIA
ØlØØ                      ORG       $1ØØ      Starting location of program
ØlØl    86 1Ø             LDAA      #TOGSW1
ØlØ3    B7 7FFE           STAA      ADRBUS    Select register $1Ø
ØlØ6    B6 7FF8           LDAA      INDATA    Accum A contains the state
                                                of the 7 toggle sw's
ØlØ9    B7 7FFA           STAA      OUTDATA   Light leds above toggle sw's

                          END
```

The Input Type Register is the state of 8 toggle switches on the control panel.

Register Label:TOGSW1
Register Type:Input
Register Address:$1Ø

Above each toggle switch is a LED indicating the state of the toggle switch. The LEDs are an Output Type Register.

Register Label:TOGLD1
Register Type:Output
Register Address:$11

The example program sets the address bus to a $1Ø. It then reads the data from the toggle switches. This automatically

increments the address bus. The data are then sent back to the LEDs, lighting the ones whose associated switch has been set.

All the hardware logic prints designate the register MSB as bit Ø and the LSB as bit 7. Motorola's convention is just the opposite. MSB equals bit 7 and LSB equals bit Ø. Motorola's convention is used in this report to describe the registers.

ACKNOWLEDGEMENT

The author is extremely greatful to Marie Grey for her patience and expertise in typing this report.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-79-156	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Programmable Video Digitizing System (PVDS) – Functional Description and Preliminary Software Considerations		5. TYPE OF REPORT & PERIOD COVERED Project Report 6. PERFORMING ORG. REPORT NUMBER Project Report ETS-46
7. AUTHOR(s) Lawrie E. Eaton		8. CONTRACT OR GRANT NUMBER(s) F19628-78-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element No. 63428F Project No. 2128
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Systems Command, USAF Andrews AFB Washington, DC 20331		12. REPORT DATE 5 June 1979 13. NUMBER OF PAGES 142
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Electronic Systems Division Hanscom AFB Bedford, MA 01731		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

digitized video
microprocessor
signal/noise ratios
object rate/direction

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A Programmable Video Digitizing System (PVDS) has been developed that will digitize selected portions of the return TV analog video from the Ebison camera at the GEODSS ETS. The digitized video can then be processed by either a microprocessor or a Modcomp Computer. From the data, the software may then determine object centroiding, night sky background calculations, signal/noise ratios and object rate/direction calculations for closed loop tracking. The system may also be used to process digitized data from future sensors, i.e., infrared or CCD.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)