INTERFACING A TV PICTURE DIGITIZER TO THE CHROMATICS COLOR-GRAPHICS COMPUTER

by

Jerome Anthony Hill

B. S., Oklahoma State University, 1966

A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

Approved by:

William J. Hankley
Major Professor

Acknowledgements:

I wish to thank Earl Harris for his help in carrying out the work of Chapter 2.

I also thank Dr. William Hankley for his continued support, encouragement, and helpful suggestions throughout this project.

Table of Contents

Figures

CHAPTER 1

Introduction

A common TV camera, such as used for closed-circuit surveillance, may be connected to a computer so that the brightness or intensity of any small spot on the TV screen may be returned as a numerical value. For example, the value 20 might mean the spot was nearly black while 214 means it was nearly white. It is customary to superimpose a grid upon the picture, forming tiny rectangles between the grid lines, which are numbered by a coordinate system. The average brightness over the rectangle located by an X-Y coordinate pair is returned as a numerical value in some specified range. This number is commonly referred to by the name "gray level", and each tiny rectangle is a "pixel", a contraction of picture element. The gray level of a pixel is then the "pixel value", and this conversion is done by a piece of electronic equipment known as a "digitizer".

The CVI Model 270A Digitizer used in this paper divides the screen into 480 pixels vertically, the Y direction, and 512 pixels horizontally, the X direction, returning the pixel values in the range of zero for pure black to 255 for pure white. The X and Y coordinates of the pixel to be read at some instant are transmitted to the digitizer via a 16-bit buss, and the pixel values are returned through an 8-bit buss.

Throughout this paper it is assumed there is some object whose image is to be digitized which will simply be called the "picture."

This paper discusses the necessary electronic modifications to a standard 16 bit parallel interface printed circuit board obtained from the Chromatics Company, to properly connect it with the CVI Digitizer.

Further, the digitizer requires that certain command bits be sent along with the coordinates, and that certain timing relationships be respected. This task is performed by a set of three subroutines written in the BASIC computer language. These are commonly known as driver or digitizer I/O routines.

Initial tests revealed that two problem areas existed: The operating speed of these routines was too slow, and the picture needed additional illumination. Possible solutions to these problems are presented in Chapters 5 and 6.

The paper concludes with a BASIC computer program to re-create an image, on the Chromatics display screen, of the digitized data when the pixel values are classed into evenly spaced intervals and all pixels falling in the same interval are displayed as some arbitrarily chosen color. This is commonly known as a "false-color image " since the data is from a black and white image. The color chosen does not necessarily need to have any relation to the colors in the original picture.

CHAPTER 2

Interfacing

A. DIGITIZER REQUIREMENTS AND OPERATION

All inputs and outputs of the digitizer are TTL level binary logic signals. A 16-bit command buss, a strobe line, and data common convey data from the computer to the digitizer. An 8-bit pixel-value buss and a status line convey data back to the computer.

To operate the digitizer, a command containing the X or Y co-ordinate of the pixel to be read and an operation code, as in Figure 1, is sent over the 16-bit command buss. When the data is stable on the buss, a short one-to-two microsecond pulse is sent on the strobe line. Normally the status line is high; however the status goes low when the strobe is received and stays low until the operation is completed, at which time it goes back high. If the operation is a read-pixel value, then the 8-bit pixel value, called the "Z value", is available on the pixel-value buss when the status goes high, and it stays there until the next read-pixel value command is executed by the digitizer.

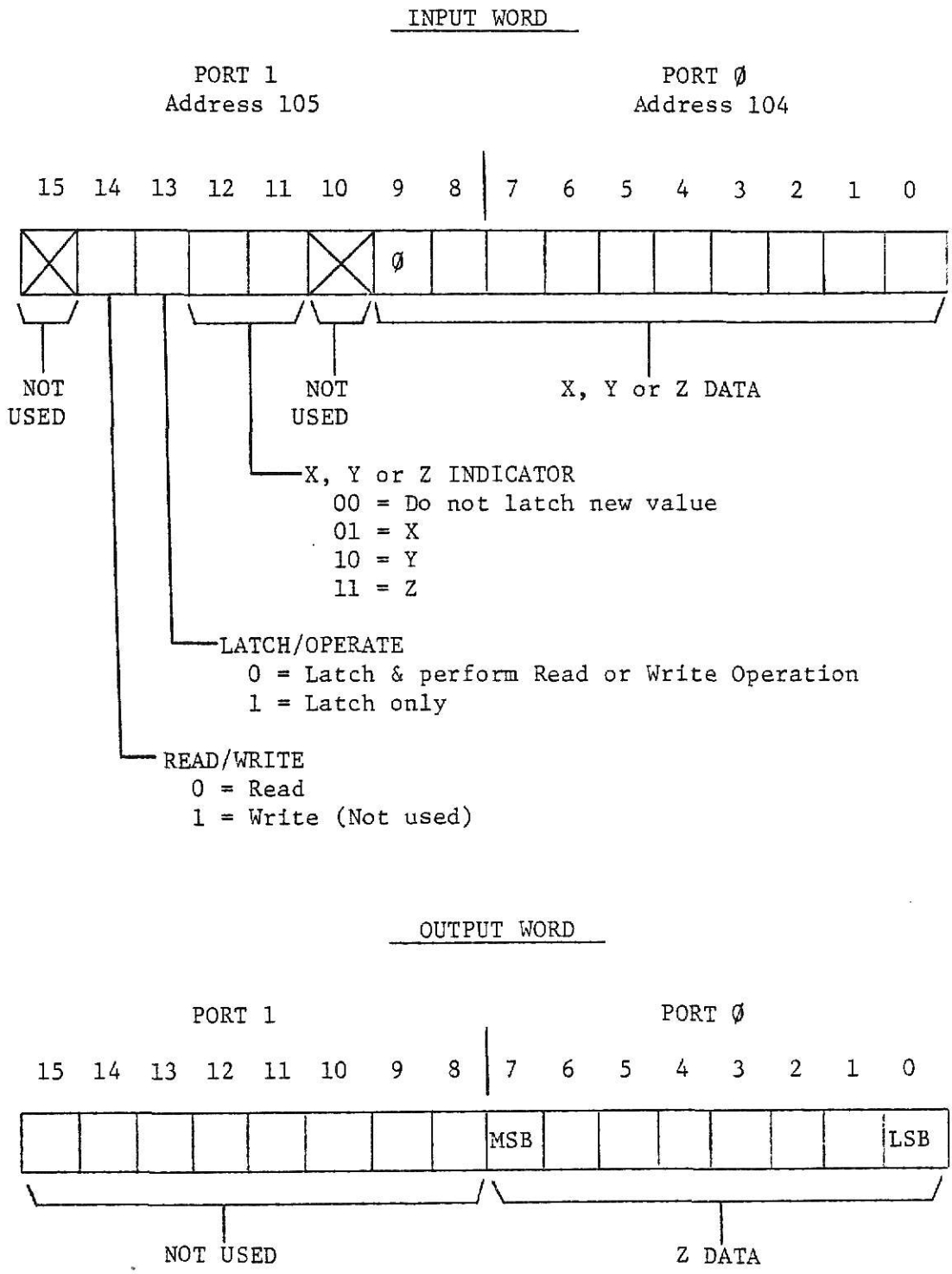
B. ORIGINAL INTERFACE BOARD CHARACTERISTICS

A standard Chromatics 16-bit parallel interface board was chosen as being most suitable to the situation. It has a dual 8-bit bi-directional data buss which can be programmed, through an escape key sequence, to

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

Figure 1. Command Word Format



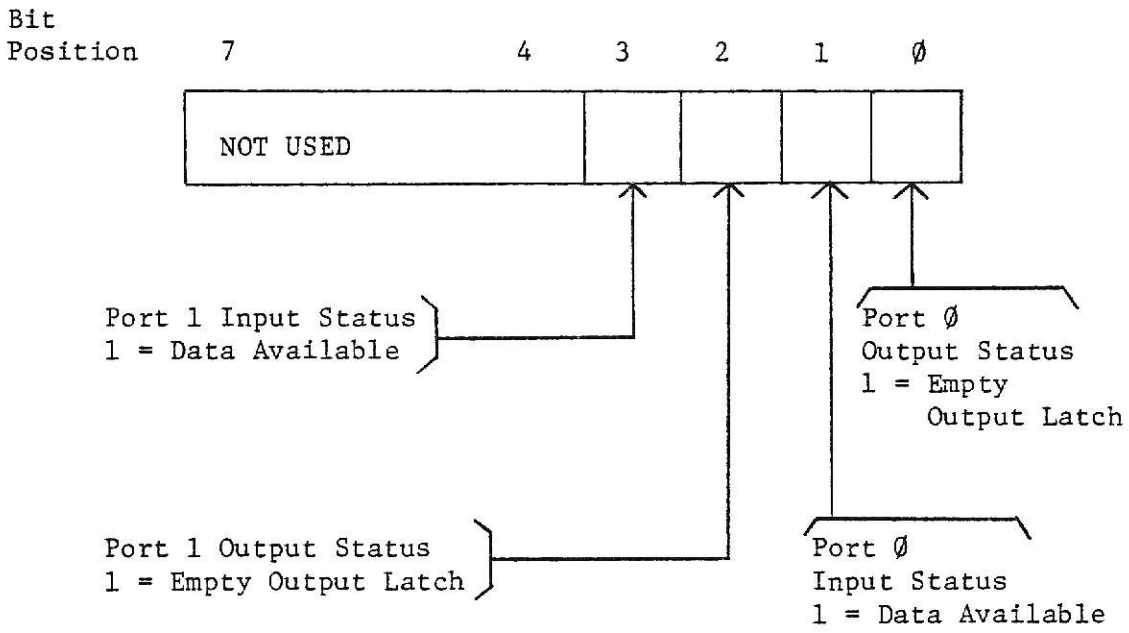
operate together as a 16-bit bi-directional buss. In addition it possesses full standard hand-shaking lines. NOT-output-data-available and NOT-output-data-enable are used for output, NOT-input-data-available and NOT-input-data-enable are used for input. In dual 8-bit mode, handshaking for both sides is provided. Three I/O port addresses are assigned: The status byte address is at 106 decimal, of which only the right 4 bits are used in dual 8-bit mode; in 16-bit mode only one set, the port Ø side, is used. See Figure 2 for bit meanings. In the dual 8-bit mode, address 104 is one 8-bit channel, referred to as port Ø, and address 105 is the other, referred to as port 1. In the 16-bit mode, two bytes are set to address 104, most significant byte first, and an on-board toggle flip-flop gates them alternately into the left and right parts of the 16-bit output when the output is wired so port 1 is the MSB and port Ø is the LSB.

The expected normal operation of this interface is as follows:

To output data from the computer to the external device: The CPU sends the value to the interface at which time two things occur: First, the NOT-output-data-available line goes low to signal the external device and to cause the output status bit in the status byte to go to zero. Next, the external device must respond with a short negative-going pulse on the NOT-output-data-enable-line when it is ready for the data. This causes three things to occur: 1. The data is made available on the bi-directional buss; 2. the NOT-output-data-available goes back high, and 3. the output status bit in the status byte goes to 1.

To input data from the external device to the computer: The external device lowers the NOT-input-data-available line, which sets the input status bit of the status byte to a 1, signaling to the computer that data is there to be

Figure 2. Status Byte



read. It is the responsibility of the computer to poll this status bit whenever input is expected and to respond by executing an IN instruction, at which time the NOT-input-data-enable line receives a negative-going pulse. On the leading edge of this pulse, the data is transferred from the bi-directional buss to the computer's internal data buss. On the trailing edge, the external device may remove its data from the buss and bring the NOT-input-data-available line back high. The latter resets the input status bit in the status byte back to zero.

C. INTERFACE BOARD MODIFICATIONS

The first major modification was prompted by the digitizer having separate command and output-pixel-value busses. It was necessary to modify the bi-directional buss structure into two uni-directional busses.

Investigation of the schematic diagram for the parallel interface (see Figure 3) disclosed that the buss was from two separate integrated circuits connected together; a tri-state data latch captured the output byte from the computer data buss and held it until the external device activated the NOT-output-data-enable line, which switched the tri-state output from open to normal and thus put the data on the buss. The other IC was a simple data gate which gated the data from the buss to the internal computer data buss when the computer pulsed the NOT-input-data-enable line.

As Chromatics socketed all their IC's, this portion of the modification was simplified. The input-gate IC on each port was pulled out, its data input pins bent out so they would clear the socket, small wires were soldered to the pins, and each IC was plugged back in. This effectively broke the

connection shown by X's in Figure 3.

It was also determined that the tri-state operation of the output latch IC's was unnecessary and, in fact, if left would leave the output buss floating much of the time. In this state interference could be picked up which could cause false operation; therefore, the pin that controls the tri-state was bent out and grounded, thus making the output always have some data present or whatever was the last output.

If it is ever deemed desirable to do so, these boards can be restored to their original specification by replacing these IC's at a cost of approximately five dollars.

The above modifications were checked out and found to work correctly; therefore the now separate busses were each wired to a RS-232C type twenty-five pin connector (a very commonly available type), the output side to the connector pin numbers shown on the schematic, and the new input buss to the corresponding pins on the other connector. The cable from the digitizer to the latter connector has the left eight bits connected to zero and the right eight bits to the pixel-value buss of the digitizer.

The strobe and status lines were handled next. The status line seemed easier to modify as only the Ø port side needed modification. It was wired to the NOT-output-data-enable line and the NOT of the status was wired to the NOT-input-data line. This NOT was obtained from the output of the IC in the interface board that was already inverting the NOT-output-data-enable. This has the effect of making both the status-byte bits (port Ø output status and port Ø input status) follow the status line. I.E., when the digitizer status line is high, the status byte bits are logic 1, and when the

Figure 3. Parallel Interface Electronic Schematic

The author apologizes, but as of publication time a legal release could not be obtained to publish this proprietary schematic. If required, a copy may be obtained from the Chromatics Company on an individual basis. See reference sheet (page 35) for this address.

digitizer status line is low, they are zero. Thus the software can issue a command to the digitizer and loop until either status-byte bit goes to logic 1.

As originally anticipated, the strobe pulse proved to be the most difficult to modify; none of the handshake lines provide the proper short pulse required. Additional scrutiny of the schematic revealed that the pulse supplied to latch the data into the output data latch, IC, had the proper shape. The timing, however, was such that it occurred simultaneously with new data appearing on the output data buss, and the digitizer specifications state that new command data must be stable before the strobe is issued.

Analysis of the digitizer schematic showed that the strobe pulse travels through two gates before actually being used for capturing the command data. Perhaps this inherent two-gate time delay and the short connecting cable would allow the data to have sufficient time to settle. This procedure has been followed and all tests made to date indicate it is successful. If it becomes necessary to lengthen the connecting cable, this fix may not work. In this event, a delay could be inserted in this connection. Some suggestions for such a delay include a hex inverter with all sections in cascade giving six additional gate times, or a dual monostable wired in the pulse delay configuration. It is not advisable to induce too much delay in the digitizer status line though as the status-byte bits do not go to zero until the strobe is received at the digitizer. If the software starts checking the status-byte bits for 1 (which should signal completion) before it has gone to zero, a false "done" signal can be given.

D. PROGRAMMING GENERAL FORMAT FOR 16-BIT MODE (obsolete)

As now modified, general programming using the modified board would follow this form:

INITIALIZE: TYPE ESC < 0 (this sets interface to 16-bit mode)

STEP 1 : output 16-bit command to port 0, address 104, most significant byte first.

STEP 2 : REPEAT UNTIL (bit 0 of status byte equals one) DO read status byte from port address 106

The above scheme was implemented using the Chromatics BASIC interpreter to send the necessary commands to the digitizer. It was discovered that the toggle flip-flop that steers the data bytes of a 16-bit word into alternate output latches would miss toggling at random. This had the effect of loading the bytes of the command word backwards until another miss occurred. During this period the digitizer would not be receiving a legal command word.

E. PROGRAMMING GENERAL FORMAT FOR DUAL 8-BIT MODE (final form)

Rather than trying to discover why the toggle flip-flop would miss, further analysis showed that dual 8-bit operation of the interface would bypass the problem and would serve just as well as 16-bit operation as long as one outputs to port 1 before port Ø. Therefore the revised general programming is:

INITIALIZE: TYPE ESC < 1 (this sets interface to dual 8-bit mode)

STEP 1 : Output the left bits of the 16-bit command word to port address 105

STEP 2 : Output the right 8 bits of the command word to port address 104

STEP 3 : REPEAT UNTIL (bit 0 of the status byte equals one) DO read status byte from port address 106.

STEP 4 : Only if this was a read-pixel command, input the pixel value from port address 104.

CHAPTER 3

Software Drivers and Demonstration Program for Driver Subroutines

A. SOFTWARE DRIVERS

Looking at the face of the TV monitor which has the image of the picture, the X co-ordinate is horizontal, increasing from left to right; the Y co-ordinate is vertical, increasing as one travels down the screen. Hence the zero-zero point is in the upper left hand corner of the screen and all X and Y values are positive integers. The maximum value of X is 511. The maximum value of Y is 479. Normally X is set to some value and Y is cycled through the values, reading a pixel at each point.

Three primitive routines are needed.

1. Initialize interface to dual 8-bit mode.
2. Send an X value to the digitizer (which it will hold until changed).
3. Send a Y value and read a pixel value.

The first routine needs only to send the sequence ESC < 1 to the operating system. It is:

```
3000   REM INITIALIZE INTERFACE TO DUAL-8
3010   PRINT CHR$(27);"<1"
3020   RETURN
```

The CHR$ is a special BASIC function that in this case outputs the unprintable character ESC(ASCII 27).

The second routine sends an X value to the digitizer. In order to accomplish this task, the digitizer needs the following command word bit settings:

BIT	VALUE	MEANING
14	Ø	Read (Write is for another unused function)
13	1	Latch only
12-11	Ø1	X-value follows
9-0	XXXXXXXXXX	Actual X-value

Notice that the actual X-value overlaps into the left hand byte but Bit 9 will always be zero for this model digitizer as X and Y can never get large enough to use it. The two bits 15 and 10 are not used and were arbitrarily set to zero.

The complete command word to transmit X is 0010100X XXXXXXXX. The left hand byte is 00101000 if X is less than 256, and 00101001 if X is equal to or greater than 256. The right hand byte is the right 8 bits of X. The left byte in decimal is 40 or 41.

It was decided to let the basic variable X% be the X co-ordinate parameter, thus the program to send an X value to the digitizer is:

```
2000   REM SEND X COORD X% TO DIGITIZER
2010   IF X% > 255 THEN OUT 105,41 ELSE OUT 105,40
2020   OUT 104, X% AND 255
2030   WAIT 106,1:REM LOOP TILL BIT Ø OF STATUS BYTE GOES 1
2040   RETURN
```

In this routine three operations unique to Chromatics BASIC have been employed. They are:

1. The "%" after a variable name makes the variable a 16-bit integer.
2. The OUT operation used in lines 2010 and 2020 issues a Z-80 OUT machine language command; the first argument is the address and the second the value to be transmitted.
3. The AND does a logical "and" between X% and 0000000011111111 thus masking out high order bits of X%.

The WAIT command implements a three step sequence:

1. Do a machine language IN instruction to the address given by the first argument which in this case is the interface status byte.
2. AND this with the bit pattern given by the second argument (00000001) thus masking out all but bit Ø.
3. If the result of the AND is 1, go on; else loop back to Step 1.

Thus the WAIT loops until bit Ø of the status byte is a one, which it does when the digitizer status line goes high, signaling operation complete.

The routine to store Y is similar except the pixel value must be converted and must be read in when the operation complete signal is given. The command code bits are:

BIT	VALUE	MEANING
14	Ø	Read
13	Ø	Latch and read pixel
12-11	1Ø	Y-value follows
9-0	XXXXXXXXXX	Actual Y-value

The complete command word to transmit Y is 0001000Y YYYYYYYY. The left hand byte is decimal 16 if Y is less than 256; otherwise it is decimal 17.

In this subroutine the input parameter is Y% (the Y co-ordinate). Output is Z% (the pixel value at point X%, Y%). The program to transmit this information to the digitizer is:

```
2045   REM RETURN PIXEL VALUE VIA Z% AT
2046   REM POINT X%,Y%.  X% PREVIOUSLY TRANSMITTED.
2050   IF Y% > 255 THEN OUT 105,17 ELSE OUT 105,16
2060   OUT 104, Y% AND 255
2070   WAIT 106,1
2080   Z% = INP(104)
2090   RETURN
```

The only new command used in this subroutine is in line 2080, i.e., INP. The INP performs a Z-80 machine language IN instruction to the address given in the argument, which in this case is the address of interface Port Ø where the pixel value data from the digitizer is stored, and converts this 8-bit value into a 16-bit value by appending eight zero bits to the left. This value is then stored into Z%.

B. DEMONSTRATION PROGRAM FOR DRIVER SUBROUTINES

The program in Figure 4 demonstrates the correct usage of the three subroutines presented in the first part of this chapter. All three subroutines must be entered before this demonstration program will run.

This demonstration program produces a replica of the TV monitor screen on the Chromatics screen, with all originally white areas displayed as green points and the black areas as blank screen. The decision is made on the basis of the pixel value; those less than 127 are displayed as green points. To increase the run speed of this program, only every fourth pixel value is read and displayed. This function is controlled by the STEP value of the FOR loops. Execution time depends heavily on the number of points plotted, but it is typically 15 to 25 minutes per picture.

```
 10    GOSUB 3010 : REM INITIALIZE INTERFACE
 20    PRINT CHR$(12);: REM CLEAR SCREEN
 30    PRINT "∼G";    : REM TURN ON PLOT MODE
 40    FOR X% = 10 TO 501 STEP 4
 50       GOSUB 2000 : REM SEND X COORD
 60       FOR Y% = 10 TO 479 STEP 4
 70          GOSUB 2050 : REM SEND Y COORD, READ Z PIXEL VALUE
 80          IF Z% < 127 THEN PLOT X%, 512 - Y%
 90       NEXT Y%
100    NEXT X%
110    MODEOFF : REM TURN OFF PLOT MODE
120    END
```

Figure 4. Demonstration program for Driver Subroutines

CHAPTER 4

Setup and Operation of Digitizer, TV Camera and Monitor and Notes on Use

The apparatus consists of a CVI Model 270A Digitizer which has two cables from the edge of printed circuit cards on the back of it, to connectors on the Chromatics, which contains the interface board inside. In addition, single coax cables to the TV camers and to the TV set monitor (which allows the user to see what the camera sees and to make adjustments) are used. Be sure these cables are properly connected.

To begin, three power switches and the auxiliary lights need to be turned on (there is no specific order for this procedure): (1) Push the push-button power switch on the left side of the digitizer's front panel (which should light up when ON); (2) push the push-button on the left under the screen of the TV set monitor; and (3) set the rocker-type switch on the end of the camera opposite the lens. (4) Turn on the auxiliary lights.

Be sure to remove the lens cap but DO NOT LOSE IT! Put it back over the lens when the session is finished.

Position the picture to be digitized on the easel and adjust the camera height, focus and picture position so the image fills and is centered in the TV set monitor. The camera height is adjusted by turning the knob between the camera and the vertical support post fastened to the back of the easel. The focus is adjusted by turning the camera lens.

On the front of the interface panel (See Figure 5b) are three switches and three knobs labeled as follows:

Switches	Knobs
Mode: Setup/Run	Black Level
Display: On/Off	Video Amplitude
Cursor: Line/Dot	Setup Position

Set the MODE SWITCH to "Setup".

Set the DISPLAY SWITCH to "On".

Set the CURSOR SWITCH to "Line".

You should see a display similar to that shown in Figure 5a. The sample line may be off the screen, therefore rotate the SETUP POSITION control knob slowly and it should appear. Position this sample line as close as possible to the center of the image but where it crosses some of the picture lines. Between the lines on the right (shown as black reference line and white reference line in Figure 5a) is a plot of the pixel value of each pixel intersected by the sample line. Now adjust the BLACK LEVEL and VIDEO AMPLITUDE control knob (Figure 5b) to make that plot just fit between the lines. The BLACK LEVEL control knob moves the graph left and right and the VIDEO AMPLITUDE controls its height. The peaks should touch the lines. Now rotate the SETUP POSITION knob left then right slowly; the display line should sweep across the image and the graph will change. Observe the graph as it changes and compromise the black level and video amplitude controls between the reference lines when the display line is over the left and right ends of the picture, and when it is over the middle.

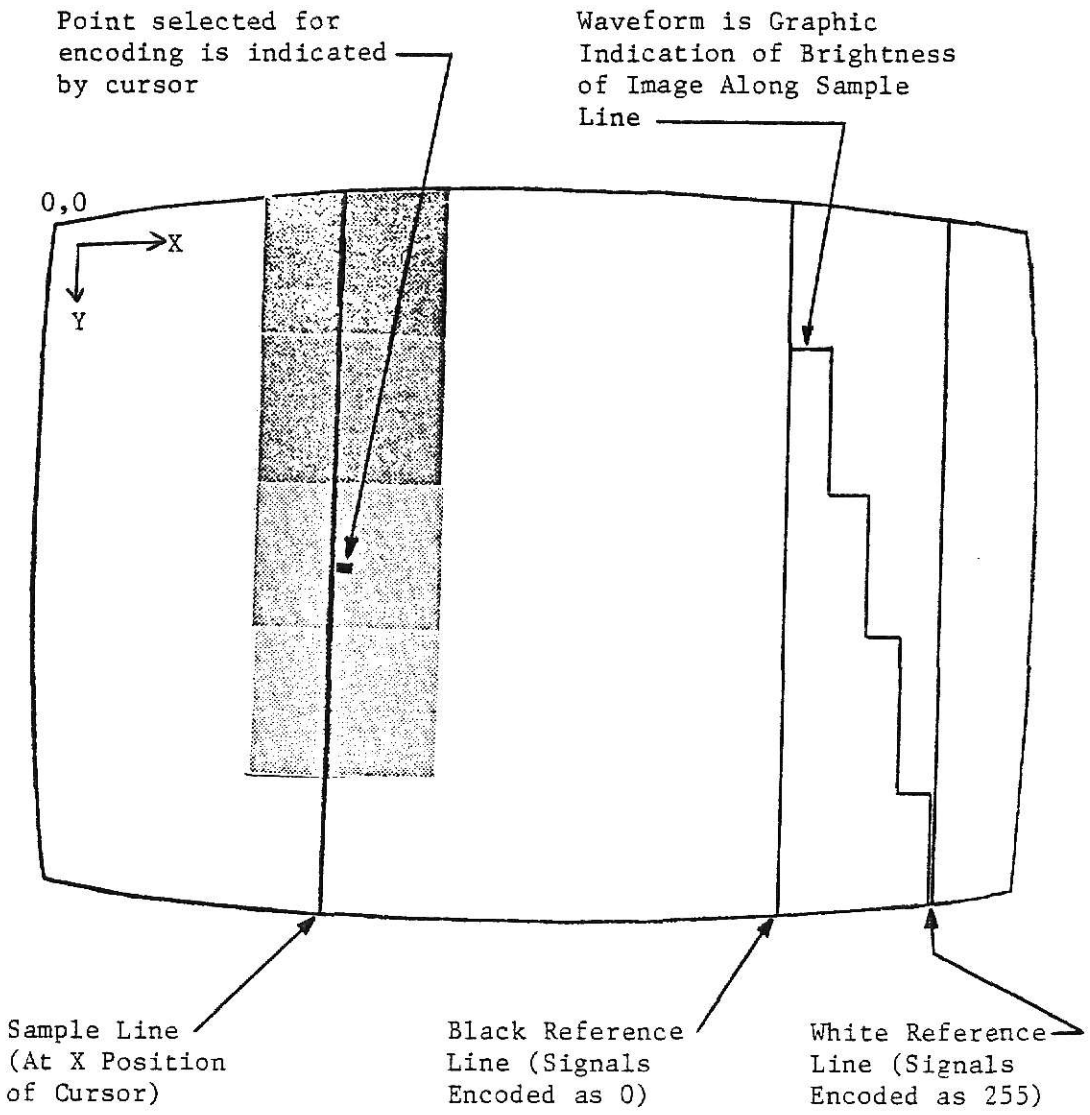
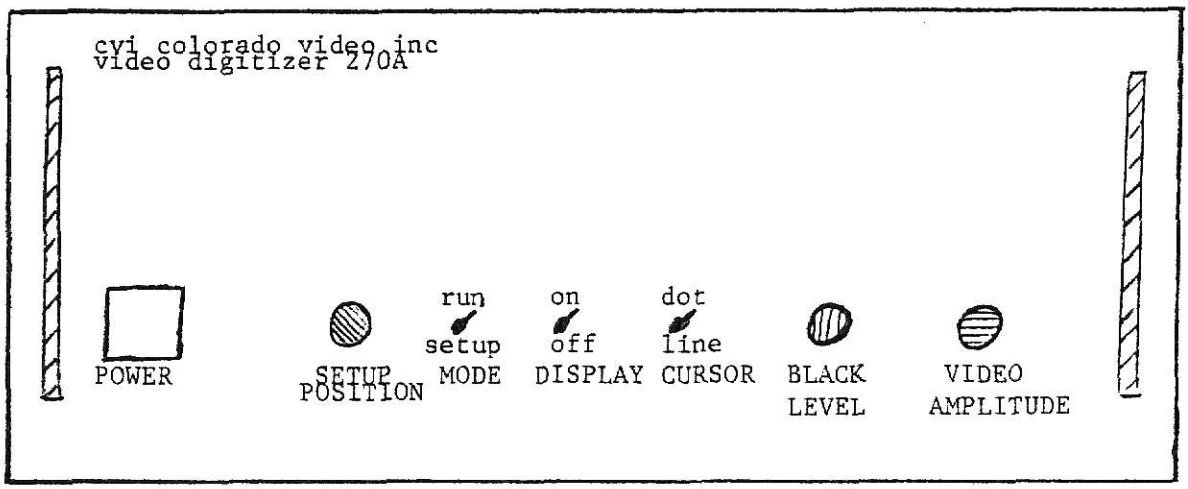


Figure 5a. Typical TV Monitor Display when Adjusting Digitizer

Figure 5b. Interface Control Panel



Next, change the switch settings as follows:

Set the MODE SWITCH to "Run".

Set the DISPLAY SWITCH to "off".

Set the CURSOR SWITCH to "Dot".

Be careful to NOT disturb the control knob settings.

Load your program into the Chromatics and begin execution. A small white dot (shown black in Fibure 5a) will mark where your program is currently digitizing.

To initialize the interface, the program should initially call Subroutine 3000 (presented in the previous chapter). For the first and for each succeeding time the X co-ordinate changes, Subroutine 2000 must be called. For each pixel value, the Y co-ordinate must be set and Subroutine 2050 called.

The digitizer's X and Y co-ordinates are (zero, zero) in the upper left hand corner whereas the Chromatics (zero, zero) point is in the lower left hand corner of the screen. The co-ordinates may be converted from the digitizer to the Chromatics by the following:

Chromatics X = Digitizer X

Chromatics Y = 512 - Digitizer Y

The range of values are 0 to 479 on X and 0 to 511 on Y. These are integer values only.

As the BASIC interpreter is so slow, it is not possible to take advantage of the synchronous operation of the digitizer. If synchronous operation is deemed necessary, these routines must be encoded into assembler (read pages 19, 21, and 22 of Reference 1).

CHAPTER 5

Correction for Illumination and Usable Gray Levels

With ordinary room light illumination of the subject picture, two pixel value anomalies occured.

The first, and most serious, anomoly discovered was that the analog picture signal sent by the TV camera was full of "noise". This "noise" is visible as random fluctuation superimposed upon the pixel value plot of Figure 5a. The effect to the program was that if the same pixel value was read twice in a row, the value could differ by as much as 30 out of the range of 255, or about 13%! In addition, since the Chromatics has 8 colors, it was deemed necessary that at least 8 levels of gray be distinguishable. With a total pixel range of 256, this meant that resolution in the levels 32 apart must be detected; with the noise figure stated above, correct pixel value detection was impossible.

After contemplation of various causes of picture signal noise, it was suggested that the low light level was forcing the internal amplifiers of the TV camera to run at maximum gain, thus producing a poor signal to noise ratio. To counteract this low light problem, two additional 20 watt fluorescent fixtures were placed parallel to the long edge of the picture. This reduced the noise to the 5 to 8 range. A test made with a gray scale strip (the image in Figure 5a is one) confirmed correct operation.

Figure 6. Program for adjusting auxiliary lighting.

```
 10    REM  READ AND PRINT PIXEL VALUES IN A 7 BY 7
 20    REM  GRID EVENLY SPACED ON SCREEN
 30    GOSUB 3010 : REM  INITIALIZE INTERFACE
 40    FOR Y%  3 TO 477 STEP 79
 50       FOR X%  4 TO 508 STEP 84
 60          GOSUB 2000 : REM SEND X CO-ORD
 70          GOSUB 2050 : REM SEND Y CO-ORD AND READ PIXEL VALUE
 80          PRINT Z%
 90       NEXT X%
100       PRINT : PRINT
110    NEXT Y%
120    END
```

Ideally the values on output should all be 255 or at least somewhat equal.

Typical output -
no auxiliary lighting:

153	196	228	232	246	219	219
176	210	215	248	217	229	209
154	243	255	255	255	235	197
173	229	255	255	246	241	193
174	218	231	239	243	249	238
160	206	254	252	231	234	215
151	162	243	255	222	290	185

Typical output -
with auxiliary lighting:

255	253	249	249	251	253	255
255	251	249	246	250	251	255
255	249	246	244	246	249	255
255	251	244	240	244	251	255
255	249	246	243	246	249	255
255	251	249	245	249	251	255
255	253	251	249	251	253	255

The second anomaly came from the change in reflectance angle for light rays from the picture into the lens. Most paper used for pictures is optimized to reflect light at a ninety degree angle to the paper. However the lens is positioned over the center of the picture, therefore the reflectance angle changes and the readability of the pixel values steadily decrease over a range of 40 when traveling a path from the center to the corner values. Under these circumstances proper gray level operation could not be assured at the corners. The additional light helped alleviate this condition somewhat but added its own variation; too bright on the edges nearest the lights.

As careful light and distance control is needed, it is suggested that a program to allow the equipment to check itself would increase efficiency. For this purpose the program in Figure 6 is included in this report. A white or gray card is to be substituted for the picture. The lights should be adjusted to provide the best illumination possible. When run, the Figure 6 program prints the values at 49 evenly spaced points in a 7 by 7 grid; the lights can then be re-adjusted so that these values are as equal as possible. No attempt was made to distinguish more than the 7 gray levels.

Chapter 6

Speed-up Techniques

To improve the pixel reading speed, the peculiar properties of the standard signal generated by the TV camera must be taken into consideration and the program must be fast enough to synchronize with scanning all 480 pixels down any fixed X co-ordinate column as it is presented by the camera in one-thirtieth of a second. The BASIC interpreter is not capable of this speed. To further complicate matters, all even numbered Y co-ordinate pixels are given before the odd ones (see Figure 7).

Accordingly, the subroutine of Figure 8, USR2, was written in Z-80 assembler language. USR2 reads all 480 pixel values into an array of 16-bit quantities, such as P%. This subroutine would be more properly invoked using a CALL operator, but this version of BASIC does not have this option. Instead the function call was pressed into service, but the value returned is garbage.

```
Example call:   DIM  P%(479)
                D% = USR2(VARPTR(P%(0)))
```

The variable D% will have unusable information (garbage) in it upon return, but the vector P% will have the pixel values in subscript positions corresponding to the Y co-ordinate.

The text of the routine from Figure 8 should be entered into the Chromatics using the text editor and saved on the file USR2. In order for this routine to assemble and interface to BASIC properly, the following

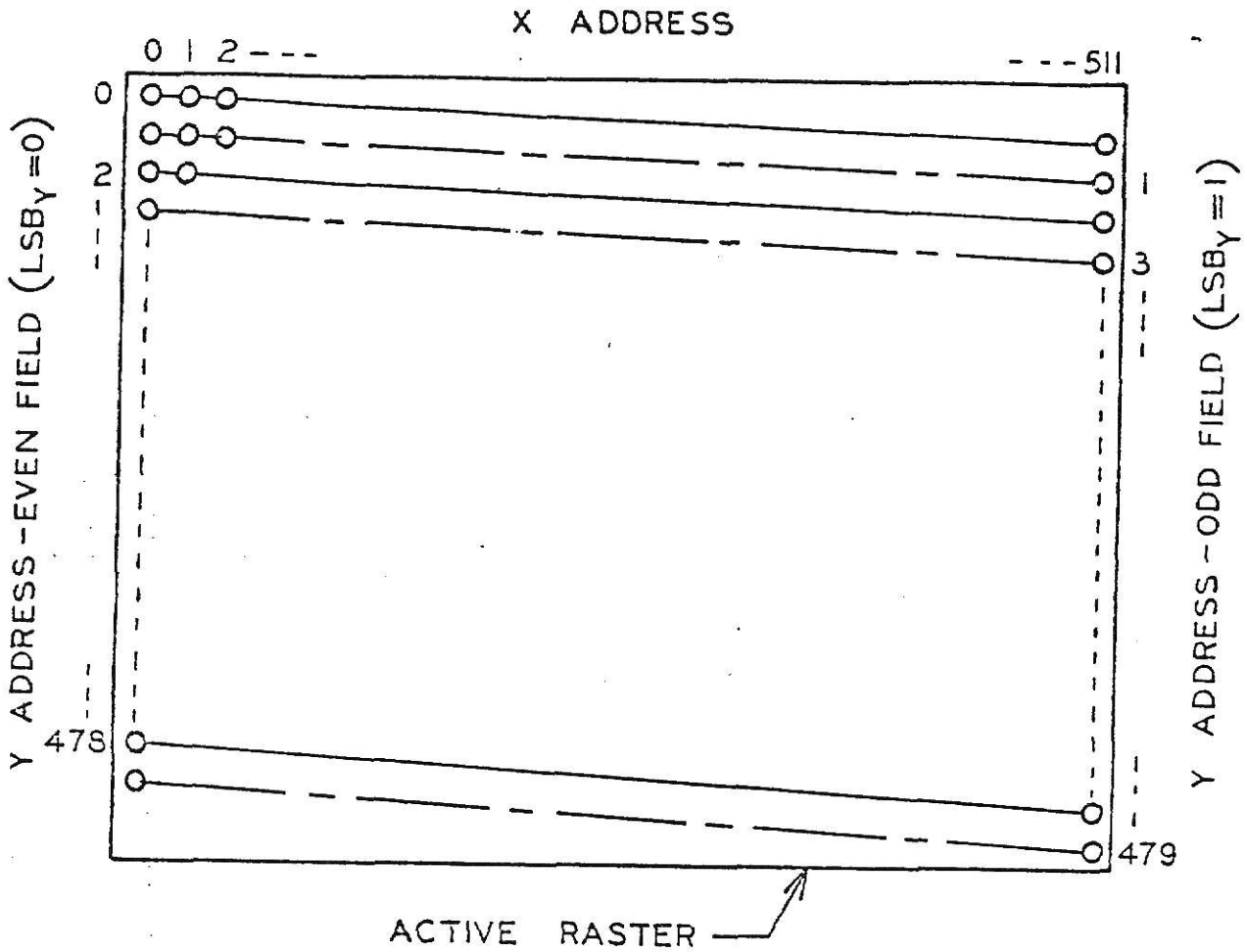


Figure 7. Digitizing Value Presentation Order

sequence must be used:

```
ASMB

BINARY

ASSEMBLE USR2
     .
     .  (assembler output)
     .
CLOSE USR2OBJ

EXIT
```

A typical program to use this subroutine.is given in Figure 9. The output of this is the same as the previous test program given in Figure 4 of Chapter 3. Note that space must be reserved in high memory by responding to the BASIC message "MEMORY SIZE" with the number 32690 instead of the customary return key.

Figure 8: USR2 Function Code

```
;   REGISTER USAGE:
;       HL      ON ENTRY, CONTAINS THE ADDRESS WHERE THE
;               LOW BYTE OF THE P VECTOR CAN BE FOUND.
;               THE HIGH BYTE IS AT HL + 1
;       AFTER INITIALIZATION
;        H      PASS FLAG 4 DOWN TO 1
;        L      RIGHT BITS OF THE Y CO-ORDINATE
;               STOPPING VALUE FOR EACH SCAN
;               WILL CONTAIN DC, DD, DE OR DF CORRESPONDING
;               TO 476, 477, 478 or 477.  476 = 01DC
;       IY      POINTER TO P ARRAY
;       DE      Y CO-ORDINATE VALUE
;       BC      INITIALLY USED TO GET P ADDRESS
;               THEREAFTER HAS CONST 8, THE
;               ADDRESS DIFFERENCE BETWEEN
;               P(0) and P(4)
;    THIS ROUTINE MAKES FOUR PASSES, FIRST FILLING
;               IN P(0), P(4),....P(476), THEN
;               P(1), P(5),....P(477), ETC.
```

(continued on next page)

(continuation)

Figure 8: USR2

```
          ORG     32691;=7FB3
USR2      LD      C,(HC)
          INC     HC
          LD      B,(HC)
          PUSH    BC
          POP     IY    ;  IY = P(0) ADDR
          PUSH    BC    ;  Save on stack
          LD      H,4   ;  Pass count
          LD      DE,0  ;  Y = 0
          LD      L, ODCH  ;  Stop value right 8 bits
          LD      BC,8  ;  BC = 8
PB        LD      A,D
          OR      10H   ;  Command Bits
          OUT     (105),A
          LD      A,E
          OUT     (104),A
U2        IN      A,(106)
          BIT     0,A
          JR      Z,U2-8
          IN      A,(104)   ;  Read pixel
          LD      (IY),A    ;  Poke to P(Y)
          LD      A,E       ;  Y = Stop?
          CP      L
          JR      Z,NX-$    ;  Jump right 8 bits match
CO        INC     DE        ;  Next point Y = Y + 4
          INC     DE
          INC     DE
          INC     DE
          ADD     IY,BC     ;  Point IY to new P(Y)
          JR      PB-$
NX        LD      A,D
          CP      Ø1H       ;  Left 8 bits = 01
          JR      NZ,CO-$   ;  Jump left 8 bits don't match
; Done with one pass, set up for next
          DEC     H
          JR      Z,END-$   ;  Jump on 4th pass
          INC     L         ;  Stop addr = +1
          POP     IY        ;  Reget P(I)  addr I = 0, 1, 2,
          INC     IY        ;  Bump to P(I+1) addr I+1 = 1, 2, 3
          INC     IY
          PUSH    IY        ;  Resave P(1)
          LD      D,0
          SUB     E,ØDBH    ;  Restart Y
          JR      PB-$
END       POP     IY        ;  Clean up stack
          RET
          END
```

Figure 9: DEMONSTRATION OF USR2 FUNCTION USAGE

```
10    DIM P%(479)
20    DOS"FETCH USER2OBJ"  :  REM READ SUBROUTINE INTO MEMORY
30    DEF USR2=32691
40    GOSUB 3010 : REM  INITIALIZE INTERFACE
50    PRINT "∩∪G";CHR$(12)  :  REM ENTER PLOT MODE, CLEAR SCREEN
60    FOR X%=0 TO 511 STEP 2
70        GOSUB 2000 : REM  SEND X CO-ORD
80        D%=USR2(VARPTR(P%(0))) : REM READ A COLUMN OF PIXELS
90        FOR Y%=0 TO 479
100           IF P%(Y%)<127 THEN PLOT X%,512-Y%
110       NEXT Y%
120   NEXT X%
130   MODEOFF
140   END
```

Chapter 7

Demonstration Program
False Color Picture From Gray Levels

This chapter concludes the report with a presentation of the program in Figure 10. The Figure 10 program reads all pixel values using the routines previously developed in Chapters 3 and 6, converts them to seven gray levels, and then displays them as points in color according to this table:

Gray Code	Pixel Value	Color
0	0-31	Black (blank screen)
1	32-63	Blue
2	64-95	Green
3	96-127	Cyan
4	128-159	Red
5	160-191	Magenta
6	192-223	Yellow
7	Over 223	White

Lines 90 and 100 read all 479 pixels in the column at X-coordinate given by X%. The loop composed of Statements 110 through 180 cycles through the values of P% and in line 120 converts them to gray levels one at a time in the variable G%. Line 130 checks to determine if the gray level changed since the previous loop and, if so, lines 150 and 160 changes the Chromatics plotting color. The divisor of line 120 was determined ad hoc by following this procedure:

1. The white card was substituted for the picture to be processed and the lights adjusted as discussed in Chapter 5.
2. The real picture was put back and positioned, and the digitizer was adjusted using the procedure outlined in Chapter 4.
3. The previous program (program used in step 1) was re-executed. It was observed that the range of the pixel values was actually only 221 for the particular picture being processed; the lightest area of the picture was actually slightly gray. This figure was divided by 7 (seven intervals between eight levels) and rounded to obtain the 32.

The backslash used in line 120 is a special Chromatics operator that performs integer division instead of floating point (floating point accuracy was unnecessary in this particular program).

Run time for this particular program is approximately 50 minutes. This excessive run time is primarily due to the number of calls of the PLOT command (131,072 calls). The program plots every other column in order to run at this speed. If all points were plotted, over 262 thousand calls would be required!

Figure 10: False Color Plot Program

```
10      REM   DISPLAY FALSE COLOR PICTURE FROM PIXEL VALUES
20      DIM P%(479)
30      DOS "FETCH USER20BJ
40      DEF USR2=32691
50      GOSUB 2010 : REM INITIALIZE INTERFACE
60      PRINT CHR$(12);"∿G";  : REM CLEAR SCREEN, TURN ON PLOT MODE
70      OG%=9 : REM OLD GRAY LEVEL LAST POINT
80      FOR X%=0 TO 511 STEP 2
90         GOSUB 2000 : REM SEND X CO-ORD
100        D%=USR2(VARPTR(P%(0)))
110        FOR Y%=0 TO 479
120           G%=P%(Y%)\32
130           IF G% = OG% THEN 170
140           REM  GRAY LEVEL CHANGED, CHANGE PLOT COLOR
150           OG% = G%
160           PRINT "∿C";CHR$(48+G%)
170           PLOT X%,512 - Y%
180        NEXT Y%
190     NEXT X%
200     MODEOFF
210     END
```

References

1. CVI Digitizer Handbook, Colorado Video Incorporated, P. O. Box 928, Boulder, CO 80306, 38 pages.

2. Option 33 - Operating Instructions - Dual Bi-Directional Parallel I/O Ports, Chromatics, Inc., 3923 Oakcliff Industrial Court, Atlanta, GA 30340, 5 pages.

3. Chromatics CG BASIC Reference Manual, Chromatics, Inc., Atlanta, GA 30340, 73 pages.

4. Preliminary Operator's Manual, CG Series Color Graphics Computers, Chromatics, Inc., Atlanta, GA 30340, 135 pages.

5. Chromatics Disk Software Reference Manual, CG Series Color Graphics Computers, Chromatics, Inc., Atlanta GA 30340, 69 pages.

6. Kodak Black and White Dark Room Data Guide, Kodak Publication No. R-20, Eastman Kodak Company, Rochester, NY 14650, 36 pages.

INTERFACING A TV PICTURE DIGITIZER TO THE CHROMATICS COLOR-GRAPHICS COMPUTER

by

JEROME ANTHONY HILL

B. S., Oklahoma State University, 1966

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

ABSTRACT

INTERFACING A TV PICTURE DIGITIZER TO THE CHROMATICS COLOR GRAPHICS COMPUTER

This paper concerns connecting the video output of a common TV camera to a color graphics computer manufactured by Chromatics Incorporated. Discussed are the electronic hardware modifications necessary, software I/O drivers, suggestions on proper operation and adjustment of the system, resolution of subject illumination difficulties, and a program is presented for false-color screen displays according ot the gray level of the digitized TV picture data.