IPLIB (IMAGE PROCESSING LIBRARY) USER'S MANUAL

Nettie D. Faulcon, James H. Monteith, and Keith Miller

December 1985

Library Copy

JAN 13 1986

Langley Research Center
Hampton, Virginia 23665
INTRODUCTION

IPLIB is a collection of HP FORTRAN 77 subroutines and functions that facilitate the use of a COMTAL image processing system driven by an HP 1000 computer. These subprograms are based on work by Nettie D. Faulcon and James H. Monteith of the Acoustics and Vibration Instrumentation Section at NASA Langley Research Center, and were put into their present form by Keith Miller of the College of William and Mary's computer science department while he was a summer fellow in the ASEE/NASA program.

IPLIB uses an HP 1000 driver, DVR41, which directs all communications between the HP 1000 and the COMTAL Vision One/20 system. DVR41 was written by M. Brown of Coulter Computer Corporation, revised by R.W. Bagdazian of Hughes Aircraft Corporation, and documented by James Monteith and Keith Miller.

This user's manual is intended for programmers who want to use the HP 1000 to drive the COMTAL image processing system. It is assumed that the programmer knows HP 1000 FORTRAN 77, or at least one FORTRAN dialect. It is also assumed that the programmer has some familiarity with the COMTAL Vision One/20 system.

The manual is divided into six sections:

1. SUBPROGRAM HANDLING:
   This section tells how to load and modify the subprograms in IPLIB as well as how to add subprograms to the library.

2. ALPHABETIC CATALOG:
   All the subprograms are listed in alphabetic order with a few words that suggest the purpose of each. Test programs are listed in alphabetic order also.

3. CROSS REFERENCE FILE:
   Each subprogram and test program is listed along with the subprograms it calls.

4. SUBPROGRAM SOURCE CODE:
   The source code for each subprogram is given in its entirety. The code for all subprograms follows a documentation outline which was designed to make each subprogram understandable without reference to any other code.

5. TEST PROGRAM SOURCE CODE:
   The source code for some programs used in the unit testing of the IPLIB subprograms are given. These test programs are not well documented, but they should be easily understood when used in conjunction with the documentation.
of the subprograms they test.

6. HP DRIVER SOURCE CODE:
The assembler code of DVR41, the HP driver for the COMTAL image processing system, is listed in its entirety.

The following references are suggested for programmers working with IPLIB:


SUBPROGRAM HANDLING

LOADING SUBPROGRAMS

When loading a program that calls IPLIB subroutines and function ("subprograms"), enter:

SEALIB::21

as part of the loading sequence. For example, the following loading sequence is used to load a program TNOTE which calls a number of IPLIB subprograms:

LOADR
RE:*TNOTE
SEALIB
SEA
D1
EN

ADDING SUBPROGRAMS

In order to add subprograms to the library, 4 files must be changed: FILES::21, XREF::21, MAKELIB::21, and %IPLIB::21.

FILES::21 is an alphabetized, annotated list of the IPLIB subprograms and test programs for those subprograms. Using the text editor, add the name of the sub-
program in alphabetic order along with a short description of its effect. Also, add the name of the unit testing program to the second list in FILES::21.

XREF::21 is a list of each subprogram and unit testing program along with the subprograms it calls. Add the new subprogram and testing program to the list, using the same format that is already there.

MAKLIB::21 is a list of the load modules of all the subprograms and test programs in IPLIB. This list is in no special order, so just add the new subprogram's load module to the list. The standard convention is to use % as the first character in load modules.

XPLIB holds all the load modules in the image processing library. If NEW is the name of the load module to be included in IPLIB, the following sequence accomplishes the addition: (HP prompts given within parentheses.) "<CR>" signifies entering a carriage return.

!(:MERGE
(ENTER DESTINATION NAMR) XPLIB<CR>
(ENTER COMMAND NAMR) I<CR>
(ENTER NAMR) XPLIB<CR>
(ENTER NAMR) NEW<CR>
(ENTER NAMR) /E<CR>

CHANGING SUBPROGRAMS

Modified programs must be tested before being changed in the library. Changes to parameters are not allowed; if such a serious modification is necessary, a NEW subprogram should be written, and added with a slightly different name. (See above for adding a subprogram to IPLIB).

If a subprogram is modified, the new source code must replace the old code. If necessary, the explanation in FILES::21 must be changed. If a different set of subprograms are called, XREF::21 must be changed. In order to replace the old load module in XPLIB with the modified version, enter the following sequence: (HP prompts given within parenthesis.)

!(:MERGE
(ENTER DESTINATION NAMR) XPLIB
(ENTER COMMAND NAMR) MAKLIB
Unless otherwise stated, these files contain subroutines.

ADD2: adds 2 images, and requires a scaling factor (p. 11).
ADDIM: adds 2 images, no scaling factor (p. 15).
BUFDIS: transfers a monochrome image to COMTAL from disk (p. 19).
CLFDS: transfers a color image to COMTAL from disk (p. 22).
CLRGR: clears a COMTAL graphics plane (p. 26).
CLRIM: clears a monochrome COMTAL image (p. 28).
CMND: sends a COMTAL command given as a constant string from HP to COMTAL (p. 33).
COUNT: computes the pixel count for each of the possible values, 0 - 255 (p. 36).
DELAY: puts HP in a busy wait for number of seconds designated (p. 39).
DIGIT: function that takes 0 - 9 integer input and returns '0' - '9' (p. 41).
DSPB: displays a monochrome COMTAL image (p. 43).
DSPCL: displays a color COMTAL image (p. 45).
DSPGR: displays a graphic plane to the COMTAL monitor (p. 48).
DSPVD: displays the COMTAL image (5) set to the video camera (p. 50).
HILO: scans a monochrome image and returns the high and low pixel values (p. 52).
HISTO: displays a scaled histogram of the designated image on the COMTAL monitor (p. 54).
ICOPY: copies one monochrome COMTAL image to another (using CMND) (p. 57).
ICPY2: copies one monochrome COMTAL image to another (using RDILN & WRILN) (p. 60).
MERGE: merges two bytes into one byte (all arguments are INTEGER) (p. 63).
NORM: finds lowest pixel value in an image, then subtracts that value from all pixels in that image; used to get light table variations (p. 65).
NOTE2: writes a line of characters into a graphics plane with a given color and size at a location; takes a constant string argument (p. 67).
NOTES: writes a line of characters into a graphics plane with a given color and size at a location; takes a character array and length (p. 72).
PAINT: interactive "painting" of square patches on COMTAL image (p. 76).
PROFIL: gives HP access to the COMTAL profiling capabilities (p. 80).
RANGE: logical function that determines if 1st argument is within 2nd & 3rd (p. 83).
RDILH: reads one horizontal line of a COMTAL graphics plane (p. 84).
RDIP: reads one point from a COMTAL graphics plane (p. 86).
RDIL2: reads one horizontal line of COMTAL pixels; 1 pixel/integer returned (p. 89).
RDILN: reads one horizontal line of COMTAL pixels; 2 pixels/integer returned (p. 92).
RDIP2: reads one pixel from a COMTAL monochrome image (p. 94).
RDIRC: reads a rectangle of pixels from a COMTAL monochrome image (p. 97).
RDUT: reads the contents of a COMTAL look-up table (p. 101).
RDP: reads the contents of a COMTAL pseudo-color table (p. 104).
RDATAB: reads the COMTAL Image/Graphics Table (p. 106).
RDTR : reads the COMTAL cursor location (p. 109).
SETV : sets a COMTAL image 5 - 9 to the video camera (p. 111).
SPLIT: splits an integer into two bytes, both bytes stored in new integers (p. 114).
SPRED: finds low and high pixel values in an image, and does a linear stretch on all pixel values to expand the range to 0 - 255 (p. 116).
SUB12 : subtracts two images with an offset of 128; differences <0 set to 0 (p. 119).
SUBIM : subtracts two images with no offset; differences <0 set to 0 (p. 123).
THRSR : sets pixels in output image to black(0) or white(255) depending on the corresponding pixel in input image and a threshold value (p. 126).
TSTl1: generates "Test Image 1", increasing pixel values right and down: display appears as a darkening slash across the screen (p. 129).
TV2C4 : digitizes 4 images from TV camera and averages them into one image (p. 131).
TV2CM : digitizes an image from TV camera into a COMTAL memory plane (p. 133).
WAIT : halts HP processing until the HP <CR> is entered (p. 136).
WPGR : removes a graphics-plane from the display (p. 137).
WRLN : writes a horizontal line of graphics bits to a COMTAL graphics plane (p. 139).
WRTPT : writes one graphics bit to a COMTAL graphics plane (p. 141).
WRL2 : writes a horizontal line of pixels to a COMTAL image memory: one pixel value / integer in the buffer (p. 144).
WRILN : writes a horizontal line of pixels to a COMTAL image memory: two pixel values / integer in the buffer (p. 147).
WRTP : writes one pixel value to a COMTAL image memory (p. 150).
WRP : writes an array of integers to a rectangle of a COMTAL image memory: one pixel value / integer in the buffer (p. 153).
WRLT : writes a look-up table to the COMTAL (p. 157).
WRPSU : writes a pseudo-color table to the COMTAL (p. 159).
WRTAR : writes a target (cursor) location to the COMTAL (p. 161).

The files that follow are test programs for the subprograms above.

TADD2 : program that tests ADDI2 (p. 163).
TADDI : program that tests ADDIM (p. 164).
TCLR : program that tests DSPCL, "Display Colo" (p. 165).
TCLOR : program that tests CLRGR, "Clear Graphics" (p. 166).
TCLRI : program that tests CLRIM, "Clear Image" (p. 167).
TCN2 : program that sends all possible single bytes to COMTAL 1 at a time (p. 168).
TCMIF : program that sends COMTAL commands via the HP keyboard: tests CMND (p. 169).
TCLNT : program that tests subroutine COUNT (p. 170).
TCNST : program that tests the string concatenation facility in HP FORTRAN 77 (p. 171).
TCPY : program that tests ICOPY, "Image COPY" (p. 172).
TEDG1 : program that tests the function DIGIT (p. 173).
TDSP : program that tests DSPBUW, "Display Black & White" and DSPCL, "Display Colo" (p. 174).
THIST : program that tests HIST, "Histogram" (p. 175).
TNORM : program that tests NORML, "NORMALize" (p. 176).
TNOTE : program that tests NOTE2 and NOTES (p. 177).
TPNT : program that tests PAINT (p. 178).
TPROF : program that tests PROFIL, "PROFiling" (p. 179).
TRANG : program that tests the function "RANGE" (p. 180).
TRDATA : program that tests RDTAR, "Read TARget" (p. 181).
TSETV : program that tests SETV, "SET Video camera" (p. 182).
TSPRD : program that tests SPRED, "SPReD pixel values" (p. 183).
TSUB1 : program that tests SUBIM and SUB12 (p. 184).
TTTRS : program that tests THRSH, "THReSHOlding" (p. 185).
TTSTI : program that tests TST1, "Test Image 1" (p. 186).
TTV2C : program that tests TV2CM and TV2C4, "TV to Color" transfers (p. 187).
TWAIT : program that tests WAIT (p. 188).
TUIPE : program that tests WIPGR and DSPGR (p. 189).
TXFDS : program that tests BWFDS and CLFDS, "Black & White From Disk" and "Color From Disk" (p. 190).
TXGLN : program that tests WRGLN and RDGLN, "Write Graphics Line" and "Read Graphics Line" (p. 191).
TXGPT : program that tests WRGPT and RDGPT, "Write Graphics Point" and "Read Graphics Point" (p. 192).
TXILN : program that tests WRILN and RDILN, "Write Image Line" and "Read Image Line" (p. 193).
TXIPT : program that tests WRPT and RDPT, "Write Image Point" and "Read Image Point" (p. 194).
TXIRC : program that tests WRIRC and RDIRC, "Write Image ReCTangle" and "Read Image ReCTangle" (p. 195).
TXLUT : program that tests WRLUT and RDLUT, "Write Look Up Table" and "Read Look Up Table" (p. 196).
TXPSU : program that tests WRPSU and RDPSU, "Write PSeUdo-color table" and "Read PSeUdo-color table" (p. 197).
TXTAR : program that tests WRTAR and RDTAR, "Write TARGET" and "Read TARGET" (p. 199).

CROSS-REFERENCE FILE

To use this file to find out which procedures or programs call a certain procedure "FRED", just use the text editor to locate all the lines that contain FRED. One of those lines is the line "FRED calls:..." The remaining lines identify which procedures call FRED.

HP FORTRAN77 intrinsic functions (which need not be loaded manually) are preceded with a *.

ADDI2 calls:
   DIGIT CMND RANGE
ADDIM calls:
CMND DIGIT RANGE

BUFD calls:
CMND DIGIT OPEN RANGE READF WRILN

CLFDS calls:
CMND DIGIT OPEN RANGE READF WRILN

CLGR calls:
CMND DIGIT RANGE

CLRIM calls:
CMND DIGIT RANGE

CMNN2 calls:

*LEN

CMND calls no other procedures.
COUNT calls:
RANGE RDIL2

DELAY calls no other procedures.

DIGIT calls:
RANGE

DSPBW calls:
RANGE

DSPCL calls:
CMND DIGIT RANGE

DSPGR calls:
CMNN2 DIGIT RANGE

DSPVD calls:
CMND DIGIT

HILO calls:
RDIL2

HISTO calls:
CMNN2 DELAY DIGIT RANGE
I COPY calls:
CMND DIGIT RANGE
ICPY2 calls:
CMND DIGIT RANGE
MERGE calls no other procedures.

NORML calls:
HILO RDIL2 WRIL2
NOTE2 calls:
ADDGR CMNN2 CMND DELAY DIGIT LEN RANGE
NOTES calls:
CMNN2 CMND DELAY DIGIT DSPGR RANGE

PAINT calls:
CMND RDTAR WAIT WRIRC

PROFL calls:
CHAR CMND DIGIT RANGE WAIT

RANGE calls no other procedures.

RDGLN calls:
RANGE
RGPT calls:
BTEST RANGE
RDIL2 calls:
*ICHAR RANGE
RDILN calls:
RANGE
RDIPT calls:
*ICHAR RANGE
RDIRC calls:
*ICHAR RANGE RDILN
RDLUT calls:
RANGE
RDPSU calls no other procedures.
RDTAB calls:
BTEST RANGE
RDTR calls no other procedures.
SETV calls:
CMND DIGIT RANGE
SPLT calls no other procedures.
SPRED calls:
FLOAT HILO IFIX RDIL2 WRIL2
SSORT calls no other procedures.
SUB12 calls:
CMND DIGIT RANGE
SUBIM calls:
CMND DIGIT RANGE
THRSH calls:
RANGE RDIL2 WRIL2
TSTII calls:
WRILN
TV2C4 calls:
ADDI2 RANGE TV2CM
TV2CM calls:
CMND DIGIT DSPBW RANGE
WAIT calls no other procedures.
UIFGR calls:
CMND2 DIGIT RANGE
URGLN calls:
RANGE
WRGPT calls:
*ICCLR *IBSET RANGE
WRIL2 calls:
CHAR RANGE
WRILN calls:
RANGE
calls: RANGE
calls: CHAR RANGE RDIL2 WRIL2
calls: RANGE
calls: no other procedures.

RANGE

The following are test programs for many of the procedures above:

TADD2 calls:
    ADDI2 CMDND DIGIT RANGE
TADDI calls:
    ADDIM CMDND DIGIT RANGE
TCLR calls:
    DSPCL
calls: CLRGR CMDND DIGIT RANGE
calls: CLRIM CMDND DIGIT RANGE
calls: CMDND WAIT
TCMM2 calls:
    CMDND
TTCNT calls:
    COUNT RANGE RDIL2
calls: no other procedures.
TCOPY calls:
    CMDND DIGIT ICOPY RANGE
calls: CMDND
TDELA calls:
    DELAY CMDND2 WAIT
TDIGI calls:
    DIGIT RANGE
calls: CMDND DSPBW DSPCL RANGE WAIT
TDSPV calls:
    DSPVD CMDND DIGIT WAIT DSPBW RANGE
THIST calls:
    CMDND2 DIGIT HISTO RANGE
TNORM calls:
    HILO NORML RANGE RDIL2 WRIL2
calls: CMDND2 CMDND DELAY DIGIT DSPGR NOTE2 NOTES RANGE URTAR
TPNT calls:
CMDND PAINT RANGE RDILN RDTAR WRILN WRIRC
TPROF coils:
CMNND DIGIT PROFL RANGE WAIT
TRANG coils:
RANGE
TRDTA coils:
RANGE RDTAB
TRSET coils no other procedures.
TSETV coils:
SETV RANGE DIGIT CMMND DSPBW
TSPRD coils:
RANGE RDIL2 SPRED WRIL2
TSSRT coils no other procedures.
TSUB1 coils:
CMNND DIGIT RANGE SUBI2 SUBIM WAIT
TTHRS coils:
RANGE RDIL2 THSH WRIL2
TTSTI coils no other procedures.
TTV2C coils:
ADDI2 CMNND DIGIT DSPBW DSPVD RANGE TV2C4 TV2CM WAIT
TWAIT coils:
WAIT
TWIPE coils:
CMNND DIGIT DSPGR RANGE WAIT WIPGR
TXFDS coils:
BLFDS CLFDS CMNND DIGIT RANGE WRILN
TXGLN coils:
RANGE RDGLN URGLN
TXGPT coils:
RANGE RDGPT RDTAR WRGPT
TXILN coils:
RANGE RDILN WRILN
TXIPT coils:
RANGE RDIPT WAIT WRIPT
TXIRC coils:
RANGE RDILN RDIRC WRILN WRIRC
TXLUT coils:
RANGE RDLUT WRlut
TXPSU coils:
RDPSU WRPSU RANGE WAIT
TXTAR coils:
RDTAR WRTAR RANGE
&ADDIM T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ADDIM(C, A, B)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER C ! the COMTAL image into which the sum of
0008 C ! image A and image B is placed by (C = A + B)
0009 INTEGER A, B ! the images whose sum is taken (C = A + B)
0010 C
0011 C***INTRODUCTION:
0012 C
0013 C The subroutine "ADD IMAGES" takes a pixel by pixel sum
0014 C of images A and B and places the resulting image into image C.
0015 C The truecolor image B is used in the processing of ADDIM, and will
0016 C be left as the combination of C, A, and B for red, green, and blue
0017 C respectively.
0018 C ADDIM does not do any scaling of the addition. If scaling is desired,
0020 C see the procedure ADDI2.
0021 C
0022 C***LANGUAGE:
0023 C
0024 C FORTRAN 77, the HP 1000 version for RTE-G/VM.
0025 C
0026 C***LIMITATIONS:
0027 C
0028 C The truecolor B image is destroyed during this operation. C is
0029 C obviously destroyed. This subroutine is accomplished using COMTAL
0030 C commands that exploit the pipeline processors. Because of this, the
0031 C processing steps are obscure. For example, there is no motivation
0032 C outside the COMTAL instructions for making the combination of C, A,
0033 C B a color image. Readers should be aware of these obscurities before
0034 C trying to understand the code.
0035 C
0036 C If the sum of any two pixels exceeds 255, the value in C is set to 255.
0037 C
0038 C ADDIM does not scale or offset the sum result. If you wish to scale
0039 C the sums, see the subroutine ADDI2.
0040 C
0041 C If any of the image numbers are out of range, an error message is printed
0042 C and no further processing takes place.
0043 C This subroutine assumes that 0 is not a legal image for the COMTAL
0044 C configuration.
0045 C
0046 C**SUBPROGRAMS CALLED:
0047 C
0048 C name source load remarks
0049 C ------- ------- ------- ----------------------------------------
0050 C CMDND &CMND $CMND Sends a command to the COMTAL as if the
0051 C command were sent from the keyboard
0052 C RANGE &RANGE $RANGE logical function that determines if the 1st
0053 C parameter is within the range of the 2nd & 3rd.
0054 C DIGIT &DIGIT $DIGIT character*1 function which returns '0'-'9'
0055 C according to integer input 0-9.
0056 C
0057 C**WRITTEN BY:
0058 C
0059 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0060 C summer fellowship.
0061 C
0062 C**REVISION HISTORY:
0063 C
0064 C
0065 C**LOCAL VARIABLES:
0066 C
0067 C INTEGER IBUF(128) ! a buffer for passing commands to COMTAL
0068 C CHARACTER*255 CBUF ! character overlay for IBUF
0069 C EQUIVALENCE (IBUF,CBUF)
0070 C
0071 C LOGICAL RANGE ! function that determines if 1st parameter
0072 C is within 2nd and 3rd parameter
0073 C
0074 C INTEGER IMLO, IMHI ! limits on COMTAL image numbers
0075 C INTEGER TERM
0076 C CHARACTER*1 DIGIT ! character*1 function that returns '0'-'9'
0077 C according to integer input 0-9.
0078 C
0079 C**INITIALIZATIONS:
0080 C
0081 C DATA IMLO/1/, IMHI/4/
0082 C DATA TERM/1/
0083 C
0084 C**PROCESSING
0085 C
0086 C IF (.NOT.(RANGE(A,IMLO,IMHI))) GOTO 8001 ! error return
0087 C IF (.NOT.(RANGE(B,IMLO,IMHI))) GOTO 8101 ! error return
0088 C IF (.NOT.(RANGE(C,IMLO,IMHI))) GOTO 8201 ! error return
0089 C
0090 C The following character string sends a series of keyboard
commands to the COMTAL. In the comments below, each command
is explained. The notation _X where X is either A, B, or C
stands for the single character that corresponds to the single
digit number associated with the parameter X.
In this notation, letters in caps were entered into CBUF, and
lower case letters are the full commands filled in by the COMTAL
NOTE: this code assumes that the digit 0 is NOT a legal value for
the parameters A, B, and C.
The "$" separates COMTAL commands.

CBUF =

1 'UN I B $'
UNassign Image B ! just in case B is already assigned.
2 'AS T B '/DIGIT(C)/' '/DIGIT(A)/' '/DIGIT(B)/' $'
ASSign Truecolor image B red _C blue _A green _B
3 'D I B $'
Display Image B
4 'SE COM G + B / 1 $' ! sets 0 offset by default, and
   '/ 1' sets no scaling. 
5 'A COM $'
Add COMbine
6 'I '/DIGIT(C)/' D R $'
Image _C = Displayed Image Red ! Red is arbitrary, since difference
   of images is monochrome
7 'D I '/DIGIT(C)/' $'
Display Image _C
8 'SU COM '
SUBtract COMbine.
CALL CMDND(IBUF, 7B)
RETURN

CALL ERROR RETURNS

WRITE(TERM, E203) A
E203 FORMAT(' THE 2ND IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE. )
GOTO 8900

WRITE(TERM, E203) B
E203 FORMAT(' THE 3RD IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE. )
GOTO 8900

WRITE(TERM, E203) C
E203 FORMAT(' THE 1ST IMAGE PARAMETER, ', I3, ', IS OUT OF RANGE. )
GOTO 8900
0137  8900  WRITE(TERM, 8901)
0138  8901  FORMAT(" ADDIM RETURNS WITHOUT FURTHER PROCESSING.")
0139       RETURN
0140  C
0141       END
0142
&ADDI2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  SUBROUTINE ADDI2(C, A, B, SCALE)
0003  C
0004  C
0005  C***PARAMETER DECLARATIONS:
0006  C
0007  INTEGER C       ! the COMTAL image into which the sum of
0008  C
0009  INTEGER A, B    ! the images whose sum is taken (C = A + B)
0010  INTEGER SCALE   ! each pixel sum divided by this number.
0011  C
0012  C***INTRODUCTION:
0013  C
0014  C  The subroutine "ADD Images" takes a pixel by pixel sum
0015  C  of images A and B, divides each sum by SCALE, and places the results
0016  C  into image C. SCALE must be between 1 and 9 inclusive.
0017  C  The truecolor image B is used in the processing of ADDI2, and will
0018  C  be left as the combination of C, A, and B for red, green, and blue
0019  C  respectively.
0020  C
0021  C***LANGUAGE:
0022  C
0023  C  FORTRAN 77, the HP 1000 version for RTE-6/VM.
0024  C
0025  C***LIMITATIONS:
0026  C
0027  C  The truecolor 8 image is destroyed during this operation. C is
0028  C  obviously destroyed. This subroutine is accomplished using COMTAL
0029  C  commands that exploit the pipeline processors. Because of this, the
0030  C  processing steps are obscure. For example, there is no motivation
0031  C  outside the COMTAL instructions for making the combination of C, A, and B
0032  C  a color image. Readers should be aware of these obscurities before trying to
0033  C  understand the code.
0034  C
0035  C
0036  C  The three image numbers must be distinct.
0037  C
0038  C  If the sum of any two pixels exceeds 255, the value in C is set to 255.
0039  C
0040  C  The SCALE factor must be between 1 and 9 inclusive.
0041  C
0042  C  If any of the image numbers are out of range, an error message is printed
0043  C  and no further processing takes place.
0044  C  This subroutine assumes that 0 is not a legal image for the COMTAL
configuration.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMND</td>
<td>CMND</td>
<td>CMND</td>
<td>Sends a command to the COMTAL as if the command were sent from the keyboard</td>
</tr>
<tr>
<td>RANGE</td>
<td>RANGE</td>
<td>RANGE</td>
<td>logical function that determines if the 1st parameter is within the range of the 2nd &amp; 3rd.</td>
</tr>
<tr>
<td>DIGIT</td>
<td>DIGIT</td>
<td>DIGIT</td>
<td>character*1 function which returns '0'-'9' according to integer input 0-9.</td>
</tr>
</tbody>
</table>

written by:

KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

| INTEGER | IBUF(128) | a buffer for passing commands to COMTAL |
| CHARACTER*255 | CBUF | character overlay for IBUF |
| EQUIVALENCE (IBUF,CBUF) |
| LOGICAL | RANGE | function that determines if 1st parameter is within 2nd and 3rd parameter |
| INTEGER | IMLO, IMHI | limits on COMTAL image numbers |
| INTEGER | TERM |
| CHARACTER*1 | DIGIT | character*1 function that returns '0'-'9' according to integer input 0-9. |

**INITIALIZATIONS:**

| DATA | IMLO/1, IMHI/4 |
| DATA | TERM/1 |

**PROCESSING**

| IF (.NOT.(RANGE(A,IMLO,IMHI))) GOTO 8001 | error return |
| IF (.NOT.(RANGE(B,IMLO,IMHI))) GOTO 8101 | error return |
| IF (.NOT.(RANGE(C,IMLO,IMHI))) GOTO 8201 | error return |
The following character string sends a series of Keyboard commands to the COMTAL. In the comments below, each command is explained. The notation #X where X is either A, B, or C stands for the single character that corresponds to the single digit number associated with the parameter X.

In this notation, letters in caps were entered into CBUF, and lower case letters are the full commands filled in by the COMTAL.

NOTE: this code assumes that the digit 0 is NOT a legal value for the parameters A, B, and C.

The "$" separates COMTAL commands.

CBUF =

1 'UN 1 8 $'/
2 'AS T 8 '/DIGIT(C)//''DIGIT(A)''DIGIT(B)'' $'/
3 'D 1 8 $'/
4 'SE COM G + B '/DIGIT(Scale)//'' $'/
5 'A COM $'/
6 'I '/DIGIT(C)//'' D R $'/
7 'D I '/DIGIT(C)//'' $'/
8 'SU COM '/
9 'SUB COMbine.]

CALL CMMND(IBUF, 77)
RETURN
C**ERROR RETURNS
C
8001 WRITE(TERM, 8003) A
8003 FORMAT(' THE 2ND IMAGE PARAMETER, ', I3, ', ', IS OUT OF RANGE. ')
8006 GOTO 8500
8101 WRITE(TERM, 8103) B
8103 FORMAT(' THE 3RD IMAGE PARAMETER, ', I3, ', ', IS OUT OF RANGE. ')
8501 GOTO 8500
8201 WRITE(TERM, 8203) C
8203 FORMAT(' THE 1ST IMAGE PARAMETER, ', I3, ', ', IS OUT OF RANGE. ')
8206 GOTO 8900
836 C
0137  B900 WRITE(TERM, B901)
0138  B901 FORMAT(* ADD12 RETURNS WITHOUT FURTHER PROCESSING.*)
0139          RETURN
0140  C
0141          END
0142
&BDFDS T=00004  IS  ON  CR0021: USING 00018 BLKS R=0000

0081 CCCCCCCCCCCCCcccccccccccccccccccccccccccccccccccccccccccccccccccc
0082 SUBROUTINE BDFDS(IIMAGE, FNAME)
0083 CCCCCCCcccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
0084 C
0085 C**PARAMETER DECLARATIONS:
0086 C
0087 INTEGER IMAGE ! the number of the COMTAL image plane to be
0088 C filled from the HP disk file.
0089 INTEGER FNAME(3)! the HP filename from which an image will be read.
0090 C
0091 C**INTRODUCTION:
0092 C
0093 The subroutine "Black and White From Disk" transfers one b&w image
0094 from the HP to the COMTAL. The image number and the name of the HP
0095 disk are given as parameters.
0096 C
0097 C**LANGUAGE:
0098 C
0099 FORTRAN 77, the HP 1000 version for RTE-6VM.
0100 C
0101 C**LIMITATIONS:
0102 C
0103 BUFDS only works for b&w images. Since a color image requires
0104 three separate b&w images, another subroutine, CLFDS is available
0105 for reading color images from the disk.
0106 C
0107 C**SUBPROGRAMS CALLED:
0108 C
0109 C
0110 C name source load remarks
0111 C RANGE @RANGE X RANG logical function that determines if the 1st
0112 C argument is within the 2nd and 3rd inclusive.
0113 C OPEN ------- ------ HP FORTRAN77 intrinsic subroutine; opens a file
0114 C and stores data block information in first param.
0115 C READF ------- ------ HP FORTRAN77 intrinsic subroutine; reads a record
0116 C from a file; requires an OPENed data block.
0117 C WRILN &WRILN XWRILN Writes a COMTAL image horizontal Line: 2 pixels
0118 C per integer in the buffer.
0119 C CMND &CMND XCMND sends commands to the COMTAL as if they were
0120 C typed at the COMTAL keyboard.
0121 C DIGIT &DIGIT XDIGIT a character*1 function that returns a single
0122 C ASCII digit when given an integer 0-9
0123 C
0124 C**WRITTEN BY:
The code on which this subprogram is based was written by NETTIE D. FAULKON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**Revision History:**

**Local Variables:**

- INTEGER DBLOCK(144) - a Data control BLOCK used in file ID.
- INTEGER IERR - holds HP 10 return code.
- INTEGER IBUF(256) - buffers one horizontal row of COMTAL pixels.
- INTEGER ROW - loop indexing which COMTAL row.
- LOGICAL RANGE - logical function that determines if 1st parameter is between 2nd and 3rd, inclusive.
- INTEGER IMLO, IMHI - limits on COMTAL image numbers.
- INTEGER TERM - logical unit for terminal output.
- CHARACTER*1 DIGIT - function that returns '0', '1', ..., or '9' according to a 0, 1, ..., or 9 integer input.
- CHARACTER*255 CBUF - overlays IBUF.
- EQUIVALENCE (CBUF, IBUF)

**Initialization:**

- DATA IMLO/1/, IMHI/4/ - Image limits.
- DATA TERM/1/ - Terminal output.

**Processing:**

- IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8801 - error return.
- CALL OPEN(DBLOCK, IERR, FLNAME)
- IF (IERR .LT. 0) GOTO 8201 - error return, open failed.
- CBUF = 'D I' //DIGIT(IMAGE) - Display the Image to be filled.
- CALL CMDND(IBUF, 5)
- DO 1000 ROW = 0, 511
- CALL READF(DBLOCK, IERR, IBUF)
- IF (IERR .LT. 0) GOTO 8301 - error return, bad read.
- CALL URILN(IMAGE, ROW, IBUF)
- 1000 CONTINUE
0091 RETURN
0092 C
0093**ERROR RETURNS
0094 C
0095 8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0096 8003 FORMAT(‘ IMAGE NUMBER,’13,’ IS OUT OF RANGE:’214.’’)
0097 GOTO 8900
0098 C
0099 8201 WRITE(TERM, 8203) IERR
0100 8203 FORMAT(‘ ERROR OCCURED DURING IMAGE FILE OPENING:’4,’’)
0101 GOTO 8900
0102 C
0103 8301 WRITE(TERM, 8303) IERR
0104 8303 FORMAT(‘ ERROR OCCURED DURING IMAGE FILE READ:’4,’’)
0105 GOTO 8900
0106 C
0107 8900 WRITE(TERM, 8901)
0108 8901 FORMAT(‘ BWFDS FAILS. NO TRANSFER TAKES PLACE.’)
0109 RETURN
0110 END
&CLFDS T=00004 IS ON CR0021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE CLFDS(RED, GREEN, BLUE, COLOR, FLNAME)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER RED, GREEN, BLUE ! colors of the COMTAL image planes
0008 C INTEGER COLOR ! the truecolor image to be formed and displayed.
0009 INTEGER FLNAME(3) ! the HP filename from which an image will be read.
0010 C
0011 C
0012 C**INTRODUCTION:
0013 C
0014 C The subroutine "Color image From Disk" transfers three b&w images
0015 C from the HP to the COMTAL, and then assigns these to a truecolor
0016 C image on the COMTAL. The filename names a single file holding all
0017 C three monochrome images. The COLOR image number is automatically
0018 C unassigned and assigned by CLFDS, and after all three component
0019 C images are transferred, the color image is displayed.
0020 C
0021 C**LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP 1000 version for RTE-6/VM.
0024 C
0025 C**LIMITATIONS:
0026 C
0027 C The three parameters RED, GREEN and BLUE must be distinct and
0028 C within the IMLO, IMHI range. COLOR must be within the TRLO, TRHI
0029 C range (which is mutually exclusive with IMLO-IMHI). The limits on
0030 C truecolor numbers are arbitrary. However, this subroutine enforces the
0031 C arbitrary limits. The HP image file must contain all three monochrome images
0032 C in the order RED, GREEN, and BLUE.
0033 C
0034 C**SUBPROGRAMS CALLED:
0035 C
0036 C
0037 C
0038 C name source load remarks
0039 C ------ ------ ------ ----------------------------------------------------
0040 C RANGE FORTRAN 77 logical function that determines if the 1st
0041 C argument is within the 2nd and 3rd inclusive.
0042 C OPEN FORTRAN 77 intrinsic subroutine; opens a file
0043 C and stores data block information in first param.
0044 C READF FORTRAN 77 intrinsic subroutine; reads a record
0045 C from a file; requires an OPENed data block.
0046 C WRILN FORTRAN 77 writes a COMTAL Image horizontal Line: 2 pixels
per integer in the buffer.

transfers a command string to the COMTAL, which
accepts it as a keyboard command.

character*1 function that returns a single digit
on legal integer inputs 0-9.

C***WRITTEN BY:

The code on which this subprogram is based was written by
NETTIE D. FAULCON, July, 1983. This subprogram was written by
KEITH MILLER, July, 1984, with the support of a NASA-ASEE
summer fellowship.

C***REVISION HISTORY:

C***LOCAL VARIABLES:

integer DBLOCK(144) ! a Data control BLOCK used in file IO.
integer IERR ! holds HP IO return code.
integer IBUF(256) ! buffers one horizontal row of COMTAL pixels.
integer ROW ! loop indexing which COMTAL row.
logical RANGE ! logical function that determines if 1st parameter
is between 2nd and 3rd, inclusive.
integer IMLO, IMHI ! limits on COMTAL b&w image numbers.
integer TRLO, TRHI ! limits on COMTAL truecolor image numbers.
integer TERM ! logical unit for terminal output
character*255 CBUF ! character overlay for sending COMTAL commands
equivalence (CBUF, IBUF)
character*1 DIGIT ! function that returns '0', '1', ..., or '9'
according to 0, 1, ..., or 9 integer input.

C***INITIALIZATIONS:

data IMLO/1/, IMHI/4/
data TRLO/5/, TRHI/9/
data TERM/1/

C***PROCESSING

if (.NOT.(range(red , IMLO. IMHI))) goto 8001 ! error return
if (.NOT.(range(GREEN , IMLO. IMHI))) goto 8101 ! error return
if (.NOT.(range(BLUE , IMLO. IMHI))) goto 8201 ! error return
if ((RED .EQ. GREEN) .OR. (GREEN .EQ. BLUE))
1 .OR. (RED .EQ. BLUE)) GOTO 8301 ! error return

IF (.NOT.(RANGE(COLOR, TRLO, TRHI))) GOTO 8401 ! error return

CALL OPEN(DBLOCK, IERR, FLNAME)
IF (IERR .LT. 0) GOTO 8501 ! error return, open failed

CBUF = "D I" //DIGIT(Redo) ! Display the RED Image as it is filled.
CALL CMMND(IBUF,5)
DO 1000 ROW = 0.511
    CALL READF(DBLOCK, IERR, I Buf)
    IF (IERR .LT. 0) GOTO 8601 ! error return, file read failed
    CALL WRILN(RED, ROW, I Buf)
1000 CONTINUE

CBUF = "D I" //DIGIT(Green) ! Display the GREEN Image as it is filled.
CALL CMMND(IBUF,5)
DO 2000 ROW = 0.511
    CALL READF(DBLOCK, IERR, I Buf)
    IF (IERR .LT. 0) GOTO 8701 ! error return, file read failed
    CALL WRILN(Green, ROW, I Buf)
2000 CONTINUE

CBUF = "D I" //DIGIT(Blue) ! Display the BLUE Image as it is filled.
CALL CMMND(IBUF,5)
DO 3000 ROW = 0.511
    CALL READF(DBLOCK, IERR, I Buf)
    IF (IERR .LT. 0) GOTO 8801 ! error return, file read failed
    CALL WRILN(Blue, ROW, I Buf)
3000 CONTINUE

Let #C, #R, #G, #B be the DIGIT associated with COLOR, RED, GREEN, and BLUE respectively; then the following CMMND calls read as follows: UNassign Image #C; ASsign Truecolor #C red #R green #G blue #B; Display Image #C

CBUF = 'UN I' //DIGIT(COLOR)
CALL CMMND(IBUF,6)

CBUF = 'AS T' //DIGIT(COLOR) // ' //DIGIT(RED) // ' //

1 DIGIT(Green) // ' //DIGIT(Blue)
CALL CMMND(IBUF,12)

CBUF = 'D I' //DIGIT(COLOR)
CALL CMMND(IBUF,5)
RETURN

C***ERROR RETURNS
0137 C
0138 8001 WRITE(TERM, 8003) RED, IMLO, IMHI
0139 8003 FORMAT(' RED IMAGE NUMBER.','3', 'IS OUT OF RANGE: ', '14', '.
0140 GOTO 8900
0141 C
0142 8101 WRITE(TERM, 8103) GREEN, IMLO, IMHI
0143 8103 FORMAT(' GREEN IMAGE NUMBER.', '3', 'IS OUT OF RANGE: ', '14', '.
0144 GOTO 8900
0145 C
0146 8201 WRITE(TERM, 8203) BLUE, IMLO, IMHI
0147 8203 FORMAT(' BLUE IMAGE NUMBER.', '3', 'IS OUT OF RANGE: ', '14', '.
0148 GOTO 8900
0149 C
0150 8301 WRITE(TERM, 8303) RED, GREEN, BLUE
0151 8303 FORMAT(' 3 MONOCHROME IMAGES MUST BE DISTINCT. YOURS: ', '314)
0152 GOTO 8900
0153 C
0154 8401 WRITE(TERM, 8403) COLOR, CLLO, CLHI
0155 8403 FORMAT(' YOUR TRUECOLOR IMAGE.', '4', 'IS OUT OF RANGE: ', '14)
0156 GOTO 8900
0157 C
0158 8501 WRITE(TERM, 8503) IERR
0159 8503 FORMAT(' ERROR WHILE OPENING IMAGE FILE: ', '5')
0160 GOTO 8900
0161 C
0162 8601 WRITE(TERM, 8603) IERR
0163 8603 FORMAT(' ERROR WHILE READING IN THE RED MONOCHROME IMAGE: ', '4')
0164 GOTO 8900
0165 C
0166 .8701 WRITE(TERM, 8703) IERR
0167 8703 FORMAT(' ERROR WHILE READING IN THE GREEN MONOCHROME IMAGE: ', '4')
0168 GOTO 8900
0169 C
0170 8801 WRITE(TERM, 8803) IERR
0171 8803 FORMAT(' ERROR WHILE READING IN THE BLUE MONOCHROME IMAGE: ', '4')
0172 GOTO 8900
0173 C
0174 8900 WRITE(TERM, 8901)
0175 8901 FORMAT(' CLFDS FAILS. NO TRANSFER TAKES PLACE. ')
0176 RETURN
0177 END
SUBROUTINE CLRGR( GRAPH )

INTEGER GRAPH ! the number of the COMTAL GRAPH to be cleared.

C***INTRODUCTION:
C Clear GRAPH clears the graphics designated by GRAPH. GRAPH should
C be within the range 1-4 for the present IRD COMTAL system.
C If GRAPH is out of range, an error message is printed and
C no COMTAL transfer takes place.

C***LANGUAGE:
C FORTRAN 77, the HP-1000 version for RTE-G/VM.

C***LIMITATIONS:
C GRAPH must be within the limits GRLO and GRHI explained below.

C***SUBPROGRAMS CALLED:
C RANGE &RANGE &RANGE logical function that determines if its 1st
C parameter is between (inclusive) its last parameters
C CMMD &CMMD &CMMD sends a COMTAL command as if it were typed at
C the COMTAL keyboard.
C DIGIT &DIGIT &DIGIT returns a character (1st parameter) which is the
C character equivalent of the integer 2nd parameter.

C***WRITTEN BY:
C Hettie D. FAULCON, July, 1983. This modification is by
C Keith MILLER, June, 1984.

C***REVISION HISTORY:

C***LOCAL VARIABLES:
INTEGER GRLO, GRHI! limits on COMTAL GRAPH numbers
CHARACTER*1 WHICH
LOGICAL RANGE! logical function that determines if its
! 1st parameter lies within last 2 parameters.
INTEGER TERM! logical unit number of the terminal output
INTEGER IBUF(128)! COMTAL command buffer
CHARACTER*255 CBUF! overlays the IBUF command buffer.
EQUIVALENCE (IBUF, CBUF)

C
***INITIALIZATIONS:
DATA GRLO/1/, GRHI/4/
DATA TERM/1/
C
***PROCESSING
IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001! error return
ELSE...clear the GRAPH
CALL DIGIT(WHICH, GRAPH)
CBUF = 'CLEAR GRAPH' // WHICH! since CBUF overlays IBUF,
! this statement loads IBUF with
! the COMTAL command
CALL CMMND(IBUF, 13)
RETURN
C
***ERROR RETURN
WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT(' GRAPH NUMBER, ', I4, ' IS OUT OF RANGE: ', 213, '.')
WRITE(TERM, 8005)
8005 FORMAT(' CLRGR FAILS. NO COMMAND SENT TO COMTAL.')
RETURN
END
**SUBROUTINE CLRIM (IMAGE)**

**INTEGER IMAGE ! the number of the COMTAL image to be cleared.**

**INTRODUCTION:**

Clear Image clears the image designated by IMAGE. IMAGE should be within the range 1-4 for the present IRD COMTAL system.

If IMAGE is out of range, an error message is printed and no COMTAL transfer takes place.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**LIMITATIONS:**

IMAGE must be within the limits IMLO and IMHI explained below.

**SUBPROGRAMS CALLED:**

name | source | load | remarks
----- | ------ | ----- | ---------------
\_\_\_ | \_\_\_ | \_\_\_ | ---------------
RANGE | \_\_\_ | \_\_\_ | logical function that determines if its 1st parameter is between (inclusive) its last parameter.
CMDND | \_\_\_ | \_\_\_ | sends a COMTAL command as if it were typed at the COMTAL keyboard.
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ | a character*1 function which returns the character associated with integer inputs 0,...,9.
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |

**WRITTEN BY:**

The code on which this subprogram is based was written by

\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |

**REVISION HISTORY:**

This modification is by

\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |
\_\_\_ | \_\_\_ | \_\_\_ | \_\_\_ |

**LOCAL VARIABLES:**

\_\_\_ | \_\_\_ | \_\_\_ | 
\_\_\_ | \_\_\_ | \_\_\_ | 
\_\_\_ | \_\_\_ | \_\_\_ | 
\_\_\_ | \_\_\_ | \_\_\_ | 
\_\_\_ | \_\_\_ | \_\_\_ | 

28
INTEGER IMLO, IMHI ! limits on COMTAL image numbers
LOGICAL RANGE ! logical function that determines if its
! 1st parameter lies within last 2 parameters.
INTEGER TERM ! logical unit number of the terminal output
INTEGER IBUF(128) ! COMTAL command buffer
CHARACTER*1 DIGIT ! function that returns the ASCII character
! associated with integer input, 0,...,9.
CHARACTER*255 CBUF ! overlays the IBUF command buffer
EQUIVALENCE (IBUF, CBUF)

C***INITIALIZATIONS:
DATA IMLO/I/ IMHI/4/
DATA TERM/I/

C***PROCESSING
IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8801 ! error return
ELSE...clear the image
  CBUF = 'CLEAR IMAGE ' // DIGIT(IMAGE) ! since CBUF overlays IBUF,
  ! this statement loads IBUF with
  ! the COMTAL command
  CALL CMDND(IBUF, 13)
RETURN

C***ERROR RETURN
WRITE(TERM, 8003) IMAGE, IMLO, IMHI
8003 FORMAT(' IMAGE NUMBER, ', 14, ' IS OUT OF RANGE: ', 213, '.')
WRITE(TERM, 8005)
8005 FORMAT(' CLRIM FAILS. NO COMMAND SENT TO COMTAL.')
RETURN
END
SUBROUTINE CMMN2(INBUF)

CHARACTER*(*) INBUF ! the characters of a COMTAL command string

This subroutine "CoMMaNd 2" allows a character string command to be sent to the COMTAL much as if the command were typed at the keyboard.

CMMN2 is designed to be sent constant strings.

The major differences are that the INBUF command string may include multiple commands, each separated by the character "$".

A character array buffer is used in equivalence with an integer array in this subroutine to illustrate the utility of the characters and still allow obvious compatibility.

The subroutine CMMND is very similar, only there an integer buffer of fixed size is used with an extra parameter to identify how many characters are valid parts of the intended command.

FORTRAN 77, the HP-1000 version for RTE-6/VM.

In order to make it easier to send single COMTAL commands to CMMND, the final blank, $, and the required zero byte are added automatically to the INBUF string.

If the last character isn't a blank, CMMND adds one. However, the caller should NOT add the final $ or zero byte to the string. Note that each $ in the string should be preceded by a blank.

WARNING: When a command is sent to the COMTAL that generates a COMTAL error, the COMTAL system is frozen until a manual <ESC> (or perhaps several) is entered from the COMTAL keyboard.

name source load remarks
****LOCAL VARIABLES:**

- `CHARACTER*1 CBUF(256)` | character buffer
- `INTEGER IBUF(128)` | the character buffer overlayed as integers
- `INTEGER IZERO` | constant value 0 for making a 0 byte (ZERO)
- `INTEGER WORDS` | counts number of words
- `INTEGER NUMCHR` | counts number of bytes
- `INTEGER LEN` | intrinsic HP FORTRAN 77 function that returns the length of a character string.
- `CHARACTER*1 BLANK, DOLLAR` | special ASCII characters
- `CHARACTER*1 ZERO` | zero is 00000000 binary.
- `EQUIVALENCE (CBUF, IBUF), (ZERO, IZERO)`

**INITIALIZATIONS:**

- `DATA BLANK/' '/, DOLLAR/'$'/, IZERO/'0'`
- `NUMCHR = LEN(INBUF)`

**PROCESSING:**

- `DO 10 INDEX = 1, NUMCHR
CBUF(INDEX) = INBUF(INDEX:INDEX)
10 CONTINUE`
- `IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30
C... ELSE...
CNUMCHR = NUMCHR + 1
CBUF(NUMCHR) = BLANK`
- `C... ADD ENDING CHARACTERS TO COMMAND
C30 CBUF(NUMCHR+1) = DOLLAR
CBUF(NUMCHR+2) = ZERO`
CBUF(NUMCHR+3) = ZERO ! safety precaution
NUMCHR = NUMCHR + 2

CHANGE BYTE COUNT TO WORD COUNT

WORDS = (NUMCHR+1) / 2 ! if N is even, intentional truncation

Programming note:
The EXEC command parameters are discussed in the HP RTE-G/VM Programmer's Reference Manual, 2-19 ff. The COMTAL parameters are discussed in section 5.2.4 of the COMTAL User's Manual.

The first parameter to EXEC identifies it as a write command.
The second parameter identifies the resident HP driver (36B) and gives a code for the operation required by this call (500B).
The third parameter is the command string, and the fourth gives the length in words of the buffer that is to be used. The fifth parameter is a code for the COMTAL interface that directs the command transfer.

CALL EXEC(2, 36B + 500B, IBUF. WORDS, 24001B)
RETURN
END
&CMMND T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE CMMND(INBUFo, INCNT)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETERS:
0006 C
0007 INTEGER INBUF(129) ! the characters of a COMTAL command string
0008 INTEGER INCNT ! the number of characters in command string
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C This subroutine "CMMaND" allows a character string command to be
0013 C sent to the COMTAL much as if the command were typed at the keyboard.
0014 C The major differences are that the INBUF command string may include
0015 C multiple commands, each separated by the character "$". Notice
0016 C that NUMCHR is in terms of characters (bytes), not words. INBUF
0017 C is an integer array to be compatible with previously written software.
0018 C but an F77 character array would probably be more appropriate.
0019 C A character array buffer is used in equivalence with an integer array
0020 C in this subroutine to illustrate the utility of the characters and
0021 C still allow obvious compatibility.
0022 C
0023 C***LANGUAGE:
0024 C
0025 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0026 C
0027 C***LIMITATIONS:
0028 C
0029 C In order to make it easier to send single COMTAL commands to CMMND,
0030 C the final blank, $, and the required zero byte are added automatically
0031 C to the INBUF string.
0032 C If the last character isn't a blank, CMMND adds one.
0033 C However, the caller should NOT add the final $ or zero byte to the
0034 C string. Note that each $ in the string should be preceded by a blank.
0035 C
0036 C WARNING: When a command is sent to the COMTAL that generates a
0037 C COMTAL error, the COMTAL system is frozen until a manual
0038 C <ESC> (or perhaps several) is entered from the COMTAL keyboard.
0039 C
0040 C***SUBPROGRAMS CALLED: NONE.
0041 C
0042 C
0043 C***WRITTEN BY:
0044 C
The original code upon which this subroutine is based was written by NETTIE D. FAULCON, JULY, 1983.

C***REVISION HISTORY:
MODIFIED BY KEITH MILLER, 6/18/84.

C***LOCAL VARIABLES:

CHARACTER*1 CBUF(256) ! character buffer
INTEGER IBUF(128) ! the character buffer overlayed as integers
INTEGER ZERO ! constant value 0 for making a 0 byte (ZERO)
INTEGER NUMCHR ! counts number of bytes
INTEGER WORDS ! counts number of words
INTEGER BLANK, DOLLAR ! special ASCII characters
CHARACTER*1 ZERO ! zero is 00000000 binary.
CHARACTER*1 ZERO ! blank is 00000000 binary.
CHARACTER*1 ZERO ! safety precaution
EQUIVALENCE (CBUF, IBUF), (ZERO, ZERO)

C***INITIALIZATIONS:

DATA BLANK/','/$, ZERO/!
NUMCHR = INCNT ! protects the input parameter, since NUMCHR is reassigned in the subroutine.

C***PROCESSING:

WORDS = (NUMCHR+1) / 2
DO 10 INDEX = 1, WORDS
   IBUF(INDEX) = INBUF(INDEX)
10 CONTINUE
IF (CBUF(NUMCHR) .EQ. BLANK) GOTO 30
ELSE...
   NUMCHR = NUMCHR + 1
   CBUF(NUMCHR) = BLANK
ADD ENDING CHARACTERS TO COMMAND
CBUF(NUMCHR+1) = DOLLAR
CBUF(NUMCHR+2) = ZERO
CBUF(NUMCHR+3) = ZERO ! safety precaution
NUMCHR = NUMCHR + 2
CHANGE BYTE COUNT TO WORD COUNT
0091  
0092  
0093  
0094  
0095  
0096  
0097  
0098  
0099  
0100  
0101  
0102  
0103  
0104  
0105  
0106  
0107  
0108  
0109  

`WORDS = (NUMCHR+1) / 2` if N is even, intentional truncation

Programming note:
The EXEC command parameters are discussed in the HP RTE-6/VM Programmer's Reference Manual, 2-19 ff. The COMTAL parameters are discussed in section 5.2.4 of the COMTAL User's Manual.

The first parameter to EXEC identifies it as a write command.
The second parameter identifies the resident HP driver (36B) and gives a code for the operation required by this call (5008).
The third parameter is the command string, and the fourth gives the length in words of the buffer that is to be used. The fifth parameter is a code for the COMTAL interface that directs the command transfer.

CALL EXEC(2, 36B + 5008, IBUF, WORDS, 240018)
RETURN
END
Subroutine COUNT examines each pixel value in the COMTAL image associated with the number IMAGE, and compiles a count of how many pixels hold the values 0-255. These 256 counts are returned in the INTEGER*4 array COUNTS.

Language:
FORTRAN 77, the HP-1000 version for RTE-6/VM.

Limitations:
Note that COUNTS is an INTEGER*4 array. The 32,767 limit for INTEGER*2 is not sufficient, since there are over 250,000 pixels in a 512 X 512 COMTAL image. COUNT does no scaling or graphing.

Subprograms Called:
- RDIL2 & RDIL2
- RANGE
- XRDIL2 reads a horizontal line of pixels from a COMTAL image
- RANGE logical function that determines if its 1st parameter is within the 2nd and 3rd parameters.

Written By:
The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

Revision History:
LOCAL VARIABLES:

LOGICAL RANGE ! logical function that determines if its 1st parameter is within the 2nd and 3rd parameters.

INTEGER INDEX ! loop index variable

INTEGER LBUF(512) ! Line Buffer holds a row of pixels.

INTEGER WHICH ! changes from pixel value to count array index

INTEGER IMLO, IMHI ! limits on COMTAL image numbers

INTEGER ROW, COL ! loop indices

INTEGER TERM ! logical unit for terminal output

INITIALIZATIONS:

DATA IMLO/1/, IMHI/4/

DATA TERM/1/

C

PROCESSING

IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return

initialize COUNTS to 0

DO 1000 INDEX = 1, 256

COUNTS(INDEX) = 0

CONTINUE

collect counts

DO 3000 ROW = 0, 511

CALL RDIL2(LBUF, IMAGE, ROW)

DO 2000 COL = 1, 512

WHICH = LBUF(COL) + 1 ! "+1" required because pixel values are 0-255, COUNTS array is indexed 1-256.

COUNTS(WHICH) = COUNTS(WHICH) + 1

CONTINUE

ERROR RETURN

WRITE(TERM, 8003) IMAGE, IMLO, IMHI

8003 FORMAT(  ' IMAGE NUMBER:', 14,  ' OUT OF RANGE:', 214)

WRITE(TERM, 8901)
0091 8901 FORMAT(' SUBROUTINE COUNT FAILS. COUNTS ARRAY NOT CHANGED.')
0092       RETURN
0093       END
&DELAY T=00004 IS ON CR00021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DELAY(SECOND)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER SECOND ! the number of seconds to delay, >= 0.
0008 C
0009 C***INTRODUCTION:
0010 C
0011 C This routine, DELAY, makes the HP busy wait for at least the
0012 C seconds given in the input parameters.
0013 C
0014 C***LANGUAGE:
0015 C
0016 C FORTRAN 77, the HP-1000 version for RTE-6.0VM.
0017 C
0018 C***LIMITATIONS:
0019 C
0020 C There is no claim that the timing here is exact. However, the HP
0021 C busy waits for AT LEAST the time required by the input parameter.
0022 C The HP EXEC for time request gives tens of milliseconds, but this
0023 C procedure uses the simpler seconds measure.
0024 C
0025 C The procedure will not work properly when the Julian calendar spins
0026 C over to 0.0.0.0.0.0.
0027 C
0028 C***SUBPROGRAMS CALLED:
0029 C
0030 C None.
0031 C
0032 C***WRITTEN BY:
0033 C
0034 C The code on which this subprogram is based was written by
0035 C NETTIE D. FAULKNER, July, 1983. This subprogram was written by
0036 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0037 C summer fellowship.
0038 C
0039 C***REVISION HISTORY:
0040 C
0041 C
0042 C***LOCAL VARIABLES:
0043 C
0044 INTEGER INTIME(6)! the milliseconds, seconds, minutes, hours.
INTEGER*4 INCNT ! and Julian day when procedure entered.
INTEGER NOW(6) ! the INTIME in units of tens of milliseconds
                        ! from 0.0.0.0.0.
INTEGER NOWCNT ! the milliseconds, seconds, minutes, hours,
                        ! and Julian day of the latest EXEC call that
INTEGER*4 NOWCNT ! NOW in units of tens of milliseconds from
                        ! 0.0.0.0.0.
INTEGER TIMREQ ! determines the time.
INTEGER TERM ! the EXEC number for a time request.
                        ! logical unit for terminal output.

C**INITIALIZATIONS:
DATA TIMREQ/,1/
DATA TERM/,1/

C**PROCESSING:
IF (SECOND .LE. 0) GOTO 8001 ! error return

CALL EXEC(TIMREQ, INTIME)
INCNT = INTIME(1) + 100*INTIME(2) + 6000*INTIME(3)
                  + 360000*INTIME(4) + 360000*365*INTIME(5)

1000 CALL EXEC(TIMREQ, NOW)
NOWCNT = NOW(1) + 100*NOW(2) + 6000*NOW(3)
                  + 360000*NOW(4) + 360000*365*NOW(5)

IF ((NOWCNT-INCNT) .LT. (SECOND*100)) GOTO 1000

RETURN

C**ERROR RETURN:
8001 WRITE(TERM,8003) SECOND
8003 FORMAT(' THE SECONDS COUNT. ',IS, '.', IS <= 0. ')
8006 WRITE(TERM,8901)
8901 FORMAT(' DELAY FAILS. NO TIMED DELAY OCCURS. ')
RETURN
END
&DIGIT T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CHARACTER FUNCTION DIGIT( INTIN )
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER INTIN  ! INTEGER Input parameter, converted to a digit.
0008 C
0009 C**INTRODUCTION:
0010 C
0011 C DIGITS converts an integer between 0 and 9 into a single
0012 C character digit. If the INTIN parameter is out of range,
0013 C an error message is printed at the terminal and DIGIT is
0014 C set to a blank.
0015 C
0016 C**LANGUAGE:
0017 C
0018 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0019 C
0020 C**LIMITATIONS:
0021 C
0022 C The INTIN parameter must be between 0 and 9.
0023 C
0024 C**SUBPROGRAMS CALLED:
0025 C
0026 C name source load remarks
0027 C -------- -------- -------------------------------
0028 C RANGE / &RANGE XRANGE logical function determines if the first parameter
0029 C lies within (inclusive) the next two parameters.
0030 C
0031 C**WRITTEN BY:
0032 C
0034 C
0035 C**REVISION HISTORY:
0036 C
0037 C
0038 C**LOCAL VARIABLES:
0039 C
0040 C CHARACTER*1 DGTARA(10)! DIGIT Array holds the digits '0'-'9'
0041 C LOGICAL RANGE  ! function that determines if its first parameter
0042 C INTEGER TERM   ! is between (inclusive) its last two parameters.
0043 C
0044 C

41
C***INITIALIZATIONS:
DATA DGTARA/'0','1','2','3','4','5','6','7','8','9'/
DATA TERM/1/

C***PROCESSING

IF (.NOT.(RANGE(INLIN, 0, 9))) GO TO 6001 ! error return
ELSE... convert to digit and return
   DIGIT = DGTARA(INLIN + 1)
RETURN

C

WRITE(TERM, 6003) INLIN
6003 FORMAT('THE INPUT TO DIGIT IS NOT A SINGLE DIGIT. DIGIT RETURNS A BLANK.')
6001 DIGIT = ''
RETURN
END
&DSPBW T=00004 IS ON CR0021 USING 00002 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DSPBW(IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! a number 1-4 designating a COMTAL image
0008 C
0009 C***INTRODUCTION:
0010 C
0011 C "Display Black and White" is a subroutine that allows the caller
0012 C to send a display command to the COMTAL from an HP program.
0013 C The call can turn on one black and white image, number 1, 2, 3,
0014 C or 4. Any previous pseudocolor or function memory commands are
0015 C nullified by a DSPBW call.
0016 C
0017 C The resident driver DVR41 is called via an EXEC to accomplish the
0018 C display.
0019 C
0020 C***LIMITATIONS:
0021 C
0022 C This subroutine does NOT display truecolor images. That is
0023 C accomplished using the subroutine DSPCL.
0024 C
0025 C***SUBPROGRAMS CALLED:
0026 C
0027 C name source load remarks
0028 ------ ------ ------ -----------------------------------------------
0029 C RANGE &RANGE XRANGE logical function that determines if the
0030 C first parameter is within the bounds defined
0031 C by the second and third parameter (inclusive).
0032 C
0033 C***WRITTEN BY:
0034 C
0035 C The code on which this subprogram is based was written by
0036 C KEITH MILLER, June, 1984.
0037 C
0038 C
0039 C***REVISION HISTORY:
0040 C
0041 C
0042 C***LOCAL VARIABLES:
0043 C
0044 C LOGICAL RANGE ! a function for determining if an integer is
0045 C * within a certain range
0046 C INTEGER TERM ! the logical unit number for the terminal
0047 C INTEGER IDUMMY! fills the place of an unused EXEC parameter
0048 C
0049 C***INITIALIZATIONS:
0050 C
0051 C DATA TERM/1/
0052 C
0053 C***PROCESSING
0054 C
0055 C IF (RANGE(IMAGE,1,4)) GOTO 2000 ! legal image number
0056 C ELSE... illegal image number
0057 C WRITE(TERM, 1001) IMAGE
0058 1001 FORMAT(' The image number ', 13, ' is out of range.‘)
0059 C WRITE(TERM, 1002)
0060 1002 FORMAT(' DSPBW fails. No action taken on command.‘)
0061 C RETURN
0062 C
0063 C 2000 CONTINUE ! send a display command to COMTAL
0064 C
0065 C In the following call, the first parameter indicates a write
0066 C operation. The second parameter is a combination of two codes:
0067 C 000B + 36B. 36B indicates the proper resident driver, and 000B
0068 C informs the driver (DRV41) that we require a display operation.
0069 C The third and fourth parameters are ignored. The final parameter
0070 C indicates the image to be displayed. The subtraction in that
0071 C final parameter is necessary because the COMTAL images are numbered
0072 C 0 to 3; the multiplication is necessary to push the image number
0073 C into the proper bits in the command word sent to the COMTAL.
0074 C
0075 C CALL EXEC(2, 000B + 36B, IDUMMY, 0, (IMAGE-1) * 2)
0076 C RETURN
0077 C END
C Changements permettant l'utilisation de l'image de couleur pour la correspondance des cartes et des fichiers d'images de la COMTAL sur le terminal COMTAL.

**INSTRUCTIONS:**

1. **INTRODUCTION:**
   - Display CoLor commands the COMTAL to display one RGB true color image. Any previous function or pseudocolor memory commands are nullified. See subroutine DSPCL to display black and white images.
   - The color display is accomplished via the CMMND subroutine, which allows COMTAL commands to be sent to the COMTAL as if they were typed on the COMTAL keyboard.

2. **LANGUAGE:**
   - FORTRAN 77, the HP-100 version for RTE-6/VM.

3. **LIMITATIONS:**
   - The three parameters RED, GREEN, and BLUE must be distinct and within the IMLO, IMHI range. TCLR must be within the TRLO, TRHI range (which is mutually exclusive with IMLO-IMHI). The limits on truecolor numbers are arbitrary. However, this subroutine enforces the arbitrary limits. The HP image file must contain all three monochrome images in the order RED, GREEN, and BLUE.

4. **SUBPROGRAMS CALLED:**
   - name | source | load | remarks
   - RANGE | &RANGE | XRANGE | logical function that determines if the 1st argument is within the 2nd and 3rd inclusive.
   - CMMND | &CMMND | XCMMND | transfers a command string to the COMTAL, which accepts it almost as a keyboard command.
C DIGIT &DIGIT &DIGIT character1 function that returns a single digit
on legal integer inputs 0-9.

C

C***WRITTEN BY:

C

C The code on which this subprogram is based was written by

C NETTIE D. FAULKON, July, 1983. This subprogram was written by

C KEITH MILLER, July, 1984, with the support of a NASA-ASEE

C summer fellowship.

C

C***REVISION HISTORY:

C

C

C

C

C***LOCAL VARIABLES:

C

C

INTEGER IBUF(256) ! buffers COMTAL command

LOGICAL RANGE ! logical function that determines if 1st parameter

is between 2nd and 3rd, inclusive.

INTEGER IMLO. IMHI ! limits on COMTAL b&w image numbers.

INTEGER TRLO. TRHI ! limits on COMTAL truecolor image numbers.

INTEGER TERM ! logical unit for terminal output

CHARACTER*255 CBUF ! character overlay for sending COMTAL commands

EQUIVALENCE (CBUF,IBUF)

CHARACTER*1 DIGIT ! function that returns '0', '1',... or '9'

according to 0, 1,... or 9 integer input.

C

C***INITIALIZATIONS:

C

DATA IMLO/1/, IMHI/4/

DATA TRLO/5/, TRHI/5/

DATA TERM/1/

C

C***PROCESSING

C

IF (.NOT.(RANGE(RD ,IMLO,IMHI))) GOTO 8001 ! error return

IF (.NOT.(RANGE(GREEN ,IMLO,IMHI))) GOTO 8101 ! error return

IF (.NOT.(RANGE(BLUE ,IMLO,IMHI))) GOTO 8201 ! error return

IF (.NOT.(RANGE(GREEN) .OR. (RED, .EQ. GREEN)) .OR. (RED .EQ. BLUE)

GOTO 8301 ! error return

IF (.NOT.(RANGE(TCLR, TRLO,TRHI))) GOTO 8401 ! error return

DISPLAY THE COLOR IMAGE

The following commands are abbreviations of the following
COMTAL commands, where #R, #G, #B, and #C stands for the single
corresponding color.

UNassign Image #C
Assign Truecolor #C red #R green #G blue #B
Display Image #C

CBUF = "UN I" //DIGIT(TCLR)
CALL CMD(INBUF,6)
CBUF = "AS T" //DIGIT(TCLR) // "DIGIT(RED)" // "DIGIT(BLUE)
I DIGIT(GREEN)" // "DIGIT(BLUE)
CALL CMD(INBUF,12)
CBUF = "D I" //DIGIT(TCLR)
CALL CMD(INBUF,5)
RETURN

**ERROR RETURNS**

WRITE(TERM, 8003) RED, IMLO, IMHI
WRITE(TERM, 8103) GREEN, IMLO, IMHI
WRITE(TERM, 8203) BLUE, IMLO, IMHI
WRITE(TERM, 8303) RED, GREEN, BLUE
WRITE(TERM, 8403) TCLR, CLLO, CLHI
WRITE(TERM, 8900)

FORMAT(" RED IMAGE NUMBER. \( I3 \) IS OUT OF RANGE: \( .214 \) .")
FORMAT(" GREEN IMAGE NUMBER. \( I3 \) IS OUT OF RANGE: \( .214 \) .")
FORMAT(" BLUE IMAGE NUMBER. \( I3 \) IS OUT OF RANGE: \( .214 \) .")
FORMAT(" 3 MONOCHROME IMAGES MUST BE DISTINCT. YOURS: \( .314 \)")
FORMAT(" YOUR TRUECOLOR IMAGE \( .14 \) IS OUT OF RANGE: \( .214 \)")
FORMAT(" DSPCL FAILS. NO DISPLAY TAKES PLACE.")
RETURN

END
*DSPGR T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE DSPGR(GRNM)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C****PARAMETER DECLARATIONS:
0006 C
0007 C    INTEGER GRNM ! a number 1-4 designating a COMTAL graphics plane.
0008 C
0009 C****INTRODUCTION:
0010 C
0011 C    "Display Graphics" is a subroutine that allows the caller
0012 C    to send a display command to the COMTAL from an HP program.
0013 C    The call can turn on one graphics plane, number 1, 2, 3,
0014 C    or 4.
0015 C
0016 C****LIMITATIONS:
0017 C
0018 C    This subroutine does not turn off previous graphics planes.
0019 C    That is accomplished using the subroutine WIPGR.
0020 C    It is OK to call DSPGR repeatedly without an intervening WIPGR.
0021 C    The extra calls have no effect, but they don't hang up the COMTAL.
0022 C
0023 C****SUBPROGRAMS CALLED:
0024 C
0025 C    name   source load   remarks
0026 C    -----  ------  -----
0027 C    RANGE &RANGE %RANGE logical function that determines if the
0028 C    first parameter is within the bounds defined
0029 C    by the second and third parameter (inclusive).
0030 C    CMHN2 &CMHN2 %CMHN2 sends a constant string to the COMTAL as if
0031 C    the string were typed on the COMTAL keyboard.
0032 C    DIGIT &DIGIT %DIGIT character*1 function that returns '0'..'9'
0033 C    according to integer input 0..9.
0034 C
0035 C****WRITTEN BY:
0036 C
0037 C    The code on which this subprogram is based was written by
0038 C    NETTIE D. FAULCON, July, 1983. This modification is by
0039 C    KEITH MILLER, June, 1984.
0040 C
0041 C****REVISION HISTORY:
0042 C
0043 C
0044 C****LOCAL VARIABLES:
0045 C LOGICAL RANGE ! a function for determining if an integer is
0047 C | within a certain range.
0048 C INTEGER TERM | the logical unit number for the terminal.
0049 C INTEGER IDUMTY | fills the place of an unused EXEC parameter.
0050 C INTEGER GRLO,GRHI | limits on a graphics plane number.
0051 C CHARACTER*1 DIGIT | function that returns '0'..'9' for input
0052 C | integers 0..9.
0053 C
0054 C***INITIALIZATIONS:
0055 C
0056 C DATA TERM/1/
0057 C DATA GRLO/1/,GRHI/4/
0058 C
0059 C***PROCESSING
0060 C
0061 C IF (.NOT.(RANGE(GRNUM,GRLO,GRHI))) GOTO 8001 ! error return
0062 C
0063 C "ADD Graphics *GRNUM", where *GRNUM stands for the digit
0064 C | corresponding to GRNUM value.
0065 C
0066 C CALL CMMN2("ADD GR '/DIGIT(GRNUM)")
0067 C RETURN
0068 C
0069 C***ERROR RETURN:
0070 C
0071 8001 WRITE(TERM, 8003) GRNUM, GRLO, GRHI
0072 8003 FORMAT( ' THE GRAPHICS NUMBER,'..13..', OUT OF RANGE:',..214)
0073 C
0074 8900 WRITE(TERM, 8901)
0075 8901 FORMAT( ' DSPGR fails. No action taken on command.' )
0076 C RETURN
0077 END
SUBROUTINE DSPVD

C****INPUT PARAMETERS:
None.

C****INTRODUCTION:
"Display Video" commands the COMTAL to display the video image which, by arbitrary convention, is always assigned to COMTAL image #5. The video must be set to image 5 previous to this call.

C****LANGUAGE:
FORTRAN 77, the HP-100 version for RTE-6/VM.

C****LIMITATIONS:
The COMTAL image #5 must have been set to video before DSPVD is called.

C****SUBPROGRAMS CALLED:

C****WRITTEN BY:
The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

C****REVISION HISTORY:

C****LOCAL VARIABLES:
INTEGER IBUF(256) ! buffers COMTAL command
0046 CHARACTER*255 CBUF ! character overlay for sending COMTAL commands
0047 EQUIVALENCE (CBUF, IBUF)
0048 INTEGER TVIMAG ! COMTAL image & for video camera
0049 CHARACTER*1 DIGIT ! function that returns '0'- '9' according
0050 C ! to integer input 0-9.
0051 C
0052 C**INITIALIZATIONS:
0053 C
0054 C DATA TVIMAG/5/ ! arbitrary choice.
0055 C
0056 C**PROCESSING
0057 C
0058 C DISPLAY THE VIDEO IMAGE
0059 C
0060 CBUF = 'DISPLAY IMAGE '//DIGIT(TVIMAG)// '
0061 C CALL CMND(IBUF,16)
0062 C
0063 C RETURN
0064 END
T=00004 IS ON CR00021 USING 00004 BLKS R=0000

**SUBROUTINE HILO(HI, LO, IMAGE)**

**PARAMETER DECLARATIONS:**

INTEGER HI, LO  ! output parameters, the high and low pixel values
INTEGER IMAGE  ! the COMTAL image number of the image that is to
                ! be searched for its high and low pixel values.

**INTRODUCTION:**

The subroutine "High and Low values #2" reads through an image and determines the highest and lowest pixel values, returning the values found. HILO scans the entire image.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**LIMITATIONS:**

As noted above, this subroutine passes through an entire image, pixel by pixel. In some applications you may want to combine other processing during that pass, but this subroutine won't let you do that.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGE</td>
<td>RANGE</td>
<td>XRANGE</td>
<td>logical function that determines if 1st argument is within 2nd &amp; 3rd inclusive.</td>
</tr>
<tr>
<td>RDIL2</td>
<td>GRDIL2</td>
<td>%RDIL2</td>
<td>reads a horizontal line of pixel values into a 512 integer array.</td>
</tr>
</tbody>
</table>

**WRITTEN BY:**

The code on which this subprogram is based was written by

NETTIE D. FAULCON, July, 1983. This subprogram was written by

KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**
**LOCAL VARIABLES:**

```plaintext
INTEGER IBUF(512) ! buffer for a horizontal row of COMTAL pixel values
INTEGER PXLO, PXHI! pixel value limits (for 8 bits, 0-255)
INTEGER ROW, COL ! indexes into the COMTAL image
INTEGER LNLLO, LNHII! limits on COMTAL image line dimension numbers
INTEGER ARALO, ARAHI! limits on buffer array dimension
INTEGER IMLO, IMHI! limits on COMTAL image numbers.
LOGICAL RANGE ! function that determines if 1st argument is within
               ! 2nd & 3rd arguments inclusive.
```

**INITIALIZATIONS**

```plaintext
DATA IMLO/1/, IMHI/4/
DATA PXLO/0/, PXHI/255/
DATA LNLLO/0/, LNHII/511/
DATA ARALO/1/, ARAHI/512/
```

**PROCESSING**

```plaintext
IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8801 ! error return

HI = PXLO ! artificially low
LO = PXHI ! artificially high

DO 2000 ROW = LNLLO, LNHII
    CALL RDIL2(IBUF, IMAGE, ROW)
    DO 1000 COL = ARALO, ARAHI
        IF (IBUF(COL) .GT. HI).HI = IBUF(COL)
        IF (IBUF(COL) .LT. LO).LO = IBUF(COL)
    1000 CONTINUE
2000 CONTINUE

RETURN
```

**FORMAT**

```plaintext
8801 WRITE(TERM,8803)IMAGE ,IMLO,IMHI
8803 FORMAT(3,' IMAGE NUMBER,'.15,' IS OUT OF RANGE:''.215,'')
GOTO 8900
8900 WRITE(TERM,8901)
8901 FORMAT(3,' HILO FAILS. HI AND LO PARAMETERS UNCHANGED.'
RETURN
END
```
SUBROUTINE HISTO(IMAGE)

INTEGER IMAGE ! the COMTAL image number to take the histogram of.

CC

**INTRODUCTION:**

The subroutine HISTOgram uses the COMTAL "function memory" (a look-up table) to construct a histogram of the given image, which is held IN A SCALED VERSION in the function memory associated with IMAGE.

This histogram is displayed by HISTO.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**LIMITATIONS:**

The function memory associated with image number IMAGE is destroyed. IMAGE must be within the boundaries for COMTAL image memories, or an error message is given and HTABLE is unchanged.

Notice that the histogram is scaled so that the largest value reaches to the top of the screen when displayed. Thus, no absolute counts can be easily deduced from the function memory values. See the subroutine COUNT if absolute pixel value counts are desired.

HISTO puts the HP into a busy wait while the COMTAL determines the histogram. Otherwise, the COMTAL ignores subsequent CMMN2 commands.

**SUBPROGRAMS CALLED:**

CMMN2 &CMNN2 XCMNN2 sends a constant string to the COMTAL, which treats it as a command typed on the COMTAL.

DELAY &DELAY XDELAY puts the HP into a busy wait for at least the given number of seconds.

DIGIT &DIGIT XDIGIT character*1 function that returns '0'-'9' according to integer input 0-9.

RANGE &RANGE XRANGE logical function that determines if its 1st
argument lies within its 2nd and 3rd, inclusive.

**WRITTEN BY:**

- The code on which this subprogram is based was written by Nettie D. Faulcon, July, 1983. This subprogram was written by Keith Miller, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

- CHARACTER*1 IMCHAR: the single digit character corresponding to input parameter IMAGE.
- CHARACTER*1 DIGIT: function that returns '0'-'9' for integer input 0-9.
- LOGICAL RANGE: function that determines if 1st argument is within 2nd and 3rd, inclusive.
- INTEGER TERM: logical unit for terminal output.
- INTEGER IMLO, IMHI: limits for IMAGE number.

**INITIALIZATIONS:**

- DATA 'IMLO/4, IMHI/4'
- DATA 'TERM/4'

**PROCESSING:**

- IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
- IMCHAR = DIGIT(IMAGE) ! delay initialization until after IMAGE has been found to be within its limits.

The following COMTAL command expands to:

- Function memory #I = Histogram of image where #I is the single digit associated with IMAGE.
- Image #I is automatically used for the histogram.

- CALL CMMN2('F '/IMCHAR/' H')

While the COMTAL compiles the histogram, it ignores all HP commands; thus, we pause until the histogram is found.

- CALL DELAY(15) ! 14 seconds experimentally determined as the
0091 C     ! time it takes to compile a histogram.
0092 CALL CMN2('D F '/IMCHAR)! "Display Function memory #1"
0093 RETURN
0094 C
0095 C*****ERROR RETURNS:
0096 C
0097 C
0098 8001 WRITE(TERM, 8003) IMAGE , IMLO, IMHI
0099 8003 FORMAT(' THE IMAGE NUMBER, ', 15, ', IS OUT OF RANGE: ', 215, ' . ')
0100 8900 WRITE(TERM, 8901)
0101 8901 FORMAT(' HISTO FAILS. NO HISTOGRAM TABLE ASSIGNMENT. ')
0102 RETURN
0103 END
&ICOPY T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ICOPY(OUTIMG, INIMG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***INPUT PARAMETERS:
0006 C
0007 INTEGER OUTIMG ! COMTAL image number for the destination
0008 INTEGER INIMG ! COMTAL image number for the source
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C "Image COPY" command the COMTAL to copy one black and
0013 C white image into another. The OUTIMG destination image
0014 C is, of course, wiped out by this exchange. The input
0015 C image for ICOPY and the output image must be associated with
0016 C a COMTAL image memory plane (1-4 currently).
0017 C The companion subroutine ICPY2 requires that the OUTIMG
0018 C be a COMTAL image memory plane, but the INIMG can be in the range
0019 C 1-9.
0020 C
0021 C***LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP-100 version for RTE-6/VM.
0024 C
0025 C***LIMITATIONS:
0026 C
0027 C ICOPY only works for the grey level images of COMTAL, not the
0028 C truecolor images.
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 ------- ------- ------- -----------------------------------------------
0034 C CMND &CMND %CMND this subroutine takes an INTEGER array which
0035 C contains a COMTAL command string, and transfers
0036 C the command to COMTAL. The second parameter
0037 C gives the character count of the command string.
0038 C RANGE &RANGE %RANGE this logical function determines if its first
0039 C argument is within the bounds formed by its
0040 C last 2 arguments, inclusive.
0041 C DIGIT &DIGIT %DIGIT character*1 function that returns a single digit
0042 C '0'-'9' according to integer input 0-9.
0043 C
0044 C***WRITTEN BY:
The code on which this subroutine is based was written by NETTIE D. FAULCON in July, 1983. This modification is by KEITH MILLER June, 1984.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

```plaintext
LOGICAL RANGE  ! function determines if 1st argument is within 2nd and 3rd argument inclusive.
CHARACTER*1 DIGIT  ! function returns '0'-'9' according to integer input 0-9.
CHARACTER*255 CCOMM  ! character buffer for building up a call to the CMMND subroutine.
INTEGER IBUF (128)  ! integer overlay of CCOMM
EQUIVALENCE (CCOMM, IBUF)
INTEGER IMLO, IMHI  ! the range of legal COMTAL image numbers
INTEGER TERM  ! terminal logical unit
```

**INITIALIZATION:**

```plaintext
DATA IMLO/1/, IMHI/4/
DATA TERM/1/
```

**PROCESSING:**

```plaintext
IF (.NOT.(RANGE(OUTIMG, IMLO, IMHI))) GOTO 0001  ! error return
IF (.NOT.(RANGE(INIMG, IMLO, IMHI))) GOTO 0101  ! error return

Legal image numbers, so do the copy

```

```plaintext
2000 CCOMM = 'IMAGE '//DIGIT(OUTIMG)//' = IMAGE '//DIGIT(INIMG)
```

**PROGRAMMING NOTE:** see COMTAL USER’S GUIDE for further information on the command string abbreviated in the string above.

```plaintext
CALL CMMND( IBUF, 17 )  ! sends copy command to COMTAL
RETURN
```

**ERROR RETURNS:**

```plaintext
WRITE(TERM, 0003)OUTIMG, IMLO, IMHI
```

58
0091  8003 FORMAT(" YOUR OUTPUT PARAMETER. 15. IS OUT OF RANGE."")
0092  GOTO 8900
0093  C
0094  8101 WRITE(TERM, 8103)INIMG, IMLO, IMHI
0095  8103 FORMAT(" YOUR INPUT PARAMETER. 15. IS OUT OF RANGE."")
0096  GOTO 8900
0097  C
0098  8900 WRITE(TERM, 8901)
0099  8901 FORMAT(" ICOPY FAILS. NO COPYING TAKES PLACE.")
0100  RETURN
0101  END
&ICPY2 T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE ICPY2(OUTIMG, INIMG)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**INPUT PARAMETERS:
0006 C
0007 INTEGER OUTIMG ! COMTAL image number for the destination
0008 INTEGER INIMG ! COMTAL image number for the source
0009 C
0010 C**INTRODUCTION:
0011 C
0012 C "Image CoPY #2" commands the COMTAL to copy one black and
0013 C white image into another. The OUTIMG destination image
0014 C is, of course, wiped out by this exchange. The input
0015 C image for ICPY2 can be any single digit number: the output
0016 C image must be a COMTAL image memory plane (1-4 currently).
0017 C The companion subroutine ICOPY requires that both images
0018 C be COMTAL image memory planes.
0019 C
0020 C**LANGUAGE:
0021 C
0022 C FORTRAN 77, the HP-100 version for RTE-6/VM.
0023 C
0024 C**LIMITATIONS:
0025 C
0026 C ICPY only works for the grey level images of COMTAL, not the
0027 C truecolor images.
0028 C
0029 C**SUBPROGRAMS CALLED:
0030 C
0031 C name   source load remarks
0032 C
0033 C CMND &CMND CMND this subroutine takes an INTEGER array which
0034 C contains a COMTAL command string, and transfers
0035 C the command to COMTAL. The second parameter
0036 C gives the character count of the command string.
0037 C RANGE &RANGE RANGE this logical function determines if its first
0038 C argument is within the bounds formed by its
0039 C last 2 arguments, inclusive.
0040 C DIGIT &DIGIT DIGIT character*1 function that returns a single digit
0041 C '0'-9' according to integer input 0-9.
0042 C
0043 C**WRITTEN BY:
0044 C
The code on which this subroutine is based was written by NETTIE D. FAUCON in July, 1983. This modification is by KEITH MILLER, June, 1984.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

**LOGICAL** RANGE  | function determines if 1st argument is within 2nd and 3rd argument inclusive.
**CHARACTER** DIGIT  | function returns '0'-'9' according to integer input 0-9.
**CHARACTER** CCOMM  | character buffer for building up a call to the CMDND subroutine.
**INTEGER** IBUF (128) | integer overlay of CCOMM

**INITIALIZATION:**

**DATA** IMLO/1/, IMHI/4/
**DATA** TERM/1/

**PROCESSING:**

IF (.NOT.(RANGE(OUTIMG, IMLO, IMHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(INIMG, 1, 9 ))) GOTO 8101 ! error return

Legal image numbers, so do the copy

**COMMD = 'IMAGE '/DIGIT(OUTIMG)/' = IMAGE '/DIGIT(INIMG)

**PROGRAMMING NOTE:** see COMTAL USERS GUIDE for further information on the command string abbreviated in the string above.

**CALL CMDND( IBUF, 17 )** ! sends copy command to COMTAL

**ERROR RETURNS:**

**WRITE(TERM, 8003)OUTIMG, IMLO, IMHI
**FORMAT('YOUR OUTPUT PARAMETER,' , ISOUTOF RANGE: ',' , 214)
0091   GOTO 8900
0092   C
0093   8101 WRITE(TERM, 8103) INIMG
0094   8103 FORMAT( ' YOUR INPUT PARAMETER ', 'IS OUT OF RANGE: 1, 9')
0095   GOTO 8900
0096   C
0097   8900 WRITE(TERM, 8901)
0098   8901 FORMAT( ' ICY2 FAILS. NO COPYING TAKES PLACE.' )
0099   RETURN
0100   END
&MERGE T=00004 IS ON CR00021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE MERGE(OUTURD, BYTE1, BYTE2)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER OUTURD ! the output, the 2 input bytes merged into 1 integer
0008 INTEGER BYTE1 ! the left, high order input byte (in lower byte).
0009 INTEGER BYTE2 ! the right, low order input byte (in lower byte).
0010 C
0011 C**INTRODUCTION:
0012 C
0013 C The subroutine MERGE takes two integer inputs which should be non-zero
0014 C in the lower byte only, and merges these lower order bytes into a single
0015 C integer output.
0016 C
0017 C**LANGUAGE:
0018 C
0019 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0020 C
0021 C**LIMITATIONS:
0022 C
0023 C Note that MERGE does not check that the upper order byte of the
0024 C input INTEGERS are zeros. This check could be added, but will slow
0025 C down MERGE execution. This speed is important, since MERGE was designed
0026 C to be a very low level routine.
0027 C
0028 C**SUBPROGRAMS CALLED:
0029 C none.
0030 C
0031 C**WRITTEN BY:
0032 C
0033 C The code on which this subprogram is based was written by
0034 C NETTIE D. FAULCON, July, 1982. This subprogram was written by
0035 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0036 C summer fellowship.
0037 C
0038 C**REVISION HISTORY:
0039 C
0040 C
0041 C**LOCAL VARIABLES:
0042 C
0043 INTEGER IHOLD ! an INTEGER interpretation of bits
0044 CHARACTER CHOLD(2) ! a CHARACTER interpretation of bits
**INTEGER IMERGE** ! an INTEGER interpretation of bits

**CHARACTER CSPLIT(2)** ! a CHARACTER interpretation of bits

**EQUIVALENCE (IHOLD, CHOLD), (IMERGE, CSPLIT)**

**INITIALIZATIONS:** none.

**PROCESSING:**

```
IHOLD = BYTE1
CSPLIT(1) = CHOLD(2)
IHOLD = BYTE2
CSPLIT(2) = CHOLD(2)
OUTURD = IMERGE
```

**RETURN**

**END**
**SUBROUTINE NORML(IMAGE)**

**PARAMETER DECLARATIONS:**

**INTEGER IMAGE** ! designates a COMTAL image to "normalize" (see below)

**INTRODUCTION:**

The subroutine NORML searches through an image to find its lowest pixel value. Then NORML replaces each pixel in the image (call that value X) with the value (X - low). This subroutine was developed to obtain an image of the variations inherent in the lighting table that should give a constant background light, but is instead giving a light with a variation of as many as 10 grey scale levels out of 255. The "normalized" background image is subtracted from the digitized image to simulate a uniform background.

**LANGUAGE:**

**FORTRAN 77, the HP-1000 version for RTE-G/VM.**

**LIMITATIONS:**

This subroutine makes two passes through the image, one to obtain the lowest pixel value, and one to write out the new pixel values. In some applications, the programmer may want to add new processes during one of those passes. Also, the subroutine HILO is used here, even though only the lowest value is required. To optimize, create a new subroutine which only determines the low value.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>READL</td>
<td>READL</td>
<td>READL</td>
<td>reads one horizontal COMTAL image line into an integer array: one integer/pixel.</td>
</tr>
<tr>
<td>WRIL2</td>
<td>WRIL2</td>
<td>WRIL2</td>
<td>writes one horizontal COMTAL image line from an integer array: one integer/pixel.</td>
</tr>
<tr>
<td>HILO</td>
<td>HILO</td>
<td>HILO</td>
<td>determines the highest and lowest pixel value in an image.</td>
</tr>
</tbody>
</table>

**WRITTEN BY:**
The code on which this subprogram is based was written by
HETTIE D. FAULCON, July, 1983. This subprogram was written by
KEITH MILLER, July, 1984, with the support of a NASA-ASEE
summer fellowship.

***REVISION HISTORY:***

***LOCAL VARIABLES:***

**INTEGER** IBUF(512) ! buffer for pixel values; one pixel / integer
**INTEGER** ARALO, ARAHI ! array bounds for a pixel buffer
**INTEGER** LNLO, LNHI ! limits on COMTAL row numbers
**INTEGER** HI, LO ! highest and lowest pixel values in IMAGE
**INTEGER** ROW, COL ! loop indices

***INITIALIZATIONS:***

**DATA** ARALO/7/, ARAHI/512/
**DATA** LNLO/0/, LNHI/511/

***PROCESSING***

**CALL** HILO(HI, LO, IMAGE)

**DO** 2000 ROW = LNLO, LNHI
**CALL** RDIL2(IBUF, IMAGE, ROW)
**DO** 1000 COL = ARALO, ARAHI
**IBUF**(COL) = IBUF(COL) - LO
**CONTINUE
**CONTINUE
**RETURN
**END
&NOTE2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
002 SUBROUTINE NOTE2(GRNUM,XCOORD,YCOORD,FACTO,NOTE)
003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
004 C
005 C***PARAMETER DECLARATIONS:
006 C
007 C INTEGER GRNUM | the number of the graphic to be written into;
008 C INTEGER XCOORD |
009 C INTEGER YCOORD | XCOORD and YCOORD define the upper left corner of the
010 C INTEGER FACTO |
011 C INTEGER COLOR |
012 C CHARACTER*1 NOTE | the message to be printed in graphics; NOTES
013 C CHARACTER*1 COLOR | designed for constant strings.
014 C CHARACTER*1 NOTE |
015 C CHARACTER*1 COLOR |
016 C***INTRODUCTION:
017 C
018 C This subroutine, NOTE2 writes a line of characters into a COMTAL graphics
019 C plane. The parameter NOTE should be a constant string.
020 C A very similar subroutine, NOTES, uses a string variable and a length
021 C parameter instead of the constant string.
022 C
023 C***LANGUAGE:
024 C
025 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
026 C
027 C***LIMITATIONS:
028 C
029 C GRNUM must be in the range 1-4 inclusive.
030 C XCOORD and YCOORD must be in the range 0-511.
031 C COLOR must be one of the following: 'S' (for "Same color"),
032 C 'R' (for "Red"), 'G' (for "Green"), 'B' (for "Blue"),
033 C 'K' (for "black"), 'W' (for "White") or 'Y' (for "Yellow")
034 C Other colors could be easily added. See TABLE 1 in the NOTES
035 C section for other colors.
036 C FACTOR must be in the range 1-16 inclusive.
037 C If a restriction is violated, NOTES fails with an error message.
038 C
039 C The graphics plane named here is turned on and all other graphics
040 C planes are turned off when this subroutine is executed.
041 C
042 C The GRNUM plane is NOT cleared.
The length of time it takes the COMTAL to write a note in graphics causes a timing problem: the COMTAL may ignore the next COMTAL command sent from the HP. Therefore, we DELAY the HP for a number of seconds proportional to the size of the NOTE characters.

**SUBPROGRAMES CALLED:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Load</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMTN2</td>
<td>&amp;CMTN2</td>
<td>XCMTN2</td>
<td>sends constant string to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>CMTND</td>
<td>&amp;CMTND</td>
<td>XCMTND</td>
<td>sends command string and length parameter to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>DELAY</td>
<td>&amp;DELAY</td>
<td>XDELAY</td>
<td>puts the HP in a busy wait; the argument to DELAY gives the number of seconds to DELAY.</td>
</tr>
<tr>
<td>DIGIT</td>
<td>&amp;DIGIT</td>
<td>XDIGIT</td>
<td>character*1 function that returns '0'-'9' according to integer input 0-9.</td>
</tr>
<tr>
<td>DSPGR</td>
<td>&amp;DSPGR</td>
<td>XDSPGR</td>
<td>adds a graphics plane to the display.</td>
</tr>
<tr>
<td>LEN</td>
<td>-------</td>
<td>------</td>
<td>HP FORTRAN intrinsic integer function that returns the length of a character string.</td>
</tr>
<tr>
<td>RANGE</td>
<td>&amp;RANGE</td>
<td>XRANGE</td>
<td>logical function that determines if 1st parameter is within the 2nd &amp; 3rd inclusive.</td>
</tr>
</tbody>
</table>

**WRITTEN BY:**

The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIGIT</td>
<td>function that returns '0'-'9' for integer</td>
</tr>
<tr>
<td>LEN</td>
<td>intrinsic HP FORTRAN function that returns the length of a character string.</td>
</tr>
<tr>
<td>RANGE</td>
<td>logical function determines if 1st argument is within 2nd and 3rd arguments, inclusive.</td>
</tr>
<tr>
<td>GRCHAR</td>
<td>the single digit that corresponds to GRNUM.</td>
</tr>
<tr>
<td>LONG</td>
<td>length of the character string.</td>
</tr>
<tr>
<td>START</td>
<td>the # of first character in CBUF that holds the first character of NOTE.</td>
</tr>
</tbody>
</table>
0091 C
0092 INTEGER GRLO,GRHI ! limits for COMTAL graphics plane number.
0093 INTEGER CMLO,CMHI ! limits on COMTAL coordinates.
0094 INTEGER FCLO,FCHI ! limits on FACTOR.
0095 INTEGER TERM ! logical unit for terminal output.
0096 C
0097 INTEGER IBUF(128) ! buffer for sending CMMND commands.
0098 CHARACTER*255 CBUF ! overlays IBUF.
0099 EQUIVALENCE (IBUF,CBUF)
0100 C
0101 C***INITIALIZATIONS:
0102 C
0103 DATA GRLO/1.,GRHI/4/
0104 DATA CMLO/0.,CMHI/511/
0105 DATA FCLO/1.,FCHI/16/
0106 DATA TERM/1/
0107 C
0108 C***PROCESSING:
0109 C
0110 IF (.NOT.(RANGE(GRNUM, GRLO,GRHI))) GOTO 8081 ! error return
0111 IF (.NOT.(RANGE(XCOORD, CMLO,CMHI))) GOTO 8101 ! error return
0112 IF (.NOT.(RANGE(YCOORD, CMLO,CMHI))) GOTO 8201 ! error return
0113 IF (.NOT.(RANGE(FACTOR,FCLO,FCHI))) GOTO 8301 ! error return
0114 C
0115 GRCHAR = DIGIT(GRNUM) ! initialization delayed until GRNUM checked.
0116 C
0117 IF ((COLOR .EQ. 'S').OR.(COLOR .EQ. 's')) GOTO 2000 ! "Same"
0118 IF ((COLOR .EQ. 'R').OR.(COLOR .EQ. 'r')) GOTO 1000 ! "Red"
0119 IF ((COLOR .EQ. 'G').OR.(COLOR .EQ. 'g')) GOTO 1100 ! "Green"
0120 IF ((COLOR .EQ. 'B').OR.(COLOR .EQ. 'b')) GOTO 1200 ! "Blue"
0121 IF ((COLOR .EQ. 'K').OR.(COLOR .EQ. 'k')) GOTO 1300 ! "Black"
0122 IF ((COLOR .EQ. 'W').OR.(COLOR .EQ. 'w')) GOTO 1400 ! "White"
0123 IF ((COLOR .EQ. 'Y').OR.(COLOR .EQ. 'y')) GOTO 1500 ! "Yellow"
0124 C
0125 ELSE...COLOR an illegal character
0126 C
0127 GOTO 8401 ! error return
0128 C
0129 1000 CALL CMMN2('CO G /'GRCHAR/' RED')
0130 C
0131 1000 CALL CMMN2('CO G /'GRCHAR/' GRN')
0132 C
0133 GOTO 2000
0134 C
0135 1200 CALL CMMN2('CO G /'GRCHAR/' BLU')
0137 GOTO 2000
0138 C
0139 C Color graphics black
0140 1300 CALL CMMN2('CO G 'GRCHAR' BLA')
0141 GOTO 2000
0142 C
0143 C Color graphics white
0144 1400 CALL CMMN2('CO G 'GRCHAR' WHT')
0145 GOTO 2000
0146 C
0147 C Color graphics white
0148 1500 CALL CMMN2('CO G 'GRCHAR' YEL')
0149 GOTO 2000
0150 C
0151 2000 CALL DSPGR(GRNUM)  ! display the selected graphics
0152 CALL WRTAR(XCOOR, YCOOR)  ! position the cursor for writing
0153 C
0154 LONG = LEN(NOTE)
0155 IF (FACTOR .GE. 10) GOTO 3000
0156 C ELSE...
0157 CBUFF(1:8) = 'G 'GRCHAR' L DIGIT(FACTOR)'
0158 START = 9
0159 LONG = LONG + 8
0160 GOTO 4000
0161 C THEN...
0162 3000 CBUFF(1:9) = 'G 'GRCHAR' L 1 DIGIT(FACTOR-10)'
0163 START = 10
0164 LONG = LONG + 9
0165 GOTO 4000
0166 C
0167 4000 CBUFF(START:LONG) = NOTE
0168 CALL CMMND(IBUF, LONG)
0169 C
0170 C Put the HP in a busy wait while the CMDAL writes the note.
0171 CALL DELAY((FACTOR/4)+1)
0172 RETURN
0173 C
0174 C******ERROR RETURNS:
0175 C
0176 8001 WRITE(TERM,8003) GRNUM, GRLO, GRHI
0177 8003 FORMAT(' THE GRAPHICS NUMBER',15,' IS OUT OF RANGE: '.214,')
0178 8000 GOTO 8900
0179 C
0180 8101 WRITE(TERM,8103)XCOOR,CMLO,CMHI
0181 8103 FORMAT(' THE X COORDINATE',15,' IS OUT OF RANGE: '.214,')
0182 8000 GOTO 8900
C0183 WRITE(TERM,8203)YCOOR,CMLO,CMHI
0184 8203 FORMAT(" THE Y COORDINATE, \',.15,\', IS OUT OF RANGE: \',.214,\'.")
0185 GOTO 8900
0186 C
0187 WRITE(TERM,8303)FACTOR,FCL0,FCHI
0188 8303 FORMAT(" THE SCALE FACTOR, \',.15,\', IS OUT OF RANGE: \',.214,\'.")
0189 GOTO 8900
0190 C
0191 WRITE(TERM,8403)COLOR
0192 8403 FORMAT(" THE COLOR PARAMETER, \',.A1,\', IS NOT S, R, G, OR B.")
0193 GOTO 8900
0194 C
0195 WRITE(TERM,8901)
0196 8901 FORMAT(" NOTE2 FAILS. NO LETTERING PLACED INTO GRAPHICS.")
0197 GOTO 8900
0198 C
0199 RETURN
0200 END
&NOTES T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE NOTES(GRUNM,XCOOR,YCOOR,COLOR,FACTOR,NOTE,LENGTH)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER GRNUM ! the number of the graphic to be written into;
0008 ! NOTES doesn't clear or display this graphic.
0009 INTEGER XCOOR !
0010 INTEGER YCOOR ! XCOOR and YCOOR define the upper left corner of the
0011 C position of the first character in the NOTE.
0012 INTEGER FACTOR ! controls size of characters plotted; 1-16.
0013 CHARACTER*1 COLOR ! signals if you wish to stay the same color ("S")
0014 ! or change to red ("R"), green("G"), or blue("B").
0015 CHARACTER*255 NOTE ! the message to be printed in graphics is in the
0016 ! first LENGTH characters of this string.
0017 INTEGER LENGTH ! the number of NOTE characters that are to be used.
0018 C
0019 C***INTRODUCTION:
0020 C
0021 C This subroutine, NOTES writes a line of characters into a COMTAL graphics
0022 C plane. The parameter NOTE should be a declared string.
0023 C A very similar subroutine, NOTE2, uses a string constant without a length
0024 C parameter.
0025 C
0026 C***LANGUAGE:
0027 C
0028 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0029 C
0030 C***LIMITATIONS:
0031 C
0032 C GRNUM must be in the range 1-4 inclusive.
0033 C XCOOR and YCOOR must be in the range 0-511.
0034 C COLOR must be one of the following: "S" (for "Same color"),
0035 C "R" (for "Red"), "G" (for "Green"), or "B" (for "Blue").
0036 C FACTOR must be in the range 1-16 inclusive.
0037 C LENGTH must be in the range 0-255 inclusive.
0038 C If a restriction is violated, NOTES fails with an error message.
0039 C
0040 C The graphics plane named by GRNUM is turned on and all other
0041 C graphics planes are turned off. GRNUM plane is NOT cleared.
0042 C
0043 C Because the COMTAL takes a while to write the note to the screen,
0044 C there can be a timing problem between the COMTAL and the HP which
causes the COMTAL to ignore the next HP command. Therefore, the routine \texttt{DELAY} is used to cause a delay proportional to the size of the characters being printed at the COMTAL.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Source</th>
<th>Load</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{CMMN2}</td>
<td>\texttt{CMMN2}</td>
<td></td>
<td>\texttt{XCMNN2} sends constant string to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>\texttt{CMND}</td>
<td>\texttt{CMND}</td>
<td></td>
<td>\texttt{XCMND} sends a fixed string command to COMTAL as if it were typed at the COMTAL keyboard.</td>
</tr>
<tr>
<td>\texttt{DELAY}</td>
<td>\texttt{DELAY}</td>
<td></td>
<td>\texttt{XDELAY} puts the HP in a busy wait for the number of seconds designated in \texttt{DELAY}'s parameter.</td>
</tr>
<tr>
<td>\texttt{DIGIT}</td>
<td>\texttt{DIGIT}</td>
<td></td>
<td>\texttt{XDIGIT} character function that returns '0'- '9' according to integer input 0-9.</td>
</tr>
<tr>
<td>\texttt{DSPGR}</td>
<td>\texttt{DSPGR}</td>
<td></td>
<td>\texttt{XDSPGR} adds a graphic plane to the display.</td>
</tr>
<tr>
<td>\texttt{RANGE}</td>
<td>\texttt{RANGE}</td>
<td></td>
<td>\texttt{XRANGE} logical function that determines if the 1st argument is within the 2nd &amp; 3rd inclusive.</td>
</tr>
</tbody>
</table>

**Written By:**

The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This subprogram was written by KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**Revision History:**

**Local Variables:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{DIGIT}</td>
<td>CHARACTER*1</td>
<td>function that returns '0'- '9' for integer input 0-9.</td>
</tr>
<tr>
<td>\texttt{RANGE}</td>
<td>LOGICAL</td>
<td>function that determines if 1st argument is within 2nd and 3rd arguments, inclusive.</td>
</tr>
<tr>
<td>\texttt{GRCHAR}</td>
<td>CHARACTER*1</td>
<td>the single digit that corresponds to GRNUM.</td>
</tr>
<tr>
<td>\texttt{LONG}</td>
<td>INTEGER</td>
<td>the length of the CMND command.</td>
</tr>
<tr>
<td>\texttt{GRLO}, \texttt{GRHI}</td>
<td>INTEGER</td>
<td>limits for COMTAL graphics plane number.</td>
</tr>
<tr>
<td>\texttt{CMLO}, \texttt{CMHI}</td>
<td>INTEGER</td>
<td>limits on COMTAL coordinates.</td>
</tr>
<tr>
<td>\texttt{FCLO}, \texttt{FCHI}</td>
<td>INTEGER</td>
<td>limits on FACTOR.</td>
</tr>
<tr>
<td>\texttt{STLO}, \texttt{STHI}</td>
<td>INTEGER</td>
<td>limits on string LENGTH.</td>
</tr>
<tr>
<td>\texttt{TERM}</td>
<td>INTEGER</td>
<td>logical unit for terminal output.</td>
</tr>
<tr>
<td>\texttt{CBUF}</td>
<td>CHARACTER*255</td>
<td>buffer for CMND commands</td>
</tr>
</tbody>
</table>
INTEGER IBUF(128) ! overlays CBUF
EQUIVALENCE (CBUF, IBUF)

C

DATA GRLO/1/,GRHI/4/
DATA CMLO/0/,CMHI/11/
DATA FCLO/1/,FCHI/16/
DATA STLO/0/,STHI/255/
DATA TERM/1/

C

C***INITIALIZATIONS:

C

C

DATA GRLO/I/,GRHI/4/
DATA CMLO/O/,CMHI/11/
DATA FCLO/I/,FCHI/16/
DATA STLO/O/,STHI/255/
DATA TERM/1/

C

C***PROCESSING:

C

IF (.NOT.(RANGE(FACTDR,FCLO,FCHI))) GOTO 8301 ! error return

C

IF (.NOT.(RANGE(YCOOR,CMLO,CMHI))) GOTO 8201 ! error return

C

IF (.NOT.(RANGE(FACTDR,FCLO,FCHI))) GOTO 8301 ! error return

C

GRCHAR = DIGIT(FACTDR) ! initialization delayed until FACTDR checked.

C

IF ((COLOR.EQ. 'S').OR.(COLOR.EQ. 's')) GOTO 2000 ! "Same"

C

IF ((COLOR.EQ. 'R').OR.(COLOR.EQ. 'r')) GOTO 1000 ! "Red"

C

IF ((COLOR.EQ. 'G').OR.(COLOR.EQ. 'g')) GOTO 1100 ! "Green"

C

ELSE...COLOR an illegal character

GOTO 6501 ! error return

C

COLOR graphics red

CALL CMN2('CO G ''/GRCHAR/'' RED')

GOTO 2000

C

COLOR graphics green

CALL CMN2('CO G ''/GRCHAR/'' GRN')

GOTO 2000

C

COLOR graphics blue

CALL CMN2('CO G ''/GRCHAR/'' BLU')

GOTO 2000

C

2000 CALL DSPGR(FACTDR) ! display the chosen graphics

CALL WRTAR(XCOOR,YCOOR) ! position the cursor for writing

C

IF (FACTOR .GE. 10) GOTO 3000

C

ELSE...

CBUF = 'G ''/GRCHAR/'' L ''/DIGIT(FACTDR)/'' ''/NOTE

74
LONG = LENGTH + 8
GOTO 4000
C THEN...
3000 CBUF = 'G'/'GRCHAR'/' L1'/'DIGIT(FACTOR-10)'/' NOTE
LONG = LENGTH + 9
GOTO 4000
C
4000 CALL CMMND(IBUF, LONG)
C
Delay for a second or two to relieve timing problem between
C the HP and the COMTAL
C
CALL DELAY((FACTOR/4)+1)
RETURN
C
C****ERROR RETURNS:
C
C
8001 WRITE(TERM,8003)GRNUM,GRID,GRID
8003 FORMAT(' THE GRAPHICS NUMBER,'.I5,'. IS OUT OF RANGE:'",214,"')
GOTO 8900
C
C
8103 FORMAT(' THE X COORDINATE,'.I5,'. IS OUT OF RANGE:'",214,"')
GOTO 8900
C
C
8203 FORMAT(' THE Y COORDINATE,'.I5,'. IS OUT OF RANGE:'",214,"')
GOTO 8900
C
C
8303 FORMAT(' THE SCALE FACTOR,'.I5,'. IS OUT OF RANGE:'",214,"')
GOTO 8900
C
C
8403 FORMAT(' THE STRING LENGTH,'.I5,'. IS OUT OF RANGE:'",214,"')
GOTO 8900
C
C
8503 FORMAT(' THE COLOR PARAMETER,'.A1,'. IS NOT S, R, G, OR B.')
GOTO 8900
C
C
8601 WRITE(TERM,8601)
8503 FORMAT(' NOTES FAILS. NO LETTERING PLACED INTO GRAPHICS.')
RETURN
END
&PAINT T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE PAINT(IMAGE, BRUSH, SHADE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 C INTEGER IMAGE ! number of the COMTAL image to be painted
0008 C INTEGER BRUSH ! size of the square brush area
0009 C INTEGER SHADE ! the pixel value to be brushed on
0010 C
0011 C***INTRODUCTION:
0012 C
0013 C The subroutine PAINT allows the interactive user to paint
0014 C onto a COMTAL image, using the trackball to guide the brush.
0015 C The user uses the HP keyboard to signal when to paint.
0016 C Each time the HP <CR> is pressed, PAINT paints a BRUSH X BRUSH
0017 C square of pixels with the cursor position in the upper left corner
0018 C of the square. The maximum size for a brush has been set (arbitrarily)
0019 C to 64 pixels square.
0020 C
0021 C***LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0024 C
0025 C***LIMITATIONS:
0026 C
0027 C The COMTAL keyboard is inaccessible to the HP. Therefore, we
0028 C must use the HP keyboard even though we use the COMTAL trackball.
0029 C Because of the HP keyboard limitations, the program requires
0030 C a pointillistic painting: one square in the image is darkened
0031 C each time the HP <CR> is pressed.
0032 C
0033 C***SUBPROGRAMS CALLED:
0034 C
0035 C name source load remarks
0036 C
0037 C DSPBW &DSPBW %DSPBW display a monochrome COMTAL image.
0038 C URIRC &URIRC %URIRC writes the contents of an array to a COMTAL image;
0039 C the values are read into a rectangle in the image.
0040 C RDTAR &RDTAR %RDTAR reads the current COMTAL cursor position.
0041 C WAIT &WAIT %WAIT halts HP processing until HP <CR> entered.
0042 C CMIND &CMIND %CMIND sends a character string to COMTAL, which
0043 C treats the string as a COMTAL keyboard command.
0044 C
The code on which this subprogram is based was written by
NETTIE D. FAULCON, July, 1983. This subprogram was written by
KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**

1. The code on which this subprogram is based was written by
   NETTIE D. FAULCON, July, 1983. This subprogram was written by
   KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**LOCAL VARIABLES:**

```plaintext
INTEGER BOX(64*64) ! the square that acts as a paint brush-
                   ! this rectangle placed into IMAGE whenever
                   ! <CR> entered to locate cursor on COMTAL.
                   ! loop index for initializing BOX.
INTEGE R XPOS    ! X and Y coordinates of last COMTAL cursor
                   ! (target) position recorded via RDTAR.
INTEGER UPLFX, UPLFY ! used as upper left corner of rectangle
                   ! to be painted.
INTEGER TERM     ! logical unit for terminal I/O.
INTEGER IMLO, IMHI ! limits on COMTAL image number.
INTEGER PXLO, PXHI ! limits on COMTAL pixel values.
INTEGER BRLO, BRHI ! limits on size of brush for painting.
CHARACTER*1 INCHAR ! character buffer for HP Keyboard input.
LOGICAL RANGE    ! function that determines if 1st argument
                   ! is within the 2nd and 3rd, inclusive.
INTEGER IBUF(128) ! buffer for sending CMND strings
CHARACTER*255 CBUF ! character overlay for IBUF
```

**INITIALIZATIONS:**

```plaintext
DATA IMLO/1/, IMHI/4/
DATA BRLO/1/, BRHI/128/
DATA PXLO/0/, PXHI/255/
DATA TERM/1/
```

**PROCESSING**

```plaintext
IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(BRUSH, BRLO, BRHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(SHADE, PXLO, PXHI))) GOTO 8201 ! error return
```

Make sure the image in question is displayed.
CALL DSPBW(IMAGE)

Initialize the paint brush (this initialization done here instead of above to avoid processing when a parameter is bad).

Programming note: the BOX array is filled as a one dimensional array, but is interpreted by WRIRC below as a two dimensional array.

DO 400 XPOS = 1, (BRUSH*BRUSH)
  BOX(XPOS) = SHADE
END CONTINUE

add the target and dump the image on the COMTAL

CBUF = 'ADD TARGET $DUMP IMAGE'
CALL CMIND(IBUF, 23)

Give instructions to user

WRITE(TERM, 501)
501 FORMAT(' ENTER <CR> TO PAINT A SQUARE.', 1 ' ENTER S<CR> TO EXIT PAINTING.')

Loop for input/painting starts here:

1000 INCHAR = ' '
READ(TERM, 1001) INCHAR
1001 FORMAT(IA1)
1002 IF (INCHAR .EQ. ' ') GOTO 2000 ! paint another square and continue.
1003 ELSE...illegal entry
WRITE(TERM, 1003) INCHAR
1004 FORMAT(' THE CHARACTER ENTERED, ', 1A1, ', IS NOT LEGAL', 1 ' FOR PAINTING. PLEASE TRY AGAIN.')
1005 GOTO 1000

Actual pointing takes place here:

2000 CALL RDTAR(UPLFX, UPLFY)
2001 CALL WRIRC(IMAGE, UPLFX, UPLFY, BOX, BRUSH, BRUSH)
2002 GOTO 1000

***ERROR RETURNS:

8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0137  8003 FORMAT(' IMAGE NUMBER.', 15, ' OUT OF RANGE:', 214, '.')
0138       GOTO 8900
0139 C
0140  8101 WRITE(TERM, 8103) BRUSH, BRLO, BRHI
0141  8103 FORMAT(' BRUSH ARGUMENT.', 15, ' OUT OF RANGE:', 214, '.')
0142       GOTO 8900
0143 C
0144  8201 WRITE(TERM, 8203) SHADE, PXLO, PXSH
0145  8203 FORMAT(' SHADE ARGUMENT.', 15, ' OUT OF RANGE:', 214, '.')
0146       GOTO 8900
0147 C
0148  8900 WRITE(TERM, 8901)
0149  8901 FORMAT(' PAINT SUBROUTINE FAILS.')
0150 C
0151  9000 CONTINUE
0152       RETURN
0153  9001 END
SUBROUTINE PROFIL(GRAPH, IMAGE)

**PARAMETER DECLARATIONS:**

INTEGER GRAPH ! the number of the COMTAL graphics plane in which
      ! the profile is to be displayed.

INTEGER IMAGE ! the number of the COMTAL image which is going to
      ! be "profiled."

**INTRODUCTION:**

The subroutine PROFIL gives HP access to the COMTAL's interactive
profiling facilities. This subroutine initializes the COMTAL for
taking profiles of IMAGE and displaying them in the specified GRAPH
plane. Note that this subroutine clears the specified graphics plane.
Control passes to COMTAL for profiling, and then a <CR> on the HP
terminal restores control to the HP terminal.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-E/VM.

**LIMITATIONS:**

GRAPH and IMAGE are checked, and if they are out of bounds, no
profiling takes place and an error message is printed. The shift
from the HP terminal to the COMTAL terminal may result in some
confusion, but this switch exploits the COMTAL circuitry much more
efficiently than would be possible using the HP terminal alone.

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
</table>
| RANGE | &RANGE | xRANGE | logical function that determines if 1st argu-
| CMND | &CMND | xCMND | sent is within 2nd and 3rd, inclusive. |
| CMND | &CMND | xCMND | sends a command to the COMTAL as if it were |
| WAIT | &WAIT | xWAIT | halted HP processing until <CR> is pressed on |
| CHAR | ------ | ------ | intrinsic HP FORTRAN77 function that converts |
|      |        |      | integers into characters |

80
C DIGIT &DIGIT %DIGIT character*1 function that returns '0'..'9' according to integer input 0..9.

C***WRITTEN BY:

C The code on which this subprogram is based was written by
C NETTIE D. FAULCON, July, 1983. This subprogram was written by
C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
C summer fellowship.

C***REVISION HISTORY:

C

C***LOCAL VARIABLES:

C CHARACTER*1 GCHAR ! the character that corresponds to the single
C digit argument, GRAPH
C CHARACTER*1 ICHAR ! the character that corresponds to the single
C digit argument, IMAGE
C CHARACTER*1 BELL ! the ASCII code for ESCAPE.
C CHARACTER*1 DIGIT ! function that returns '0'..'9' according to
C integer argument 0..9.
C CHARACTER CHAR ! intrinsic FORTRAN?? function for integer to
C character conversion.

C INTEGER IBUF(128)! integer buffer for sending COMTAL commands
C CHARACTER*255 CBUF ! overloads IBUF
C EQUIVALENCE (IBUF,CBUF)

C INTEGER GRLO,GRHI! limits on numbers of graphics planes in COMTAL
C INTEGER IMLO,IMHI! limits on numbers of image planes in COMTAL
C LOGICAL RANGE ! logical function that determines if 1st argu-
C ment is within 2nd and 3rd argument, inclusive.
C INTEGER TERM ! logical unit for terminal output

C***INITIALIZATIONS:

C DATA GRLO/)GRHI/4/
C DATA IMLO/)IMHI/4/
C DATA TERM/1/
C BELL = CHAR(7) ! HP bell

C***PROCESSING

C IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8881
C IF (.NOT.(RANGE(IMAGE,IMLO,IMHI))) GOTO 8181
set up COMTAL display for profiling

GCHAR = DIGIT(GRAPH)
ICHAR = DIGIT(IMAGE)

the following COMTAL command reads as follows (*G stands for
the single digit associated with GRAPH; and *I, with IMAGE):

Display Image *I; Add Graphics *G; Clear Graphics *G;
COLOR Graphics *G RED.

CBUF = 'D I //ICHAR// SADD G //GCHAR// SCL G //GCHAR//
      ' SCO G //GCHAR// RED'
CALL CMNND(IBUF, 34)

the following COMTAL command reads: Add TARGET:
Graphics *G = PROFILE of image *I.

CBUF = 'A TA *G //GCHAR// PRO //ICHAR
CALL CMNND(IBUF, 15)

print out instructions for the COMTAL keyboard interaction

WRITE(TERM, 7001)
7001 FORMAT( 'COMTAL function switches control profiling."
      1 /"Switch 2 toggles X profiling on and off."
      2 /"Switch 3 toggles Y profiling on and off."
      3 /"Switch 1 requests new profiles."
      4 /"Press <ESC> on the COMTAL keyboard and then press"
      5 /"<CR> on the HP keyboard to end profiling.")
CALL WAIT
RETURN

ERROR RETURNS:

WRITE(TERM, 8003)GRAPH, GRLO, GRHI
8003 FORMAT( 'THE GRAPH PARAMETER, ",I4," IS OUT OF RANGE:" ,213)
GOTO 8900

WRITE(TERM, 8103)IMAGE, IMLO, IMHI
8103 FORMAT( 'THE IMAGE PARAMETER, ",I4," IS OUT OF RANGE:" ,213)
GOTO 8900

8900 WRITE(TERM, 8901)
8901 FORMAT('PROFL FAILS. NO PROFILING DONE.")
RETURN
END
LOGICAL FUNCTION RANGE(OBJECT, LOW, HIGH)

PARAMETER DECLARATIONS:
INTEGER OBJECT    ! the number to be examined
INTEGER LOW, HIGH  ! the limits on the number

INTRODUCTION:
This function returns .TRUE. if the OBJECT is within the range
between LOW and HIGH (inclusive), and .FALSE. if it is outside
that range. If LOW .GT. HIGH, an error message is printed,
and .FALSE. is returned.

LIMITATIONS:
No error code is returned. Only a message is printed out.

WRITTEN BY:
Keith Miller, NASA-Langley ASEE fellow, 1984

REVISION HISTORY:

LOCAL VARIABLES:
INTEGER TERM  ! logical unit for terminal output

INITIALIZATIONS:
DATA TERM/1/

PROCESSING:
IF (LOW .GT. HIGH) WRITE(TERM, 1001) LOW, HIGH
1001 FORMAT(LOW,' , IS , ,GT. HIGH, , IS , , RANGE fails.')
RANGE = ((OBJECT .GE. LOW) .AND. (OBJECT .LE. HIGH))
RETURN
END
**RDGLN** T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDGLN (ONOFFS, GRAPH, LINE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER ONOFFS(32) ! 16 bits per integer, 512 bits in a line.
0008 INTEGER GRAPH    ! COMTAL image graphics to be written to.
0009 INTEGER LINE      ! which horizontal line to be written to:
0010 C
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C This subroutine Reads a Graphics Line from the COMTAL. The line of
0015 C bits is coded as 1 for on and 0 for off.
0016 C
0017 C***LANGUAGE:
0018 C
0019 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0020 C
0021 C***LIMITATIONS:
0022 C
0023 C If GRAPH or LINE is out of range, an error message is printed
0024 C and no transfer takes place.
0025 C
0026 C***SUBPROGRAMS CALLED:
0027 C
0028 C name source load remarks
0029 C
0030 C RANGE &RANGE XRANGE logical function that determines if its
0031 C first parameter is within the last two parameters.
0032 C
0033 C***WRITTEN BY:
0034 C
0035 C The code on which this subprogram is based was written by
0036 C NETTIE D. FRALCON, July, 1983. This modification is by
0037 C KEITH MILLER, June, 1984.
0038 C
0039 C***REVISION HISTORY:
0040 C
0041 C
0042 C***LOCAL VARIABLES:
0043 C
0044 C LOGICAL RANGE       ! function that ascertains if its first parameter
INTEGER TERM | the logical unit for terminal output
INTEGER GRLO, GRHI | the limits on COMTAL monochrome image numbers
INTEGER LNLO, LNHI | the limits on COMTAL image line numbers

***INITIALIZATIONS:

DATA TERM/1/
DATA GRLO/1/, GRHI/4/
DATA LNLO/0/, LNHI/511/

***PROCESSING

IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return

Programming note:
The EXEC call is explained in detail in the
HP Programmer's Reference Manual for RTE-6/VM, p. 2-19ff. This
transfer function for the COMTAL is discussed in the
COMTAL User's Manual. Section 5.2.2.1. In the EXEC call
that follows, the HP resident driver, DVR41, is called as
follows: the first parameter (1) signifies a read; the
second parameter is in two parts: 3GB identifies the resident
DVR41 driver, and 1008 identifies the line transfer operation
of that driver; the third parameter (ONOFFS) holds the data to be
transferred, and the fourth parameter gives ONOFFS' length in words
(32); and the final parameter is a COMTAL command code for the transfer
CALL EXEC(1,3GB+1008,ONOFFS,32,(GRAPH-1)*2048 + LINE + 512)

***ERROR RETURNS

WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT(' GRAPHICS NUMBER, ' , 13, ' OUT OF RANGE: ' , 212, ' ')
GOTO 8900

8101 WRITE(TERM, 8103) LINE, LNLO, LNHI
8103 FORMAT(' LINE NUMBER, ' , 14, ' OUT OF RANGE: ' , 213, ' ')
GOTO 8900

8900 WRITE(TERM, 8901)
8901 FORMAT(' RDGLN FAILS. NO TRANSFER. ')
RETURN
END
**SUBROUTINE RDGPT(VALUE, GRAPH, XCOORD, YCOORD)**

**PARAMETER DECLARATIONS:**

- INTEGER VALUE ! 0-1 graphics value to be retrieved.
- INTEGER GRAPH ! COMTAL graph number to be read from.
- INTEGER XCOORD, YCOORD ! point where value is to be read from.

**INTRODUCTION:**

This subroutine Reads a Graph Point from the COMTAL. The value of the graphics is an on/off decision, represented by an integer 1 or 0. If the graph number or coordinates are out of range, an error message is printed and no transfer takes place.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**LIMITATIONS:**

If GRAPH, XCOORD, YCOORD, or VALUE are out of range, an error message is printed at the terminal and no transfer takes place.

**SUBPROGRAMS CALLED:**

- RANGE &RANGE XRANGE logical function that determines if its first parameter is within the last two parameters.
- BTTEST logical function that takes an integer argument and returns .TRUE. or .FALSE. according to the bit selected by its second argument: bit = 0, FALSE returned.

**WRITTEN BY:**

The code on which this subprogram is based was written by NETTIE D. FAULCON. July, 1983. This modification is by KEITH MILLER. June, 1984.
**LOCAL VARIABLES:**

0045 C

0046 C

0047 C

0048 C

0049 C

0050 C

0051 C

0052 LOGICAL RANGE

0053 is between (inclusive) its last 2 parameters

0054 C

0055 LOGICAL BTEST

0056 is a certain bit on or off.

0057 C

0058 INTEGER TERM

0059 the logical unit for terminal output

0060 C

0061 INTEGER GRLO, GRHI

0062 the limits on COMTAL monochrome graph numbers

0063 C

0064 INTEGER LINLO, LINHI

0065 the limits on COMTAL graph line numbers

0066 C

0067 INTEGER BTLO, BTHI

0068 the limits on COMTAL graphics values

0069 C

0070 INTEGER BITS(32)

0071 a buffer to read & write a COMTAL graphics line

0072 C

0073 INTEGER WORD

0074 which word of BITS holds the bit selected by XCOORD.

0075 C

0076 INTEGER BIT

0077 which bit in BITS(WORD) holds the bit selected by

0078 C

0079 XCOORD; bits numbered 0-15, right to left.

0080 C

0081 **INITIALIZATIONS:**

0082 C

0083 DATA TERM/1/

0084 DATA GRLO/1/, GRHI/4/

0085 DATA LNLO/0/, LINHI/511/

0086 C

0087 **PROCESSING**

0088 C

0089 IF (.NOT.(RANGE(GRAPH, GRLO, GRHI))) GOTO 8001 ! error return

0090 IF (.NOT.(RANGE(XCOORD, LNLO, LINHI))) GOTO 8101 ! error return

0091 IF (.NOT.(RANGE(YCOORD, LNLO, LINHI))) GOTO 8201 ! error return

0092 C

0093 Programming note:

0094 The EXEC calls below are to the DVR41 driver. The first call

0095 is identical to the call made in RDILN. The second EXEC call

0096 is identical to the one in URILN. See the documentation for

0097 those subroutines for details on these calls.

0098 C

0099 Read the COMTAL line (horizontal) that contains the point in question:

0100 C

0101 CALL EXEC(1,368+1003,BITS.32,(GRAPH-1)*2048 + 512 + YCOORD)

0102 C

0103 Find the single bit that has been selected:

0104 C

0105 WORD = (XCOORD/16) + 1

0106 BIT = (16*WORD) - XCOORD - 1

0107 C

0108 VALUE = 0 ! bit is clear until proven set.

0109 IF (BTEST(BITS(WORD),BIT)) VALUE = 1
RETURN

***ERROR RETURNS

8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT('GRAPH NUMBER: ', I3, 'OUT OF RANGE: ', 212, '.')
GOTO 8900

8101 WRITE(TERM, 8103)XCOORD, LNLO, LNHI
8103 FORMAT('X COORDINATE: ', I4, 'OUT OF RANGE: ', 214, '.')
GOTO 8900

8201 WRITE(TERM, 8203)YCOORD, LNLO, LNHI
8203 FORMAT('Y COORDINATE: ', I4, 'OUT OF RANGE: ', 214, '.')
GOTO 8900

8900 WRITE(TERM, 8901)
8901 FORMAT('RDGPT FAILS. NO TRANSFER.')
RETURN
END
&RDIL2 T=00004 IS ON CR00021 USING 00010 BLKS R=0000

0001 CCCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCCCDCCCCC
**LOCAL VARIABLES:**

**INTEGER** PIXELS(256) ! 512 bytes, 1 pixel/byte, transferred to COMTAL

**CHARACTER*1** CPIX(512) ! overlays PIXELS

**EQUIVALENCE** (PIXELS, CPIX)

**LOGICAL RANGE** ! function that ascertains if its first parameter

**is between** (inclusive) its last 2 parameters

**INTEGER TERM** ! the logical unit for terminal output

**INTEGER IMLO, IMHI** ! the limits on COMTAL monochrome image numbers

**INTEGER LNLO, LNHI** ! the limits on COMTAL image line numbers

**INTEGER INDEX** ! loop index for stepping through the arrays.

**INITIALIZATIONS:**

**DATA** TERM/1/

**DATA** IMLO/1/, IMHI/4/

**DATA** LNLO/0/, LNHI/511/

**PROCESSING**

**IF** (.NOT.(RANGE(Image, IMLO, IMHI))) **GOTO** 8001 ! error return

**IF** (.NOT.(RANGE(Line, LNLO, LNHI))) **GOTO** 8101 ! error return

**Programming note:**

The EXEC call is explained in detail in the HP Programmer's Reference Manual for RTE-6/VM, p.2-19ff. This transfer function for the COMTAL is discussed in the COMTAL User's Manual, Section 5.2.2.1. In the EXEC call that follows, the HP resident driver, DVR41, is called as follows: the first parameter (I) signifies a read; the second parameter is in two parts: 3GB identifies the resident DVR41 driver, and 100B identifies the line transfer operation of that driver; the third parameter (PIXELS) holds the data to be transferred, and the fourth parameter gives PIXELS' length in words (256): and the final parameter is a COMTAL command code for the transfer.

**CALL EXEC(I, 3GB+100B, PIXELS, 256, (IMAGE-I)*2048 + LINE)**

transform the 512 8 bit values into 512 16 bit integers

**DO** 1000 INDEX = 1, 512

**INTS(INDEX) = ICHAR(CPIX(INDEX))** ! CPIX overlays PIXELS

**CONTINUE**

**RETURN**
0091 C
0092 C***ERROR RETURNS
0093 C
0094 8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0095 8003 FORMAT(' IMAGE NUMBER.', i3, ' OUT OF RANGE: ', 212,'.')
0096    GOTO 8900
0097 C
0098 8101 WRITE(TERM, 8103) LINE, LNLLO, LNHI
0099 8103 FORMAT(' LINE NUMBER.', i4, ' OUT OF RANGE: ', 213, '.')
0100    GOTO 8900
0101 C
0102 8900 WRITE(TERM, 8901)
0103 8901 FORMAT(' RDIL2 FAILS. NO TRANSFER.')
0104    RETURN
0105    END
&RDILN T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDILN(PIXELS, IMAGE, LINE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER PIXELS(256)! 512 bytes (pixels) to be read
0008 INTEGER IMAGE ! COMTAL image number to be read from
0009 INTEGER LINE ! which horizontal line to be read from
0010 C
0011 C
0012 C**INTRODUCTION:
0013 C
0014 C This subroutine reads an image line from the COMTAL. The line of
0015 C pixels is made up of 8 bit (0-255) grey scale intensities. The
0016 C PIXELS array will be filled to capacity by RDILN.
0017 C
0018 C**LANGUAGE:
0019 C
0020 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0021 C
0022 C**LIMITATIONS:
0023 C
0024 C IMAGE must be a monochrome image. The LINE parameter must be between
0025 C 0 and 511. If IMAGE or LINE is out of range, an error message is printed
0026 C and no transfer takes place.
0027 C
0028 C**SUBPROGRAMS CALLED:
0029 C
0030 C name source load remarks
0031 C ------- ------- ------- ------------------------------------------
0032 C RANGE ORANGE X RANGE logical function that determines if its
0033 C first parameter is within the last two parameters.
0034 C
0035 C**WRITTEN BY:
0036 C
0037 C The code on which this subprogram is based was written by
0038 C NETTIE D. FAULCON, July, 1983. This modification is by
0039 C KEITH MILLER, June, 1984.
0040 C
0041 C**REVISION HISTORY:
0042 C
0043 C**LOCAL VARIABLES:
0044 C
0045 LOGICAL RANGE  ! function that ascertains if its first parameter
0046 C  ! is between (inclusive) its last 2 parameters
0047 INTEGER TERM  ! the logical unit for terminal output
0048 INTEGER IMLO, IMHI  ! the limits on COMTAL monochrome image numbers
0049 INTEGER LNLO, LNHI  ! the limits on COMTAL image line numbers
0050 C
0051 C***INITIALIZATIONS:
0052 C
0053 DATA TERM/1/
0054 DATA IMLO/1/, IMHI/4/
0055 DATA LNLO/0/, LNHI/511/
0056 C
0057 C***PROCESSING
0058 C
0059 IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8081 ! error return
0060 IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8181 ! error return
0061 C
0062 C Programming note:
0063 C The EXEC call is explained in detail in the
0064 C HP Programmer’s Reference Manual for RTE-6/VM.p.2-19ff. This
0065 C transfer function for the COMTAL is discussed in the
0066 C COMTAL User’s Manual, Section 5.2.2.1. In the EXEC call
0067 C that follows, the HP resident driver, DVR41, is called as
0068 C follows: the first parameter (I) signifies a read; the
0069 C second parameter is in two parts: 3GB identifies the resident
0070 C DVR41 driver, and 1008 identifies the line transfer operation
0071 C of that driver; the third parameter (PIXELS) holds the data to be
0072 C transferred, and the fourth parameter gives PIXELS’ length in words
0073 C (256); and the final parameter is a COMTAL command code for the transfer
0074 C CALL EXEC(1, 3GB+1008, PIXELS, 256, (IMAGE-1)*2048 + LINE)
0075 RETURN
0076 C
0077 C***ERROR RETURNS
0078 C
0079 8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0080 8003 FORMAT(’ IMAGE NUMBER:’, 13, ’ OUT OF RANGE:’, 212,’.’)
0081 GOTO 8900
0082 C
0083 8101 WRITE(TERM, 8103) LINE, LNLO, LNHI
0084 8103 FORMAT(’ LINE NUMBER:’, 14, ’ OUT OF RANGE:’, 213,’.’)
0085 GOTO 8900
0086 C
0087 8900 WRITE(TERM, 8901)
0088 8901 FORMAT(’ RDILN FAILS. NO TRANSFER.’)
0089 RETURN
0090 END
&RDIP T-00004 IS ON CR00021 USING 00018 BLKS R-0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 C
0003 C
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER VALUE ! receives pixel value, converted from byte
0008 C ! to integer.
0009 INTEGER IMAGE ! COMTAL image number to be read from.
0010 INTEGER XCOORD, YCOORD ! point where value is to be read from.
0011 C
0012 C***INTRODUCTION:
0013 C
0014 C This subroutine ReadS an Image Point from the COMTAL. The value
0015 C of the pixel is an 8 bit (0-255) gray scale intensity. If the
0016 C image number or coordinates are out of range, an
0017 C message is printed and no transfer takes place.
0018 C
0019 C***LANGUAGE:
0020 C
0021 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0022 C
0023 C***LIMITATIONS:
0024 C
0025 C The IMAGE designated
0026 C must be a monochome image. The YCOORD parameter must be between
0027 C 0 and 511. If IMAGE or YCOORD is out of range, an error message is printed
0028 C and no transfer takes place.
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C ------- ------- ------- ---------------------------------
0034 C RANGE &RANGE XRANGE logical function that determines if its
0035 C first parameter is within the last two parameters.
0036 C ICHAR ----- ----- HP FORTRAN77 intrinsic function: converts a byte
0037 C into its integer code.
0038 C
0039 C***WRITTEN BY:
0040 C
0041 C The code on which this subprogram is based was written by
0042 C NETTIE D. FAULCON, July, 1983. This modification is by
0043 C KEITH MILLER. June, 1984.
REVISI0N HISTORY:

LOCAL VARIABLES:

LOGICAL RANGE ! function that ascertains if its first parameter is between (inclusive) its last 2 parameters

INTEGER TERM ! the logical unit for terminal output

INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers

INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers

INTEGER PIXELS(256) ! a buffer to read a COMTAL line

CHARACTER*1 BYTES(512) ! overlay for PIXELS buffer

EQUIVALENCE (PIXELS, BYTES)

INITIALIZATIONS:

DATA TERM/1/

DATA IMLO/1/, IMHI/4/

DATA LNLO/0/, LNHI/511/

PROCESSING

IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return

IF (.NOT.(RANGE(XCOORD, LNLO, LNHI))) GOTO 8201 ! error return

Programming note:

Read the COMTAL line (horizontal) that contains the point in question:

CALL EXEC(1, 36B+100B, PIXELS, 256, (IMAGE-1)*2048 + YCOORD)

convert the byte that is to be read:

VALUE = ICHAR(BYTES(XCOORD+1)))! ICHAR is an intrinsic F77 function

which converts a character into its integer code.

! The '+1' changes from pixels, which are 0-511 to the FORTRAN array, 1-512

RETURN

ERROR RETURNS
0091  003 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0092  003 FORMAT(‘ IMAGE NUMBER.’, I3, ‘ OUT OF RANGE:’, 212,’.’)
0093          GOTO 8900
0094  C
0095  8101 WRITE(TERM, 8103)YCOORD, LNLO, LNH1
0096  8103 FORMAT(‘ Y COORDINATE,’, I4, ‘ OUT OF RANGE:’, 214,’.’)
0097          GOTO 8900
0098  C
0099  8201 WRITE(TERM, 8203)XCOORD, LNLO, LNH1
0100  8203 FORMAT(‘ X COORDINATE,’, I4, ‘ OUT OF RANGE:’, 214,’.’)
0101          GOTO 8900
0102  C
0103  8901 WRITE(TERM, 8901)
0104  8901 FORMAT(‘ RD1PT FAILS. NO TRANSFER.’)
0105          RETURN
0106          END
SUBROUTINE RDIRC(OUTARA, XDIM, YDIM, IMAGE, UPLFX, UPLFY)

PARAMETER DECLARATIONS:

INTEGER XDIM, YDIM ! dimensions of the output array, OUTARA
INTEGER OUTARA(XDIM, YDIM) ! the array to be filled
INTEGER IMAGE ! the number of the COMTAL image from
INTEGER UPLFX, UPLFY ! the image coordinates of the upper left
corner of the rectangle of pixels that is
! to be read into OUTARA.

INTRODUCTION:

The subroutine Read Image Rectangle transfers pixel values from a
designated section of an image to an integer array. Note that although
pixel values are generally stored with 1 byte/pixel, RDIRC places each
numeric value into a 2 byte integer in OUTARA.

XDIM, YDIM, IMAGE, UPLFX, and UPLFY are all checked for possible out
of range errors before any transfer is attempted.

LANGUAGE:

FORTRAN 77, the HP-1000 version for RTE-6/VM.

LIMITATIONS:

RDIRC does a great deal of error checking before initiating the
transfer. If the programmer can verify that all calls to RDIRC
will be legal, these checks could be commented out to improve
machine efficiency. Also, the call to the subroutine RDILN could
be replaced inline by the RDILN code (which is short).

SUBPROGRAMS CALLED:

name     source     load     remarks
--------- ------- ------- --------------------------
RDILN     &RDILN    &RDILN transfers a line of bytes from a COMTAL image
RANGE     &RANGE    &RANGE logical function that determines if its first
parameter is within the last two parameters
ICHAR     ------- ------- HP FORTRAN77 intrinsic function; converts a
byte into its integer code.

---

**WITTEN BY:**

*The code on which this subprogram is based was written by*

**NETTIE D. FAULCON, July, 1983. This code was written by***

**KEITH MILLER, July, 1984.***

---

**REVISION HISTORY:**

---

**LOCAL VARIABLES:**

---

**INITIALIZATIONS:**

---

**PROCESSING**

---

```
0045 C
0046 C
0047 C**WRITTEN BY:**
0048 C
0049 C The code on which this subprogram is based was written by
0050 C NETTIE D. FAULCON. July, 1983. This code was written by
0052 C
0053 C**REVISION HISTORY:**
0054 C
0055 C
0056 C**LOCAL VARIABLES:**
0057 C
0058 C INTEGER IMROW, IMCOL ! a location in the image
0059 C INTEGER ARAROW, ARACOL ! a location in OUTARA
0060 C INTEGER ICHAR ! intrinsic byte to integer conversion
0061 C LOGICAL RANGE ! function
0062 C INTEGER IMLO, IMHI ! limits on COMTAL image numbers
0063 C INTEGER LNLO, LNHI ! limits on COMTAL pixel coordinates
0064 C INTEGER LNCT ! LNHI-LNLO+1, # of pixels in an image line
0065 C INTEGER TERM ! logical unit for terminal output
0066 C INTEGER IBUF(256) ! buffer to hold COMTAL horizontal line
0067 C CHARACTER*(512) CBUF(512) ! overlay for IBUF
0068 C EQUIVALENCE (IBUF, CBUF)
0069 C
0070 C**INITIALIZATIONS:**
0071 C
0072 C
0073 C
0074 DATA TERM/1/
0075 DATA IMLO/0/, IMHI/4/
0076 DATA LNLO/0/, LNHI/511/, LNCT/512/
0077 C
0078 C**PROCESSING**
0079 C
0080 IF (.NOT.(RANGE(Image, IMLO, IMHI )))GOTO 8001 ! error return
0081 IF (.NOT.(RANGE(XDIM, 1, LNCT)))GOTO 8101 ! error return
0082 IF (.NOT.(RANGE(YDIM, 1, LNCT)))GOTO 8201 ! error return
0083 IF (.NOT.(RANGE(UPLFX, LNLO, LNHI )))GOTO 8301 ! error return
0084 IF (.NOT.(RANGE(UPLFY, LNLO, LNHI )))GOTO 8401 ! error return
0085 IF (.NOT.(RANGE(UPLFX*XDIM-1, LNLO, LNHI )))GOTO 8501 ! error return
0086 IF (.NOT.(RANGE(UPLFY*YDIM-1, LNLO, LNHI )))GOTO 8601 ! error return
0087 C
0088 C we get to this point if the transfer is to take place
0089 C
0090 IMROW = UPLFY
```
DO 2000 ARAROW = 1, YDIM
CALL RDILN(IBUF, IMAGE, IMROW)
IMROW = IMROW + 1 ! increment for next pass thru 2000 loop
C the next line initializes the column pointer:
IMCOL = UPLFX + 1 ! the "+1" is necessary because COMTAL image
C coordinates range from 0 to 511 and the
C FORTRAN array indices range from 1 to 512.
DO 1000 ARACOL = 1, XDIM
OUTARA(ARAROW, ARACOL) = ICHAR(CBUF(IMCOL))
IMCOL = IMCOL + 1
1000 CONTINUE
2000 CONTINUE
RETURN ! successful termination
C C**ERROR RETURNS:
C
6661 WRITE(TERM, 9003) IMAGE, IMLO, IMHI
9003 FORMAT(' IMAGE NUMBER:', I3, ' OUT OF RANGE:', 214, '.')
6662 GOT0 8900
C
6101 WRITE(TERM, 8103) XDIM, 1, LNCNT
8103 FORMAT(' X DIMENSION:', I4, ' OUT OF RANGE:', 215, '.')
6102 GOT0 8900
C
6201 WRITE(TERM, 8203) YDIM, 1, LNCNT
8203 FORMAT(' Y DIMENSION:', I4, ' OUT OF RANGE:', 215, '.')
6202 GOT0 8900
C
8301 WRITE(TERM, 8303) UPLFX, LNLO, LNHI
8303 FORMAT(' X COORDINATE FOR CORNER:', I4, ' OUT OF RANGE:', 215, '.')
8304 GOT0 8900
C
8401 WRITE(TERM, 8403) UPLFY, YDIM, LNLLO, LNHI
8403 FORMAT(' Y COORDINATE FOR CORNER:', I4, ' OUT OF RANGE:', 215, '.')
8404 GOT0 8900
C
8501 WRITE(TERM, 8503) UPLFX, XDIM, LNLO, LNHI
8503 FORMAT(' X COORDINATE FOR THE CORNER AND THE X DIMENSION ',
1 ' OF THE ARRAY', ' OVERFLOW IMAGE BOUNDARIES.',
2 ' X COORDINATE = ', I4, ' X DIMENSION = ', I4, ' .',
3 ' IMAGE COORDINATE LIMITS ARE ', 215, '. ')
8504 GOT0 8900
C
8601 WRITE(TERM, 8603) UPLFX, YDIM, LNLO, LNHI
8603 FORMAT(' Y COORDINATE FOR THE CORNER AND THE Y DIMENSION ')
0137 - 1 'OF THE ARRAY',/,'OVERFLOW IMAGE BOUNDARIES'.
0138 - 2 'X COORDINATE =',14,'X DIMENSION =',14,/
0139 - 3 'IMAGE COORDINATE LIMITS ARE ',215,'
0140          GOTO 8900
0141 C
0142 8900 WRITE(TERM, 8903)
0143 8963 FORMAT('RDIRC FAILS. NO TRANSFER TAKES PLACE.')
0144          RETURN
0145          END
0146
0147
&RLUT T=00004 IS ON CR00021 USING 00018 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RLUT(TABLE, LUTNUM)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER TABLE(256) ! the values of the specified COMTAL look-up
0008 C ! table are read into this array. The 0 entry
0009 C ! goes into TABLE(1), .... the 255 entry goes
0010 C ! into TABLE(256).
0011 INTEGER LUTNUM ! the number of the COMTAL look-up table (called
0012 C ! "function memory" in the COMTAL literature).
0013 C
0014 C***INTRODUCTION:
0015 C
0016 C The subroutine Read Look-Up Table (LUT) reads the COMTAL mapping from
0017 C the integers 0-255 into the integer array TABLE. This LUT can be used
0018 C for grey level enhancements in the COMTAL. A similar subroutine
0019 C called RDPSU is used to read from a pseudocolor look-up table. This
0020 C routine is only used for grey scale look-up tables.
0021 C
0022 C***LANGUAGE:
0023 C
0024 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0025 C
0026 C***LIMITATIONS:
0027 C
0028 C Although 8 bits are sufficient for the look up table values, full
0029 C integers are used in TABLE. This format is dictated by the COMTAL
0030 C conventions as given in section 5.2.3.1.
0031 C
0032 C***SUBPROGRAMS CALLED:
0033 C
0034 C name source load remarks
0035 C ------ ----- ------- -----------------------------------------
0036 C RANGE &RANGE &RANGE logical function which determines if its 1st
0037 C parameter is within its 2nd and 3rd inclusive.
0038 C
0039 C***WRITTEN BY:
0040 C
0041 C The code on which this subprogram is based was written by
0042 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0043 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0044 C summer fellowship.

0045
C***REVISION HISTORY:
C
C***LOCAL VARIABLES:
C
LOGICAL RANGE  ! logical function which determines if its 1st
parameter is within its 2nd and 3rd inclusive.

INTEGER TERM  ! logical unit for terminal output

INTEGER LUTLO,LUTHI ! limits for COMTAL function memories

C***INITIALIZATIONS:
C
DATA  TERM/1/
DATA  LUTLO/1/, LUTHI/4/

C***PROCESSING
C
IF (.NOT.(RANGE(LUTNUM, LUTLO, LUTHI))) GOTO 8001 ! error return

Programming notes:
The EXEC command parameters are discussed in the HP RTE-6/VM
are discussed in section 5.2.3 of the COMTAL User's Manual.

The first parameter to EXEC identifies the EXEC command as
a read command. The second parameter identifies the resident
HP driver (36B) and gives the code (200B) that identifies this
operation, a transfer to a COMTAL function memory (Look-Up Table).
The third parameter gives the Look-Up Table values (TABLE),
and the fourth parameter gives the length of TABLE in words.
The fifth parameter is a COMTAL code that is described bit by
bit in the User's Manual. In short, bit 15 signifies write to
COMTAL, bit 14 designates function memory instead of pseudocolor,
bit 12 signifies standard replacement, and bits 8&9 identify the
function memory to be used. (Bits are numbered 15 high, 0 low).

CALL EXEC(1, 36B+200B, TABLE, 256, ((LUTNUM-1)*256) )
RETURN

C***ERROR RETURN
C
8001 WRITE(TERM, 8003) LUTNUM, LUTLO, LUTHI
8003 FORMAT( ' THE FUNCTION MEMORY NUMBER,,14, . IS OUT OF RANGE: ',
1   214, ' ')
8090 WRITE(TERM, 8901)
0091 8901 FORMAT(' RDLUT FAILS. NO TRANSFER FROM COMTAL.')
0092      END
**RDP5U T=00004 IS ON CR00021 USING 00018 BLKS R=0000**

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE RDP5U(TABLE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C****PARAMETER DECLARATIONS:
0006 C
0007 INTEGER TABLE(768);! the values of the specified COMTAL look-up
0008 C ! table are read into this array. The RED table
0009 C ! is in TABLE(1:256); the GREEN, in TABLE(257:512);
0010 C ! and the BLUE, in TABLE(513:768).
0011 C
0012 C****INTRODUCTION:
0013 C
0014 C The subroutine Read the Pseudocolor table reads the 3 COMTAL mappings
0015 C from 0-255 which comprise the pseudocolor table. Note that the values
0016 C are placed into TABLE in the order RED, GREEN, and BLUE.
0017 C
0018 C****LANGUAGE:
0019 C
0020 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0021 C
0022 C****LIMITATIONS:
0023 C
0024 C Although 8 bits are sufficient for the look-up table values, full
0025 C integers are used in TABLE. This format is dictated by the COMTAL
0026 C conventions as given in section 5.2.3.1.
0027 C
0028 C****SUBPROGRAMS CALLED:
0029 C
0030 C NONE.
0031 C
0032 C****WRITTEN BY:
0033 C
0034 C The code on which this subprogram is based was written by
0035 C NETTIE B. FAULCON, July, 1983. This subprogram was written by
0036 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0037 C summer fellowship.
0038 C
0039 C****REVISION HISTORY:
0040 C
0041 C
0042 C****LOCAL VARIABLES:
0043 C
0044 C NONE.

---

104
**initializations:**

```
0045 C
0046 C***initializations:
0047 C
0048 C  none
0049 C
0050 C***processing
0051 C
0052 C  Programming notes:
0053 C  The EXEC command parameters are discussed in the HP RTE-6/VM
0055 C  are discussed in section 5.2.5 of the COMTAL User's Manual.
0056 C
0057 C  The first parameter to EXEC identifies the EXEC command as
0058 C  a READ command. The second parameter identifies the resident
0059 C  HP driver (36B) and gives the code (300B) that identifies this
0060 C  operation, a transfer from the COMTAL pseudocolor table.
0061 C  The third parameter gives the array that will hold the values,
0062 C  and the fourth parameter gives the length of TABLE in words.
0063 C  The fifth parameter is a COMTAL code that is described bit by
0064 C  bit in the User's Manual. The DVR41 driver takes care of all the
0065 C  bits except 8&9 which identify the color to be transferred.
0066 C
0067 C  Note that we make three separate calls to EXEC. Each call fills a
0068 C  different section of TABLE with a different color of the COMTAL's
0069 C  pseudocolor table.
0070 C
0071 C  call exec( 1, 36B+300B, table(1), 256, 1*256 ) ! red
0072 C  call exec( 1, 36B+300B, table(257), 256, 0*256 ) ! green
0073 C  call exec( 1, 36B+300B, table(513), 256, 2*256 ) ! blue
0074 C  return
0075 C  end
```
SUBROUTINE RDTAB(TABLE, CNUMB, IMOGR)

INTEGER TABLE(16) ! holds the image/graphics table from COMTAL.
INTEGER CNUMB ! COMTAL image or graph number.

This subroutine Reads the TABLE associated with the image or graphics
memory plane identified by CNUMB (the COMTAL image or graphics number)
and IMOGR (which is either a 0, indicating an image, or 1, indicating
a graphics plane. The 16 word, 32 byte table goes into TABLE.

FORTRAN 77, the HP-1000 version for RTE-6/VM.

LIMITATIONS:
IMOGR must be either a 0 or a 1, and CNUMB must be between 1 and 9
inclusive. If either number is out of range, no transfer takes place.

SUBPROGRAMS CALLED:

name   source   load   remarks
-------- ------ ------ -----------------------------
RANGE   &RANGE XRANGE logical function that determines if its
first parameter is within the last two parameters.
BTEST   ------ ------ HP FORTRAN?? intrinsic function that takes an
integer argument and returns .TRUE. or .FALSE.

WRITTEN BY:
The code on which this subprogram is based was written by
NETTIE D. FAULCON, July, 1983. This modification is by
KEITH MILLER, June, 1984.
LOCAL VARIABLES:

LOGICAL RANGE ! function that ascertains if its first parameter is between (inclusive) its last 2 parameters.

LOGICAL BTEST ! is a certain bit on or off.

INTEGER TERM ! the logical unit for terminal output.

INTEGER DGLO, DGHI ! the limits on single Digit image/graphics numbers.

INTEGER BTLO, BTHI ! the limits on COMTAL graphics values.

xCOOR; bits numbered 0-15, right to left.

INITIALIZATIONS:

DATA TERM/1/
DATA DGLO/1/, DGHI/9/
DATA BTLO/0/, BTHI/1/

PROCESSING

IF (.NOT.(RANGE(CNUMB,DGLO,DGHI))) GOTO 8001 ! error return

IF (.NOT.(RANGE(IMOGR,BTLO,BTHI))) GOTO 8101 ! error return

Programming note:

The EXEC call below is to the DVR41 driver.

The first argument, "1", identifies the operation as a read.

The second argument has two parts: "368" identifies the DVR41 driver, and "5005" selects a transfer code = 3 operation of that driver.

The third argument, "TABLE", is the buffer that will hold the IGT Image/Graphics Table information after the EXEC is completed.

The fourth argument, "16", gives the length of TABLE in words.

The final parameter is a code to the COMTAL which identifies the mode of the transfer ("7\*4096"), mode 7; selects either image or graphics ("IMOGR\*128"); and gives the number of the image/graphics memory plane ("CNUMB-1").

CALL EXEC(1,368\*5005,TABLE,16, 7\*4096 + IMOGR\*128 + CNUMB-1)

RETURN

ERROR RETURNS

WRITE (TERM, 8003) CNUMB, DGLO, DGHI

FORMAT ('COMTAL NUMBER: ', 13, ' OUT OF RANGE: ', 212, '..')

GOTO 8900

WRITE (TERM, 8103) IMOGR, BTLO, BTHI
0091 8103 FORMAT(' IMAGE/GRAPHICS ARGUMENT MUST BE 0 OR 1. NOT ',14,'')
0092      GOTO 8900
0093 C
0094 8900 WRITE(TERM, 8901)
0095 8901 FORMAT(' RDTAB FAILS. NO TRANSFER. ')
0096      RETURN
0097      END
&RDTAR T=00004 IS ON CB00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER XCOORD ! X coordinate of the COMTAL target location.
0008 INTEGER YCOORD ! Y coordinate of the COMTAL target location.
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C This subroutine ReaDs the TArget (cursor) location from the COMTAL.
0013 C
0014 C***LANGUAGE:
0015 C
0016 C FORTRAN 77, the HP-1000 version for RTE-G/VM.
0017 C
0018 C***LIMITATIONS: NONE
0019 C
0020 C***SUBPROGRAMS CALLED: NONE
0021 C
0022 C***WRITTEN BY:
0023 C
0024 C The code on which this subprogram is based was written by
0025 C NETTIE D. FAULCON, July, 1983. This modification is by
0026 C KEITH MILLER, June, 1984.
0027 C
0028 C***REVISION HISTORY:
0029 C
0030 C
0031 C***LOCAL VARIABLES:
0032 C
0033 INTEGER IBUF(2) ! the buffer to hold the COMTAL data transfer
0034 C
0035 C***INITIALIZATIONS: NONE
0036 C
0037 C***PROCESSING
0038 C
0039 C Programming note:
0040 C The EXEC call is explained in detail in the
0041 C HP Programmer's Reference Manual for RTE-G/VM, p.2-19ff. This
0042 C transfer function for the COMTAL is discussed in the
0043 C COMTAL User's Manual, Section 5.2.4. In the EXEC call
0044 C that follows, the HP resident driver, DVR41, is called as
0045 C follows: the first parameter (1) signifies a read; the
0046 C second parameter is in two parts: 36B identifies the resident
0047 C DVR41 driver, and 4008 identifies the target transfer operation
0048 C of that driver; the third parameter (IBUF) will hold the COMTAL data
0049 C to be transferred, and the fourth parameter gives the length in words
0050 C (2); the final parameter is a COMTAL command code for the transfer,
0051 C (00000B).
0052 C
0053 CALL EXEC(1, 36B+4308, IBUF, 2, 00000B)
0054 XCOORD = IBUF(1)
0055 YCOORD = IBUF(2)
0056 RETURN
0057 END
&SETV T=00004 IS ON CR0021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SETV(IMAGE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER IMAGE ! the COMTAL image number to be assigned to the
0008 C ! video camera; 5 is traditional.
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C The subroutine "SET Video" establishes the image number that is to be
0013 C associated with the video camera, and displays the camera input.
0014 C
0015 C***LANGUAGE:
0016 C
0017 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0018 C
0019 C***LIMITATIONS:
0020 C
0021 C SETV should only be called once during a COMTAL session. If the
0022 C camera is already set to IMAGE and SETV is called, the COMTAL
0023 C freezes. Manually reset the COMTAL with the SHIFT- to release any
0024 C previous SETV command.
0025 C
0026 C Note that IMAGE should not be an image number used for COMTAL memory.
0027 C For programming convenience, we insist on a single digit. 5 is
0028 C traditional in this lab, but 5-9 will do.
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C
0034 C CMNDC CMNDC %CMNDC sends character strings to the COMTAL where
0035 C they are much like COMTAL keyboard commands.
0036 C DIGITG DIGITG %DIGIT character*1 function that returns '0'-'9'
0037 C
0038 C RANGE RANGE %RANGE logical function which determines if its 1st
0039 C argument is within the 2nd and 3rd, inclusive.
0040 C
0041 C***WRITTEN BY:
0042 C
0043 C The code on which this subprogram is based was written by
0044 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

<table>
<thead>
<tr>
<th>LOGICAL</th>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGE</td>
<td>function which determines if its 1st argument is within the 2nd &amp; 3rd arguments, inclusive.</td>
<td></td>
</tr>
<tr>
<td>CHARACTER*1</td>
<td>DIGIT</td>
<td>function that returns '0'-'9' according to integer input 0-9.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>IBUF(128)</td>
<td>holds COMTAL keyboard command strings for use with CMMND.</td>
</tr>
<tr>
<td>CHARACTER*255</td>
<td>CBUF</td>
<td>overlays IBUF.</td>
</tr>
<tr>
<td>EQUIVALENCE</td>
<td>(IBUF,CBUF)</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>VDLO,VDHI</td>
<td>the limits on video image number: one digit, but not a 0 reserved for image memory.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>TERM</td>
<td>logical unit number for terminal input</td>
</tr>
</tbody>
</table>

**INITIALIZATIONS:**

| DATA | VDLO/5/, VDHI/9/ |
| DATA | TERM/I/ |

**PROCESSING:**

```
IF (.NOT.(RANGE(IMAGE,VDLO,VDHI))) GOTO 0001 | error return
```

the following COMTAL command is expanded to:

```
Set Video image #I; Display Image #I
```

where #I is the digit equal to IMAGE.

This command will hang up the COMTAL if the camera is already set to IMAGE. However, we can't do a RELEASE just to be sure because if the camera is NOT set to IMAGE, then a RELEASE also hangs up! (catch-22.) If the mode 7 IGP table transfer is incorporated into DVR41, perhaps the table can be inquired about SET or not SET. However, the COMTAL Users Manual is not clear on that matter.

```
CBUF = 'S V '+'/'+DIGIT(IMAGE)+' " SD I '+'/'+DIGIT(IMAGE)
```

**ERROR RETURNS:**

112
0091 C
0092 8001 WRITE(TERM, 8003) IMAGE, VDLO, VDHI
0093 8003 FORMAT(' YOUR IMAGE ARGUMENT,,' I4,,' IS OUT OF RANGE:,' 2I4)
0094 C
0095    WRITE(TERM, 8901)
0096 8901 FORMAT(' SETV FAILS. NO ACTION TAKEN. ')
0097 C
0098    RETURN
0099    END
0100
&SPLIT T=00004 IS ON CR00021 USING 00005 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC.
0002 SUBROUTINE SPLIT(BYTE1, BYTE2, INBYTE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER BYTE1 ! on output, this integer holds the left (high order)
0008 INTEGER BYTE2 ! byte of INBYTE in its right (low order) byte.
0009 INTEGER BYTE2 ! on output, this integer holds the right (low order)
0010 INTEGER INBYTE ! the input integer whose two bytes are to be SPLIT.
0011 C
0012 C
0013 C***INTRODUCTION:
0014 C
0015 C The subroutine SPLIT takes each of the two bytes of INBYTE and
0016 C interprets each byte as a separate integer. The two resulting
0017 C numbers are assigned to BYTE1 (which gets the high order byte of
0018 C INBYTE) and to BYTE2 (which gets the low order byte).
0019 C
0020 C***LANGUAGE:
0021 C
0022 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0023 C
0024 C***LIMITATIONS:
0025 C
0026 C None.
0027 C
0028 C***SUBPROGRAMS CALLED:
0029 C
0030 C None.
0031 C
0032 C***WRITTEN BY:
0033 C
0034 C The code on which this subprogram is based was written by
0035 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0036 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0037 C summer fellowship.
0038 C
0039 C***REVISION HISTORY:
0040 C
0041 C
0042 C***LOCAL VARIABLES:
0043 C
0044 INTEGER IHOLD ! an INTEGER interpretation of bits
CHARACTER CHOLD(2) ! a CHARACTER interpretation of bits

INTEGER IMERGE ! an INTEGER interpretation of bits

CHARACTER CSPLIT(2) ! a CHARACTER interpretation of bits

EQUIVALENCE (IHOLD, CHOLD), (IMERGE, CSPLIT)

IMERGE = INBYTE
IHOLD = 0 ! zero out high order bits

CHOLD(2) = CSPLIT(1)
BYTE1 = IHOLD

CHOLD(2) = CSPLIT(2)
BYTE2 = IHOLD

RETURN

END
SUBROUTINE SPREAD(IMAGE)

**PARAMETER DECLARATIONS:**

INTEGER IMAGE ! the number of the COMTAL image to be contrast spread

**INTRODUCTION:**

The subroutine SPREAD takes as input and output a COMTAL image. The subroutine requires two passes through the image. On the first pass, SPREAD determines the high and low pixel values in the image. On the second pass, SPREAD replaces each pixel value \( X \) with \( (X - \text{lowest}) \times (255/(\text{highest-lowest})) \). If the lowest and highest value are identical, no pixels are replaced and no message is printed. If the lowest value is 0 and the highest 255, no pixels are replaced and no message is printed.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**LIMITATIONS:**

The present implementation reads from the COMTAL image twice. This could be speeded up by placing the pixel values into a virtual array during the first pass. Also, the decision not to change a uniform grey image at all is arbitrary (but defensible...how do you stretch a constant value?).

**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDIL2</td>
<td>&amp;RDIL2</td>
<td>&amp;RDIL2</td>
<td>reads a horizontal line of pixels from the COMTAL and places the values in an INTEGER array</td>
</tr>
<tr>
<td>WRIL2</td>
<td>&amp;WRIL2</td>
<td>&amp;WRIL2</td>
<td>writes a horizontal line of pixels from an integer array to a COMTAL image</td>
</tr>
<tr>
<td>HILO</td>
<td>&amp;HILO</td>
<td>&amp;HILO</td>
<td>passes through a COMTAL image and returns the highest and lowest pixel value found</td>
</tr>
<tr>
<td>IFIX</td>
<td>------</td>
<td>------</td>
<td>FORTRAN IV intrinsic function: converts from real to integer</td>
</tr>
<tr>
<td>FLOAT</td>
<td>------</td>
<td>------</td>
<td>FORTRAN IV intrinsic function: converts from real to integer</td>
</tr>
</tbody>
</table>

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.
C INTEGER to real.

C

C****WRITTEN BY:

C

The code on which this subprogram is based was written by

NETTIE D. FAULCON, July, 1983. This subprogram was written by

Keith Miller, July, 1984, with the support of a NASA-ASEE summer fellowship.

C

C****REVISION HISTORY:

C

C

C****LOCAL VARIABLES:

C

INTEGER IBUF(512) ! holds pixel values read/written to/from COMTAL
INTEGER HIGH, LOW ! high and low pixel values in the IMAGE
INTEGER PXHI, PXLO ! limits of pixel values.
INTEGER LNLO, LNHI ! limits on COMTAL line numbers
INTEGER ROW, COL ! indices for COMTAL images and the arrays.
INTEGER TERM ; logical unit for terminal output
REAL FACTOR ! the scaling factor for doing the contrast spread

C

C****INITIALIZATIONS:

C

DATA PXLO/0/, PXHI/255/
DATA LNLO/0/, LNHI/511/
DATA TERM/1/

C

C****PROCESSING

C

CALL HILO(HIGH, LOW, IMAGE) ! get highest and lowest pixel values
! in the image

IF (HIGH .LE. LOW) GOTO 9999 ! no processing required
WRITE(TERM, 3501) HIGH, LOW
3501 FORMAT('SPRED DIAGNOSTIC. HIGH AND LOW ARE ', 214, '.*)
IF ((HIGH .EQ. PXHI) .AND. (LOW .EQ. PXLO)) GOTO 9999 ! no processing

FACTOR = 255.0 / FLOAT(HIGH-LOW)

DO 4888 ROW = 1,512
   CALL RDIL2(IBUF, IMAGE, ROW)
DO 3000 COL = 1,512
IBUF(COL) = IFIX(FLOAT(IBUF(COL)-LOW) * FACTOR)

CONTINUE

CALL URIL2(IMAGE,ROW,IBUF)

CONTINUE

C

RETURN

END
&SUBI2 T=00004 IS ON CR00021 USING 00024 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE SUBI2(C, A, B)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER C ! the COMTAL image into which the difference between
0008 C ! image A and image B is placed by SUBI2 (C = A - B)
0009 INTEGER A, B ! the images whose difference is taken
0010 C
0011 C**INTRODUCTION:
0012 C
0013 C The subroutine "SUBtract Images #2" takes a pixel by pixel difference
0014 C of images A and B and places the resulting image into image C.
0015 C The truecolor image B is used in the processing of SUBI2, and will
0016 C be left as the combination of C, A, and B for red, green, and blue
0017 C respectively.
0018 C
0019 C Unlike SUBIM, which does no scaling or offsetting, SUBI2 offsets the
0020 C result of the subtraction by adding 128 to each pixel difference.
0021 C Thus, a 128 pixel value in image C after the call means that
0022 C the true value of the difference was 0. This offset can be handy when
0023 C many of the values of the difference are less than 0. After the offset
0024 C is added, any pixel values less than 0 are set to 0.
0025 C
0026 C**LANGUAGE:
0027 C
0028 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0029 C
0030 C**LIMITATIONS:
0031 C
0032 C The truecolor B image is destroyed during this operation. C is
0033 C obviously destroyed. This subroutine is accomplished using COMTAL
0034 C commands that exploit the pipeline processors. Because of this, the
0035 C processing steps are obscure. For example, there is no motivation
0036 C outside the COMTAL instructions for making the combination of C, A, and B
0037 C a color image. Readers should be aware of these obscurities before trying
0038 C to understand the code.
0039 C
0040 C If any offset difference is less than 0, the pixel value is set to 0.
0041 C
0042 C If any image number is out of range, an error message is printed and
0043 C no further processing takes place.
0044 C This subroutine assumes that 0 is not a legal image for the COMTAL
C configuration.

C

C

C

C

integer IBUF(128) ! a buffer for passing commands to COMTAL

character*255 CBUF ! character overlay for IBUF

equivalence (IBUF,CBUF)

logical RANGE ! function that determines if 1st parameter

is within 2nd and 3rd parameter

character*1 DIGIT ! returns a single digit character '0'

! to '9' for integer input 0-9.

-- INIALIZATIONS:

data IMLO/1/, IMHI/4/

data TERM/1/

data TERM/1/

if (.not.(range(A,IMLO,IMHI))) goto 8001 ! error return

if (.not.(range(B,IMLO,IMHI))) goto 8101 ! error return

if (.not.(range(C,IMLO,IMHI))) goto 8201 ! error return

C
The following character string sends a series of keyboard commands to the COMTAL. In the comments below, each command is explained. The notation $X$ where $X$ is either A, B, or C stands for the single character that corresponds to the single digit number associated with the parameter $X$.

In this notation, letters in caps were entered into CBUF, and lower case letters are the full commands filled in by the COMTAL.

NOTE: this code assumes that the digit 0 is NOT a legal value for the parameters A, B, and C.

The "$" separates COMTAL commands.

```
CBUF =
1 'UN I 8 $'
2 'AS T B $' 'DIGIT(C)/' 'DIGIT(A)/' 'DIGIT(B)/' $'
3 'D I 8 $'
4 'SE COM G - B $'
5 'A COM $'
6 'I $' 'DIGIT(C)/' D R $'
7 'D I $' 'DIGIT(C)/'$'
8 'SU COM '
9 'SE COM $'
```

CALL CMDND(IBUTF, 74)
RETURN
C
```
C***ERROR RETURNS
C
GOTO 8900
```

```
GOTO 8900
```

GOTO 8900
```
C
0138 0900 WRITE(TERM, 0901)
0139 0901 FORMAT(' SUB12 RETURNS WITHOUT FURTHER PROCESSING. ')
0140        RETURN
0141        END
0142
SUBIM T=00004 IS ON CR00021 USING 00004 BLKS R=0000

C CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE SUBIM(C, A, B)

C

INTEGER C ! the COMTAL image into which the difference between
INTEGER A, B ! image A and image B is placed by SUBIM (C = A - B)

C

INTRODUCTION:

The subroutine "SUBtract Images" takes a pixel by pixel difference
of images A and B and places the resulting image into image C.

The truecolor image B is used in the processing of SUBIM, and will
be left as the combination of C, A, and B for red, green, and blue
respectively.

LANGUAGE:

FORTRAN 77, the HP-1000 version for RTE-6/VM.

LIMITATIONS:

The truecolor B image is destroyed during this operation. C is
obviously destroyed. This subroutine is accomplished using COMTAL
commands that exploit the pipeline processors. Because of this, the
processing steps are obscure. For example, there is no motivation
outside the COMTAL instructions for making the combination of C, A, and B
a color image. Readers should be aware of these obscurities before trying
to understand the code.

The 3 images C, A, and B must be distinct.

If any difference is less than 0, the pixel value is set to 0.

SUBIM does no scaling or offsetting. SUBI2 does an automatic
scale and offset.

If any image number is out of range, an error message is printed and
no further processing takes place.

This subroutine assumes that 0 is not a legal image for the COMTAL
configuration.
**SUBPROGRAMS CALLED:**

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMND &amp;CMND</td>
<td>&amp;CMND</td>
<td>CMND Sends a command to the COMTAL as if the command were sent to the keyboard</td>
<td></td>
</tr>
<tr>
<td>RANGE &amp;RANGE</td>
<td>&amp;RANGE</td>
<td>RANGE logical function that determines if the 1st parameter is within the range of the 2nd &amp; 3rd.</td>
<td></td>
</tr>
<tr>
<td>DIGIT &amp;DIGIT</td>
<td>&amp;DIGIT</td>
<td>DIGIT character*1 function that returns '0'- '9' according to integer input 0-9.</td>
<td></td>
</tr>
</tbody>
</table>

**WRITTEN BY:**

KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.

**REVISION HISTORY:**

**LOCAL VARIABLES:**

| INTEGER  | IBUF(128)  | a buffer for passing commands to COMTAL |
| CHARACTER*255 | CBUF  | character overlay for IBUF |
| EQUIVALENC | (IBUF,CBUF)  |
| LOGICAL  | RANGE  | function that determines if 1st parameter is within 2nd and 3rd parameter |
| INTEGER  | IMLO, IMHI  | limits on COMTAL image numbers |
| INTEGER  | TERM  |
| CHARACTER*1 | DIGIT  | returns a single digit character '0' to '9' for integer input 0-9. |

**INITIALIZATIONS:**

| DATA  | IMLO/1, IMHI/4 |
| DATA  | TERM/1 |

**PROCESSING**

IF (.NOT. (RANGE(A,IMLO,IMHI))) GOTO 8801 ! error return
IF (.NOT. (RANGE(B,IMLO,IMHI))) GOTO 8811 ! error return
IF (.NOT. (RANGE(C,IMLO,IMHI))) GOTO 8821 ! error return

The following character string sends a series of keyboard commands to the COMTAL. In the comments below, each command is explained. The notation *A* where *X* is either A, B, or C stands for the single character that corresponds to the single
digit number associated with the parameter X.
In this notation, letters in caps were entered into CBUF, and lower case letters are the full commands filled in by the COMTAL.
NOTE: this code assumes that the digit 0 is NOT a legal value for the parameters A, B, and C.
The "$" separates COMTAL commands.

CBUF =

1 'UN I B $'/
2 'AS T E '/DIGIT(C)'/DIGIT(A)'/DIGIT(B) $'/
3 'D I E $'/
4 'SE COM G - B + B $'/
5 'A COM $'/
6 'I '/DIGIT(C) D R $'/
7 'D I '/DIGIT(C) $'/
8 'Su COM '
9 'Su COM '
10 'Su COM '
11 'Su COM '
12 'Su COM '
13 'Su COM '
14 'Su COM '
15 'Su COM '
16 'Su COM '
17 'Su COM '
18 'Su COM '
19 'Su COM '
20 'Su COM '
21 'Su COM '
22 'Su COM '
23 'Su COM '
24 'Su COM '
25 'Su COM '
26 'Su COM '
27 'Su COM '
28 'Su COM '
29 'Su COM '
30 'Su COM '
31 'Su COM '
32 'Su COM '
33 'Su COM '
34 'Su COM '
35 'Su COM '
36 'Su COM '
37 'Su COM '
38 'Su COM '
39 'Su COM '
40 'Su COM '
41 'Su COM '
42 'Su COM '
43 'Su COM '
44 'Su COM '
45 'Su COM '
46 'Su COM '
47 'Su COM '
48 'Su COM '
49 'Su COM '
50 'Su COM '
51 'Su COM '
52 'Su COM '
53 'Su COM '
54 'Su COM '
55 'Su COM '
56 'Su COM '
57 'Su COM '
58 'Su COM '
59 'Su COM '
60 'Su COM '
61 'Su COM '
62 'Su COM '
63 'Su COM '
64 'Su COM '
65 'Su COM '
66 'Su COM '
67 'Su COM '
68 'Su COM '
69 'Su COM '
70 'Su COM '
71 'Su COM '
72 'Su COM '
73 'Su COM '
74 'Su COM '
75 'Su COM '
76 'Su COM '
77 'Su COM '
78 'Su COM '
79 'Su COM '
80 'Su COM '
81 'Su COM '
82 'Su COM '
83 'Su COM '
84 'Su COM '
85 'Su COM '
86 'Su COM '
87 'Su COM '
88 'Su COM '
89 'Su COM '
90 'Su COM '
91 'Su COM '
92 'Su COM '
93 'Su COM '
94 'Su COM '
95 'Su COM '
96 'Su COM '
97 'Su COM '
98 'Su COM '
99 'Su COM '
100 'Su COM '
101 'Su COM '
102 'Su COM '
103 'Su COM '
104 'Su COM '
105 'Su COM '
106 'Su COM '
107 'Su COM '
108 'Su COM '
109 'Su COM '
110 'Su COM '
111 'Su COM '
112 'Su COM '
113 'Su COM '
114 'Su COM '
115 'Su COM '
116 'Su COM '
117 'Su COM '
118 'Su COM '
119 'Su COM '
120 'Su COM '
121 'Su COM '
122 'Su COM '
123 'Su COM '
124 'Su COM '
125 'Su COM '
126 'Su COM '
127 'Su COM '
128 'Su COM '
129 'Su COM '
130 'Su COM '
131 'Su COM '
132 'Su COM '
133 'Su COM '
134 'Su COM '
135 'Su COM '
136 'Su COM '
137 'Su COM '
138 'Su COM '
139 'Su COM '
140 'Su COM '
141 'Su COM '
142 'Su COM '
143 'Su COM '
144 'Su COM '
145 'Su COM '
146 'Su COM '
147 'Su COM '
148 'Su COM '
149 'Su COM '
150 'Su COM '
151 'Su COM '
152 'Su COM '
153 'Su COM '
154 'Su COM '
155 'Su COM '
156 'Su COM '
157 'Su COM '
158 'Su COM '
159 'Su COM '
160 'Su COM '
161 'Su COM '
162 'Su COM '
163 'Su COM '
164 'Su COM '
165 'Su COM '
166 'Su COM '
167 'Su COM '
168 'Su COM '
169 'Su COM '
170 'Su COM '
171 'Su COM '
172 'Su COM '
173 'Su COM '
174 'Su COM '
175 'Su COM '
176 'Su COM '
177 'Su COM '
178 'Su COM '
179 'Su COM '
180 'Su COM '
181 'Su COM '
182 'Su COM '
183 'Su COM '
184 'Su COM '
185 'Su COM '
186 'Su COM '
187 'Su COM '
188 'Su COM '
189 'Su COM '
190 'Su COM '
191 'Su COM '
192 'Su COM '
193 'Su COM '
194 'Su COM '
195 'Su COM '
196 'Su COM '
197 'Su COM '
198 'Su COM '
199 'Su COM '
200 'Su COM '
201 'Su COM '
202 'Su COM '
203 'Su COM '
204 'Su COM '
205 'Su COM '
206 'Su COM '
207 'Su COM '
208 'Su COM '
209 'Su COM '
210 'Su COM '
211 'Su COM '
212 'Su COM '
213 'Su COM '
214 'Su COM '
215 'Su COM '
216 'Su COM '
217 'Su COM '
218 'Su COM '
219 'Su COM '
220 'Su COM '
221 'Su COM '
222 'Su COM '
223 'Su COM '
224 'Su COM '
225 'Su COM '
226 'Su COM '
227 'Su COM '
228 'Su COM '
229 'Su COM '
230 'Su COM '
231 'Su COM '
232 'Su COM '
233 'Su COM '
234 'Su COM '
235 'Su COM '
236 'Su COM '
237 'Su COM '
238 'Su COM '
239 'Su COM '
240 'Su COM '
241 'Su COM '
242 'Su COM '
243 'Su COM '
244 'Su COM '
245 'Su COM '
246 'Su COM '
247 'Su COM '
248 'Su COM '
249 'Su COM '
250 'Su COM '
251 'Su COM '
252 'Su COM '
253 'Su COM '
254 'Su COM '
255 'Su COM '
256 'Su COM '
257 'Su COM '
258 'Su COM '
259 'Su COM '
260 'Su COM '
261 'Su COM '
262 'Su COM '
263 'Su COM '
264 'Su COM '
265 'Su COM '
266 'Su COM '
267 'Su COM '
268 'Su COM '
269 'Su COM '
270 'Su COM '
271 'Su CO
&THRSH T=E9004 IS ON CR00021 USING 00018 BLKS R=0000

0001  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004  C
0005  C
0006  C
0007  C INTEGER  OUTIMG  ! the COMTAL image number for the thresholded image.
0008  C INTEGER  INIMG ! the COMTAL image & for the image to be thresholded.
0009  C INTEGER  THRESH! the threshold pixel value: < threshold -> 0, 
0010  C >= threshold -> 255.
0011  C
0012  C
0013  C
0014  C The subroutine THRESHold creates a new image OUTIMG by examining the
0015  C pixel values in INIMG. If an INIMG pixel value is < THRESH, then the
0016  C corresponding pixel value in OUTIMG is 0. If the INIMG pixel value
0017  C is >= THRESH, the corresponding pixel value in OUTIMG is 255.
0018  C OUTIMG and INIMG need not be distinct.
0019  C
0020  C
0021  C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0022  C
0023  C
0024  C
0025  C
0026  C OUTIMG and INIMG must be COMTAL memory planes (1-4). If not, an
0027  C error message is printed and no thresholding takes place.
0028  C THRESH must be a value between 0-255, or a message is printed instead
0029  C of any thresholding.
0030  C
0031  C
0032  C
0033  C
0034  C
0035  C
0036  C
0037  C
0038  C
0039  C
0040  C
0041  C
0042  C
0043  C
0044  C

The code on which this subprogram is based was written by
C NETTIE D. FAULCON, July, 1983. This subprogram was written by
C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
C summer fellowship.
C
C***REVISION HISTORY:
C
C
C***LOCAL VARIABLES:
C
C INTEGER IBUF(512) | buffer for read/write of horizontal line of
C COMTAL pixels.
C INTEGER LINE | loop index that keeps track of the line
C in INIMG and OUTIMG being processed.
C INTEGER PXCNT | loop index that counts pixels in a line.
C LOGICAL RANGE | function that determines if its 1st argument
C lies within its 2nd and 3rd inclusive.
C INTEGER IMLO, IMHI | range of legal COMTAL image numbers.
C INTEGER PXLO, PXHI | range of legal COMTAL pixel values.
C INTEGER LNLO, LNHI | range of legal COMTAL line numbers.
C INTEGER ARALO, ARAHI | range of array holding a line of pixels.
C INTEGER TERM | logical unit for terminal output.
C
C***INITIALIZATIONS:
C
C DATA IMLO/1/, IMHI/4/
C DATA PXLO/0/, PXHI/255/
C DATA LNLO/0/, LNHI/511/
C DATA ARALO/1/, ARAHI/512/
C DATA TERM/1/
C
C***PROCESSING:
C
C IF (.NOT.(RANGE(OUTIMG, IMLO, IMHI))) GOTO 8001 | error return
C IF (.NOT.(RANGE(INIMG, IMLO, IMHI))) GOTO 8101 | error return
C IF (.NOT.(RANGE(THRESH, PXLO, PXHI))) GOTO 8201 | error return
C
C DO 2000 LINE = LNLO, LNHI
C CALL RDIL2(IBUF, INIMG, LINE)
C DO 1000 PXCNT = ARALO, ARAHI
C IF (IBUF(PXCNT) .LT. THRESH) GOTO 500
C ELSE...
C IBUF(PXCNT) = 255
C GOTO 1000
C
C THEN...
C IBUF(PXCNT) = 0
C
C 500
0001 1083  CONTINUE
0002         CALL WRIL2(OUTIMG, LINE, IBUF)
0003 2083  CONTINUE
0004         RETURN
0005 C
0006 C*ERROR RETURNS:
0007 C
0008 8081 WRITE(TERM, 8003) OUTIMG, IMLO, IMHI
0009 8083 FORMAT( 'THE OUTPUT IMAGE NUMBER', ',I5', ' IS OUT OF RANGE: ', 214)
0100     GOTO 8900
0101 C
0102 8181 WRITE(TERM, 8103) INIMG, IMLO, IMHI
0103 8183 FORMAT( 'THE INPUT IMAGE NUMBER', ',I5', ' IS OUT OF RANGE: ', 214)
0104     GOTO 8900
0105 C
0106 8281 WRITE(TERM, 8203) THRESH, PXLO, PXHI
0107 8283 FORMAT( 'THE THRESHOLD VALUE', ',I5', ' IS OUT OF RANGE: ', 214)
0108     GOTO 8900
0109 C
0110 8900 WRITE(TERM, 8901)
0111 8901 FORMAT( 'THRESH FAILS. OUTIMG NOT CHANGED.')
0112     RETURN
0113     END
&TSTII T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE TSTII(WHICH)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 INTEGER WHICH ! designates the COMTAL image in which the test
0008 C image is to be generated
0009 C
0010 C***INTRODUCTION:
0011 C
0012 C The subroutine Test Image I generates a distinctive pattern in one
0013 C of the COMTAL image memories. The pattern is generated with ascending
0014 C pixel values as you move to the right and down in the image. However,
0015 C when 255 is reached in either the x or y directions, the pixel values
0016 C restart at 0. This gives a slash across the screen appearance at the
0017 C discontinuity, and a gradual change in grey scale elsewhere.
0018 C
0019 C***LANGUAGE:
0020 C
0021 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0022 C
0023 C***LIMITATIONS:
0024 C
0025 C The test pattern developed here is always the same. An enhancement
0026 C might be to vary the repetition of the pixel values (here, each pixel
0027 C is identical to one of its horizontal neighbors and one of its vertical
0028 C neighbors).
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C ------- ------- ------- -----------------------------
0034 C WRILN &WRILN %WRILN given an integer buffer of at least 512 bytes.
0035 C WRILN writes a horizontal line of pixels to a
0036 C designated COMTAL image.
0037 C
0038 C***WRITTEN BY:
0039 C
0040 C The code on which this subprogram is based was written by
0041 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0042 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0043 C summer fellowship.
**REVISION HISTORY:**

**LOCAL VARIABLES:**

```
INTEGER BIGBUF(512) ! 1024 bytes of pixel values
INTEGER I ! pointer into BIGBUF
INTEGER ROW ! designates a COMTAL image row
```

**INITIALIZATIONS:**

```
DO 100 1 = 0,255
BIGBUF(I+1) = I * 257 ! I*256 numbers the high byte
BIGBUF(I+257) = BIGBUF(I+1) ! facilitates the wraparound effect
100 CONTINUE
```

**PROCESSING**

```
DO 200 ROW = 0,511
CALL WRILN( WHICH, ROW, BIGBUF((ROW/2)+1) )
200 CONTINUE
```

**END**
&TV2C4 T=00004 IS ON CR00021 USING 00006 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE TV2C4
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C***PARAMETER DECLARATIONS:
0006 C
0007 C None.
0008 C
0009 C***INTRODUCTION:
0010 C
0011 C The subroutine "TV to Comtal image transfer #4" digitizes four
0012 C copies of the current TV image, and averages them together into a
0013 C single image stored in COMTAL image #1. COMTAL images 1, 2, and 3
0014 C are used for storage. Image 5 must be set to the video camera image
0015 C before TV2C4 is called.
0016 C
0017 C***LANGUAGE:
0018 C
0019 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0020 C
0021 C***LIMITATIONS:
0022 C
0023 C Images 1, 2, 3, and 4 are changed by this routine.
0024 C
0025 C This subroutine assumes that image 5 has been set to the video
0026 C camera previous to the call. If 5 is not set to video, the COMTAL
0027 C hangs up.
0028 C
0029 C
0030 C***SUBPROGRAMS CALLED:
0031 C
0032 C name source load remarks
0033 C-------- ------- ------- -----------------------------------------------
0034 C RANGE &RANGE XRANGE logical function that determines if its 1st
0035 C argument lies within 2nd & 3rd argument, inclusive.
0036 C TV2CM &TV2CM xTV2CM digitizes the camera associated with COMTAL image
0037 C 5 into the 1st argument, an image memory.
0038 C ADD12 &ADD12 xADD12 adds two images, pixel by pixel, and divides the
0039 C sums by last argument, to produce a new image.
0040 C
0041 C***WRITTEN BY:
0042 C
0043 C The code on which this subprogram is based was written by
0044 C NETTIE D. FAUCON, July, 1983. This subprogram was written by

131
**KEITH MILLER, July, 1984, with the support of a NASA-ASEE summer fellowship.**

---

**REVISION HISTORY:**

---

**LOCAL VARIABLES:**

- **LOGICAL RANGE**: function that determines if its 1st argument is within its 2nd & 3rd, inclusive.
- **INTEGER TVIMAG**: COMTAL image number associated with the video camera.

**INITIALIZATIONS:**

- **DATA TVIMAG**: arbitrary convention used at our lab.
- **DATA TERM**:

**PROCESSING:**

- **CALL DSPVD**
- **CALL TV2CM(1)**
- **CALL TV2CM(2)**
- **CALL ADDI2(3,1,2,2)**: put average of images 1&2 into image 1
- **CALL TV2CM(1)**
- **CALL TV2CM(2)**
- **CALL ADDI2(4,1,2,2)**: put average of images 2&3 into image 2
- **CALL ADDI2(1,3,4,2)**: put average of images 1&2 into image 1
- **CALL DSPBW(1)**

**RETURN**

**END**
**TV2CM**

**INTEGER DECLARATIONS:**

The subroutine TV2CM digitizes a "snapshot" of the current TV image into the COMTAL image number CMIMAG. Previous to the TV2CM call, image 5 must be set to the video camera.

**INTRODUCTION:**

FORTRAN 77, the HP-1000 version for RTE-G/VM.

**LIMITATIONS:**

The video camera must be associated with TVIMAGE before TVCM is called. TVIMAGE is 5 as an arbitrary convention in this lab.

CMIMAG must identify a COMTAL image memory (IMLO to IMHI).

**SUBPROGRAMS CALLED:**

*DIGIT* &*DIGIT* &*DIGIT* character*^1* function that returns '0'..'9' according to integer input 0..9.

**WRITTEN BY:**

The code on which this subprogram is based was written by

**writes:**

NETTIE D. FAULCON. July, 1983. This subprogram was written by

**writes:**

KEITH MILLER. July, 1984, with the support of a NASA-ASEE
C summer fellowship.

C

C***REVISION HISTORY:

C

C

C***LOCAL VARIABLES:

C

C INTEGER IMLO, IMHI ! limits on COMTAL image memory numbers.

C INTEGER TVIMAGE ! the COMTAL image & associated with the

C video camera.

C LOGICAL RANGE ! function that determines if the 1st argument

C is within the 2nd and 3rd arguments, inclusive.

C CHARACTER*1 DIGIT ! function that returns '0'..'9' according

C to integer input 0..9.

C INTEGER TERM ! logical unit for terminal output.

C INTEGER IBUF(128)! buffer for CMMND COMTAL command strings.

C CHARACTER*255 CBUS ! overlays IBUF.

C EQUIVALENCE : (IBUF.CBUS)

C INTEGER TVIMAG ! COMTAL image & associated with video image:

C ! arbitrarily set to 5 in this lab.

C

C***INITIALIZATIONS:

C

C DATA IMLO/1/, IMHI/4/ ! arbitrary convention for our lab.

C DATA TVIMAGE/5/ !

C DATA TERM/1/

C

C***PROCESSING:

C

C IF (.NOT.(RANGE(CMIMAG,IMLO,IMHI))) GOTO 8001 ! error return

C

C

C let @C and @V be the digits associated with CMIMAG and TVIMAGE:

C then the following CMMND string is expanded by the COMTAL into:

C Display Image @V: Image @C = Displayed image Red

C

C CBUF = 'D I '/DIGIT(TVIMAGE)/ ' SI '/DIGIT(CMIMAG)/ ' D R '

C

C CALL CMMND(IBUF, 15)

C CALL DSPBLU(CMIMAG)

C RETURN

C

C***ERROR RETURNS:

C

C

C 8001 WRITE(TERM, 8003) CMIMAG, IMLO, IMHI

C 8003 FORMAT(' THE COMTAL IMAGE NUMBER,'14,' IS OUT OF RANGE:'214)
0091   GOTO 8900
0092   C
0093   C
0094  8900 WRITE(TERM,8901)
0095  8901 FORMAT(' TV2CM FAILS. NO DIGITIZING TAKES PLACE. ')
0096   C
0097    RETURN
0098    END
**INTRODUCTION:**

WAIT pauses until <CR> is pressed on the HP keyboard.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**SUBPROGRAMS CALLED:** NONE

**WRITTEN BY:**

KEITH MILLER, NASA-Langley ASEE fellow, 1984

**REVISION HISTORY:** NONE

**LOCAL VARIABLES:**

INTEGER TERM ! logical unit of the terminal
INTEGER IDUMMY ! facilitates the read that forces a pause

**INITIALIZATIONS:**

DATA TERM/1/

**PROCESSING:**

WRITE(TERM, 1001)
1001 FORMAT(' PUSH <CR> TO CONTINUE.' )
READ(TERM, 2001) IDUMMY
2001 FORMAT(I2)
RETURN
END
&WIPGR T=00004 IS ON CR00021 USING 00012 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0045  C        INTEGER      TERM            ! within a certain range.
0046  C        INTEGER      IDUMMY          ! the logical unit number for the terminal.
0047  C        INTEGER      GRLO,GRHI      ! fills the place of an unused EXEC parameter.
0049  C        INTEGER      GRLO,GRHI      ! limits on a graphics plane number.
0050  C        CHARACTER*1 DIGIT         ! function that returns '0'-'9' for input
0051  C        INTEGER      TERM          ! integers 0-9.
0053  C        C***INITIALIZATIONS:
0055  C        DATA      TERM/1/
0056  C        DATA      GRLO/1/,GRHI/4/
0057  C        C***PROCESSING
0059  C        IF (.NOT.(RANGE(GRNUM,GRLO,GRHI))) GOTO 8001 ! error return
0060  C        "Subtract Graphics GRNUM", where GRNUM stands for the digit
0062  C        corresponding to GRNUM value.
0063  C        CALL CMMN2( 'SUB GR' //DIGIT(GRNUM) )
0065  RETURN
0066  C        C***ERROR RETURN:
0066  C        8001 WRITE(TERM, 8003) GRNUM, GRLO, GRHI
0070  8003 FORMAT( 'THE GRAPHICS NUMBER',I3,', OUT OF RANGE:',2I4)
0071  C        8900 WRITE(TERM, 8901)
0073  8901 FORMAT( 'DSPGR fails. No action taken on command.' )
0074  RETURN
0075  END
&WRGLN T=00004 IS ON CR00021 USING 00016 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE WRGLN(GRAPH, LINE, ONOFFS)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER GRAPH ! COMTAL image graphics line to be written to.
0008 INTEGER LINE ! which horizontal line to be written to;
0009 C ! lines numbered from 1 (screen top) to 512.
0010 INTEGER ONOFFS(32) ! 16 bits per integer, 512 bits in a line.
0011 C
0012 C**INTRODUCTION:
0013 C
0014 C This subroutine writes a Graphics Line to the COMTAL. The line of
0015 C bits is coded as 1 for on and 0 for off. Each bit in ONOFFS must be
0016 C set by the caller of WRGLN.
0017 C
0018 C**LANGUAGE:
0019 C
0020 FORTRAN 77, the HP-1000 version for RTE-6/VM.
0021 C
0022 C**LIMITATIONS:
0023 C
0024 C If GRAPH or LINE is out of range, an error message is printed
0025 C and no transfer takes place.
0026 C
0027 C**SUBPROGRAMS CALLED:
0028 C
0029 name source load remarks
0030 ------ ------ ------ -----------------------------
0031 C RANGE &RANGE %RANGE logical function that determines if its
0032 C first parameter is within the last two parameters.
0033 C
0034 C**WRITTEN BY:
0035 C
0036 C The code on which this subprogram is based was written by
0037 C KETTIE D. FAULCON, July, 1983. This modification is by
0038 C KEITH MILLER, June, 1984.
0039 C
0040 C**REVISION HISTORY:
0041 C
0042 C**LOCAL VARIABLES:
0043 C
0044 LOGICAL RANGE ! function that ascertains if its first parameter
C INTEGER TERM  ! is between (inclusive) its last 2 parameters
C INTEGER GRLO, GRHI  ! the limits on COMTAL monochrome image numbers
C INTEGER LNLO, LNHI  ! the limits on COMTAL image line numbers
C
C***INITIALIZATIONS:
C
DATA TERM/1/
DATA GRLO/1/, GRHI/4/
DATA LNLO/0/, LNHI/511/
C
C***PROCESSING
C
IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return
C
Programming note:
C
The EXEC call is explained in detail in the
C
HP Programmer's Reference Manual for RTE-6/VM.p.2-19ff. This
C
transfer function for the COMTAL is discussed in the
C
COMTAL User's Manual, Section 5.2.2.1. In the EXEC call
C
that follows, the HP resident driver called DVR41 is called as
C
follows: the first parameter (2) signifies a write; the
C
second parameter is in two parts: 36B identifies the resident
C
DVR41 driver, and 1088 identifies the line transfer operation
C
of that driver; the third parameter (ONOFFS) holds the data to be
C
transferred, and the fourth parameter gives ONOFFS' length in words
C
(32); and the final parameter is a COMTAL command code for the transfer.
C
CALL EXEC(2,36B+1088,ONOFFS.32,(GRAPH-1)*2048 + LINE + 512)
C
RETURN
C
C***ERROR RETURNS
C
C
8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 8003 FORMAT(' GRAPHICS NUMBER:, I3, ' OUT OF RANGE:, 212, '.')
8001 GOTO 8900
C
8101 WRITE(TERM, 8103) LINE, LNLO, LNHI
8103 8103 FORMAT(' LINE NUMBER:, I4, ' OUT OF RANGE:, 213, '.')
8005 GOTO 8900
C
8900 WRITE(TERM, 8901)
8901 FORMAT(' WRGLN FAILS. NO TRANSFER.')
C
RETURN
C
END
**Introduction:**

This subroutine writes a graph point to the COMTAL. The value of the graphics is an on/off decision, represented in the call by an integer that must be a 1 or 0. If the graph number, coordinates, or on/off value are out of range, an error message is printed and no transfer takes place.

**Language:**

FORTRAN 77, the HP-1000 version for RTÉ-6/VM.

**Limitations:**

If GRAPH, XCOORD, YCOORD, or VALUE are out of range, an error message is printed at the terminal and no transfer takes place.

**Subprograms Called:**

- RANGE &RANGE XRANGE logical function that determines if its first parameter is within the last two parameters.
- IGSET ------- ------- HP FORTRAN77 intrinsic function that takes an integer argument and returns that integer with one bit set, according to the second argument.
- IBCLR ------- ------- HP FORTRAN77 intrinsic function that takes an integer argument and returns that integer with one bit cleared, according to the second argument.

**Written By:**

The code on which this subroutine is based was written by NETTIE D. FAULCON, July, 1983. This modification is by
**LOCAL VARIABLES:**

```
LOGICAL RANGE ! function that ascertains if its first parameter
    is between (inclusive) its last 2 parameters

INTEGER TERM ! the logical unit for terminal output

INTEGER GRLO, GRHI ! the limits on COMTAL monochrome graph numbers

INTEGER LNLO, LNHI ! the limits on COMTAL graph line numbers

INTEGER BTLO, BTHI ! the limits on COMTAL graphics values

INTEGER BITS(32) ! a buffer to read & write a COMTAL graphics line

INTEGER WORD ! which word of BITS holds the bit selected by XCOORD.

INTEGER BIT ! which bit in BITS(WORD) holds the bit selected by

XCOORD; bits numbered 0-15, right to left.
```

**INITIALIZATIONS:**

```
DATA TERM/1/
DATA GRLO/1/, GRHI/4/
DATA LNLO/0/, LNHI/511/
DATA BTLO/0/, BTHI/1/
```

**PROCESSING**

```
IF (.NOT.(RANGE(GRAPH,GRLO,GRHI))) GOTO 8001 ! error return

IF (.NOT.(RANGE(XCOORD,LNLO,LNHI))) GOTO 8101 ! error return

IF (.NOT.(RANGE(VALUE,BTLO,BTHI))) GOTO 8301 ! error return
```

Programming note:

The EXEC calls below are to the DVR41 driver. The first call
is identical to the call made in RDILN. The second EXEC call
is identical to the one in URILN. See the documentation for
those subroutines for details on these calls.

Read the COMTAL line (horizontal) that contains the point in question:

```
CALL EXEC(1,3GB+100B,BITS.32,(GRAPH-1)*2048 + 512 + YCOORD)
```

Change the single bit that has been selected:

```
WORD = (XCOORD/16) + 1
BIT = (16*WORD) - XCOORD - 1
```
C
IF (VALUE .EQ. 0) GOTO 1000
C ELSE...VALUE .EQ. I
BITS(WORD) = IBSET(BITS(WORD),BIT)
GOTO 2000
C THEN...VALUE .EQ. 0
1000 BITS(WORD) = IBCLR(BITS(WORD),BIT)
GOTO 2000
C
Write the graph line with one changed graphics to COMTAL
2000 CALL EXEC(2,368+1008,BITS,32,(GRAPH-1)*2048 + 512 + YCOORD)
RETURN
C
***ERROR RETURNS
8001 WRITE(TERM, 8003) GRAPH, GRLO, GRHI
8003 FORMAT(' GRAPH NUMBER:', 13, ' OUT OF RANGE: ', 212, '.')
GOTO 8900
8101 WRITE(TERM, 8103) XCOORD, LNLO, LNHI
8103 FORMAT(' X COORDINATE:', 14, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8201 WRITE(TERM, 8203) YCOORD, LNLO, LNHI
8203 FORMAT(' Y COORDINATE:', 14, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8301 WRITE(TERM, 8303) VALUE, BTLO, BTHI
8303 FORMAT(' BIT VALUE:', 14, ' OUT OF RANGE: ', 214, '.')
GOTO 8900
8900 WRITE(TERM, 8901)
8901 FORMAT(' WRGPT FAILS. NO TRANSFER.')
RETURN
END
SUBROUTINE WRIL2(IMAGE, LINE, INTS)

INTEGER IMAGE ! COMTAL image number to be written to
INTEGER LINE ! which horizontal line to be written to:
INTEGER INTS(512) ! 512 values, one integer per pixel, to

INTRODUCTION:
This subroutine, WRITE Image LINE 2, writes a line of pixels to the
COMTAL. The input array INTS has a two byte integer for each pixel,
but the COMTAL only uses the lower order byte of each integer. Therefore,
WRIL2 strips off the upper byte before sending the pixels to the COMTAL.
WRIL2 is very similar to WRILN, which writes out a line of bytes.

LANGUAGE:
FORTRAN 77, the HP-1000 version for RTE-G/VM.

LIMITATIONS:
The INTS array must be completely "full". The IMAGE designated
must be a monochrome image. The LINE parameter must be between
0 and 511. If IMAGE or LINE is out of range, an error message is printed
and no transfer takes place.

SUBPROGRAMS CALLED:

RANGE &RANGE &RANGE logical function that determines if its
first parameter is within the last two parameters,
CHAR ----- ----- intrinsic HP FORTRAN77 function that strips off
the upper byte of an integer and returns the lower
byte as a character.

WRITTEN BY:
The code on which this subprogram is based was written by
**REVISION HISTORY:**

**LOCAL VARIABLES:**

- INTEGER PIXELS(256) ! holds lower order bytes of INTS values
- CHARACTER*1 CPIX (512) ! overlays PIXELS
- EQUIVALENCE (PIXELS, CPIX)

- LOGICAL RANGE ! function that ascertains if its first parameter
  is between (inclusive) its last 2 parameters
- INTEGER TERM ! the logical unit for terminal output
- INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers
- INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers
- INTEGER INDEX ! indexes into arrays in a loop

**INITIALIZATIONS:**

- DATA TERM/I/
- DATA IMLO/I/, IMHI/4/
- DATA LNLO/0/, LNHI/511/

**PROCESSING**

- IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8001 ! error return
- IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return

- strip off the upper bytes of the pixel values in INTS
- DO 1000 INDEX=1,512
  - CPIX(INDEX) = CHAR(INTS(INDEX)) ! CPIX overlays PIXELS
- 1000 CONTINUE

Programming note:

The EXEC call is explained in detail in the HP Programmer's Reference Manual for RTE-6/M, p.2-19ff. This transfer function for the COMTAL is discussed in the COMTAL User's Manual, Section 5.2.2.1. In the EXEC call that follows, the HP resident driver called DVR41 is called as follows: the first parameter (2) signifies a write; the second parameter is in two parts: 368 identifies the resident DVR41 driver, and 1008 identifies the line transfer operation transferred, and the fourth parameter gives PIXELS' length in words
C (256); and the final parameter is a COMTAL command code for the transfer.

CALL EXEC(2, 368+100B, PIXELS, 256, (IMAGE-1)*2048 + LINE)

RETURN

C**ERROR RETURNS

C

WRITE (TERM, 0003) IMAGE, IMLO, IMHI

FORMAT(" IMAGE NUMBER: ", I3, ", OUT OF RANGE: ", 212, ",")

GOTO 8900

C

WRITE (TERM, 0103) LINE, LNLO, LNHI

FORMAT(" LINE NUMBER: ", I4, ", OUT OF RANGE: ", 213, ",")

GOTO 8900

C

WRITE (TERM, 8901)

FORMAT(" WRIL2 FAILS, NO TRANSFER.")

RETURN

END
SUBROUTINE WRILN(IMAGE, LINE, PIXELS)

PARAMETER DECLARATIONS:

INTEGER IMAGE ! COMTAL image number to be written to
INTEGER LINE ! which horizontal line to be written to:
INTEGER PIXELS ! lines numbered from 1 (screen top) to 512.
INTEGER PIXELS(256)! 512 bytes (pixels) to be transferred

INTRODUCTION:
This subroutine WRites an Image Line to the COMTAL. The line of pixels is made up of 8 bit (0-255) grey scale intensities. The PIXELS array is assumed to be completely full.

LANGUAGE:
FORTRAN 77, the HP-1000 version for RTE-6/VM.

LIMITATIONS:
The PIXELS array must be completely "full". The IMAGE designated must be a monochrome image. The LINE parameter must be between 8 and 511. If IMAGE or LINE is out of range, an error message is printed and no transfer takes place.

SUBPROGRAMS CALLED:

name       source     load     remarks
-----------------------
RANGE       xRANGE    xRANGE   logical function that determines if its first parameter is within the last two parameters.

WRITTEN BY:
The code on which this subprogram is based was written by
NETTIE D. FAULCON, July, 1983. This modification is by
KEITH MILLER, June, 1984.
LOCAL VARIABLES:

LOGICAL RANGE ! function that ascertains if its first parameter
is between (inclusive) its last 2 parameters

INTEGER TERM ! the logical unit for terminal output

INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers

INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers

C***INITIALIZATIONS:

DATA TERM/1/
DATA IMLO/1/, IMHI/4/
DATA LNLO/0/, LNHI/511/

C***PROCESSING

IF (.NOT.(RANGE(IMAGE, IMLO, IMHI))) GOTO 8801 ! error return
IF (.NOT.(RANGE(LINE, LNLO, LNHI))) GOTO 8101 ! error return

Programming note:
The EXEC call is explained in detail in the
HP Programmer's Reference Manual for RTE-6/VM p.2-19ff. This
transfer function for the COMTAL is discussed in the
COMTAL User's Manual, Section 5.2.2.1. In the EXEC call
that follows, the HP resident driver called DVR41 is called as
follows: the first parameter (2) signifies a write; the
second parameter is in two parts: 36B identifies the resident
DVR41 driver, and 100B identifies the line transfer operation
of that driver; the third parameter (PIXELS) holds the data to be
transferred, and the fourth parameter gives PIXELS' length in words
(256); and the final parameter is a COMTAL command code for the transfer.

CALL EXEC(2, 36B+100B, PIXELS, 256, (IMAGE-1)*2048 + LINE)

RETURN

C***ERROR RETURNS

WRITE(TERM, 8003) IMAGE, IMLO, IMHI
WRITE(TERM, 8103) LINE, LNLO, LNHI

RETURN
0091   FORMAT(' WRILN Fails. NO TRANSFER.')
0092       RETURN
0093       END
**SUBROUTINE WRIPT** (IMAGE, XCOORD, YCOORD, VALUE)

**PARAMETER DECLARATIONS:**

**IMAGE** ! COMTAL image number to be written to.

**XCOORD, YCOORD** ! point where new value is to be written to.

**VALUE** ! 0-255 pixel value to be written to image pt.

**INTRODUCTION:**

This subroutine writes an Image Point to the COMTAL. The value of the pixel is an 8 bit (0-255) grey scale intensity. If the image number, coordinates, or value are out of range, an error message is printed and no transfer takes place.

**LANGUAGE:**

FORTRAN 77, the HP-1000 version for RTE-6/VM.

**LIMITATIONS:**

The PIXELS array must be completely "full". The IMAGE designated must be a monochrome image. The LINE parameter must be between 0 and 511. If IMAGE, XCOORD, or YCOORD are out of range, an error message is printed at the terminal and no transfer takes place.

**SUBPROGRAMS CALLED:**

**RANGE & RANGE X RANGE** logical function that determines if its first parameter is within the last two parameters.

**WRITTEN BY:**

The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This modification is by KEITH MILLER, June, 1984.

**REVISION HISTORY:**
LOCAL VARIABLES:

LOGICAL RANGE   ! function that ascertains if its first parameter
                ! is between (inclusive) its last 2 parameters
INTEGER TERM    ! the logical unit for terminal output
INTEGER IMLO, IMHI ! the limits on COMTAL monochrome image numbers
INTEGER LNLO, LNHI ! the limits on COMTAL image line numbers
INTEGER PXLO, PXHI ! the limits on COMTAL pixel values
INTEGER PIXELS(256)   ! a buffer to read & write a COMTAL line
CHARACTER*1 BYTES(512) ! overlay for PIXELS buffer
EQUIVALENCE (PIXELS,BYTES)
INTEGER IHOLD   ! hold an integer for byte conversion
CHARACTER*1 CHAR(2) ! overlay for IHOLD
EQUIVALENCE (IHOLD,CHARS)

INITIALIZATIONS:

DATA TERM/1/
DATA IMLO/1/, IMHI/4/
DATA LNLO/0/, LNHI/511/
DATA PXLO/0/, PXHI/255/

PROCESSING

IF (.NOT.(RANGE(IMAGE,IMLO,IMHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(XCOORD,LNLO,LNHI))) GOTO 8101 ! error return
IF (.NOT.(RANGE(YCOORD,LNLO,LNHI))) GOTO 8201 ! error return
IF (.NOT.(RANGE(VALUE,PXLO,PXHI))) GOTO 8301 ! error return

Programming note:

The EXEC calls below are to the DVR41 driver. The first call
is identical to the call made in RDILN. The second EXEC call
is identical to the one in WRILN. See the documentation for
those subroutines for details on these calls.

Read the COMTAL line (horizontal) that contains the point in question:

CALL EXEC(1, 36B+130D, PIXELS, 255, (IMAGE-1)*2048 + YCOORD)

Change the single byte that needs changing:

IHOLD = VALUE ! transfers the pixel VALUE to CHAR(2).
BYTES(XCOORD+1) = CHAR(2) ! transfers VALUE to PIXELS in proper position
                           ! '++' converts from 0-255 pixels to 1-256
                           ! FORTRAN array.
0091 C Write the image line with one changed pixel to COMTAL
0092 C
0093 CALL EXEC(2, 368+100B, PIXELS, 256, (IMAGE-1)*2048 + YCOORD)
0094 RETURN
0095 C
0096 C***ERROR RETURNS
0097 C
0098 8001 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
0099 8003 FORMAT(\"IMAGE NUMBER:, I3, \"OUT OF RANGE:, 212,\")
0100 GOTO 8900
0101 C
0102 8101 WRITE(TERM, 8103)XCOORD, LNLO, LNHI
0103 8103 FORMAT(\"X COORDINATE:, I4, \"OUT OF RANGE:, 214,\")
0104 GOTO 8900
0105 C
0106 8201 WRITE(TERM, 8203)YCOORD, LNLO, LNHI
0107 8203 FORMAT(\"Y COORDINATE:, I4, \"OUT OF RANGE:, 214,\")
0108 GOTO 8900
0109 C
0110 8301 WRITE(TERM, 8303) VALUE, PXLO, PXHI
0111 8303 FORMAT(\"PIXEL VALUE:, I4, \"OUT OF RANGE:, 214,\")
0112 GOTO 8900
0113 C
0114 8900 WRITE(TERM, 8901)
0115 8901 FORMAT(\"WRIT FAILS, NO TRANSFER.\")
0116 RETURN
0117 END
SUBROUTINE WRIRC(IMAGE, UPLFX, UPLFY, INPARA, XDIM, YDIM)

INTEGER IMAGE ! the number of the COMTAL image to which
INTEGER UPLFX, UPLFY ! the image coordinates of the upper left
INTEGER XDIM, YDIM ! dimensions of the input array, INPARA
INTEGER INPARA(XDIM, YDIM) ! the array holding the new pixel values

The subroutine WRite Image ReCtangle transfers pixel values from an
integer array to a portion of a COMTAL image. Note that although
pixel values are generally stored with 1 byte/pixel, WRIRC takes as
input an array of integers in which each integer holds one pixel value.
XDIM, YDIM, IMAGE, UPLFX, and UPLFY are all checked for possible out
of range errors before any transfer is attempted.

FORTRAN 77, the HP-1000 version for RTE-6/VM.

WRIRC does a great deal of error checking before initiating the
transfer. If the programmer can verify that all calls to WRIRC
will be legal, these checks could be commented out to improve
machine efficiency. Also, the calls to the subroutines RDIL2
and WRIL2 could be replace by inline code, which is not long.

SUBPROGRAMS CALLED:

<table>
<thead>
<tr>
<th>name</th>
<th>source</th>
<th>load</th>
<th>remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDIL2</td>
<td>&amp;RDIL2</td>
<td></td>
<td>WRIL2 transfers a line of bytes from a COMTAL image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>into an HP array of integers.</td>
</tr>
<tr>
<td>WRIL2</td>
<td>&amp;WRIL2</td>
<td></td>
<td>WRIL2 transfers a buffer of integers to a COMTAL image</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>as a line of bytes (1 byte/integer).</td>
</tr>
</tbody>
</table>
C. RANGE  & RANGE  & RANGE logical function that determines of its first parameter is within the last two parameters.

C CHAR  ------ ------ HP FORTRAN intrinsic function converts a 2 byte integer into a one byte char (removes high order byte).

C***WRITTEN BY:
C
C The code on which this subprogram is based was written by NETTIE D. FAULKON, July, 1983. This code was written by KEITH MILLER, July, 1984.

C***REVISION HISTORY:

C
C
C***LOCAL VARIABLES:

C
C INTEGER IMROW, IMCOL | a location in the image
C INTEGER ARAROW, ARACOL | a location in INPARA
C INTEGER ICHAR | intrinsic byte to integer conversion
C LOGICAL RANGE | function that ascertains if the first parameter is within the last two parameters.
C INTEGER IMLO, IMHI | limits on COMTAL image numbers
C INTEGER LNLO, LNHI | limits on COMTAL pixel coordinates
C INTEGER DIMLIM, | limit on the dimensions of INPARA
C INTEGER TERM | logical unit for terminal output
C INTEGER IBUF(512) | buffer to hold COMTAL horizontal line

C***INITIALIZATIONS:

C
C DATA TERM/1/
C DATA IMLO/1/, IMHI/4/
C DATA LNLO/6/, LNHI/511/
C DATA DIMLIM/64/

C***PROCESSING

C

IF (.NOT. (RANGE (IMAGE, IMLO, IMHI )))GOTO 8001 | error return
IF (.NOT. (RANGE (XDIM, 1, DIMLIM )))GOTO 8101 | error return
IF (.NOT. (RANGE (YDIM, 1, DIMLIM )))GOTO 8201 | error return
IF (.NOT. (RANGE (UPLFX, LNLO, LNHI )))GOTO 8301 | error return
IF (.NOT. (RANGE (UPLFY, LNLO, LNHI )))GOTO 8401 | error return
IF (.NOT. (RANGE (UPLFX+DIM-1, LNLO, LNHI )))GOTO 8501 | error return
IF (.NOT. (RANGE (UPLFY+YDIM-1, LNLO, LNHI )))GOTO 8601 | error return
we get to this point if the transfer is to take place

IMROW = UPLFY

DO 2000 ARAROW = 1, YDIM
   CALL RDIL2(IBUF, IMAGE, IMROW)
   ! the next line initializes the column pointer:
   IMCOL = UPLFX + 1 ! the "+1" is necessary because COMTAL image
   ! coordinates range from 0 to 511 and the
   ! FORTRAN array indices range from 1 to 512.

DO 1000 ARACOL = 1, XDIM
   IBUF(IMCOL) = INPARA(ARAROW, ARACOL)
   IMCOL = IMCOL + 1 ! increment for next 1000 loop pass

CONTINUE

CALL WRIL2(IMAGE, IMROW, IBUF)
IMROW = IMROW + 1 ! increment for next 2000 loop pass

CONTINUE

RETURN ! successful termination

***ERROR RETURNS:

6061 WRITE(TERM, 8003) IMAGE, IMLO, IMHI
6062 8003 FORMAT(IMAGE NUMBER, , I3, " OUT OF RANGE:", 214,'.'
6063 GOTO 8900

6101 WRITE(TERM, 8103) XDIM, 1, DIMLIM
6102 8103 FORMAT(" X DIMENSION", , I4, " OUT OF RANGE:", 215,'.'
6103 GOTO 8900

6191 WRITE(TERM, 8203) YDIM, 1, DIMLIM
6192 8203 FORMAT(" Y DIMENSION", , I4, " OUT OF RANGE:", 215,'.'
6193 GOTO 8900

8301 WRITE(TERM, 8303) UPLFX, LNLO, LNHI
8302 8303 FORMAT(" X COORDINATE FOR CORNER", , I4, " OUT OF RANGE:", 215,'.'
8303 GOTO 8900

8401 WRITE(TERM, 8403) UPLFY, LNLO, LNHI
8402 8403 FORMAT(" Y COORDINATE FOR CORNER", , I4, " OUT OF RANGE:", 215,'.'
8403 GOTO 8900

8501 WRITE(TERM, 8503) UPLFX, XDIM, LNLO, LNHI
8502 8503 FORMAT(" X COORDINATE FOR CORNER AND THE X DIMENSION ",
8503 1 " OF THE ARRAY",/ " OVERFLOW IMAGE BOUNDARIES.",
8504 2 " X COORDINATE =", 14, "X DIMENSION =", 14, /
8505 3 " IMAGE COORDINATE LIMITS ARE ", 215, ")
8505 GOTO 8900

GOTO 8900
0137 C-
0138 8661 WRITE(TERM, 8603) UPLFY, YDIM, LNLO, LNHI
0139 8603 FORMAT(' Y COORDINATE FOR THE CORNER AND THE Y DIMENSION ',
0140      1 ' OF THE ARRAY',/, ' OVERFLOW IMAGE BOUNDARIES.',
0141      2 ' X COORDINATE = ', 14, ' X DIMENSION =', 14, '/,
0142      3 ' IMAGE COORDINATE LIMITS ARE ', 215, '.')
0143     GOTO 8900
0144 C-
0145 8900 WRITE(TERM, 8903)
0146 8903 FORMAT(' WRIRC FAILS. NO TRANSFER TAKES PLACE. ')
0147     RETURN
0148     END
0149
0150

156
&URLUT T=00004 IS ON CR00021 USING 00004 BLKS R=0000

0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE URLUT(LUTNUM, TABLE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C###PARAMETER DECLARATIONS:
0006 C
0007 INTEGER LUTNUM ! the number of the COMTAL look-up table (called
0008 ! "function memory" in the COMTAL literature).
0009 INTEGER TABLE(256) ! the values to be placed in the look-up table;
0010 ! notice that 0 maps to TABLE(1), 1 maps to
0011 ! TABLE(2), ..., and 255 maps to TABLE(256).
0012 C
0013 C###INTRODUCTION:
0014 C
0015 C The subroutine WRITE Look-Up Table (LUT) establishes a COMTAL mapping
0016 C from the integers 0-255 to the elements in TABLE. This LUT can be
0017 C used for grey level enhancements in the COMTAL. A similar subroutine
0018 C called WRPSU is used to establish a pseudocolor look-up table. This
0019 C routine is only used for grey scale look-up tables.
0020 C
0021 C###LANGUAGE:
0022 C
0023 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0024 C
0025 C###LIMITATIONS:
0026 C
0027 C Although 8 bits are sufficient for the look-up table values, full
0028 C integers are used in TABLE. This format is dictated by the COMTAL
0029 C conventions as given in section 5.2.3.1.
0030 C
0031 C###SUBPROGRAMS CALLED:
0032 C
0033 C name source load remarks
0034 C ------- ------ ------- -----------------------------------------------
0035 C RANGE $RANGE $RANGE logical function which determines if its 1st
0036 C parameter is within its 2nd and 3rd inclusive.
0037 C
0038 C###WRITTEN BY:
0039 C
0040 C The code on which this subprogram is based was written by
0041 C NEETIE D. FAULCON, July, 1983. This subprogram was written by
0042 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0043 C summer fellowship.
0044 C
LOGICAL RANGE ! logical function which determines if its 1st parameter is within its 2nd and 3rd inclusive.
INTEGER TERM ! logical unit for terminal output
INTEGER LUTLO, LUTHI ! limits for COMTAL function memories

DATA TERM/1/
DATA LUTLO/1/, LUTHI/4/

IF (.NOT.(RANGE(LUTNUM, LUTLO, LUTHI))) GO TO 8801 ! error return

Programming notes:
The EXEC command parameters are discussed in the HP RTE-G/VM Programmer's Reference Manual, 2-19ff. The COMTAL parameters are discussed in section 5.2.3 of the COMTAL User's Manual.

The first parameter to EXEC identifies the EXEC command as a write command. The second parameter identifies the resident HP driver (36B) and gives the code (268B) that identifies this operation, a transfer to a COMTAL function memory (Look-Up Table).
The third parameter gives the Look-Up Table values (TABLE), and the fourth parameter gives the length of TABLE in words.
The fifth parameter is a COMTAL code that is described bit by bit in the User's Manual. In short, bit 15 signifies write to COMTAL, bit 14 designates function memory instead of pseudocolor, bit 12 signifies standard replacement, and bits 8 & 9 identify the function memory to be used. (Bits are numbered 15 high, 0 low).

CALL EXEC( 2, 36B+268B, TABLE, 256, ((LUTNUM-1)*256) )
RETURN

WRITE(TERM, 8803) LUTNUM, LUTLO, LUTHI
8823 FORMAT(' THE FUNCTION MEMORY NUMBER','14, '.', IS OUT OF RANGE:',
1 214, ',')
8900 WRITE(TERM, 8901)
8901 FORMAT(' LUT FAILS. NO TRANSFER TO COMTAL.')
END
0001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0002 SUBROUTINE WRPSLI (TABLE)
0003 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0004 C
0005 C**PARAMETER DECLARATIONS:
0006 C
0007 INTEGER TABLE(768) ! the values of the COMTAL look-up
0008 C ! table are read from this array. The RED table
0009 C ! is in TABLE(1:256); the GREEN, in TABLE(257:512);
0010 C ! and the BLUE, in TABLE(513:768).
0011 C
0012 C**INTRODUCTION:
0013 C
0014 C The subroutine WRITE the PSEudocolor table writes the 3 COMTAL mappings
0015 C from 0-255 which comprise the pseudocolor table. Note that the values
0016 C are placed into TABLE in the order RED, GREEN, and BLUE.
0017 C
0018 C**LANGUAGE:
0019 C
0020 C FORTRAN 77, the HP-1000 version for RTE-6/VM.
0021 C
0022 C**LIMITATIONS:
0023 C
0024 C Although 8 bits are sufficient for the look-up table values, full
0025 C integers are used in TABLE. This format is dictated by the COMTAL
0026 C conventions as given in section 5.2.3.1.
0027 C
0028 C**SUBPROGRAMS CALLED:
0029 C
0030 C NONE.
0031 C
0032 C**WRITTEN BY:
0033 C
0034 C The code on which this subprogram is based was written by
0035 C NETTIE D. FAULCON, July, 1983. This subprogram was written by
0036 C KEITH MILLER, July, 1984, with the support of a NASA-ASEE
0037 C summer fellowship.
0038 C
0039 C**REVISION HISTORY:
0040 C
0041 C
0042 C**LOCAL VARIABLES:
0043 C
0044 C NONE.
C**INITIALIZATIONS:
C
C
C
C
C**PROCESSING

Programming notes:
The EXEC command parameters are discussed in the HP RTE-6/VM Programmer’s Reference Manual, 2-19ff. The COMTAL parameters are discussed in section 5.2.3 of the COMTAL User’s Manual.

The first parameter to EXEC identifies the EXEC command as a write command. The second parameter identifies the resident HP driver (36B) and gives the code (300B) that identifies this operation, a transfer from the COMTAL pseudocolor table.
The third parameter gives the array that will hold the values, and the fourth parameter gives the length of TABLE in words.
The fifth parameter is a COMTAL code that is described bit by bit in the User’s Manual. The DVR41 driver takes care of all the bits except B&9 which identify the color to be transferred.

Note that we make three separate calls to EXEC. Each call fills a different section of the pseudocolor table from the TABLE array.

CALL EXEC( 2, 36B+300B, TABLE(1), 256, 1*256 ) ! red
CALL EXEC( 2, 36B+300B, TABLE(257), 255, 0*256 ) ! green
CALL EXEC( 2, 36B+300B, TABLE(513), 256, 2*256 ) ! blue
RETURN
SUBROUTINE URTAR(XCOORD, YCOORD)

INTEGER XCOORD  ! X coordinate of the desired target location.
INTEGER YCOORD  ! Y coordinate of the desired target location.

This subroutine writes a target location to the COMTAL, thereby "positioning" the COMTAL cursor.

FORTRAN 77, the HP-1000 version for RTE-6/VM.

Both the XCOORD and YCOORD must be within the range 0 to 511. If not, an error message is printed and no transfer takes place to the COMTAL.

The code on which this subprogram is based was written by NETTIE D. FAULCON, July, 1983. This modification is by KEITH MILLER, June, 1984.

The code on which this subprogram is based was written by

The logical unit for terminal output
**INITIALIZATIONS:**

```plaintext
DATA TERM/1/
DATA LNLO/0/, LNHI/511/
```

**PROCESSING**

```plaintext
IF (.NOT.(RANGE(XCOORD, LNLO, LNHI))) GOTO 8001 ! error return
IF (.NOT.(RANGE(YCOORD, LNLO, LNHI))) GOTO 8101 ! error return

Programming note:
The EXEC call is explained in detail in the
HP Programmer's Reference Manual for RTE-6/VM, p.2-19ff. This
transfer function for the COMTAL is discussed in the
COMTAL User's Manual, Section 5.2.4. In the EXEC call
that follows, the HP resident driver called DVR41 is called as
follows: the first parameter (2) signifies a write; the
second parameter is in two parts: 36B identifies the resident
DVR41 driver, and 400B identifies the target transfer operation
of that driver; the third parameter (IBUF) contains the two coordinates
to be transferred, and the fourth parameter gives the length of the
buffer in words. The last parameter is ignored.

```plaintext
IBUF(1) = XCOORD
IBUF(2) = YCOORD
CALL EXEC(2, 36B+400B, IBUF, 2, IDUMMY)
RETURN
```

**ERROR RETURNS**

```plaintext
0801 WRITE(TERM, 8003) XCOORD, LNLO, LNHI
0803 FORMAT(1X, 'XCOORD:', 5, ' OUT OF RANGE:1', 214, 1X)
GOTO 8900
0802 C
0803 WRITE(TERM, 8103) YCOORD, LNLO, LNHI
0805 FORMAT(1X, 'YCOORD:', 5, ' OUT OF RANGE:', 214, 1X)
GOTO 8900
0806 C
0807 WRITE(TERM, 8901)
0809 FORMAT('WRITE FAILS. NO TRANSFER.1')
0810 RETURN
0811 END
```
&TADD2.T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001  PROGRAM TADD2
0002   C
0003   C  TEST THE ADDPROGRAM ADDI2 (ADD IMAGES)
0004   C
0005   C  KEITH MILLER 7/17/84
0006   C
0007   C  LOAD MODULES: XTADD2, XADDI2, XCMDND, XRANGE, XDIGIT
0008   C
0009   CALL ADDI2(0,1,2,1) ! SHOULD GIVE OUT OF RANGE ERROR.
0010   CALL ADDI2(1,0,2,1) ! *
0011   CALL ADDI2(1,2,0,1) ! *
0012   C
0013   CALL ADDI2(3,1,2,1) ! SHOULD WORK.
0014   C
0015   STOP
0016   END
&TADDI T=60004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TADDI
0002 C
0003 C TEST THE ADDPROGRAM ADDIM (ADD IMAGES)
0004 C
0005 C KEITH MILLER 7/17/84
0006 C
0007 C LOAD MODULES: XTADDI, XADDIM, %CMND, %RANGE, %DIGIT
0008 C
0009 C CALL ADDIM(0,1,2,1) ! SHOULD GIVE OUT OF RANGE ERROR
0010 CALL ADDIM(1,0,2,1) ! *
0011 CALL ADDIM(1,2,0,1) ! *
0012 C
0013 C CALL ADDIM(3,1,2,1)
0014 C
0015 STOP
0016 END
&TCLR  T=00004 IS ON CR00021 USING 0001 BLKS R=0000

0001       PROGRAM TCLR
0002       C
0003       C    tests the DSPCL subroutine
0004       C
0005       CALL DSPCL(0,1,2,3)
0006       STOP
0007       END
0008
0009
0010
0011
0012
0013
0014
0001 PROGRAM TCLRG
0002 C
0003 C TEST CLEAR GRAPHICS
0004 C
0005 C KEITH MILLER, 7/2/84
0006 C
0007 C LOAD MODULES: %TCLRG, %CLRGR, %RANGE, %CMMND, %DIGIT
0008 C
0009 C INTEGER IBUF(128)
0010 C CHARACTER*20 CBUF ! OVERLAYS THE FIRST 10 ELEMENTS OF IBUF
0011 C EQUIVALENCE (IBUF, CBUF)
0012 C
0013 C CBUF = 'CLEAR GR 2'
0014 C
0015 C CALL CLRGR(0)
0016 C CALL CLRGR(1)
0017 C CALL CMMND(IBUF, 10)
0018 C CALL CLRGR(5)
0019 C
0020 C STOP
0021 C END
0022
0023
0024
0025
0026
PROGRAM TCLRI

TEST CLEAR IMAGE

KEITH MILLER, 7/2/84

LOAD MODULES: %TCLRI, %CLRIM, %RANGE, %CMDND, %DIGIT

INTEGER IBUF(128)
CHARACTER*20 CBUF ! OVERLAYS THE FIRST 10 ELEMENTS OF IBUF
EQUIVALENCE (IBUF, CBUF)

CBUF = 'CLEAR IMAGE 2'

CALL CLRIM(0)
CALL CLRIM(1)
CALL CMDND(IBUF, 13)
CALL CLRIM(5)

STOP
END
&TCMM2 T=60004 IS ON CR00021 USING 00003 BLKS R=0000

0001 PROGRAM TCMM2
0002 C
0003 C TEST CMNND AGAIN
0004 C
0005 C KEITH MILLER, 7/12/84
0006 C
0007 C LOAD MODULES TCMM2, CMNND, WMNND
0008 C
0009 INTEGER INNUM, I, IBUF(128)
0010 CHARACTER*255 CBUF
0011 EQUVALENCE (IBUF, CBUF)
0012 C
0013 DO 1000 I = 0.256
0014 WRITE(1,500) I
0015 500 FORMAT('TESTING CHARACTER *', I4)
0016 CALL WAIT
0017 CBUF = 'G 2 PRO 1'
0018 CALL CMNND(IBUF,9)
0019 CBUF = CHAR(I)
0020 CALL CMNND(IBUF,1)
0021 1000 CONTINUE
0022 STOP
0023 END
&TCMMN T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001       PROGRAM  TCMMN
0002 C      Keith Miller, 6/16/84
0003 C      Testing SUBROUTINE CMMND
0004 C
0005 C      LOAD MODULES REQUIRED: %TCMMN, %CMMND
0006 C
0007 C      INTEGER    IBUF(20)
0008 C
0009 C      WRITE(1,1) -
0010 C      FORMAT(’ TO EXIT THIS TEST, ENTER AN EMPTY STRING’)
0011 C
0012  1      WRITE(1,10)
0013 C      FORMAT(’ ENTER ASCII STRING’)
0014  5      READ(1,20) IBUF
0015  10     FORMAT(128A2)
0016 C      N = ITLOG() ! ITLOG gives the number of characters typed in
0017 C      IF (N .EQ. 0) GOTO 999
0018 C      CALL CMMND(IBUF,N)
0019 C      GO TO 5
0020 C      999   END
&TCHT T=03004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TCNT
0002 C
0003 C TEST THE SUBROUTINE COUNT
0004 C
0005 C KEITH MILLER 7/11/84
0006 C
0007 C LOAD MODULES: TCNT, COUNT, RANGE, RDIL2
0008 C
0009 INTEGER 4 I4BUF(256)
0010 INTEGER INDEX
0011 C
0012 CALL COUNT(I4BUF, 1)
0013 DO 1000 INDEX = 0, 31
0014 WRITE(1,999) (I4BUF((INDEX+J)+J), J=1,8)
0015 999 FORMAT(819)
0016 1000 CONTINUE
0017 C
0018 STOP
0019 END
&TCONS T=03004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TCONS
0002 C
0003 C TEST CONCATENATION OF CHARACTER STRING, FORTRAN 77
0004 C
0005 INTEGER IHOLD
0006 CHARACTER*1 CHOLD(2)
0007 EQUIVALENCE (IHOLD, CHOLD)
0008 CHARACTER*5 STRING
0009 C
0010 IHOLD = 6 + 600
0011 STRING = '***' // CHOLD(2) // '***'
0012 WRITE(1,1000) CHOLD(2), STRING
0013 1000 FORMAT( ' LETTER=', A1, '. STRING=', A5, '.')
0014 STOP
0015 END
TCOPY T=O2004 IS ON CR80021 USING 80001 BLKS R=GO99

0001 PROGRAM TCOPY
0002 C TEST ICOPY
0003 C LOAD MODULES REQUIRED: XTCOPY, XICOPY, XCMND, XRANGE,
0004 C XDIGIT
0005 C CALL ICOPY(1,2) ! SHOULD GIVE OUT OF RANGE ERROR
0006 CALL ICOPY(5,1) ! SHOULD GIVE OUT OF RANGE ERROR
0009 CALL ICOPY(2,-3) ! SHOULD GIVE OUT OF RANGE ERROR
0010 STOP
0011 END
&TDIGI T=E0004 IS ON C00021 USING 00002 BLKS R=0000

0001    PROGRAM TDIGI
0002    C
0003    C      TEST DIGIT
0004    C      KEITH MILLER, 7/2/84
0005    C
0006    C      LOAD MODULES: &TDIGI, &DIGIT, &RANGE
0007    C
0008    C      INTEGER INT
0009    C      CHARACTER*1 HCHAR
0010    C
0011    DO 1000 INT = -1, 12
0012        CALL DIGIT(HCHAR, INT)
0013        WRITE(1,501) INT, HCHAR
0014      501 FORMAT(' INTEGER INPUT AND CHARACTER OUTPUT:',1I3,1A1,'.'
0015 1000 CONTINUE
0016    C
0017    STOP
0018  END
0019
0020
0021
0022
0023
0024
0025
&TDSP T=6004 IS ON CR00021 USING 00001 BLKS R=0000

0001   PROGRAM TDSP
0002   C
0003   C TEST DSPBW AND DSPCL
0004   C
0005   C LOAD MODULES REQUIRED: %TDSP, %DSPBW, %DSPCL, %CMMND,
0006   C %WAIT, %RANGE, %DIGIT
0007   C
0008   CALL DSPBW(1)
0009   CALL WAIT
0010   CALL DSPBW(2)
0011   CALL WAIT
0012   CALL DSPBW(3)
0013   CALL WAIT
0014   CALL DSPBW(0)
0015   CALL WAIT
0016   CALL DSPCL(1,2,3,8)
0017   STOP
0018   END
0019
0020
0021
PROGRAM THIST

C TEST THE SUBROUTINE HISTO

C KEITH MILLER, 7/31/84

C LOAD MODULES: %THIST, %HISTO, %DIGIT, %CMMN2, %RANGE

INTEGER TABLE(256)

INTEGER I ! implicit do loop index

CALL HISTO(1)

STOP

END
PROGRAM THORM

TEST THE PROCEDURE NORML.

KEITH MILLER JULY 6, 1983

LOAD MODULES: THORM, NORML, HILO, RDIL2, WRIL2, RANGE

CALL NORML(2)

STOP

END
0001  PROGRAM TNOTE
0002  C
0003  C TEST THE SUBROUTINES NOTES AND NOTE2
0004  C
0005  C KEITH MILLER, 7/26/84
0006  C
0007  C LOAD MODULES: TNOTE, NOTES, NOTE2, DSPGR, WRTAR,
0008  C CMN2, CMNND, DIGIT, RANGE, DELAY
0009  C
0010  CHARACTX255 CBUF
0011  INTEGER IBUF(128)
0012  EQUIVALENCE (CBUF, IBUF)
0013  C
0014  CALL NOTE2(1,100,100,'B',3,'1.BLUE')
0015  CALL NOTE2(1,100,200,'R',3,'2.RED')
0016  CALL NOTE2(1,100,300,'G',3,'3.GREEN')
0017  CALL NOTE2(1,100,400,'S',3,'4.SAME')
0018  C CBUF(I:II) = 'USING NOTES'
0019  CALL NOTES(1,10,10,'R',1,CBUF,11)
0020  CALL NOTES(5,10,10,'R',1,CBUF,11)
0021  CALL NOTES(1,512,10,'R',1,CBUF,11)
0022  CALL NOTES(-1,10,'R',1,CBUF,11)
0023  CALL NOTES(1,10,512,'R',1,CBUF,11)
0024  CALL NOTES(1,10,10,'R',1,CBUF,11)
0025  CALL NOTES(1,10,10,'R',0,CBUF,11)
0026  CALL NOTES(1,10,10,'R',17,CBUF,11)
0027  CALL NOTES(1,10,10,'R',17,CBUF,11)
0028  CALL NO
0029  CALL DSPGR(1)
0030  CALL DSPGR(2)
0031  STOP
0032  END
PROGRAM TPNT

TEST THE SUBROUTINE PAINT

KEITH MILLER, 7/16/84

LOAD MODULES: TPNT, PAINT, RANGE, URINC, RDTAR,
RDIL2, URIL2, CMMND, DSPBW

CALL PAINT(1,35, 200)

STOP

END
&TPROF T=0004 IS ON CR00021 USING 00002 BLKS R=0000

0001 PROGRAM TPROF
0002 C
0003 C TEST PROFL SUBROUTINE
0004 C
0005 C KEITH MILLER, 7/12/84
0006 C
0007 C LOAD MODULES: %TPROF, %PROFL, %RANGE, %CMDND, %WAIT,
0008 C       %DIGIT
0009 C
0010 C CALL PROFL(1,1)
0011 C WRITE(1,1001)
0012 1001 FORMAT(' SUCCESSFULLY RETURNED TO CALLER OF PROFILER')
0013 C STOP
0014 C END
PROGRAM TRANG
C TESTS THE RANGE FUNCTION
C LOAD MODULES REQUIRED: TRANG, RANGE
C
LOGICAL RANGE, ANSWER
C
WRITE(1,1)
1 FORMAT(' SHOULD BE T F ERROR-F')
C
WRITE(1,2001) RANGE(2,1,3)
2001 FORMAT(' ', L1)
WRITE(1,2001) RANGE(1,2,3)
ANSWER = RANGE(3,2,1)
WRITE(1,2001) ANSWER
C
STOP
END
&TRDTA T=60004 IS ON CRO0021 USING 02002 BLKS R=0000

0301 PROGRAM TRDTA
0302 C TEST THE SUBROUTINE RDTAB
0303 C KEITH MILLER, 7/19/84
0304 C LOAD MODULES: %TRDTA, %RDTAB, %RANGE
0305 C INTEGER TABLE(16), INDEX, WHICH
0306 C
0307 DO 2000 WHICH = 1,4
0308 CALL RDTAB(TABLE, WHICH, 0)
0309 WRITE(1,999)WHICH, TABLE(1), TABLE(2), TABLE(3), TABLE(4)
0310 999 FORMAT(I2, ', ', 409)
0311 2000 CONTINUE
0312 C
0313 STOP
0314 END
program tsetv

* test the setv subroutine

keith miller, 7/17/84

load modules: tsetv, setv, range, digit, cmd, dspbw

call setv(2) !should be an error
call setv(11) !should be an error
call setv(5) !should work

stop
end
PROGRAM TSPRD

TESTS THE PROGRAM TSPRED.

KEITH MILLER, JULY 5, 1984

LOAD MODULES: TSPRD, XSPPRED, XRDIL2, XWRIL2, XRENAGE

INTEGER IBUF(512), I1, I2, I3, INDEX

WRITE(1, 1001)
C1001 FORMAT(' GIVE THREE SHADES, USING THE FORMAT 314: ')
READ (1, 1003) I1, I2, I3
C1003 FORMAT(314)

DO 2000 INDEX=1, 170
IBUF(INDEX) = I1
IBUF(INDEX+17) = I2
IBUF(INDEX+340) = I3
C2000 CONTINUE

IBUF(511) = I3
IBUF(512) = I3

DO 3000 INDEX=0, 511
CALL WRIL2(I, INDEX, IBUF)
C3000 CONTINUE

CALL SPRED(1)

STOP

END
0001 PROGRAM TSUBI
0002 C TESTS SUBROUTINES SUBIM AND SUBI2
0003 C KEITH MILLER 7/23/64
0004 C LOAD MODULES: %TSUBI, %SUBIM, %SUBI2, %RANGE, %DIGIT, %WAIT, %CMD
0005 C
0006 C
0007 C
0008 C
0009 C
0010 C
0011 CALL SUBIM(1,3,2) ! no offset!
0012 CALL WAIT
0013 CALL SUBI2(1,3,2) ! 128 offset!
0014 C
0015 STOP
0016 END
PROGRAM TTHRS

C TESTS THE SUBROUTINE THRESHOLD

C KEITH MILLER 7/25/84

C LOAD MODULES: TTHRS, THRSH, RANGE, RDIL2, WRIL2

INTEGER THRESH! THRESHOLD FROM USER

INTEGER IN, OUT! TWO IMAGES

WRITE(1,1001)

1001 FORMAT(' GIVE THE OUTPUT IMAGE NUMBER:')

READ(1,1003) OUT

1003 FORMAT(I)

WRITE(1,1005)

1005 FORMAT(' GIVE THE INPUT IMAGE NUMBER:')

READ(1,1003) IN

WRITE(1,1007)

1007 FORMAT(' GIVE THE THRESHOLD PIXEL VALUE:')

READ(1,1009) THRESH

1009 FORMAT(I)

CALL THRSH(OUT, IN, THRESH)

STOP

END
&TTSTI T=0004 IS ON CR0021 USING 00001 BLKS R=0000

0001       PROGRAM TTSTI
0002       C
0003       C       TESTS THE TEST IMAGES
0004       C
0005       C       LOAD MODULES: $TTSTI, $TSTII, $WRLN, $RANGE
0006       C
0007       CALL TSTII(1)
0008       STOP
0009       END
&TTV2C T=0004 IS ON CR00021 USING 00002 BLKS R=0000

0001               PROGRAM TTV2C
0002              C
0003              C    TEST THE SUBROUTINES TV2CM AND TV2C4
0004              C
0005              C    KEITH MILLER, 7/17/84
0006              C
0007              C    LOAD MODULES: _TTV2C, _TV2C4, _TV2CM, _CMND, _DSPBW,
0008              C    _RANGE, _DIGIT, _WAIT, _ADD12, _DSPVD
0009              C
0010              C    WRITE(1,1001)
0011               1001 FORMAT(’MAKE SURE TV CAMERA IS SET TO IMAGE 5.’)
0012              C
0013              C    CALL WAIT
0014              C    CALL TV2C4
0015              C    STOP
0016              C
0017              C    END
PROGRAM TWAIT

TESTING THE WAIT SUBROUTINE

KEITH MILLER, JUNE 8, 1984

INTEGER IERR, IX, IY

WRITE(1,1000)

1000 FORMAT('START WAIT TEST')

CALL WAIT(IERR)

CALL KMRTA(IX,IY,IERR)

WRITE(1,2000)

2000 FORMAT('END WAIT TEST')

STOP

END
&TWIPE T=20004 IS ON CR88821 USING 02002 BLKS R=0000

0001    PROGRAM TWIPE
0002    C
0003    C    TESTS THE PROCEDURES DSPGR AND WIPGR
0004    C
0005    C    KEITH MILLER 7/30/84
0006    C
0007    C    LOAD MODULES: %TWIPE, %DSPGR, %WIPGR, %WAIT,
0008    C    %RANGE, %DIGIT, %CMMP2
0009    C
0010    CALL DSPGR(1)
0011    CALL DSPGR(2)
0012    CALL DSPGR(3)
0013    C
0014    CALL WAIT
0015    C
0016    CALL WIPGR(3)
0017    CALL WAIT
0018    CALL WIPGR(2)
0019    CALL WAIT
0020    CALL WIPGR(1)
0021    CALL WAIT
0022    CALL WIPGR(1)
0023    STOP
0024    END
&TXFDS J=00004 IS ON CR00021 USING 00001 BLKS R=0000

0001     PROGRAM TXFDS
0002     C
0003     C     TEST PROGRAM FOR BWFDS, CLFDS
0004     C
0005     C     KEITH MILLER, 7/12/84
0006     C
0007     C     LOAD MODULES: TXFDS, BWFDS, CLFDS, RANGE, WRILN, CMD, DIGIT
0008     C
0009     INTEGER FLNAME(3)
0010     CHARACTER*6 CNAME
0011     EQUIVALENCE (FLNAME,CNAME)
0012     C
0013     CNAME = 'CFXRAY"
0014     CALL BWFDS(1, FLNAME)
0015     C
0016     CNAME = 'CFM1AND'
0017     CALL CLFDS(1,3,8, FLNAME)
0018     STOP
0019     END
PROGRAM TXGLN

C TEST URGLN AND RDGLN

C LOAD MODULES REQUIRED: XTGLN, XRDGLN, XURGLN, X RANGE

INTEGER LINE ! number of line in image being processed

INTEGER IBUF(32) ! buffer to hold ONOFF values, one line/time

! note that 16 ONOFF bits fit in one INTEGER

DO 2000 LINE = 1, 512

CALL RDGLN(IBUF, 2, LINE)

CALL WRGLN(1, LINE, IBUF)

2000 CONTINUE

END
PROGRAM TXGPT

TEST RDGPT AND WRGPT

KEITH MILLER, 6/22/84

LOAD MODULES: TXGPT, RDGPT, WRGPT, RANGE, RDTAR

INTEGER INDEX, LOOP INDEX

DO 1000 INDEX = 1, 20
    CALL WRGPT(I, INDEX, INDEX, 1)
    CALL WRGPT(I, INDEX, 21-INDEX, 1)
1000 CONTINUE

CALL WAIT

DO 2000 INDEX = 1, 20
    CALL WRGPT(I, INDEX, INDEX, 0)
    CALL WRGPT(I, INDEX, 21-INDEX, 0)
2000 CONTINUE

DO 3000 INDEX = 1, 5
    CALL WAIT
    CALL RDTAR(I, IY)
    CALL RDGPT(IVALUE, 1, IX, IY)
    WRITE(1, 2001) IX, IY, IVALUE
3000 CONTINUE

STOP

END
PROGRAM TXILN

TEST URILN AND RDILN

LOAD MODULES REQUIRED: %TXILN,%RDILN,%URILN,%RANGE

INTEGER LINE ! number of line in image being processed
INTEGER IBUF(256) ! buffer to hold pixel values, one line/time

DO 2000 LINE = 0, 511
    CALL RDILN(IBUF, 1, LINE)
    CALL WRILN(2, LINE, IBUF)
2000 CONTINUE
&TXIPT T=00004 IS ON CR00021 USING 00002 BLKS R=0000

0001      PROGRAM TXIPT
0002      C
0003      C    TEST RDIPT AND WRIPT
0004      C
0005      C    KEITH MILLER, 6/21/84
0006      C
0007      C    LOAD MODULES: %TXIPT, %RDIPT, %WRIPT, %RANGE, %WAIT
0008      C
0009      C    INTEGER XCOORD, YCOORD, VALUE
0010      C
0011      CALL RDIPT(VALUE, 1, 200, 100)
0012      WRITE(1,1001) VALUE
0013      1001 FORMAT(1, ' VALUE =', I4)
0014      CALL WRIPT(1,200,100,005)
0015      CALL RDIPT(VALUE, 1, 200, 100)
0016      WRITE(1,1001) VALUE
0017      C
0018      STOP
0019      END
0020
0021
0022
0023
$TXIRC T=00004 IS ON CR00021 USING 00002 BLKS R=0000$

0001            PROGRAM TXIRC  
0002            TEST RDIRC (READ IMAGE RECTANGLE) & WRIRC (WRITE IMAGE RECTANGLE)  
0003            KEITH MILLER, JULY 3, 1984  
0004            LOAD MODULES: %TXIRC, %RDIRC, %WRIRC, %RDLN, %WRILN  
0005            INTEGER BUFFER(10,10)  
0006            INTEGER ROW, COL  
0007            CALL RDIRC(BUFFER, 10, 10, 1, 0, 0)  
0008            CALL WRIRC(1, 100, 100, BUFFER, 10, 10)  
0009            STOP  
0010            END  
0011  
0012

0013

0014

0015

0016

0017

0018
PROGRAM TXLUT

C TEST WRLUT AND RDLUT

C KEITH MILLER, 7/10/84

C LOAD MODULES: %TXLUT, %WRLUT, %RDLUT, %RANGE

INTEGER TABLE(256), TABLE2(256)
INTEGER INDEX

DO 1000 INDEX = 1, 256
    TABLE(INDEX) = 256-INDEX
1000 CONTINUE

CALL WRLUT(1, TABLE)

CALL RDLUT(TABLE2, 1)

DO 2000 INDEX = 1, 256
    WRITE (1, 1999) TABLE2(INDEX)
2000 CONTINUE

1999 FORMAT(14)

STOP
END
&TXPSU T=0004 IS ON CR00021 USING 05300 BLKS R=0000

0001 PROGRAM TXPSU
0002 C
0003 C TEST PSEUDO COLOR TABLE TRANSFERS: RDPSU AND WRPSU
0004 C
0005 C LOAD MODULES: %TXPSU, %RDPSU, %WRPSU, %RANGE, %WAIT
0006 C
0007 C KEITH MILLER, 7/10/84
0008 C
0009 C INTEGER TABLE(768)
0010 C INTEGER INDEX
0011 C
0012 C CALL RDPSU(TABLE)
0013 C
0014 C WRITE(1,999)
0015 999 FORMAT(' THE REDS:')
0016 DO 2000 INDEX=1,1256
0017 WRITE(1,999) TABLE(INDEX)
0018 1999 FORMAT(14)
0019 2000 CONTINUE
0020 C
0021 C CALL WAIT
0022 C
0023 C WRITE(1,2999)
0024 2999 FORMAT(' THE GREENS:')
0025 DO 4000 INDEX=257,512
0026 WRITE(1,3999) TABLE(INDEX)
0027 3999 FORMAT(14)
0028 4000 CONTINUE
0029 C
0030 C CALL WAIT
0031 C
0032 C WRITE(1,4999)
0033 4999 FORMAT(' THE BLUES:')
0034 DO 6000 INDEX=513,768
0035 WRITE(1,5999) TABLE(INDEX)
0036 5999 FORMAT(14)
0037 6000 CONTINUE
0038 C
0039 C DO 1000 INDEX = 1,1256
0040 TABLE(INDEX) = 256-INDEX
0041 TABLE(INDEX+256) = INDEX
0042 TABLE(INDEX+512) = 122
0043 1000 CONTINUE
0044 C
0045    CALL WRPSU(TABLE)
0046    STOP
0047    END
PROGRAM TXTAR

TEST RDTAR AND WRTAR

KEITH MILLER, 6/21/84

LOAD MODULES: %TXTAR, %RDTAR, %WRTAR, %RANGE

INTEGER XCOORD, YCOORD ! CURSOR COORDINATES
INTEGER INDEX ! LOOP INDEX

CALL URTAR(12,34) ! INITIAL POSITION OF TARGET

DO 1000 INDEX = 1, 20
   CALL RDTAR(XCOORD, YCOORD)
   WRITE(1,500)XCOORD, YCOORD
   500 FORMAT(215)
   CALL URTAR(INDEX, INDEX)
1000 CONTINUE

CALL WRTAR(513,1) ! SHOULD BE AN ERROR ON X
CALL WRTAR(0,1) ! SHOULD BE AN ERROR ON X
CALL WRTAR(512,-1) ! SHOULD BE AN ERROR ON Y
STOP
END
BP DRIVER, DVR41, SOURCE CODE

DOCUMENTATION ADDED AUG. 1984, JIM MONTEITH & KEITH MILLER.

TO INVOKE THIS DRIVER FROM A FORTRAN ROUTINE, MAKE AN EXEC
CALL LIKE THE ONE SHOWN BELOW. FOR MORE INFORMATION ON HOW
THE EXEC CALL WORKS, SEE CHAPTER 2 IN THE HP MANUAL
"RTE-6/VM PROGRAMMER'S REFERENCE MANUAL". FOR A LIST OF THE
EQUIPMENT TABLE WORDS AND BITS, SEE SECTION L OF THE QUICK
REFERENCE GUIDE FOR THE HP-1000.

SAMPLE CALL:

CALL EXEC (ICODE, LU, IFUNC, IBUFF, ILEN, ICMND)

ICODE: THIS PARAMETER IS EITHER A 1, 2, OR 3.
1: SIGNIFIES A READ OPERATION. (COMTAL -> HP)
2: SIGNIFIES A WRITE OPERATION. (HP -> COMTAL)
3: SIGNIFIES A CONTROL OPERATION: THIS MODE DOESN'T
   SEEM TO WORK IN OUR SYSTEM.

THE EXEC CALL SETS BITS 0 AND 1 OF WORD 6 OF THE
EQUIPMENT TABLE (EOT6 IN THE CODE BELOW) ACCORDING
TO THE ICODE VALUE.

LU : LOGICAL UNIT NUMBER. THE LOGICAL UNIT NUMBER FOR
THE COMTAL IS SET AT SYSTEM GENERATION. AT THE MOMENT
IT IS 368.

IFUNC: THIS PARAMETER SIGNALS THE DRIVER AS TO THE TYPE OF
HP <-> COMTAL COMMUNICATION THAT IS BEING REQUESTED.
ALTHOUGH THERE IS SOMETHING OF A RELATIONSHIP BETWEEN THE
IFUNC CODES AND THE TRANSFER CODES OF SECTION 5.2 IN THE
COMTAL USERS MANUAL, THE RELATIONSHIP IS HARD TO EXPLAIN.
SO WE WON'T TRY. INSTEAD, WE'LL GIVE A CHART THAT GIVES
THE IFUNC VALUES AND THE RELEVANT COMTAL USER MANUAL
SECTION NUMBERS WHERE FURTHER DETAILS ON THE TRANSFERS
ARE AVAILABLE. THE EXEC CALL STORES THE IFUNC CODE IN
IN BITS 6-10 OF WORD 6 OF THE EQUIPMENT TABLE (EQT6 IN
THIS DRIVER).

<table>
<thead>
<tr>
<th>IFUNC</th>
<th>COMTAL</th>
<th>MANUAL</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>000B</td>
<td>5.2.1</td>
<td>DISPLAY COMMANDS</td>
<td></td>
</tr>
<tr>
<td>100B</td>
<td>5.2.2</td>
<td>IMAGE/GRAPHIC TRANSFERS</td>
<td></td>
</tr>
<tr>
<td>200B</td>
<td>5.2.3</td>
<td>FUNCTION MEMORY (LOOK UP TABLE) TRANSFERS</td>
<td></td>
</tr>
<tr>
<td>300B</td>
<td>5.2.3</td>
<td>PSEUDO-COLOR TABLE TRANSFERS</td>
<td></td>
</tr>
<tr>
<td>400B</td>
<td>5.2.4.1</td>
<td>CURSOR AND TRACKBALL REQUESTS</td>
<td></td>
</tr>
<tr>
<td>500B</td>
<td>5.2.4.3</td>
<td>MACRO &amp; COMMAND BLOCK TRANSFERS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.2.4.4</td>
<td>IMAGE/GRAPHICS PARAMETER BLOCK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.2.4.6</td>
<td>CODE BLOCK TRANSFERS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.2.4.8</td>
<td>IMAGE/GRAPHICS TABLE READ</td>
<td></td>
</tr>
<tr>
<td>600B</td>
<td>5.2.2.3</td>
<td>LINK SUBFUNCTION ESTABLISHES A CORRESPONDENCE BETWEEN THE NUMBERS USED TO IDENTIFY HP CONTROLLED IMAGES AND THE INTERNAL COMTAL IMAGES. SINCE WE USE THE COMTAL 1 USERS AT A TIME, WE DON'T USE THIS SUBFUNCTION.</td>
<td></td>
</tr>
</tbody>
</table>

IBUFF: INTEGER ARRAY BUFFER WHICH IS USED TO FERRY DATA BETWEEN COMTAL AND THE HP. SOMETIMES THIS BUFFER PARAMETER IS IGNORED. THE EXEC CALL PLACES THE IBUFF ADDRESS IN SLOT 7 OF THE EQUIPMENT TABLE, AND THIS DRIVER REFERS TO IBUFF USING EQT7.

ILEN: INTEGER LENGTH, IN WORDS, OF IBUFF. THIS LENGTH IS STORED BY THE EXEC CALL IN WORD 8 OF THE EQUIPMENT TABLE, EQT8 IN THIS DRIVER.

ICMND: INTEGER CODE THAT IS USED TO GIVE CODED INFORMATION TO THE COMTAL. IN THE COMTAL MANUAL, THE BITS DESCRIBED IN THE RELEVANT 5.2 SUBSECTION ARE CODED INTO ICMND. THIS PARAMETER IS SOMETIMES IGNORED. ICMND IS STORED IN WORD 9 OF THE EQUIPMENT TABLE, EQT9 IN THIS DRIVER.

NOTES:
1. USES STANDARD INTERRUPTS FOR TRANSFER OF DEVICE COMMANDS AND TERMINATION SEQUENCES.
2. USES D.M.A. FOR TRANSFER OF DATA BLOCKS IF NECESSARY

3. THE TWO INTERFACE CARDS FOR THE COMTAL MUST BE MOUNTED IN CONSECUTIVE I/O SLOTS.

4. THE COMMAND CARD MUST BE THE LOW I/O SLOT OR SELECT CODE.

5. MANY BUGS WERE FOUND IN THE ORIGINAL AND THESE HAVE BEEN REMOVED AS WELL AS MAKING THE CODE A MORE EFFICIENT ROUTINE. R.W.B.

ENT I.41.C.41

INITIATION SECTION FOLLOWS

NOP
JSB SETIO
CLB
LDA EOT6.1 GET CONTROL SUBFUNCTION
AND =B3700
STA FUNC SAVE FUNCTION MEMORY
LDA EOT6.1 GET CONTROL WORD OF REQUEST
AND =B3 ISOLATE THE FUNCTION
CPA =B2 CHECK FOR WRITE (SKIP IF NOT WRITE)
JMP RORWI IF REQUEST FUNCTION = 2 (WRITE)
LDB BIT15 SET READ FLAG FOR ACTUAL BIT USED
CPA =B1 CHECK FOR READ (SKIP IF NOT READ)
JMP RORWI IF REQUEST FUNCTION = 1 (READ)

* ANYTHING ELSE IS CONSIDERED A CONTROL REQUEST
LDA FUNC GET CONTROL WORD
SZA
JMP REJCT FOR NOW ONLY CONTROL REQUEST "0" IS VALID
CLR1 CLC DAT.C
CLR2 CLC CST.C
LDA =B4
JMP I.41.I

REJCT LDA =B2 SET A=2 FOR ILLEGAL CONTROL REQUEST
JMP I.41.I RETURN TO IOC

*
0137 ******** SETIO ROUTINE FOR INITIATION SECTION FOLLOWS
0138 SETIO NOP
0139 STA B   SAVE SELECT CODE FROM A (STATUS CARD SC)
0140 IOR CLC  OR IN "CLC" INSTRUCTION
0141 STA CLR2
0142 STA RORW1
0143 INA    INCREMENT TO HIGHER SELECT CODE (DATA CARD SC)
0144 STA CLR1
0145 STA RORW
0146 LDA LIA "LIA INSTRUCTION" TO A
0147 IOR B   OR IN SELECT CODE
0148 STA RORW2
0149 LDA STC "STC INSTRUCTION" TO A
0150 IOR B
0151 STA IO1L3 STORE INSTRUCTION AT LABEL "IO1L3"   
0152 LDA OTA
0153 IOR B
0154 STA IO1L2
0155 LDA OTB
0156 INB    INCREMENT SELECT CODE IN B
0157 IOR B
0158 STA IO1L1
0159 JMP SETIO.I RETURN
0160 ************************************************************
0161 ********** READ OR WRITE REQUEST ****************************
0162 ************************************************************
0163 * CHECK FOR COMTAL IN REMOTE
0164 *
0165 RORW1 STB RFLAG
0166 RORU CLC DAT.C CLEAR CONTROL AND FLAG
0167 RORU1 CLC CST.C
0168 RORW2 LIA CST GET STATUS
0169 AND NBSY FIX FOR COMTAL VO/20 R.W.B.
0170 CPA NBSY FIX FOR COMTAL VO/20 R.W.B.
0171 RSS SKIP IF YES
0172 JMP RORW2 FIX FOR COMTAL VO 20 R.W.B.
0173 ************************************************************
0174 **************************** DETERMINE WHICH SUBFUNCTION ****************************
0175 ****************************
0176 *** NOTE THAT THE DISPLAY COMMAND AND TRACK BALL REQUESTS ARE
0177 * PERFORMED USING DEVICE INTERRUPTS.
0178 * HOWEVER IMAGE LINE, FUNCTION, AND PSEUDO-COLOR TRANSFERS ARE
0179 * PERFORMED USING DEVICE INTERRUPTS TO TRANSMIT THE COMMAND AND
0180 * ENDING SEQUENCE, BUT WITH A DMA TRANSFER IN BETWEEN.
0181 *
0182 LDA FUNC GET CONTROL WORD
0183 * CPA DSPLV DISPLAY COMMAND REQUEST?
0184 JMP DSPLY
0185 * CPA ILTRV IMAGE LINE TRANSFER?
0186 JMP CHDCP
0187 * CPA FMTRV FUNCTION MEMORY TRANSFER?
0188 JMP CHDCP
0189 * CPA PCMTV PSEUDO-COLOR TRANSFER
0190 JMP CHDCP
0191 * CPA TBALV TRACK BALL REQUEST
0192 JMP TBALL
0193 * CPA SUPTV CHECK FOR TYPE 3 DMA SUPPORT CONTROL
0194 JMP CHDCP
0195 * CPA LINKV CHECK FOR LINK SUBFUNCTION
0196 JMP LINK
0197 * CPA LSRTV PROCESS DISPLAY COMMAND REQUEST ***************
0198 * PROCESS TRACK BALL REQUEST ***************
0199 * PROCESS DISPLAY COMMAND REQUEST ***************
0200 DSPLY LDA =D1 SET STATE CONTROL VARIABLE
0201 STA EOT12.I WHAT TO DO
0202 LDB EOT9.I GET OPTION
0203 CLA CLEAR OUT A
0204 JMP 101 GO TO START 10 ROUTINE
0205 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0206 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0207 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0208 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0209 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0210 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0211 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0212 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0213 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0214 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0215 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0216 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0217 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0218 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0219 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0220 TBALL LDB RFLAG
0221 CPB BIT15 READ FLAG SET? (SKIP IF WRITE)
0222 JMP RTBAL JUMP IF SET FOR READ
0223 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0224 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0225 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0226 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0227 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0228 **** TRANSMIT WRITE COMMAND AND X-POSITION.
0229 LDA B.I CONTENTS OF BUFFER TO A
0230 AND =B0777 MASK FOR BITS 0-8
0231 STA B
0232 LDA =B3 TRANSFER CODE OF 3 TO A
0233 JMP 101 GO START I/O
0234 *
0235 **************************************** READ TRACK BALL ****************************************
0236 *** ALSO READS LITTLE RED SWITCHES ON TRACK BALL APPARATUS. 
0237 *
0238 RTBAL LDA =D5 SET FLAG TO TELL CONT. SECTION WHAT TO DO
0239 STA EQT12.I
0240 LDA =B3 TRANSFER CODE=3
0241 JMP 101 GO TO START I/O
0242 *
0243 *
0244 ******** CHECK FOR DCPC CHANNEL ASSIGNED **************
0245 *
0246 CHDCP DLD INTBA.I ACCESS DCPC ASSIGNMENT WORDS FROM THE
0247 * THE INTERRUPT TABLE WHOSE ADDRESS IS IN THE BASE PAGE
0248 * COMMUNICATIONS AREA.
0249 CPA EQT1 IS CHANNEL 1 ASSIGNED TO THIS DRIVER
0250 JMP CH1 IF YES
0251 CPB EQT1 IS CHANNEL 2 ASSIGNED TO THIS DRIVER
0252 JMP CH2 IF YES
0253 LDA =B5 SET RETURN CODE TO 5
0254 JMP 1.41.I RETURN TO IOC REQUESTING A DCPC CHANNEL
0255 CH1 LDB =B2 LOW SELECT CODE OF CHANNEL 1 TO B
0256 RSP SKIP
0257 CH2 LDB =B3 LOW SELECT CODE OF CHANNEL 2 TO B
0258 STB EQT11.I STORE IN EQUIPMENT TABLE FOR USE BY COMPLETION SECTION
0259 *
0260 ******** WHICH OF THE DMS SUBFUNCTIONS ***********
0261 *
0262 CHFTN LDA FUNC CONTROL WORD TO A
0263 CPA FMTRV
0264 JMP FMTR FUNCTION MEMORY TRANSFER
0265 CPA PCMTR
0266 JMP PCMTR PSEUDO COLOR TRANSFER
0267 CPA SUPTV
0268 JMP SUPPT SUPPORT TRANSFER
0269 *
0270 ******** IMAGE LINE TRANSFER ROUTINE **********************
0271 *
0272 ILTR LDA =D8 SET STATE CONTROL VARIABLE
0273 STA EQT12.I
0274 LDA EQT9.I GET INSTRUCTIONS FOR LINE TRANSFER
0275 IOR RFLAG SET READ OR WRITE AS PER READ FLAG
0276 ILTR2 STA B
0277 LDA =B1 SET TRANSFER CODE TO 1
0278 STA TRAN SAVE TRANSFER CODE
0279 JMP 101 GO START I/O
0280 * FUNCTION MEMORY OR PSUEDO COLOR TRANSFER
0281 * CLEAR A FOR PSUEDO COLOR TRANSFER
0282 PCMTR CLA CONTINUE TRANSFER
0283 STAB
0284 JMP PCMT2 CONTINUE TRANSFER
0285 *
0286 FMTR LDA BIT14 SET FUNCTION MEMORY BIT
0287 *
0288 PCMT2 LDB =D8 SET STATE VARIABLE
0289 STB EOT12,I SAVE STATE FOR CONTINUATION
0290 IOR EOT9,I MERGE IN IOPT1 WORD
0291 IOR BIT13 MERGE IN 8000S COMPATIBILITY BIT
0292 IOR RFLAG MERGE IN READ OR WRITE BIT
0293 FMTR2 STA B PLACE CONTROL WORD IN B
0294 LDA =B2 SET TRANSFER CODE
0295 STA TRAN SAVE TRANSFER CODE FOR LATER
0296 JMP 101
0297 * SUPPORT FUNCTION BLOCK TRANSFER (TR CODE = 3)
0298 * SET DMA TRANSFER RETURN STATUS FOR CONTINUATION SECTION
0299 *
0300 SUPPT LDA =D8 SET DMA TRANSFER RETURN STATUS FOR CONTINUATION SECTION
0301 STA EOT12,I AND SAVE IT
0302 LDA EQT9,I GET OPTIONS
0303 IOR RFLAG SET READ OR WRITE BIT
0304 STA B SAVE IN B
0305 LDA =B3 SET TRANSFER CODE
0306 STA TRAN SAVE TRANSFER CODE
0307 JMP 101 AND GO DO IT
0308 *
0309 * SPECIAL PROCESSING FOR TR 1 MODE 6 CALL
0310 *
0311 LINK LDA =D1 SET NO BLOCK TRANSFER
0312 STA EOT12, I SAVE THIS
0313 LDB EQT9,I GET COMMAND OPTIONS
0314 LDA =B1 SET TRANSFER CODE
0315 JMP 101 GO DO IT
0316 *
0317 *
0318 *
0319 * THIS ROUTINE STARTS THE DATA TRANSFER OF A COMMAND
0320 * TO THE COMTAL

206
0321 *
0322 *
0323 101  NOP
0324 I01L1 OTB DAT  PUT TRANSFER COMMAND ON DATA LINES
0325 IOR GOBIT
0326 I01L2 OTA CST  PUT TRANSFER CODE ON COMMAND LINES
0327 I01L3 STC CST.C  SEND THE GO PULSE, START TRANSFER
0328 CLA  NOW RETURN TO IOC WITH
0329 JMP I.41.1  OPERATION INITIATED (A=0=OK)
0330 *  WHEN THE COMMAND IS RECEIVED BY THE COMTAL, AN
0331 *  INTERRUPT FROM THE COMTAL WILL CALL CIC AND THIS
0332 *  DRIVERS COMPLETION AND CONT. SECTION WILL EXECUTE.
0333 *
0334 *
0335 *
0336 **************  COMPLETION SECTION FOLLOWS **************
0337 *
0338 *
0339 *
0340 C.41  NOP
0341 JSB CS10
0342 LDA EQT1.1  CHECK FOR SPURIOUS INTERRUPT
0343 AND *B?????  GET I/O REQUEST LIST POINTER
0344 S2A  IS A REQUEST IN PROGRESS
0345 JMP COMP2  IF YES GO PROCESS REQUEST
0346 STA EQT15.1  NO, ITS SPURIOUS SO ZERO TIME-OUT CLOCK TO PREVENT TIME-OUT
0347 SPUR2 I5Z C.41  ADJUST RETURN TO P+2 (CONT.)
0348 JMP C.41.1  MAKE CONTINUATION RETURN TO CIC
0349 *
0350 *
0351 COMP2  NOP
0352 LDA CTABA  STATE CONTROL TABLE BASE ADDRESS TO A
0353 ADA EQT12.1  ADD THE STATE CONTROL VARIABLE
0354 JMP A.I  JUMP TO ADDRESS JUMP TABLE
0355 *
0356 ** DISPLAY COMMAND COMPLETION.
0357 *
0358 SCA1 LDA IDSC  INTERRUPTING DEVICE SELECT CODE TO A
0359 CPA SSC  IS INTERRUPTING SELECT CODE THE STATUS CARD?
0360 RSS  SKIP IF YES?
0361 JMP SPUR2  JUMP TO SPURIOUS INTERRUPT IF NOT
0362 LDB =B1  SET B FOR TRANS. LOG
0363 JMP CEND1
0364 *
0365 ** TRACK BALL 1
0366 *

0325 IOR GOBIT
0326 I01L2 OTA CST
0327 I01L3 STC CST.C
0328 CLA
0329 JMP I.41.1
0330 *
0331 *
0332 *
0333 *
0334 *
0335 *
0336 **************
0337 *
0338 *
0339 *
0340 C.41  NOP
0341 JSB CS10
0342 LDA EQT1.1
0343 AND *B?????
0344 S2A
0345 JMP COMP2
0346 STA EQT15.1
0347 SPUR2 I5Z C.41
0348 JMP C.41.1
0349 *
0350 *
0351 COMP2  NOP
0352 LDA CTABA
0353 ADA EQT12.1
0354 JMP A.I
0355 *
0356 ** DISPLAY COMMAND COMPLETION.
0357 *
0358 SCA1 LDA IDSC
0359 CPA SSC
0360 RSS
0361 JMP SPUR2
0362 LDB =B1
0363 JMP CEND1
0364 *
0365 ** TRACK BALL 1
0366 *

207
0367 SCA2 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0368 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0369 RSS SKIP IF YES?
0370 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT RETURN IF NOT
0371 LDA =D3
0372 STA EOT12.I SET STATE CONTROL VAR.
0373 LDB EOT7.I ADDRESS OF BUFFER TO B
0374 INB INCREMENT B (BUFFER ADDRESS)
0375 LDA B=1 CONTENTS OF BUFFER+1 TO A
0376 AND =B0777 SIGNIFYS Y-POSITION
0377 IOR BIT14
0378 STA B
0379 LDA =B3
0380 IOR GOBIT
0381 JMP C101 TRANSFER CODE OF 3
0382 *
0383 *** TRACK BALL 2
0384 *
0385 SCA3 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0386 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0387 RSS SKIP IF YES?
0388 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT SET
0389 LDA =D4
0390 STA EOT12.I SET STATE CONTROL VARIABLE
0391 JMP TBDUN
0392 *
0393 *** TRACK BALL 3
0394 *
0395 SCA5 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0396 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0397 RSS SKIP IF YES?
0398 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0399 LDA =D6
0400 STA EOT12.I SET STATE CONTROL VARIABLE
0401 JMP C102 JUMP TO START A READ
0402 *
0403 *** TRACK BALL 4
0404 *
0405 SCA6 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0406 CPA DSC IS INTERRUPTING SELECT CODE THE DSCA CARD?
0407 RSS SKIP IF YES?
0408 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0409 LDA =D7
0410 STA EOT12.I SET STATE CONTROL VARIABLE
0411 GET1 LIA DAT DATA FROM DATA LINES TO A (X-POSITION &SWITCH 1)
0412 ** TRACK BALL 5
0413 *
0413 * AND =B0777
0414 LDB EQT7.1 ADDRESS OF BUFFER TO B
0415 STA B.1 STORE X-POSITION IN BUFFER
0416 JMP CI02
0417 *
0418 *** TRACKBALL 5
0419 *
0420 SCA7 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0421 CPA DSC IS INTERRUPTING SELECT CODE THE DSCA CARD?
0422 RSS SKIP IF YES?
0423 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0424 LDA =D4
0425 STA EQT12.1 SET STATE CONTROL VARIABLE
0426 GET2 LIA DAT DATA FROM DATA LINES TO A
0427 LDB EQT7.1 GET BUFFER ADDRESS
0428 INB POINT TO WORD TWO
0429 STA B.1 STORE Y-POSITION IN BUFFER+1
0430 JMP TBDUN
0431 *
0432 TBDUN LDA =B3 THIS ADDED P0ER RUB 6/15/81
0433 IOR CLBIT
0434 IOR ENBIT
0435 JMP CI03
0436 *
0437 *** TRACK BALL TRANSMIT END
0438 *
0439 SCA4 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0440 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0441 RSS SKIP IF YES?
0442 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0443 LDB =B2 SET TRANS. LOG FOR WRITE
0444 LDA EQT6.1 GET CONTROL WORD
0445 AND =B3 ISOLATE FUNCTION CODE
0446 CPA =B1 IS IT A READ?
0447 LDB =B4 SET TRANS. LOG TO 4 FOR READ
0448 JMP CEND1
0449 *
0450 *
0451 *
0452 CI01 NOP
0453 OTB DAT
0454 CI03 OTA.CST
0455 STC CST.C
0456 ISZ C.41
0457 JMP C.41.I
0458 *
0459  *
0460  *
0461 C102 NOP
0462 CLC DAT.C
0463 STC DAT.C
0464 ISZ C.41
0465 JMP C.41.I
0466  *
0467 ** DMA TRANSFER 1
0468  *
0469 SCA8 LDA IDSC INTERRUPTING DEVICE SELECT CODE TO A
0470 CPA SSC IS INTERRUPTING SELECT CODE THE STATUS CARD?
0471 RSS SKIP IF YES?
0472 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
0473 LDA =D9
0474 STA EOT12.I SET STATE CONTROL VARIABLE
0475  *
0476  *
0477  *
0478 ** THIS ROUTINE MODIFIES THE DCPC INITIALIZATION INSTRUCTIONS
0479  TO SPECIFY THE SELECT CODES OF THE ASSIGNED DCPC
0480  *
0481 STDMA LDB EOT1.I GET LOW SELECT CODE FROM EOT
0482 LDA OTA "OTA INSTRUCTION" TO A
0483 IOR B OR IN THE LOWER SELECT CODE
0484 STA D3 STORE AT LABEL D3
0485 STA D5 STORE AT LABEL D5
0486 ADA =B4 ADD 4 TO INSTR. TO CHANGE TO HIGHER SELECT CODE
0487 STA D1
0488  *
0489 LDA CLC "CLC INSTR." TO A
0490 IOR B OR IN LOWER SELECT CODE
0491 STA D2
0492 LDA STC
0493 IOR B
0494 STA D4
0495 ADA =B4
0496 STA D6
0497 STA D7
0498 LDA CLC
0499 IOR DSC
0500 STA D6+1
0501 LDA STF
0502 IOR DSC
0503 STA D6+2
0504 LDA STC
8505 IOR DSC
8506 STA D7+1
8507 */
8508 */
8509 */
8510 ******** THIS ROUTINE INITIATES THE DCPC DATA TRANSFERS ********
8511 */
8512 */
8513 */
8514 */
8515 GOIO LDA EDT4.I
8516 AND =B77 MASK FOR COMMAND SELECT CODE
8517 INA INCREMENT FOR DATA SELECT CODE
8518 IOR BIT15 TURN ON HANDSHAKE BIT (STC)
8519 D1 OTA 6B PUT CONTROL WORD 1 TO DCPC CHANNEL (SELECT 6 OR 7)
8520 D2 CLC 2B PREPARE MEM. ADDR. REGISTER FOR CW2 (SELECT CODE 2 OR 3)
8521 LDA EDT6.I GET CONTROL WORD
8522 AND =B3 MASK FOR FUNCTION
8523 STA B STORE FUNCTION IN B
8524 LDA EDT7.I ADDRESS OF BUFFER TO A
8525 CPB =B1 IS FUNCTION A READ?
8526 IOR BIT15 TURN ON "IN" BIT FOR A READ, IF YES
8527 D3 OTA 2B CONTROL WORD 2 TO DCPC CHANNEL (SC 2 OR 3)
8528 D4 STC 2B PREPARE MEM. ADDR. REGISTER FOR CW3 (SC 2 OR 3)
8529 LDA EDT8.I GET LENGTH OF TRANSFER
8530 CLC, INA MAKE TWO'S COMPLEMENT
8531 D5 OTA 2B CONTROL WORD 3 TO DCPC CHANNEL (SELECT CODE 2 OR 3)
8532 CPB =B1 IS IT A READ?
8533 JMP D7 JUMP IF YES
8534 D6 STC 6B ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
8535 CLC DAT CLEAR DEVICE
8536 STF DAT ACTIVATE DEVICE?
8537 JMP GOEND
8538 D7 STC 6B ACTIVATE DCPC CHANNEL (SELECT CODE 6 OR 7)
8539 STC DAT,C ACTIVATE DEVICE
8540 GOEND ISZ C.41 NOW RETURN TO CIC WITH
8541 JMP C.41.I CONTINUATION
8542 */
8543 *** DMA TRANSFER 2
8544 */
8545 SCA9 LDA EDT11.I GET LOW SELECT CODE OF DCPC CHANNEL
8546 ADA =B4 ADD 4 TO CONVERT LOW DCPC SELECT CODE TO HIGH S.C.
8547 CHKDF CPA IDSC IS INTERRUPTING SELECT CODE THAT OF THE
8548 */
8549 RSS SKIP IF YES
8550 JMP SPUR2 JUMP TO SPURIOUS INTERRUPT IF NOT
LDA =D10      SET CONTROL STATE VARIABLE
STA EQT12.1

LDA TRAN
IOR CLBIT   * SET CLEAR BIT
IOR ENBIT   * ADD END BIT
JMP C103    * GO TO IT

** DMA TRANSFER 3

LDA IDC8   INTERRUPTING DEVICE SELECT CODE TO A
CPA SSC    IS INTERRUPTING SELECT CODE THE STATUS CARD?
RSS        SKIP IF YES?
JMP SPUR2  IF NOT GO TO SPURIOUS INTERRUPT
LDB EQT11.I RETRIEVE LOWER SELECT CODE OF ASSIGNED DCPC CHANNEL
LDA CLC    "CLC INSTRUCTION" TO A
IOR B      OR IN THE CHANNELS LOWER SELECT CODE
STA CD1    STORE INSTRUCTION AT LABEL CD1
ADA =B4    ADD 4 TO INSTRUCTION IN A TO ADJUST TO HIGHER SELECT CODE
STA CD2    STORE INSTRUCTION AT LABEL CD2

CD1 CLC 2B.C CLEAR LOW DCPC SELECT CODE
CD2 CLC 6B.C CLEAR HIGH DCPC SELECT CODE
LDB EQT8.I LENGTH OF BUFFER TO B (TRANSMISSION LOG)
CEND1 LIA CST GET STATUS WORD FROM COMTAL
AND =B37    STRIP OFF UNUSED BITS
STA SAVE1   SAVE IN SAVE1 TEMPORARILY
LDA EQT5.I REMOVE PREVIOUS STATUS
AND =B177480 BITS IN EQT WORD 5
IOR SAVE1   OR IN NEW BITS
STA EQT5.I AND RESET INTO EQT WORD 5

CEND2 CLC DAT.C CLEAR DEVICE DATA SELECT CODE
CLC CST.C CLEAR DEVICE COMMAND SELECT CODE

CLA        SET A = 0 = OK RETURN CODE
IOR =B100000 SET BIT TO RETURN DCPC CHANNEL
JMP C.41.I MAKE COMPLETION RETURN TO CIC

SETIO SUBROUTINE FOR COMPLETION/CONTINUATION SECTION *****
0597 *
0598 CSIO NOP
0599 STA IDSC  SAVE SELECT CODE OF INTERRUPTING DEVICE
0600 LDA EOT4.I
0601 AND #B77  MASK FOR STATUS SELECT CODE
0602 STA SSC  SAVE STATUS SELECT CODE AT SSC
0603 INA    ADD 1 TO A FOR DATA SELECT CODE
0604 STA DSC  SAVE DSCA SELECT CODE AT DSC
0605 LDA OTA  "OTA INSTRUCTION" TO A
0606 IOR SSC  OR IN STATUS SELECT CODE
0607 STA CI03  STORE INSTRUCTION IN A AT LABEL CI03
0608 LDA STC
0609 IOR SSC
0610 STA CI03+1
0611 INA
0612 *
0613 STA CI02+2
0614 LDA CLC
0615 IOR SSC
0616 STA CEND2+1
0617 INA
0618 STA CEND2
0619 STA CI02+1
0620 LDA LIA
0621 IOR SSC
0622 STA CEND1
0623 INA
0624 STA GET1
0625 STA GET2
0626 LDA OTB
0627 IOR DSC
0628 STA CI01+1
0629 JMP CSIO.1  RETURN
0630 *
0631 *
0632 *
0633 *
0634 *
0635 *
0636 *******  CONSTANTS AND STORAGE AREA  *******
0637 *
0638 RFLAG BSS .1
0639 FUNC  BSS 1
0640 TRAN BSS 1
0641 SAVE1 BSS 1
0642 SAVE2 BSS 1
**0661** **THE FOLLOWING VALUES (SUFFIX V) DETERMINE WHICH DRIVER**
**0662** **SUBFUNCTIONS ARE INVOKED, PARAMETER IFUNC IN SAMPLE CALL ABOVE.**
**0663** **THE FOLLOWING VALUES (SUFFIX V) DETERMINE WHICH DRIVER SUBFUNCTIONS ARE INVOKED, PARAMETER IFUNC IN SAMPLE CALL ABOVE.**

**0664** DSPLV OCT 0
**0665** ILTRV OCT 100
**0666** FMTRV OCT 200
**0667** PCMTV OCT 300
**0668** TBALV OCT 400
**0669** SUPTV OCT 500
**0670** LINKV OCT 600

**0671** **THE FOLLOWING VALUES (SUFFIX V) DETERMINE WHICH DRIVER SUBFUNCTIONS ARE INVOKED, PARAMETER IFUNC IN SAMPLE CALL ABOVE.**

**0672** CST EQU 24 **USED ONLY FOR DOCUMENTATION. ACTUAL INSTRUCTION IS**
**0673** DAT EQU 25 **SET UP BY A SETIO ROUTINE.**
**0674** A EQU 0
**0675** D EQU 1
**0676** OTA OTA 0
**0677** CLC CLC 0.C
**0678** STC STC 0.C
**0679** LIA LIA 0
**0680** OTB OTB 0
**0681** SFS SFS 0
**0682** STF STF 0

**0683** **THE FOLLOWING VALUES (SUFFIX V) DETERMINE WHICH DRIVER SUBFUNCTIONS ARE INVOKED, PARAMETER IFUNC IN SAMPLE CALL ABOVE.**

**0684** . EQU 16508
**0685** *
**0686** **THE FOLLOWING CONSTANTS ARE USED TO READ THE EQUIPMENT TABLE**
**0687** **ASSOCIATED WITH THIS DRIVER. SEE THE HP-1000 QUICK REFERENCE**
** Manual for Diagrams and Tables on Which These Constants Are Based. **

691 INTBA EQU .+4
692 E0T1 EQU .+8
693 E0T2 EQU .+9
694 E0T3 EQU .+10
695 E0T4 EQU .+11
696 E0T5 EQU .+12
697 E0T6 EQU .+13
698 E0T7 EQU .+14
699 E0T8 EQU .+15
700 E0T9 EQU .+16
701 E0T10 EQU .+17
702 E0T11 EQU .+18
703 E0T12 EQU .+81
704 E0T13 EQU .+82
705 E0T14 EQU .+83
706 E0T15 EQU .+84
707 *
708 *
709 *
710 ****** State Control Address Table ******
711 CTABA DEF CTAB
712 CTAB NOP
713 JMP SCA1 DISPLAY COMMAND COMPLETION
714 JMP SCA2 TRACK BALL 1
715 JMP SCA3 TRACK BALL 2
716 JMP SCA4 TRACK BALL TRANSMIT END
717 JMP SCA5 TRACK BALL 3
718 JMP SCA6 TRACK BALL 4
719 JMP SCA7 TRACK BALL 5
720 JMP SCA8 DMA TRANSFER 1
721 JMP SCA9 DMA TRANSFER 2
722 JMP SCA10 DMA TRANSFER 3
723 *
724 BSS 10
725 END
IPLIB is a collection of HP FORTRAN 77 subroutines and functions that facilitate the use of a COMTAL image processing system driven by an HP-1000 computer. It is intended for programmers who want to use the HP 1000 to drive the COMTAL Vision One/20 system. It is assumed that the programmer knows HP 1000 FORTRAN 77 or at least one FORTRAN dialect. It is also assumed that the programmer has some familiarity with the COMTAL Vision One/20 system.